Unit-3 Analysis

Analysis

- Analysis is the first systems development life cycle (SDLC) phase where you begin to understand, in depth, the need for system changes. Systems analysis involves a substantial amount of effort and cost, and is therefore undertaken only after management has decided that the systems development project under consideration has merit and should be pursued through this phase. Most observers would agree that many of the errors in developed systems are directly traceable to inadequate efforts in the analysis and design phases of the life cycle.

The purpose of analysis is to determine what information and information processing services are needed to support selected objectives and functions of the organization. Gathering this information is called requirements determination.

The results of the requirements determination can be structured according to three essential views of the current and replacement information systems:

Process: The sequence of data movement and handling operations within the system.

Logic and timing. The rules by which data are transformed and manipulated and an indication of what triggers data transformation.

Data: The inherent structure of data independent of how or when they are processed

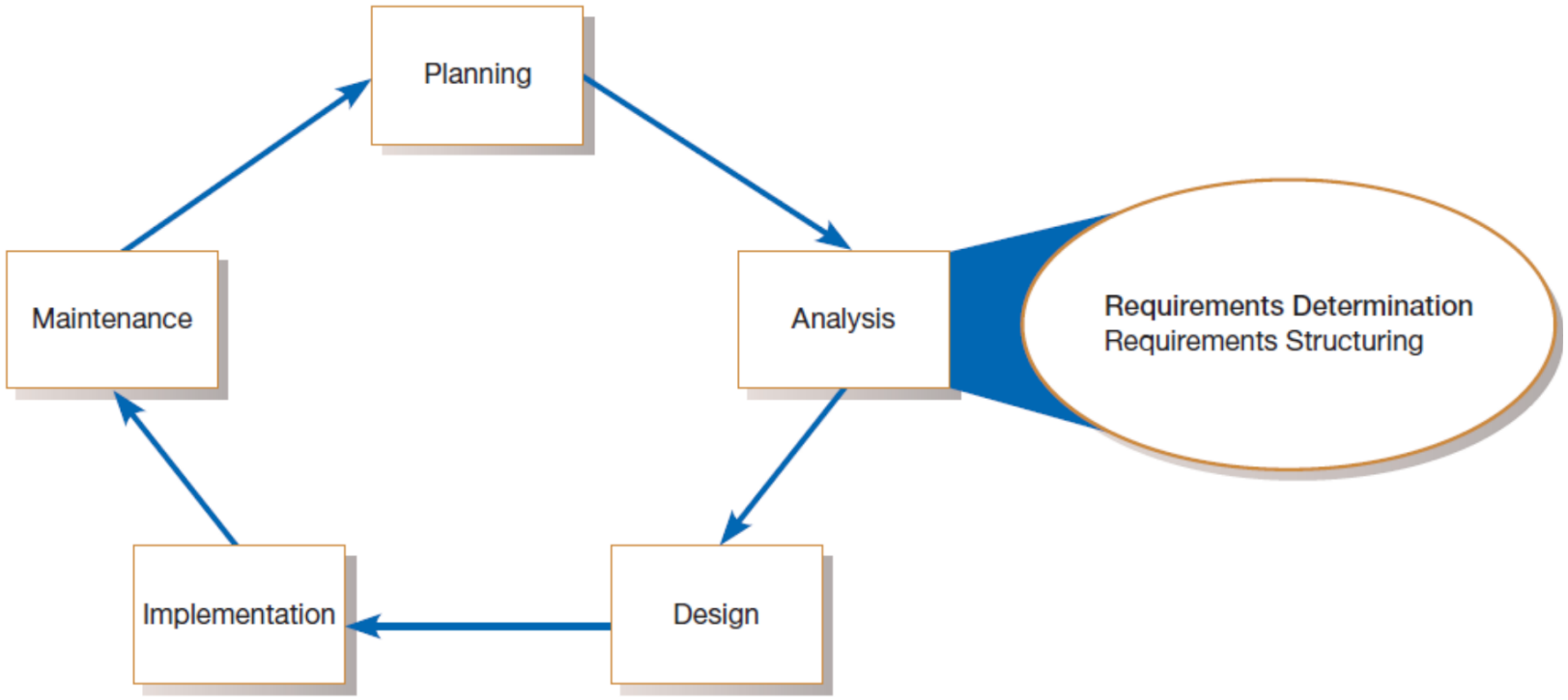


FIGURE 6-1
Systems development life cycle with analysis phase highlighted.

Determining System Requirements

Introduction:

Systems analysis is the part of the systems development life cycle in which you determine how the current information system functions and assess what users would like to see in a new system.

Analysis has two sub phases: requirements determination and requirements structuring.

- Requirements determination and requirements structuring are two core components of system analysis.
- Traditionally, interviewing, questionnaires, directly observing and analyzing documents are four main methods adopted by system analysts to collect information.
- JAD and prototyping are two modern requirements determination methodologies, which are developed and based on the previous traditional methods.
- A well-structured representation of system requirements can dramatically improve the communication among analysts, designers, users, and programmers .
- DFD, structured English, decision tables, decision trees, and E-R diagrams are traditional primary requirements structuring tool

Characteristic	Interviews	Questionnaires	Observation	Document analysis	JAD	Prototyping
Information Richness	High	Medium to low	High	Low (passive) and old	High	Medium to High
Time Required	Can be extensive	Low to moderate	Can be extensive	Low to moderate	Dedicated period of time of all kinds of involved people	Moderate and can be extensive
Expense	Can be high	Moderate	Can be high	Low to moderate	High	High
Chance for Follow-up and probing	Good	Limited	Good	Limited	Good	Good
Confidentiality	Interviewee is known to interviewer	Respondent can be unknown	Observee is known to interviewer	Depends on nature of document	All the people know each other	Usually know each other
Involvement of Subject	Interviewee is involved and committed	Respondent is passive, no clear commitment	Interviewees may or may not be involved and committed depending on whether they know if they are being observed	None, no clear commitment	All kinds of people are involved and committed	Users are involved and committed
Potential Audience	Limited numbers, but complete responses from those interviewed	Can be quite large, but lack of response from some can bias results	Limited numbers and limited time of each	Potentially biased by which documents were kept or because document not created for this purpose	Potentially biased by the subordinator intentionally don't want to directly point out his superior's errors.	Limited numbers; it is difficult to diffuse or adapt to other potential users

The Process Of Determining Requirements

Once management has granted permission to pursue(follow or start) development of a new system (this was done at the end of the project identification and selection phase of the SDLC) and a project is initiated and planned, you begin determining what the new system should do. During requirements determination, you and other analysts gather information on what the system should do from as many sources as possible: from users of the current system; from observing users; and from reports, forms, and procedures. All of the system requirements are carefully documented and prepared for structuring. In many ways, gathering system requirements is like conducting any investigation.

These characteristics include the following:

• **Impertinence:** You should question everything. You need to ask questions such as: Are all transactions processed the same way? Could anyone be charged something other than the standard price? Might we someday want to allow and encourage employees to work for more than
one department?

• Impartiality: Your role is to find the best solution to a business problem or opportunity. It is not, for example, to find a way to justify the purchase of new hardware or to insist on incorporating what users think they want into the new system requirements. You must consider issues raised by all parties and try to find the best organizational solution.

• Relax constraints: Assume that anything is possible and eliminate the infeasible. For example, do not accept this statement: “We’ve always done it that way, so we have to continue the practice.” Traditions are different from rules and policies. Traditions probably started for a good reason but, as the organization and its environment change, they may turn into habits rather than sensible procedures.

• Attention to details: Every fact must fit with every other fact. One element out of place means that even the best system will fail at some time. For example, an imprecise (not accurate or exact) definition of who a customer is may mean that you purge (remove) customer data when a customer has no active orders, yet these past customers may be vital contacts for future sales.

• Reframing: Analysis is, in part, a creative process. You must challenge yourself to look at the organization in new ways. You must consider how each user views his or her requirements. You must be careful not to jump to the following conclusion: “I worked on a system like that once—thisnew system must work the same way as the one I built before.

TABLE 6-1 Deliverables for Requirements Determination

1. Information collected from conversations with or observations of users: interview transcripts, notes from observation, meeting minutes
2. Existing written information: business mission and strategy statements, sample business forms and reports and computer displays, procedure manuals, job descriptions, training manuals, flowcharts and documentation of existing systems, consultant reports
3. Computer-based information: results from JAD sessions, CASE repository contents and reports of existing systems, and displays and reports from system prototypes

Traditional methods for determining requirements

TABLE 6-2 Traditional Methods of Collecting System Requirements

- Individually interview people informed about the operation and issues of the current system and future systems needs
- Interview groups of people with diverse needs to find synergies and contrasts among system requirements
- Observe workers at selected times to see how data are handled and what information people need to do their jobs
- Study business documents to discover reported issues, policies, rules, and directions as well as concrete examples of the use of data and information in the organization

Contemporary Methods For Determining system Requirements

TABLE 6-5 Contemporary Methods for Collecting System Requirements

- Bringing session users, sponsors, analysts, and others together in a JAD session to discuss and review system requirements
- Using *CASE tools* during a JAD to analyze current systems to discover requirements that will meet changing business conditions
- Iteratively developing system *prototypes* that refine the understanding of system requirements in concrete terms by showing working versions of system features

Requirements structuring

- Requirements structuring is the process to use some kind of systematical and standard, well-structured methods to model the real world.

- Traditionally, we use data flow diagram for process modeling, decision table or decision tree for logic modeling, and Entity-relationship diagram for data modeling.

- These modeling tools usually separately model only one face of the real world. So, when we try to show the integral picture of a system, we usually choose more than one of the above requirements structuring methods.

Traditionally, there are three basic methods for requirements structuring:

- Data flow diagrams, it is widely used for process modeling
- Structure English, decision tables, and decision trees, they are used for logic modeling
- Entity and relationship diagram, it is used for data modeling

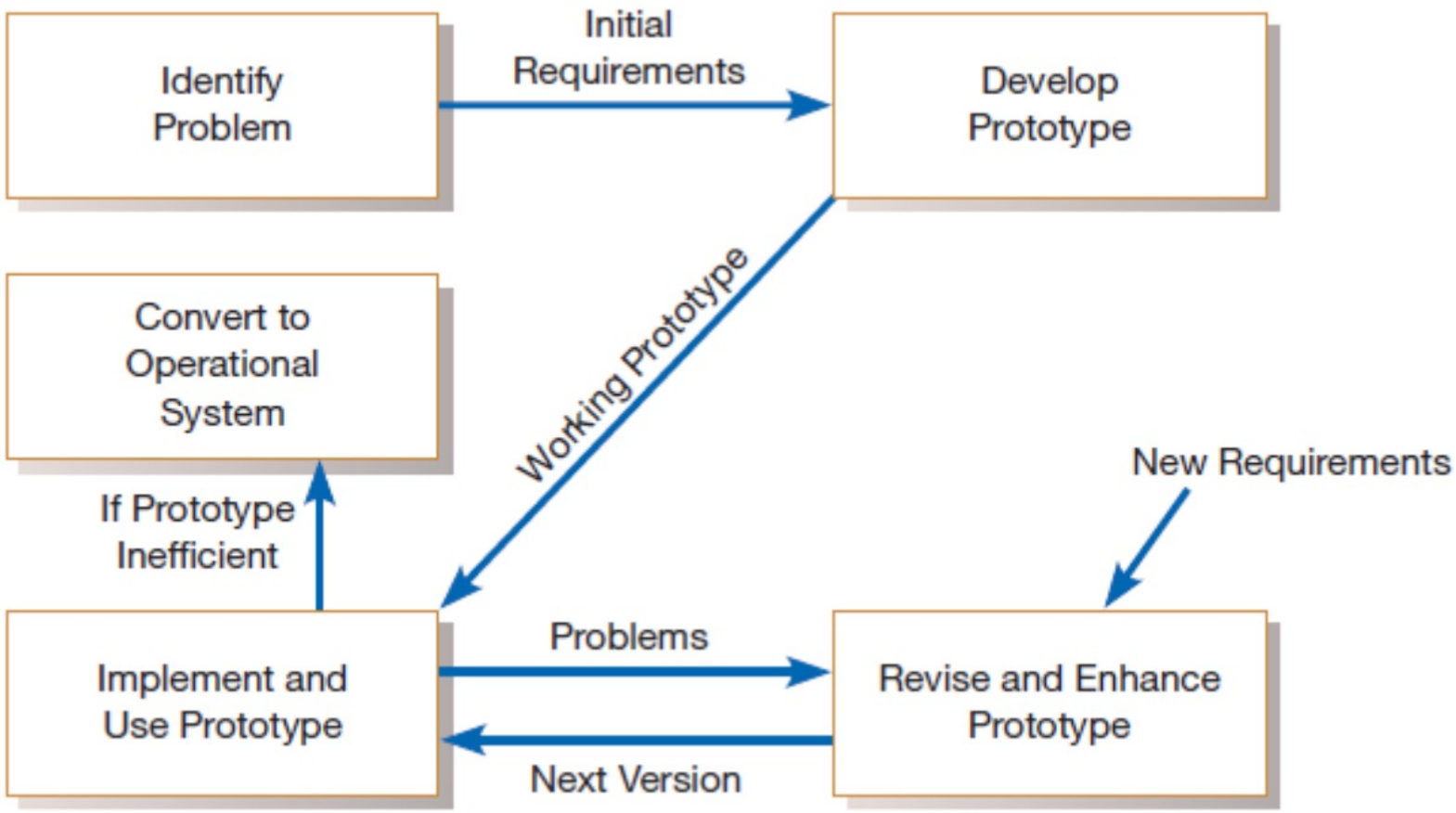
Joint Application Design (JAD)

A structured process in which users, managers, and analysts work together for several days in a series of intensive meetings to specify or review system requirements.

The main idea behind JAD is to bring together the key users, managers, and systems analysts involved in the analysis of a current system. The primary purpose of using JAD in the analysis phase is to collect systems requirements simultaneously from the key people involved with the system. The result is an intense (extreme and forceful or (of a feeling) very strong) and structured, but highly effective, process. As with a group interview, having all the key people together in one place at one time allows analysts to see where there are areas of agreement and where there are conflicts. Meeting with all of these important people for over a week of intense sessions allows you the opportunity to resolve conflicts, or at least to understand why a conflict may not be simple to resolve.

Prototyping

An iterative process of systems development in which requirements are converted to a working system that is continually revised through close collaboration between an analyst and users.



Logic Modeling

- A logic modeling is a graphic roadmap that presents the shared relationships among the resources, activities, outcomes, and impact of your program.
- DFD don't show the logic inside the processes – what occurs within a process? How input data is converted into output information.
- **Logic modeling** can also be used to show when processes on a DFD occur.
- Logic model has four components:
- **1. Needs** are about the problem and why it's important to address it. What issue are we trying to address?
- **2. Inputs** are the things that contribute to addressing the need (usually a combination of money, time, expertise and resources). What resources are we investing?
- **3. Activities** describe the things that the inputs allow to happen. What are we doing with these resources?
- **4. Outcomes** are usually expressed in terms of measures of success. What difference are we hoping to make?

Logic Modeling

- Each Process on the lowest level DFD will be represented by one or more of the following:

- **1. Structured English**
- **2. Decision Tables**
- **3. Decision Trees**
- **4. State-transition diagrams**
- **5. Sequence diagrams**
- **6. Activity diagrams**

Food Prompt ?	NO	Yes	NO	Yes
Food Tasty ?	No	No	Yes	Yes

Actions

Complain	Yes	No	NO	No
Tell Friends	Yes	NO	Yes	Yes
Write a review	No	No	No	Yes

Modeling Logic With Decision Tables

- Decision tables are a concise / short / brief visual representation for specifying which actions to perform depending on given conditions.
- They are algorithms whose output is a set of actions.
- A **decision table** is an excellent tool to use in both testing and requirements management.
- **Decision Table** is a structured exercise to formulate requirements when dealing with complex business rules.
- **Decision Tables** are used to model complicated logic.
- Lets take an example scenario for an ATM where a decision table would be of use.
- A customer requests a cash withdrawal. One of the business rules for the ATM is that the ATM machine pays out the amount if the customer has sufficient funds in their account or if the customer has the credit granted.

Modeling Logic With Decision Tables

- This simple example of a business rule is quite complicated to describe in text. A decision table makes the same requirements clearer to understand:

Conditions	R1	R2	R3
Withdrawal amount <= balance	T	F	F
Credit granted	-	T	F
Actions			
Withdrawal granted	T	T	F

- In a decision table, conditions are usually expressed as true (T) or false (F). Each column in the table corresponds to a rule in the business logic that describes the unique combination of circumstances that will result in the actions.

Modeling Logic With Decision Tables

- Decision tables can be used in all situations where the outcome depends on the combinations of different choices, and that is usually very often.
- In many systems there are tons of business rules where decision tables add a lot of value.
- **Steps to create decision tables**
- **Step 1 – Analyze the requirement and create the first column**
- Requirement: "Withdrawal is granted if requested amount is covered by the balance or if the customer is granted credit to cover the withdrawal amount".
- Express conditions and resulting actions in a list so that they are either TRUE or FALSE.
- In this case there are two conditions, "withdrawal amount <= balance" and "credit granted".

Modeling Logic With Decision Tables

- There is one result, the withdrawal is granted.

Conditions
Withdrawal amount <=balance
Credit granted
Actions
Withdrawal granted

- **Step 2: Add Columns**
- Calculate how many columns are needed in the table
- The number of columns depends on the number of conditions and the number of alternative for each condition.
- Number of column that is needed:

Number of conditions	Number of Columns
1	2
2	4
3	8
4	16
5	32

Modeling Logic With Decision Tables

- **Step 2: Add Columns**
- To fill True (T) and False (F) for the conditions, the simplest way is:
- Row 1: TF
- Row 2: TTFF
- Row 3: TTTTFFFF
- For each row, there is twice as many T and F as the previous line.

Modeling Logic With Decision Tables

- **Step 3: Reduce the table**
- Make insignificant values with "-". If the requested amount is less than or equal to the account balance it does not matter if credit is granted.
- In the next step, you can delete the column that have become identical.

Conditions				
Withdrawal amount<=balance	T	F	T	F
Credit granted	-	T	-	F
Actions				
Withdrawal granted				

- Check for invalid combinations.
- Invalid combinations are those that cannot happen, for example, that someone is both an infant and senior. Mark them somehow, e.g. with "X". In this example, there are no invalid combinations.
- Finish by removing duplicate columns. In this case, the first and third column are equal therefore one of them is removed.

Modeling Logic With Decision Tables

- **Step 4: Determining actions**
- Enter actions for each column in the table. Name the columns (the rules) be named R1/ Rule 1, R2/Rule 2 and so on, but you can also give them descriptive names.

Conditions			
Withdrawal amount<=balance	T	F	F
Credit granted	-	T	F
Actions			
Withdrawal granted	T	T	F

Modeling Logic With Decision Tables

- **Step 5: Write test cases**
- Write test cases based on the table. At least one test case per column gives full coverage of all business rules.
- **Test Case for R1:** Balance = 200, requested withdrawal = 200. Expected result: Withdrawal granted.
- **Test Case for R2:** Balance = 100, requested withdrawal = 200. Credit granted. Expected result: Withdrawal granted.
- **Test Case for R3:** Balance = 100, requested withdrawal = 200. No credit. Expected result: Withdrawal denied.

Advantages and Disadvantages of Decision Tables

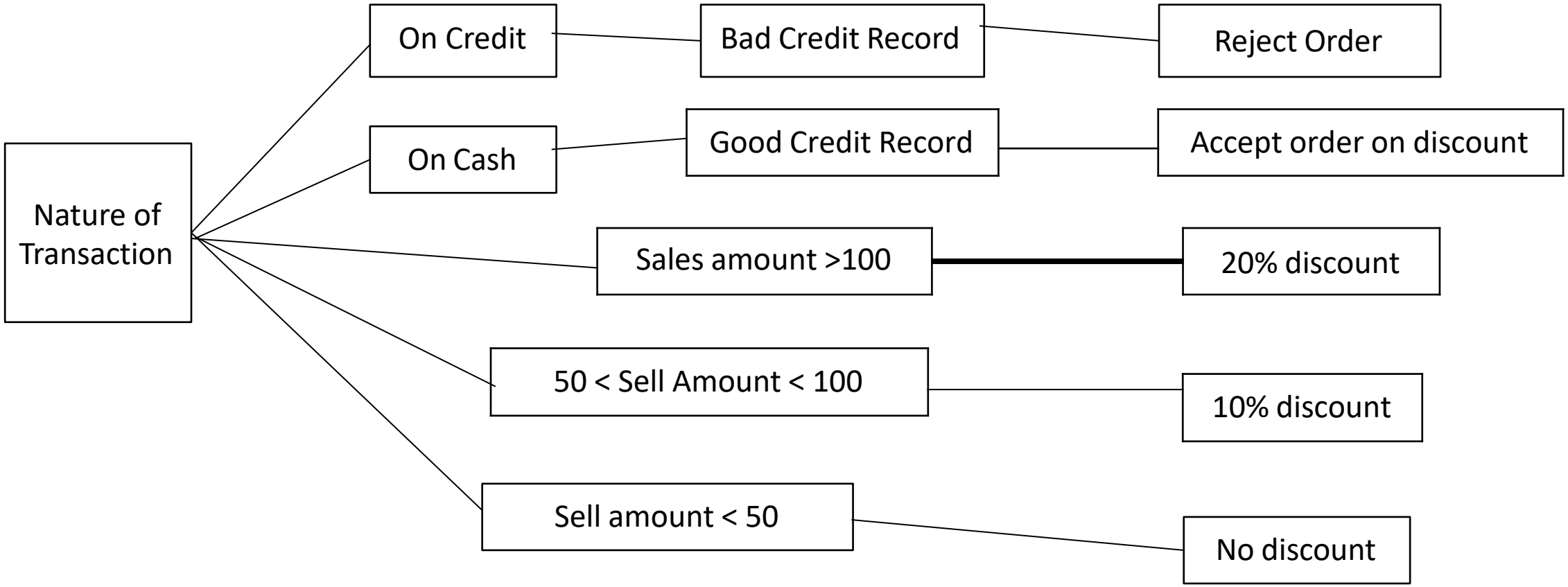
- **Advantages:**
- 1. Decision tables make it possible to detect combinations of conditions that would otherwise not have been found and therefore not tested or developed.
- 2. The requirements become much clearer and you often realize that some requirements are illogical/ irrational/ unreasonable, something that is hard to see when the requirements are only expressed in text.
- **Disadvantages:**
- 1. A decision table is not equivalent to complete test cases containing step-by-step instructions of what to do in what order.
- 2. When decision tables detail is required, the decision table has to be further detailed into test cases.

Decision Tree

- It is the most powerful and popular tool for classification and prediction.
- A **Decision Tree** is a flowchart like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.
- Decision trees are a method for defining complex relationships by describing decisions and avoiding the problems in communication.
- A **Decision Tree** is a diagram that shows alternative actions and conditions within horizontal tree framework.
- A square node indicates an action and a circle indicates a condition.
- It forces analysts to consider the sequence of decisions and identifies the actual decision that must be made.

Decision Tree

- The table below shows the decision tree:



Decision Tree

- Advantages of decision tree:
- 1. It expresses the logic of if then else in pictorial form
- 2. It is useful to express the logic when a value is variable or an action is dependent.
- 3. It helps the analyst to identify the actual decision to be made.
- 4. It is used to verify logic and problems that involve a few complex decision and limited number of actions.

- Disadvantages of decision tree:
- 1. There is absent of information in its format to take for conditions to take.
- 2. Large number of branches with many paths will confuse rather than help in analysis.

Pseudo Code

```
// Input: Prompt the user to enter two numbers
input firstNumber
input secondNumber

// Calculate the sum of the two numbers
sum = firstNumber + secondNumber

// Output: Display the result to the user
output sum
```

Pseudo Codes

- Pseudo-code is an informal way of programming description that does not require any strict programming language syntax or underlying technology considerations.
- It is used for creating an outline or a rough draft of a program.
- Pseudo-code summarizes a program's flow but excludes underlying details.
- System designers write pseudo-code to ensure that programmers understand a software project's requirements and align code accordingly.
- It has no syntax like any of the programming language and thus can't be compiled or interpreted by the computer.
- It's simply an implementation of an algorithm and informative text written in plain English.

Pseudo Codes

- Example of program to print Fibonacci up to n numbers.
- Void function Fibonacci
- Get value of n;
- Set value of a to 1;
- Set value of b to 1;
- Initialize I to 0;
- For(i=0; i<n; i++)
- {
- if a greater than b
- {

Pseudo Codes

- increase b by a;
- print b;
- }
- Else if b greater than a
- {
- increase a by b;
- print a;
- }
- }

0 1 1 2 3 5 8 13

Pseudo Codes

- Advantages of Pseudo-code
- 1. Improves the readability of any approach.
- 2. It is one of the best approach to start implementation of an algorithm.
- 3. Acts as a bridge between the program and the algorithm or flowchart.
- 4. The main goal of a pseudo code is to explain what exactly each line of a program should do, hence making the code construction phase easier for the programmer.

Structuring System Data Requirements

- Structuring system and database requirements concentrates on the definition, structure and relationships within data.
- The characteristics of data captured during data modeling are crucial in the design of databases, programs, computer screens and printed reports. This information is essential in ensuring data integrity in an information system.
- The most common format used for data modeling in entity-relationship diagramming.
- Data modeling explains the characteristics and structure of data independent of how the data may be stored in computer memories and is usually developed iteratively.

Conceptual Data Modeling

- A conceptual schema or conceptual data model is a map of concepts and their relationships used for databases.
- A conceptual data model is a representation of organizational data.
- Its purpose is to show as many rules about the meaning and interrelationships among data as possible.
- It is typically done in parallel with other requirements analysis and structuring steps during systems analysis.
- Analysts develop a conceptual data model for the current system and then build or refine a purchased conceptual data model that supports the scope and requirements for the proposed or enhanced system.
- Conceptual data modeling, using either the ER or UML approach, is particularly useful in the early steps of the database life cycle, which involve requirements analysis and logical design.

The process of Conceptual Data Modeling

- This process usually begins with developing a conceptual data model for the existing system.
- Then a new CDM, which includes all of the data requirements for the new system is built.
- E-R diagrams can be translated into wide variety of technical architecture for data, such as relational, network and hierarchical.

Deliverables and Outcomes

- The primary deliverable from the conceptual data modeling is the E-R diagram.
- Another deliverable from CDM is a full set of entries about data objects to be stored in the project dictionary or repository.
- The repository is the mechanism to link data, process and logic models of an information system.

Gathering Information for Conceptual Data Modeling

- A data model explains what the organization does and what rules govern how work is performed in the organization.
- The first perspective is generally called the top-down approach. This perspective derives the business rules for a data model from an intimate understanding of the nature of the business, rather than from any specific information requirements in computer displays, reports or business forms.
- The bottom-up approach is on the contrary a process of gaining an understanding of data by reviewing specific business documents handled within the system.

Introduction to E-R Modeling

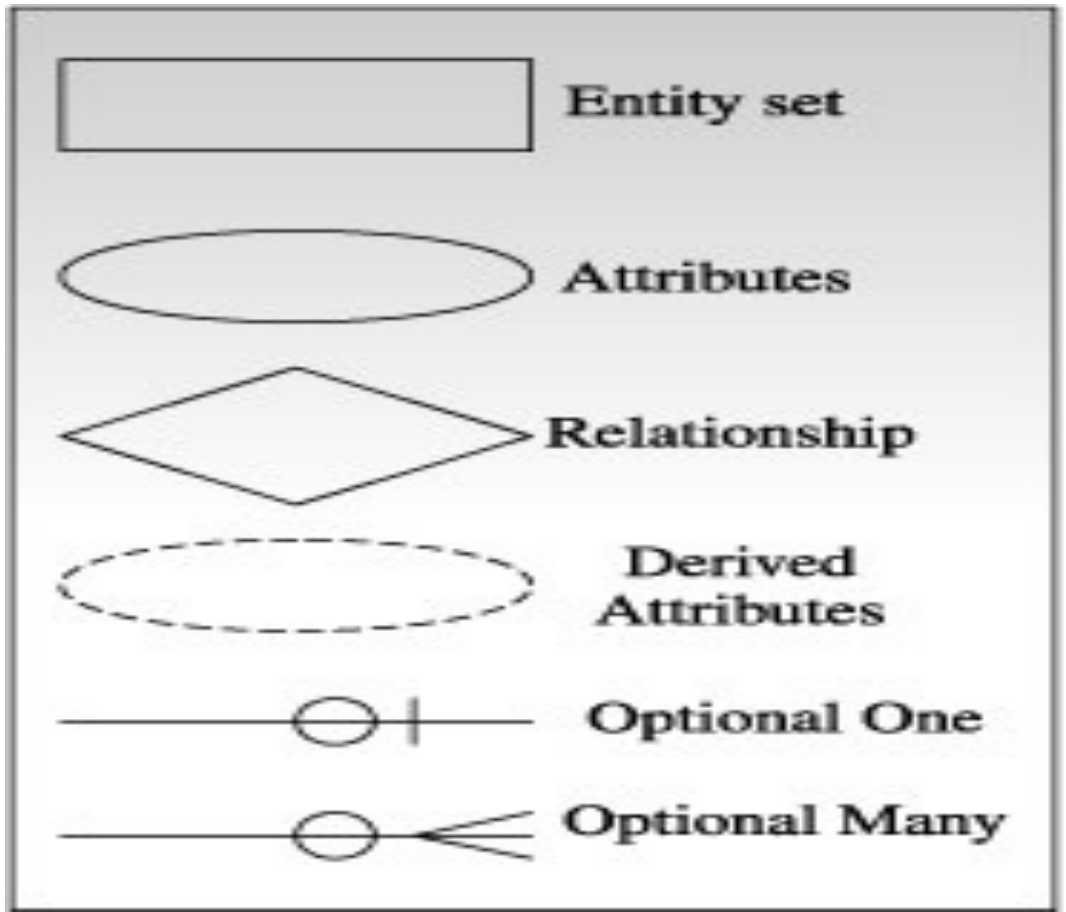
- An E-R data model is a detailed, logical representation of the data for an organization or for a business area.
- An entity is a person, place, object, event, or concept in the user environment about which the organization wishes to maintain data.
- An entity has its own identity, which distinguishes it from other entities.
- There is an important distinction between **entity types** and **entity instances**.
- An **entity type** is a single occurrence of an entity type.
- For example, there is usually one EMPLOYEE type but there may be hundreds of instances of this entity type stored in a database.
- Each entity type has a set of attributes associated with it.
- For example, for an entity STUDENT we have such attributes like: STUDENT NO, NAME, ADDRESS, PHONE NO.
- Every entity must have an attribute or set of attributes that distinguishes one instance from other instances of the same type – and that is called a **candidate key**.
- A **primary key** is a candidate key that has been selected to be used as the identifier for the entity type.

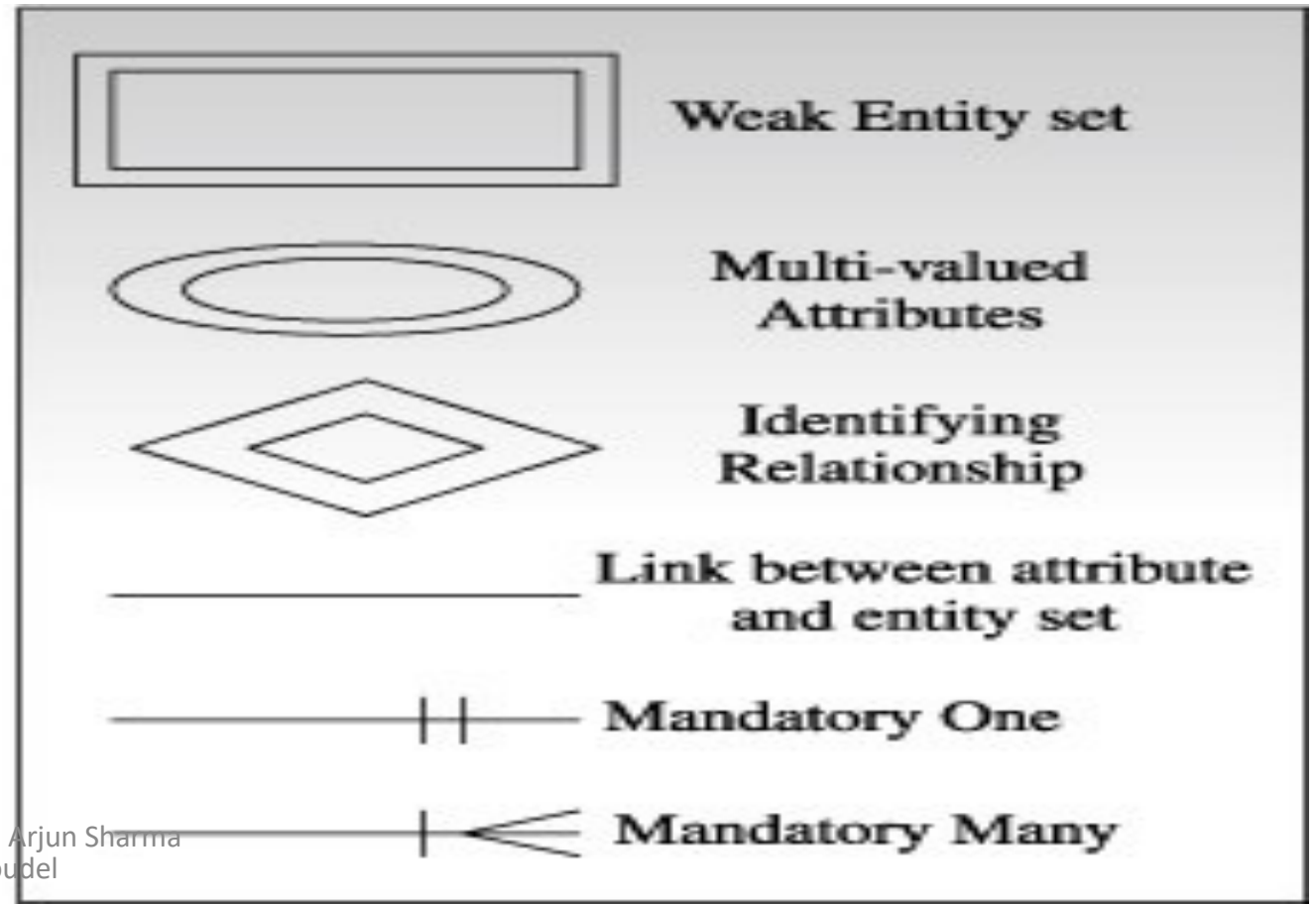
Introduction to E-R Modeling

- **Entity Relationship Modeling (ER Modeling)** is a graphical approach to database design.
- It uses Entity/Relationship to represent real world objects.
- An entity is a thing or object in real world that is distinguishable from surrounding environment. For example each employee of an organization is a separate entity.
- It is a technique used in database design that helps describe the relationship between various entities of an organization. Terms used in E-R model are:
- **Entity** – It specifies distinct real world items in an application. For example: vendor, item, student course, teachers etc.
- **Relationship** – They are the meaningful dependencies between entities. For example, vendor supplies items, teacher teaches courses, then supplies and teachers are relationships.
- **Attributes** – It specifies the properties of relationships. For example, vendor code, student name.

Introduction to E-R Modeling

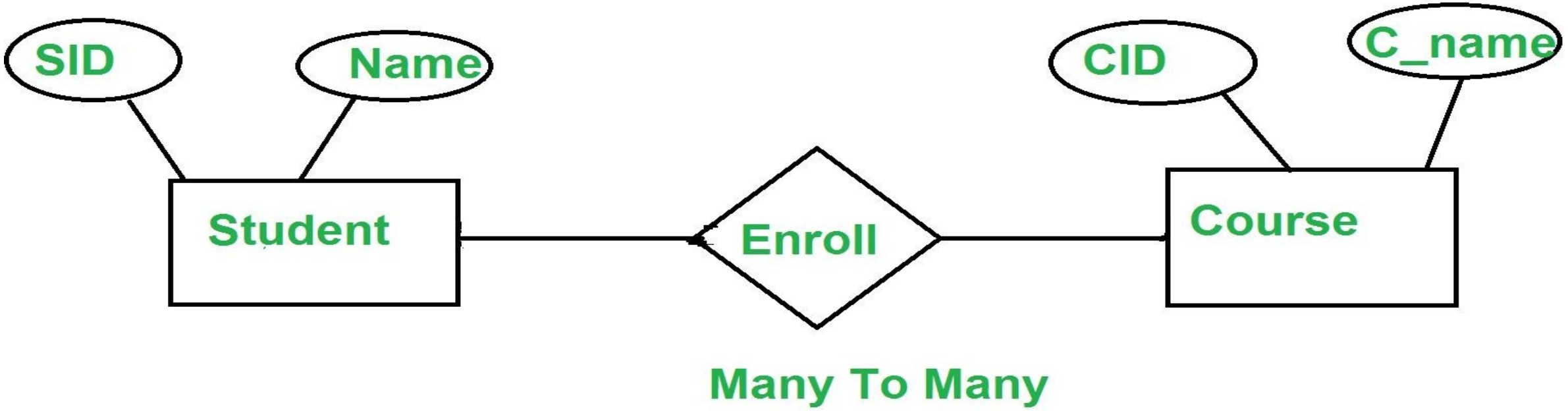
- **Entity-relationship diagram (ERD)** are essential to modeling anything from simple to complex databases, but the shapes and notations used can be very confusing.
- There are various types of symbols used for ER diagram, some of well-defined symbols are listed below;





Introduction to E-R Modeling

- **Entity-relationship diagram (ERD)** are essential to modeling anything from simple to complex databases, but the shapes and notations used can be very confusing.
- There are various types of symbols used for ER diagram, some of well-defined symbols are listed below;

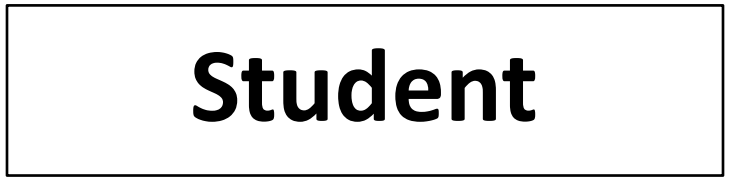


Elements of E-R / ER Design Issues

- **ENTITY:**
- An entity is a real-world thing either living or non-living that is easily recognizable and non-recognizable.
- An entity can be place, person, object, event or a concept, which stores data in the database.
- The characteristics of entities are must have an attribute, and a unique key.
- For example: Tuple1/ row1 / record1 contains information about Hari (id, name, age, address) which has existence in real world. So, the tuple1 is an entity. So, we may say each tuple is an entity. Let "Hari" is a particular member of entity type student.
-
-

Entity

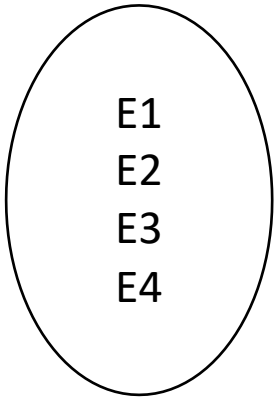
Elements of E-R / ER Design Issues

- **ENTITY TYPE:**
- It is collection of entities having common attribute.
- As in student table, each row is an entity and has common attributes.
- So, Student is a entity type which contains entities having attributes id, name and age.
- Also each entity type in a database is described by a name and a list of attribute.
- For example: Collection of entities with similar properties.
-
- 

Entity type

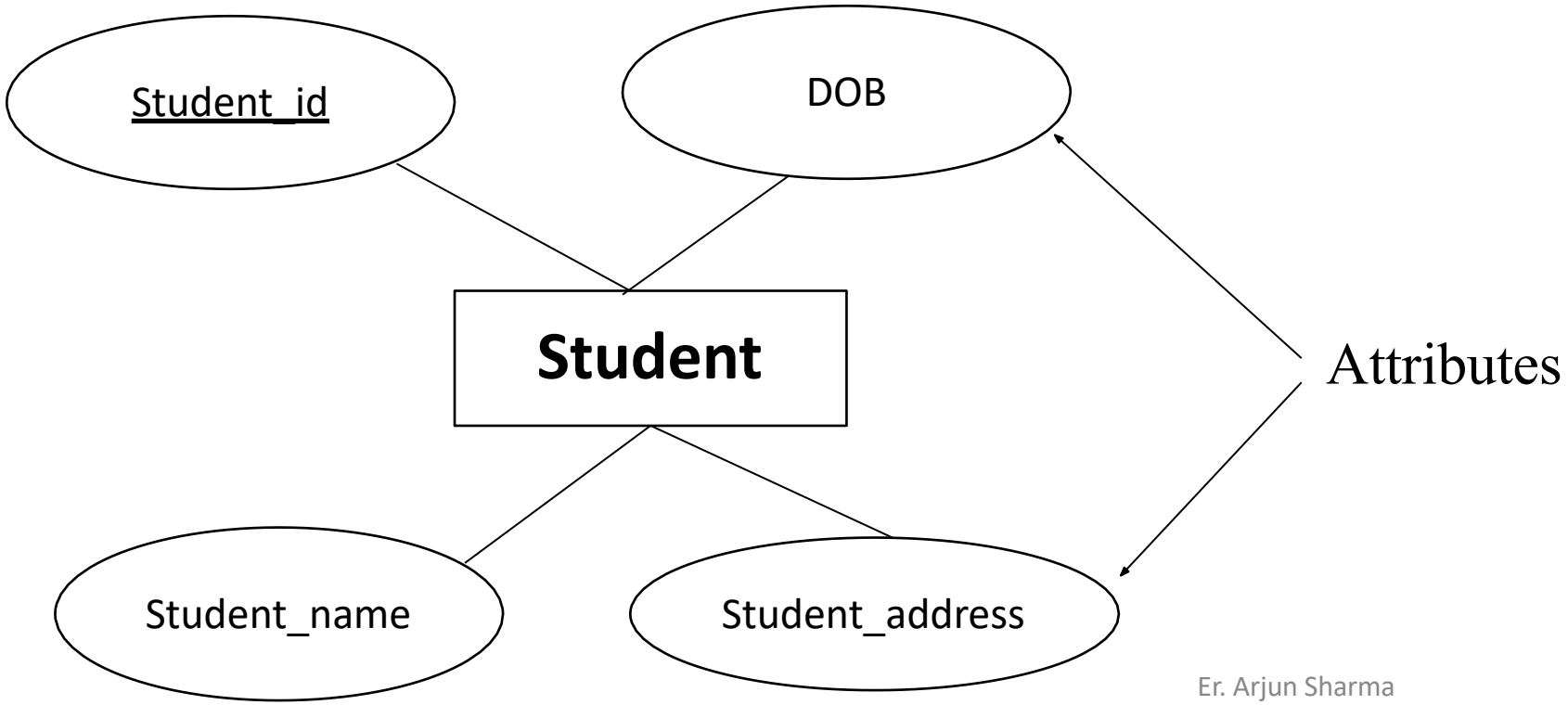
Elements of E-R / ER Design Issues

- **ENTITY SET:**
- It is same as entity type, but defined at a particular point in time such as students enrolled in a class on the first day.
- Other example: customer who purchased last month, cars currently registered in Florida. A related term is instance, in which the specific person or car would be an instance of the entity set.
- For example: let E1 is an entity having entity type student and set of all student is called entity set.



Entity set

Elements of E-R / ER Design Issues

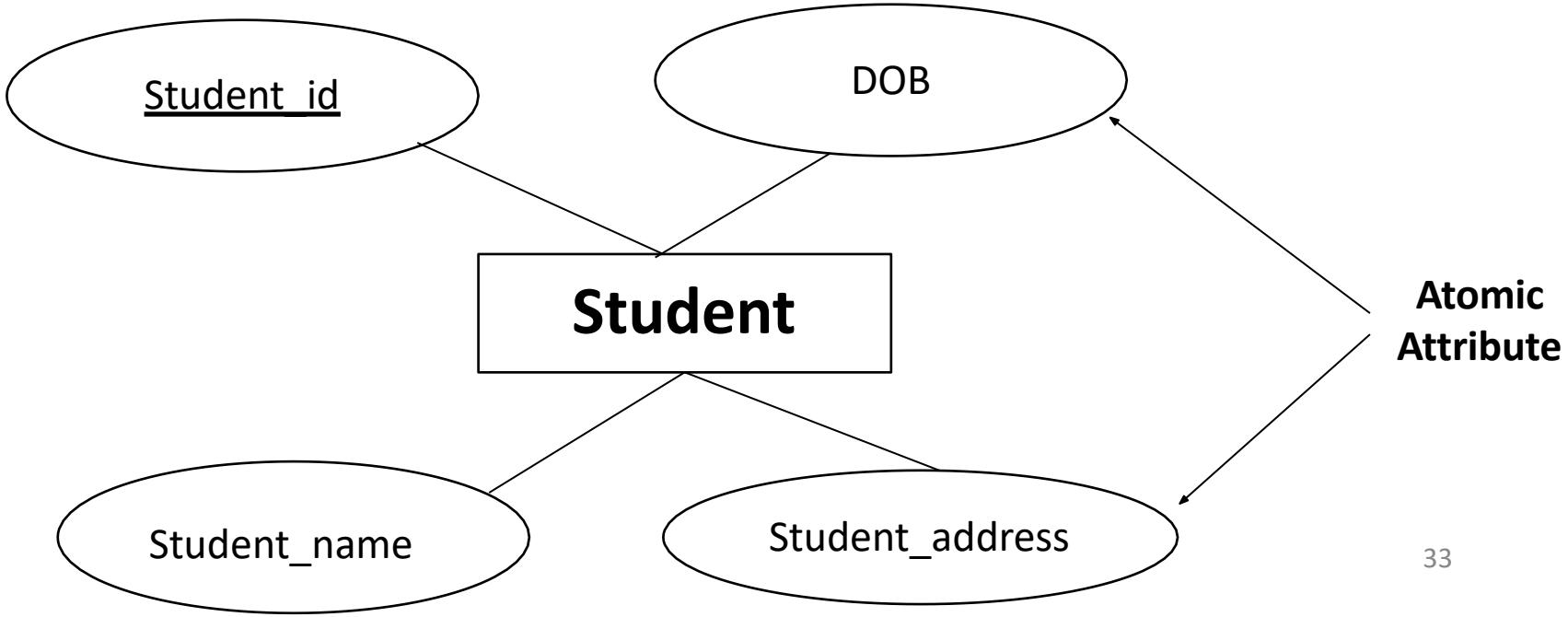
- **Attributes:**
- Attributes are the properties which define the entity type. For example, student_id, student_name, DOB, age, address, mobile_no etc. are the attributes which define entity type student.
- In ER diagram, attribute is represented by an oval.
- 

Elements of E-R / ER Design Issues

- **Types of Attributes:**
- Attributes of an entity type can be further divided into following types.
- A. Atomic Vs. Composite attributes
- B. Single valued Vs. Multi valued attributes
- C. Stored Vs. Derived attributes
- D. NULL value attribute
- E. Key attribute
-

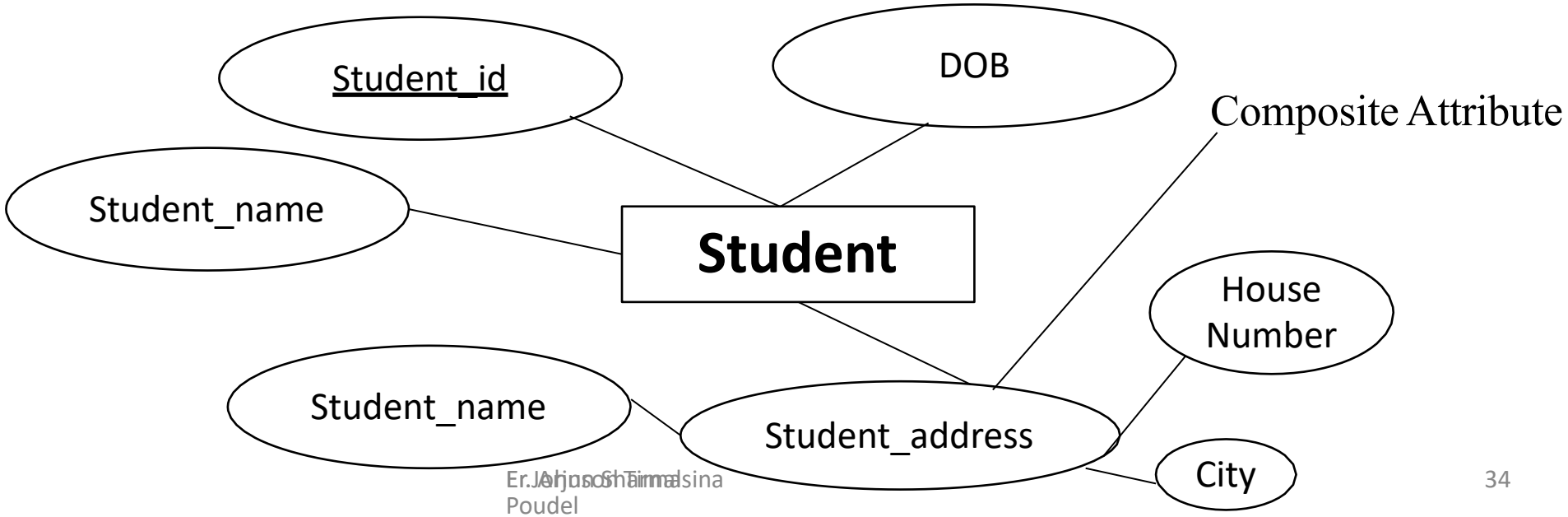
Types of Attributes

- **A. Atomic Vs. Composite attributes**
- An attribute that cannot be divided into smaller independent attribute is known as **atomic attribute**. For example: Assume, student is an entity and its attributes are name, age, DOB, address and phone number.
- Here the student_id, DOB attribute of students (entity) cannot further divide. In this example, Student_id and DOB are atomic attribute.
-



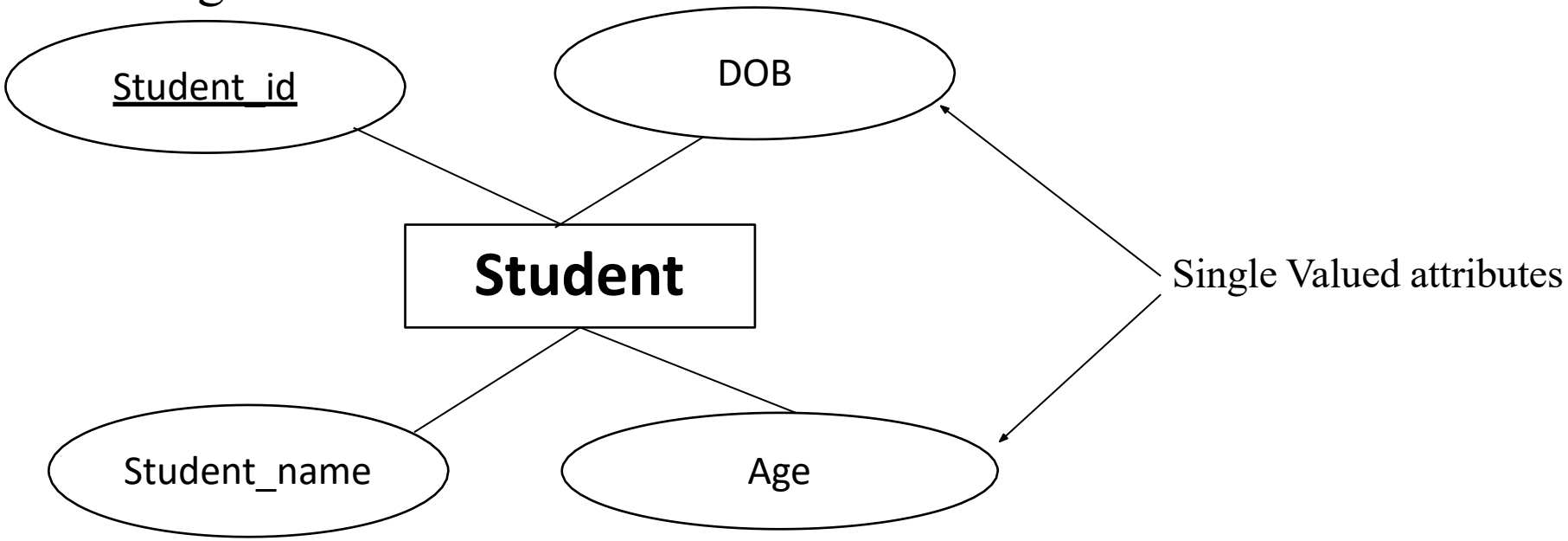
Types of Attributes

- **A. Atomic Vs. Composite attributes**
- An attribute that can be divided into smaller independent attribute is known as **composite attribute**. For example: Assume, student is an entity and its attributes are student_id, name, age, DOB, address and phone number.
- Here the address (attribute) of student (entity) can be further divided into house number, city and so on. In this example, address is composite attribute.
-
-



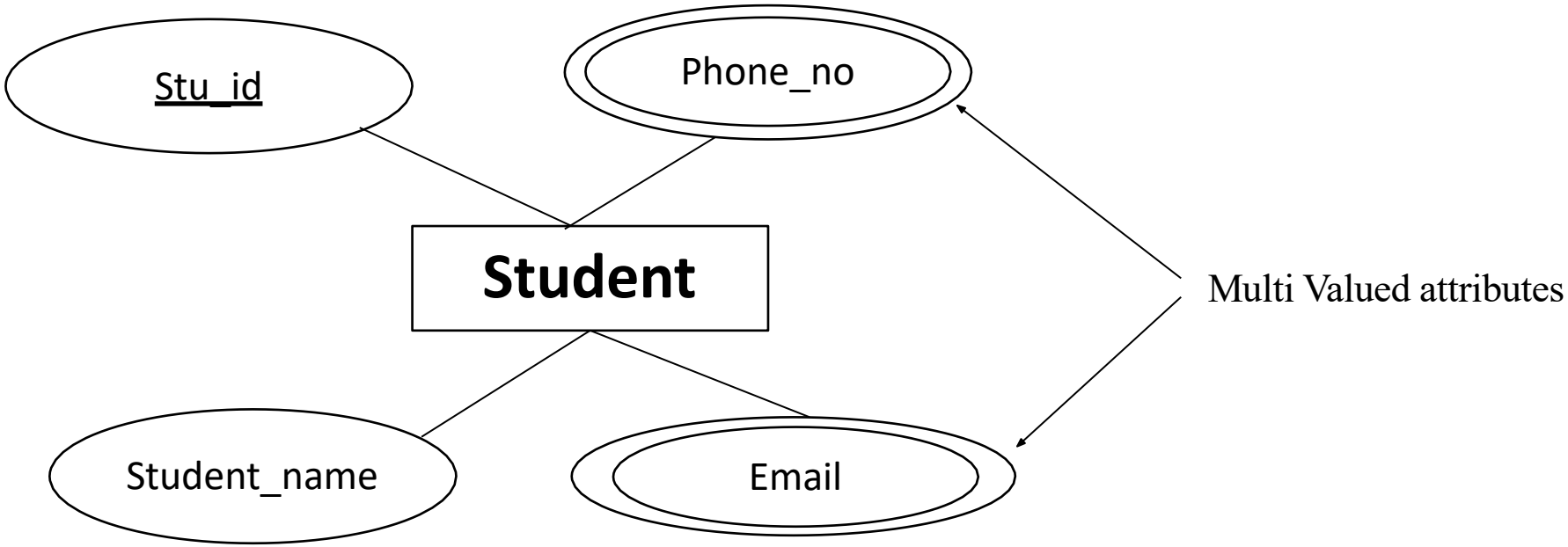
Types of Attributes

- **B. Single Valued Vs. Multi Valued attributes**
- An attribute that has only single value for an entity is known as single valued attribute. For example: Assume, student is an entity and its attributes are Stu_id, Name, age, DOB, address and phone number.
- Here the age and DOB(attribute) of student (entity) can have only one value. In this example, age and DOB are single valued attribute.



Types of Attributes

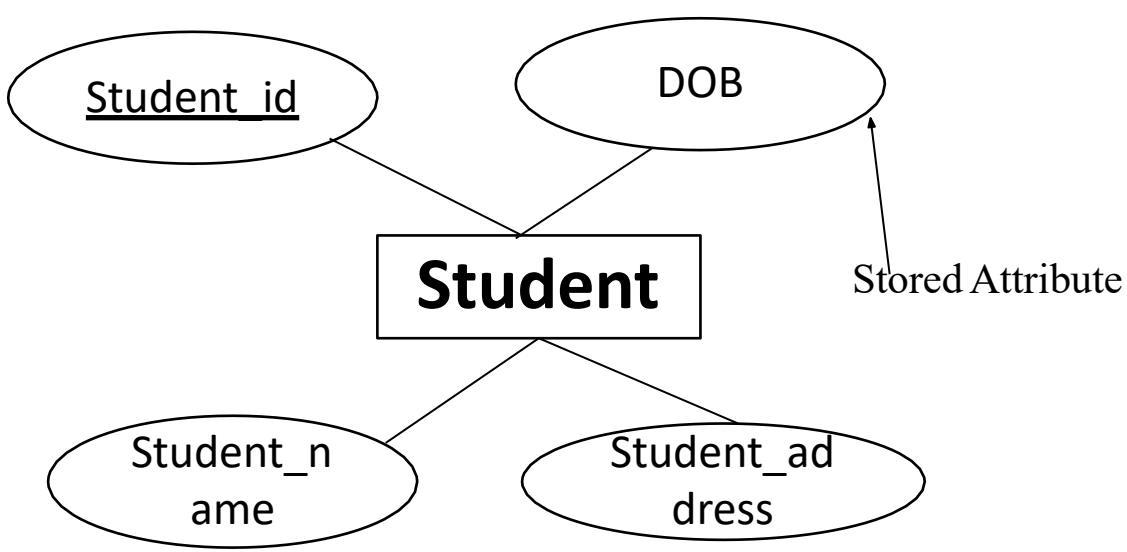
- **B. Single Valued Vs. Multi Valued attributes**
- An attribute that can have multiple values for an entity is known as multi valued attribute. For example: Assume, Student is an entity and its attributes are Stu_id, Name, age, DOB, address, email and phone no.
- Here the phone no and Email (attribute) of Student (entity) can have multiple values because a student may have many phone numbers. In this example, Email and Phone_no are multi valued attributes.

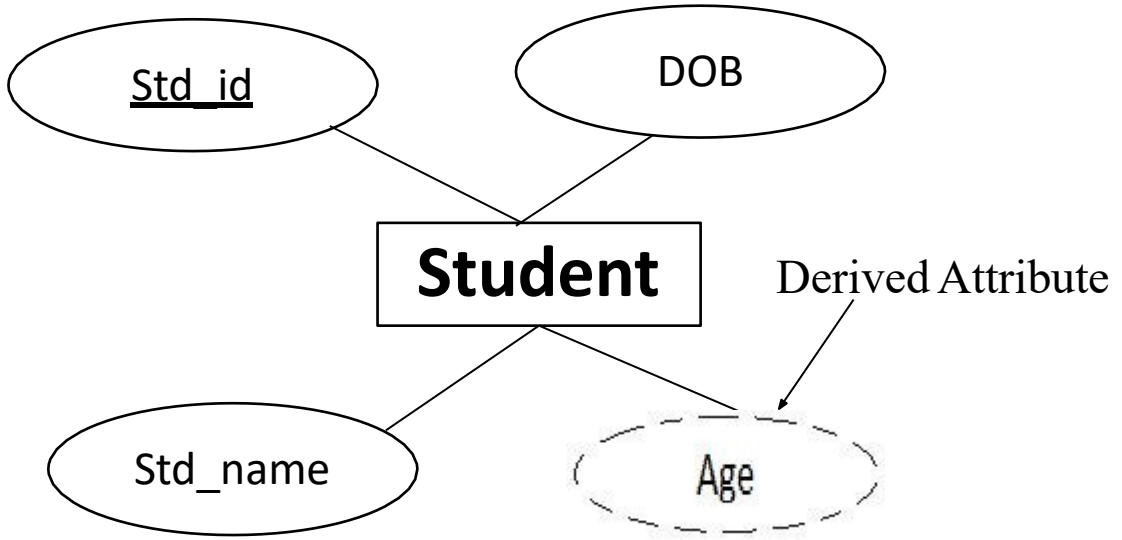


- `

Types of Attributes

- **C. Stored Vs. Derived attributes**
- An attribute that cannot be derived from another attribute and we need to store their value in database is known as stored attribute. For example, DOB cannot derive from age of student.
- An attribute that can be derived from another attribute and we do not need to store their value in the database due to dynamic nature is known as derived attribute.
- It is denoted by dotted oval. For example, age can derive from birth date of student.





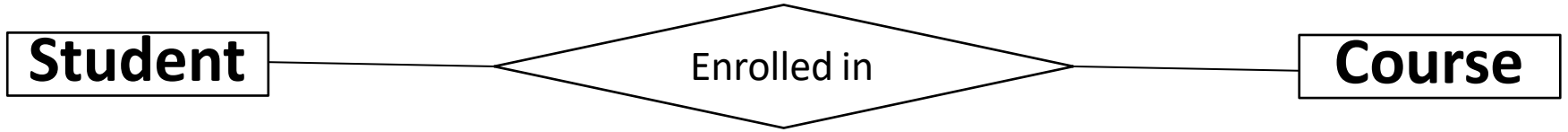
Types of Attributes

- **D. NULL value attributes**
- An attribute, which has not any value for an entity is known as null value attribute. For example, assume Student is an entity and its attributes are Name, Age, Address and Phone_no.
- There may be chance when a student has no phone no. In this case, phone number is called NULL valued attributes.

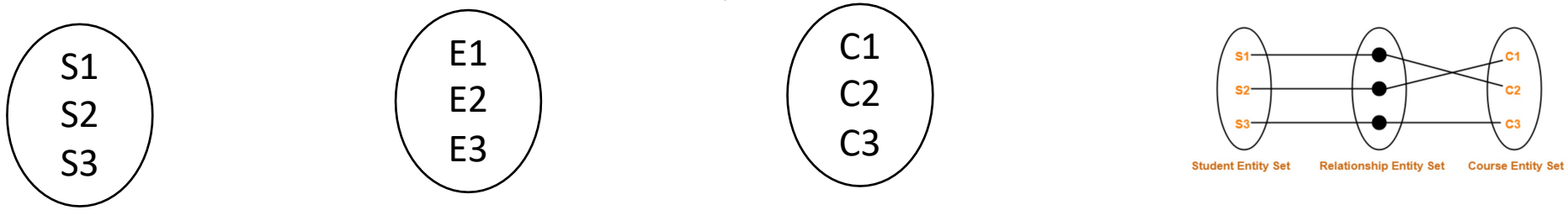
- **E. Key attributes**
- An attribute that has unique value of each entity is known as key attribute. For example, every student has unique roll no. Here roll no is key attribute.

Relationship Type and Relationship Set

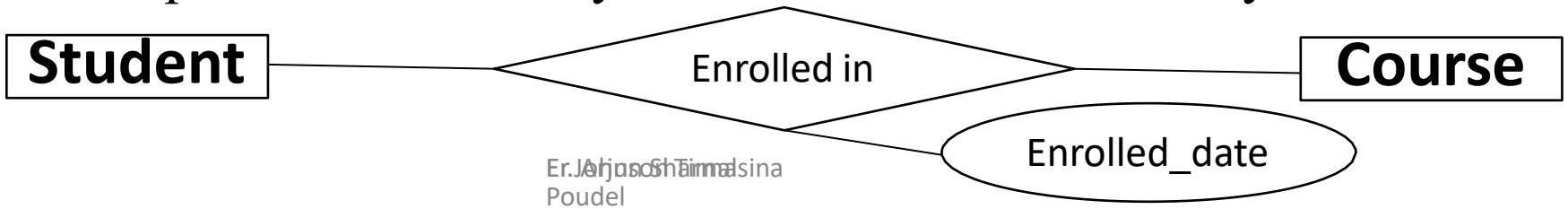
- A relationship type represents the association between entity types. For example, "Enrolled in" is a relationship type that exists between entity type Student and Course. In ER diagram, relationship type is represented by a diamond and connecting the entities with lines.



- A set of relationships of same type is known as relationship set. The following relationship set depicts / represents / illustrates S1 is enrolled in C2, S2 is enrolled in C1 and S3 is enrolled C3.



- In another way, we can say that association between two entity sets is called relationship set. A relationship set may also have attributes called descriptive attributes. For example, the enrolled_in relationship set between entity sets student and course may have the attribute enrolled_date.



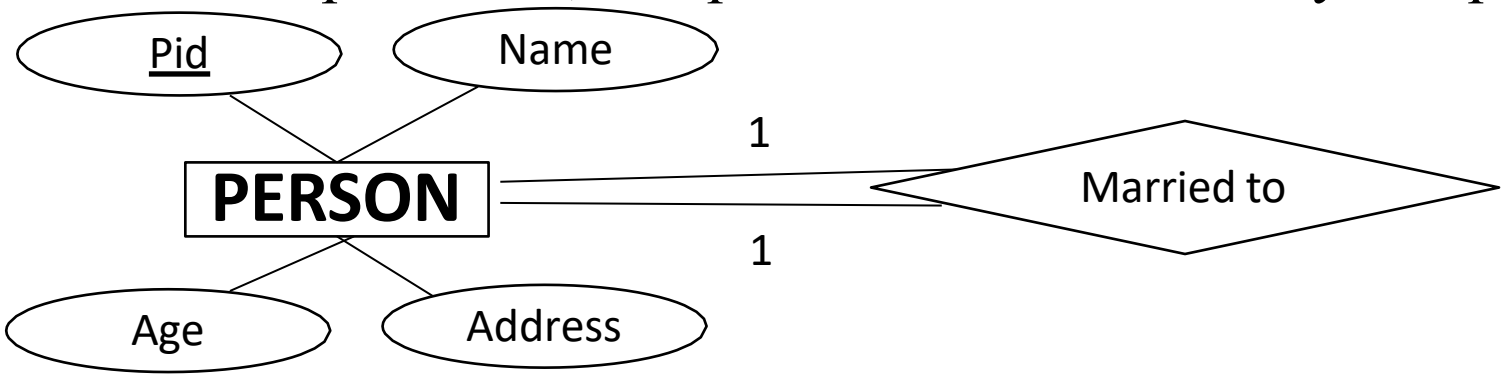
Degree of a Relationship

- Number of entity sets that participate in a relationship set is called degree of the relationship set. On the basis of degree, relationships can be divided as below:
- Unary Relationship
- Binary Relationship
- N-ray Relationship

- **Unary Relationship**
- If only one entity set participates in a relation, the relationship is called as unary relationship. Here same entity set participates in relationship twice with different roles. Role names are specified above the link joining entity set and relationship set. This type of relationship set is sometimes called a recursive relationship set. There are three types of unary relationships.
- 1:1 unary relationship
- 1:M unary relationship
- M:N unary relationship

One to One (1:1) Unary Relationship

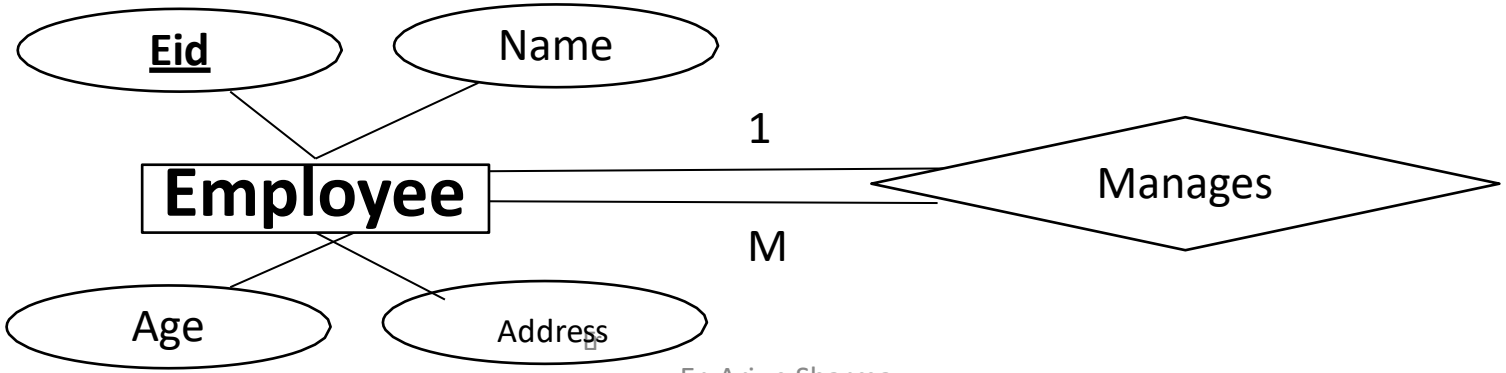
- In the example below, one person is married to only one person.



One to Many (1:M) Unary Relationship

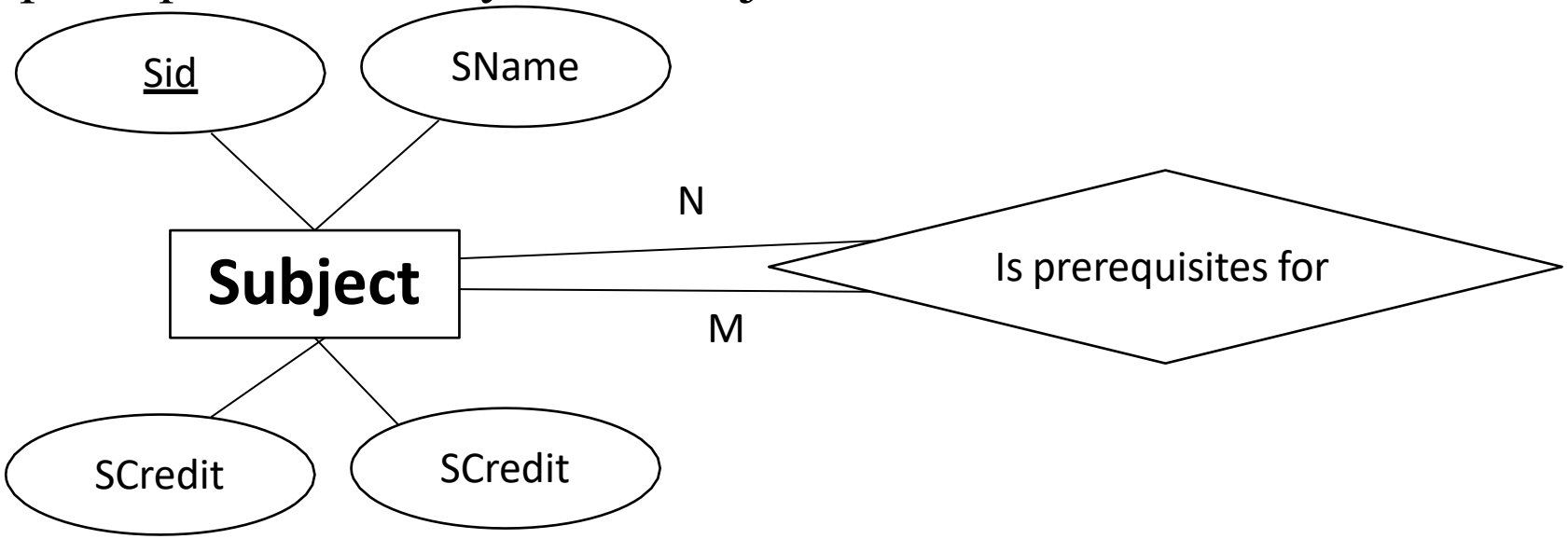
Address

- An employee may manage many employee but an employee is managed by only employee. This types of relationship with employee relationship set itself is called 1:M unary relationship as shown in below:



Many to Many (M:M) Unary Relationship

- A subject may have many other subjects as prerequisites and each subject may be a prerequisite to many other subjects.

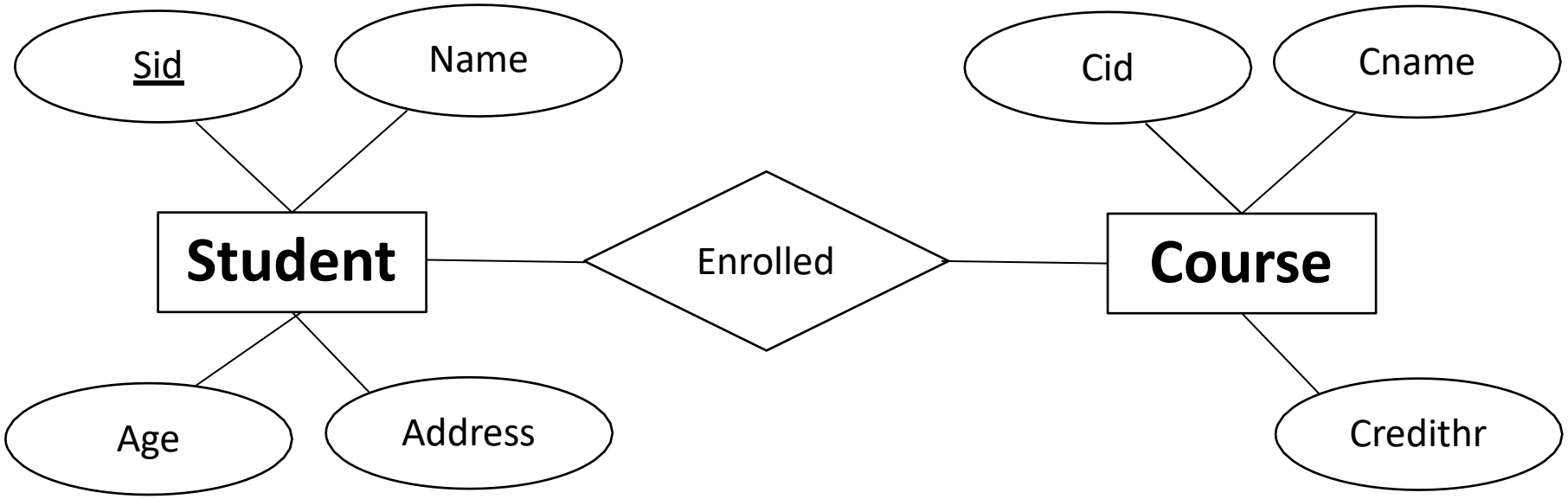


Binary Relationship

- When there are two entities set participating in a relation, the relationship is called as binary relationship. For example, Student is enrolled in Course. This is the most common type of relationship in database systems.

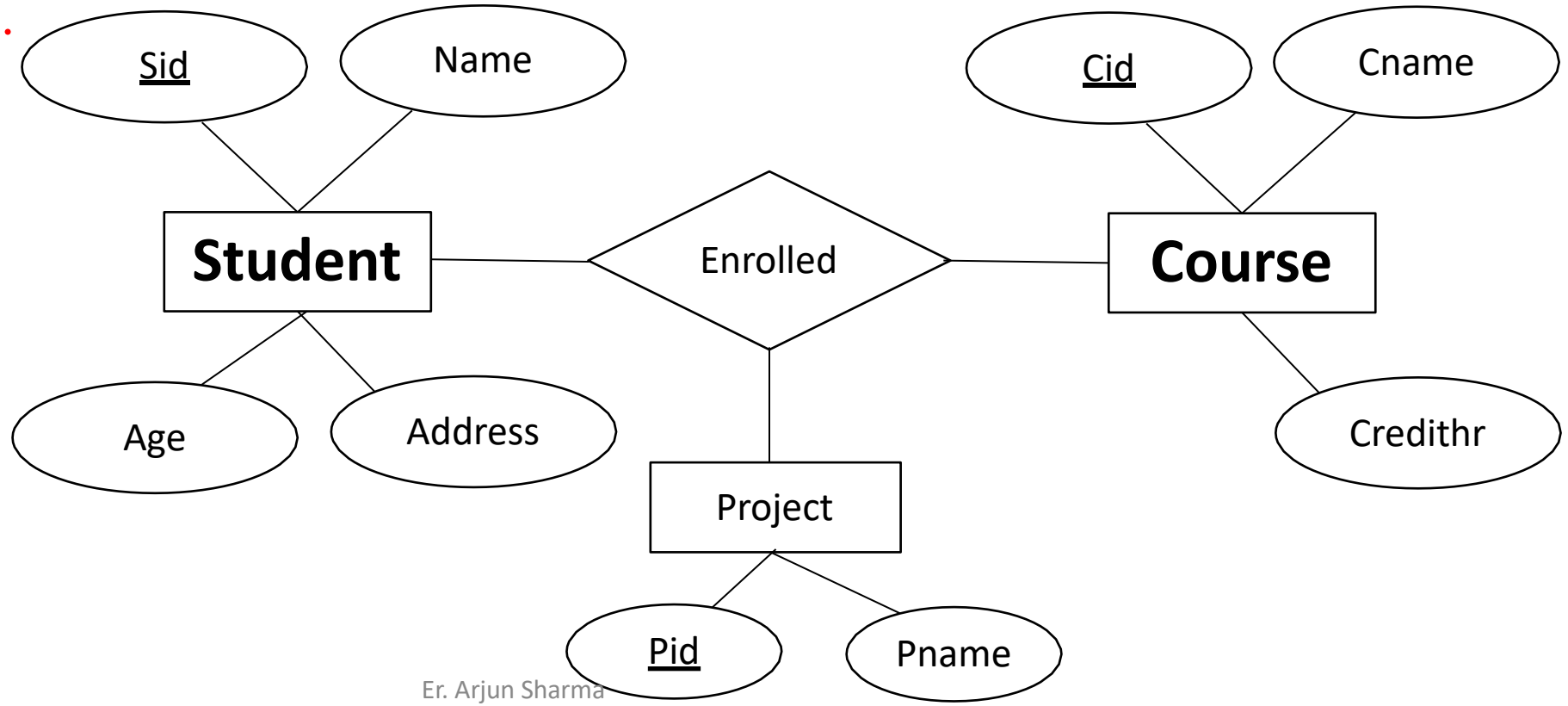
Binary Relationship

- When there are two entities set participating in a relation, the relationship is called as binary relationship. For example, Student is enrolled in Course. This is the most common type of relationship in database systems.



N-ary Relationship

- When there are n entities set participating in a relation, the relationship is called n-ary relationship. For example, if n = 1 then it is called unary relationship, if n=2 then it is called binary relationship.
- Generally in N-ary relationship there are more then two entities participating with a single relationship i.e. n>2.

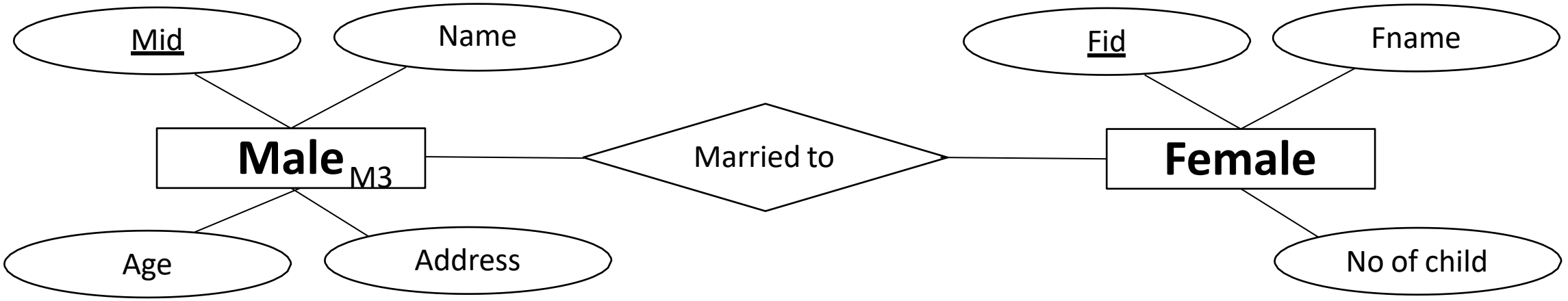


Constraints on ER Model/Structural Constraints in ER

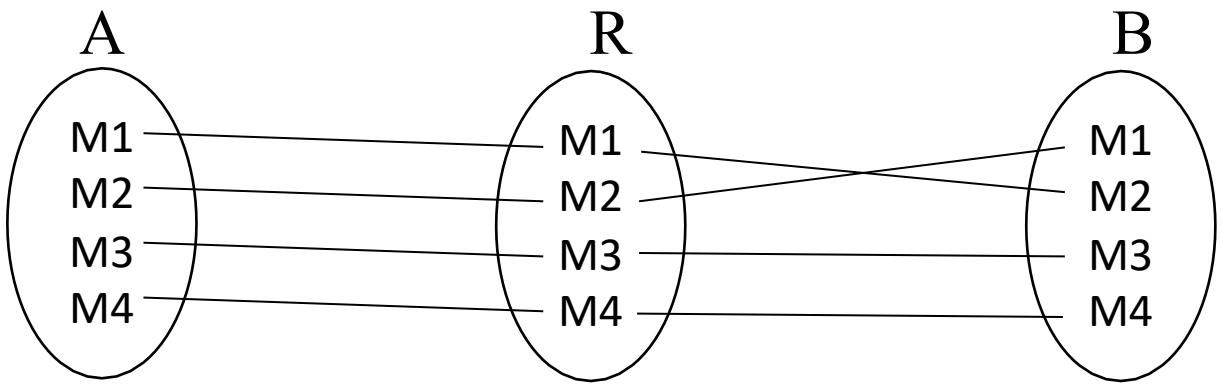
- Relationship sets in ER model usually have certain constraints that limit the possible combinations of entities that may involve in the corresponding relationship set. Database content must confirm these constraints. The most important structural constraints in ER are listed below:
- Mapping cardinalities and
- Participation constraints
- **Mapping Cardinality Constraints**
- The number of times an entity of an entity set participated in a relationship set is known as cardinality. Cardinality can be of different types:
- One-to-One
- One-to-Many
- Many-to-One
- Many-to-Many
- We express cardinality constraints by drawing either a directed line(→), signifying "one", or an undirected line (-), signifying "many", between relationship set and the entity set.

One-to-One Mapping Cardinality Constraints

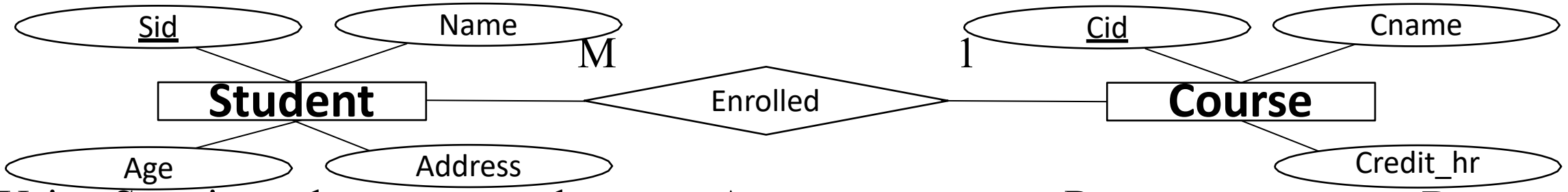
- In One-to-One mapping, an entity in E1 is associated with at most one entity in E2, and an entity in E2 is associated with at most one entity in E1. Let us assume that a male can marry to one female and a female can marry to one male. So the relationship will be one to one.



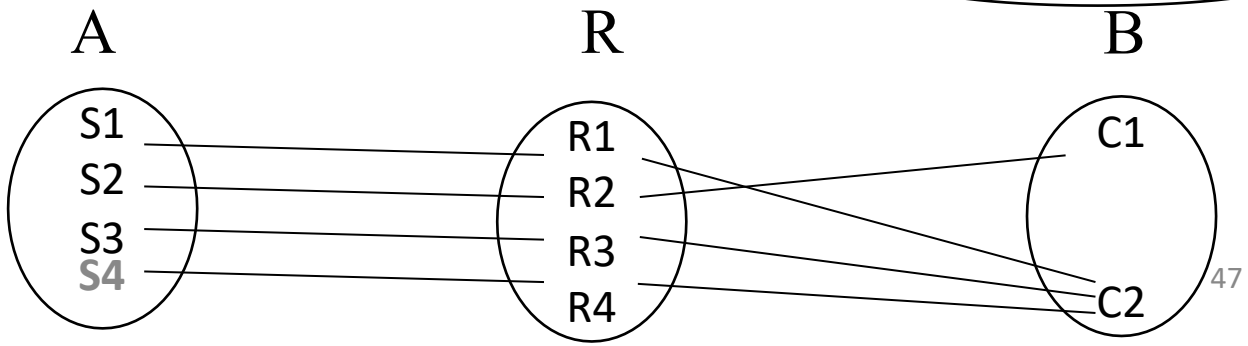
- Using Sets, it can be represented as:



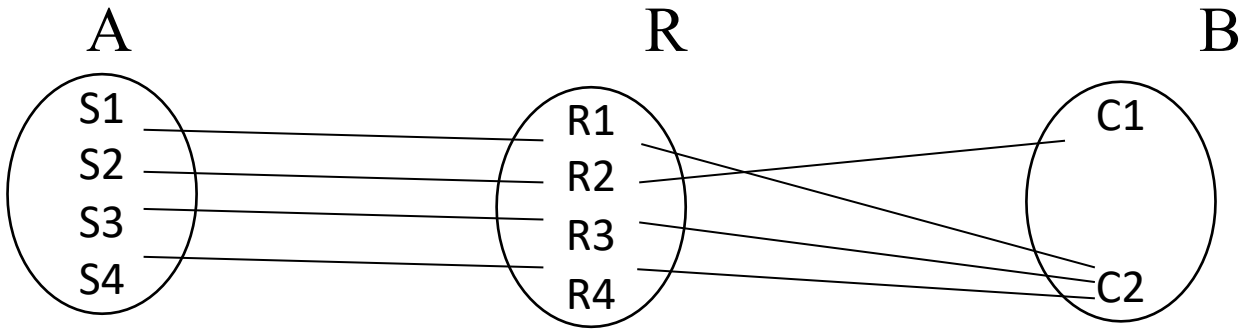
One-to-Many or Many-to-One Mapping Constraints

- In one-to-many mapping, an entity in E1 is associated with any number of entities in E2, and an entity in E2 is associated with at most one entity in E1.
- In this mapping, an entity in E1 is associated with at most one entity in E2, and an entity in E2 is associated with any number of entities in E1.
- Let us assume that a student can take only one course but one course can be taken by many students. So the cardinality will be n to 1. It means that for one course there can be n students but for one student, there will be only one course.
- 

- Using Sets, it can be represented as:



Many-to-Many Mapping Constraints

- In Many-to-Many mapping, an entity in E1 is associated with any number of entities in E2, and an entity in E2 is associated with any number of entities in E1. Let us assume that a student can take more than one course and one course can be taken by many students. So, the relationship will be many to many.
- Using Sets, it can be represented as:
- 

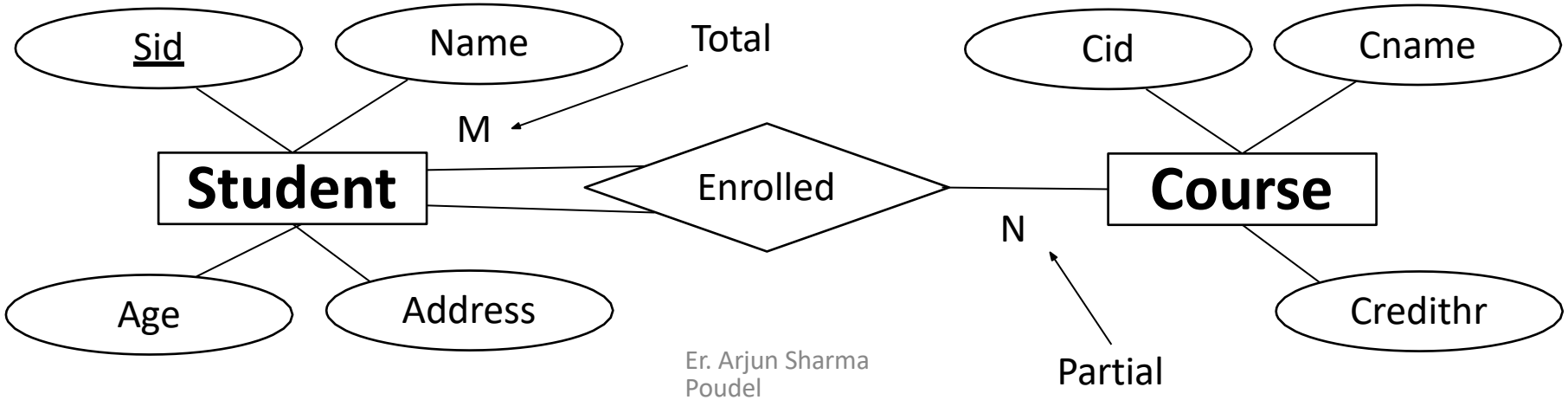
- In this example, student S1 is enrolled in C1 and C3 and Course C2 is enrolled by S1, S2 and S4. So, it is many to many relationships.

Participation Constraints

- Participation Constraint is applied on the entity participating in the relationship set. Constraint on ER model that determines whether all or only some entity occurrences participate in a relationship is called participation constraint.
- It specifies whether the existence of an entity depends on its being related to another entity via the relationship type.
- There are two types of participation constraints:
- Total Participation Constraints and
- Partial Participation Constraints.
- **Total Participation Constraints**
- It specifies that each entity in the entity set must compulsorily participate in at least one relationship instance in that relationship set. That is why; it is also called as mandatory participation. Total participation is represented using a double line between the entity set and relationship set in ER diagram. If each student must enroll in a course, the participation of student will be total.

Participation Constraints

- **Partial Participation Constraints**
- It specifies that each entity in the entity set may or may not participate in the relationship instance in that relationship set. That is why; it is also called as optional participation. Partial participation is represented using a single line between the entity set and relationship set in ER diagram. If some courses, are not enrolled by any of the student, the participation of course will be partial.
- The diagram depicts the 'Enrolled in' relationship set with Student Entity set having total participation and Course Entity set having partial participation.
-



Multiple Choice Questions

- 1. In a DFD external entities are represented by a ……..
- a. Rectangle b. Ellipse c. Diamond shaped box d. Circle
- 2. The final step of the system analysis phase in the SDLC is to ……
- a. Gather data b. Write system analysis report c. Propose changes d. Analyze data
- 3. The key considerations involved in the feasibility analysis is / are
- a. Economic b. Technical c. Behavioral d. All of them
- 4. In ER diagrams, rectangles are used to denote
- a. Entity types b. Attribute types c. Key types d. Structure types
- 5. In constructing ER diagrams, double ovals are used to denote
- a. Multi-value table b. Multi-value entity c. Multi-value attribute d. Multi-value key
- 6. Weak entities are used in entity relationship diagrams and are denoted by
- a. Double rectangles b. Double square c. Double ovals d. Double squares

Multiple Choice Questions

- 7. Which of the following is not a requirement of structured design?
- a. It should use many GOTO statements b. It should be made up of a hierarchy of modules
- c. The code should be executed in a top-to-bottom fashion within each module
- d. Each module should be independent as possible of all other modules, except its parent
- 8. The data flow diagram (DFD) shows ……..
- a. The flow of data b. The processes c. The area where they are storedd. All of them
- 9. In a DFD, an originator or data receiver is usually designed by …….
- a. A square box b. A circle c. A rectangle d. None of them
- 10. At highest level, a DFD is referred to as ……
- a. Scope diagram b. Context diagram c. Level 1 DFD d. Level 2 DFD

Answer the following Questions

- 1. Name four traditional techniques for collecting information during analysis. Give the advantages and disadvantages of each method.
- 2. What is JRP and who takes part in a JRP session?
- 3. How has computing been used to support JRP?
- 4. What benefits do GSS provide?
- 5. Which types of CASE tools are appropriate for use during requirements determination?
- 6. What are the advantages and disadvantages of discovery prototyping?
- 7. Explain how conceptual data modeling is different when you start with a prepackaged data model rather than a clean sheet of paper.
- 8. List the deliverables from the conceptual data modeling part of the analysis phase of the systems development.
- 9. How are E-R diagrams similar to and different from decision trees? In what ways are data and logic modeling techniques complementary? What problems might be encountered if either data or logic modeling techniques were not performed well or not performed at all as part of the systems development process?
- 10. Discuss why some systems developers believe that a data model is one of the most important parts of the statement of information system requirements.

Answer the following Questions

- 12. What is a business process? Why is business process diagramming important?
- 13. Choose a transaction that you are likely to encounter, perhaps ordering a cap and gown for graduation, and develop a high-level DFD or a context diagram. Decompose this to a level-0 diagram.
- 14. Describe systems analysis and the major activities that occur during this phase of the systems development life cycle.
- 15. Describe four traditional techniques for collecting information during analysis. When might one be better than another?
- 16. What is JAD? How is it better than traditional information gathering techniques? What are its weaknesses?
- 17. Describe how prototyping can be used during requirements determination. How is it better or worse than traditional methods?
- 18. What are disruptive technologies and how do they enable organizations to radically change their business processes?
- 19. Why is continual user involvement a useful way to discover system requirements? Under what conditions might it be used? Under what conditions might it not be used?

Er. Arjun Sharma Poudel