

DATABASE ANALYSIS AND DESIGN 1

Week 2

Outline

- Database design phases
- The entity relationship model
 - Entities
 - Attributes
 - Relationships

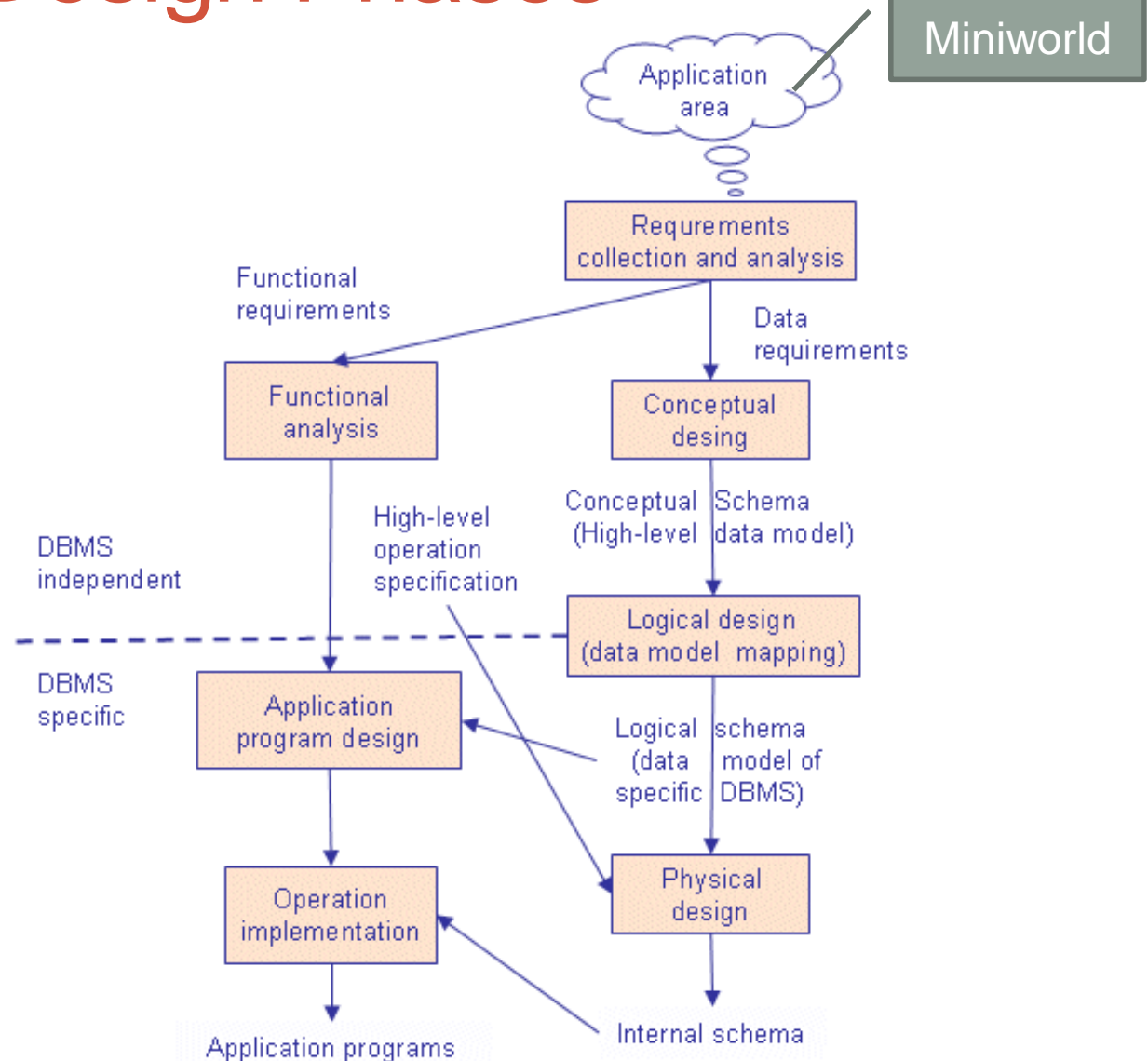
Three Main Stages of Database Development



- Requirements Analysis
 - Understand the data problem through user interviews, forms, reports, observations etc.
- Component Design Stage
 - Create a data model that is a graphical representation of what will be finally will be implemented
- Implementation Stage
 - Actual development of the database
 - Leads to database actually being used in the real environment



Database Design Phases



After requirements
are gathered...

Database Designs

Conceptual Design

- Identify all entities
- Define attributes and their properties
- Define Relationships between entities

Logical Design

- Transfer conceptual design into relational form
- Transform entities as tables
- Transform entity attributes into table columns
- Normalization

Physical Design

- Specify internal storage structure and access paths
- Assign one or more indexes
- Tune indexes

Requirements

Data Model

(Entity-Relationship Model)



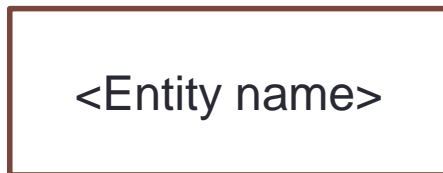
Entity-Relationship Model (E-R Model)

- It is based on the notion of real-world entities and relationships among them.
- ER Model is used for the conceptual design of a database.
- ER Model consists of
 - **Entities**
 - **Relationships** between entities
 - **Attributes** of entities
- ER models are represented by ER Diagrams (ERD).

Entity

- An entity in an ER Model is a real-world entity having properties called attributes.
- E.g., in a university database, a student is considered as an entity. Student has various attributes like name, age, degree, etc.

- ERD Notation



E.g.



The name of the entity is placed inside the box.

Identifying Entities in a Scenario

- Look for the people, places, things, organizations, concepts, and events that an organization needs to capture, store, or retrieve information about.
- There are three general categories of entities:
 - **Physical entities** - are tangible and easily understood
 - **Conceptual entities** - are not tangible and are less easily understood. They are often defined in terms of other entity-types.
 - **Event/State entities** - are typically incidents that happen. They are very abstract and are often modelled in terms of other entity-types as an associative entity.

Identifying Entities in a Scenario

- Physical entities

- *people*, for example, doctor, patient, employee, customer,
- *property*, for example, equipment, land and buildings, furniture and fixtures, supplies,
- *products*, such as goods and services.

- Conceptual entities

- *organizations*, for example, corporation, church, government,
- *agreements*, for example, lease, warranty, mortgage,
- *abstractions*, such as strategy and blueprint.

- Event/State entities

- *events*, for example, purchase, negotiation, service call, and deposit.
- *states*, for example, ownership, enrolment, and employment.

When an Entity is Not an Entity

- There are a number of things that may appear to be entities about which facts are kept, but which should not be defined as such.
- Processes - Processes may actually perform actions on entities but are not, themselves, entities.
 - Examples are: payroll deduction, budgeting (an action on an organization unit).
- Calculations - Calculations are derived from the attributes of an entity.
 - Examples are: inventory level, average age, net worth.
- Reports - Reports present facts about one or more entities.
 - Examples are: Project schedule, Income statement.
- Facts about entities - Facts about entities describe characteristics of an entity and should be modelled as attributes.
 - Examples are: telephone number, date of hire.

Rules for Defining Entities

- Name each entity using a noun in the singular form (e.g., Employee not Employees).
- Use a word that is precise and clearly identifies the object.
- When appropriate and needed to distinguish similar entities, use an adjective to further describe the noun (e.g., Permanent Employee, Temporary Employee).
- Use a term that is familiar to the business or is commonly used in everyday language.
- The entity name is representative of the characteristics or attributes of the entity.
 - E.g., use a term such as Inventory Item rather than Item.

Important to Note that...

System is Not an Entity



Example

- A company has several departments. Each department has a supervisor and at least one employee. Employees must be assigned to at least one, but possibly more departments. At least one employee is assigned to a project, but an employee may be on vacation and not assigned to any projects. The important data fields are the names of the departments, projects, supervisors and employees, as well as the supervisor and employee number and a unique project number.
- **Find Entities**

Example

- A **company** has several departments. Each department has a supervisor and at least one employee. Employees must be assigned to at least one, but possibly more departments. At least one employee is assigned to a project, but an employee may be on vacation and not assigned to any projects. The important data fields are the names of the departments, projects, supervisors and employees, as well as the supervisor and employee number and a unique project number.
- **Find Entities**

Example

Identify Entities

- The entities in this system are Department, Employee, Supervisor and Project.
- One is tempted to make Company an entity, but it is a false entity because it has only one instance in this problem.
- **True entities must have more than one instance.**


Example 2

- An inventory software used in a retail shop will have a database that monitors elements such as item type, item source and item price for each item and demographic details of buyers.
- **Identify entities**

Example 2

- An **inventory software** used in a retail shop will have a database that monitors elements such as item type, item source and item price for each item and demographic details of buyers.
- **Identify entities:**
- Buyer, item

Attributes

- Entities have attributes that together describes the entity
- E.g. Project entity has following attributes;
 - projectName
 - startDate
 - projectType
 - projectDescription

When you fill in values to each attribute, you will get entity instance
- Each attribute has a
 - **Data type** (e.g. *employeeId* as integer, *employeeName* as varchar with max 50 characters) and
 - **Other properties** (e.g. *whether or not null values can be stored*)

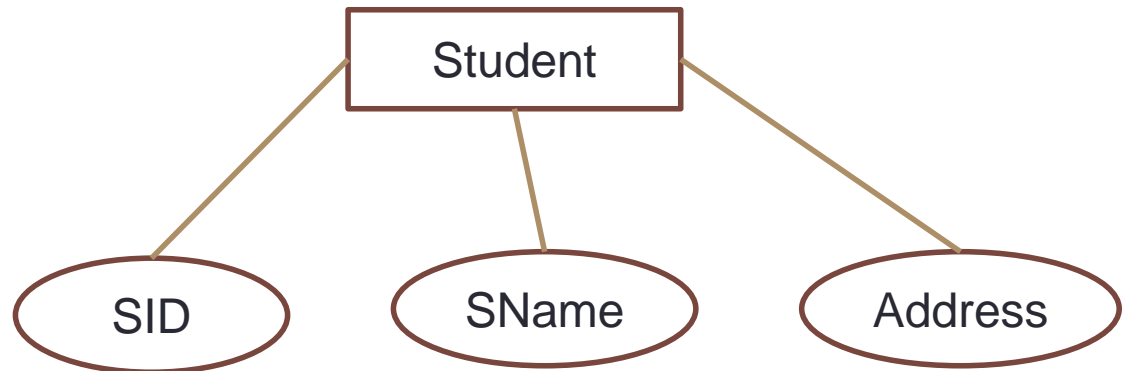
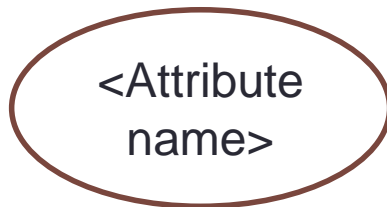
Attributes

- Each attribute is a characteristic of the entity.
- Attributes have
 - A name
 - An associated entity
 - Values for each instance of the entity they are belong to

Attributes

- ERD Notation

E.g.



Types of Attributes in ER Model

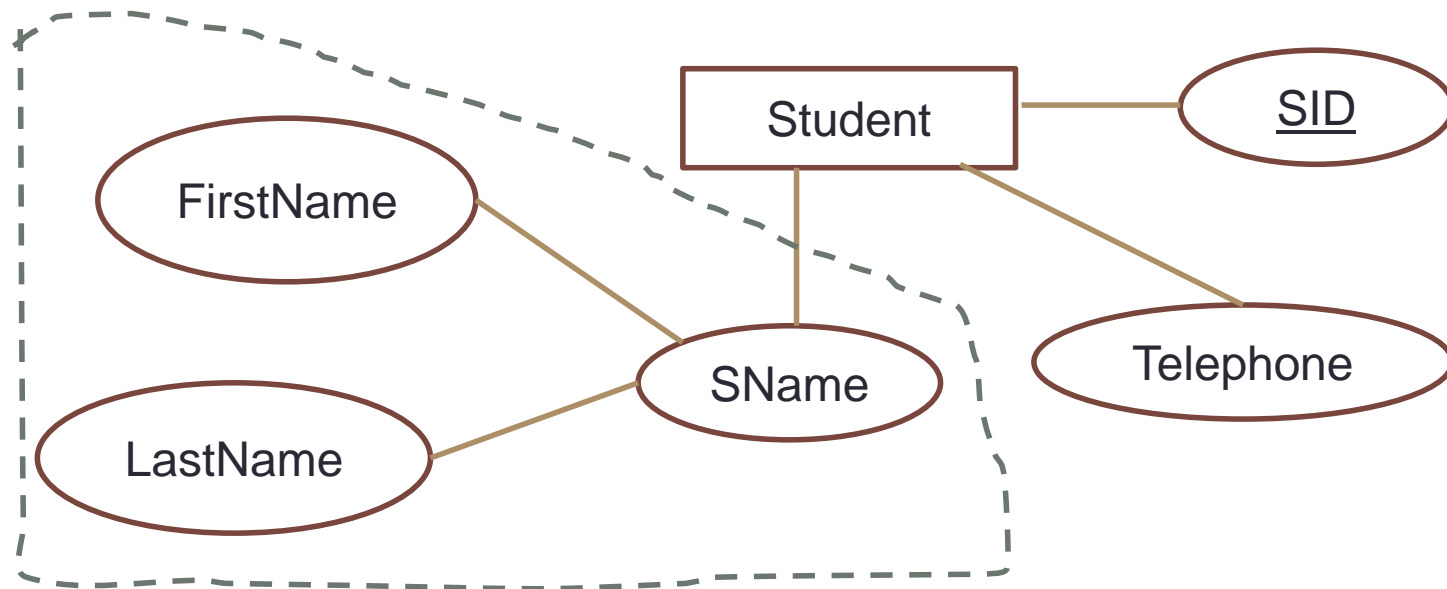
- Simple Vs. Composite
- Single valued Vs. Multivalued
- Stored Vs. Derived

Simple (atomic) Attribute

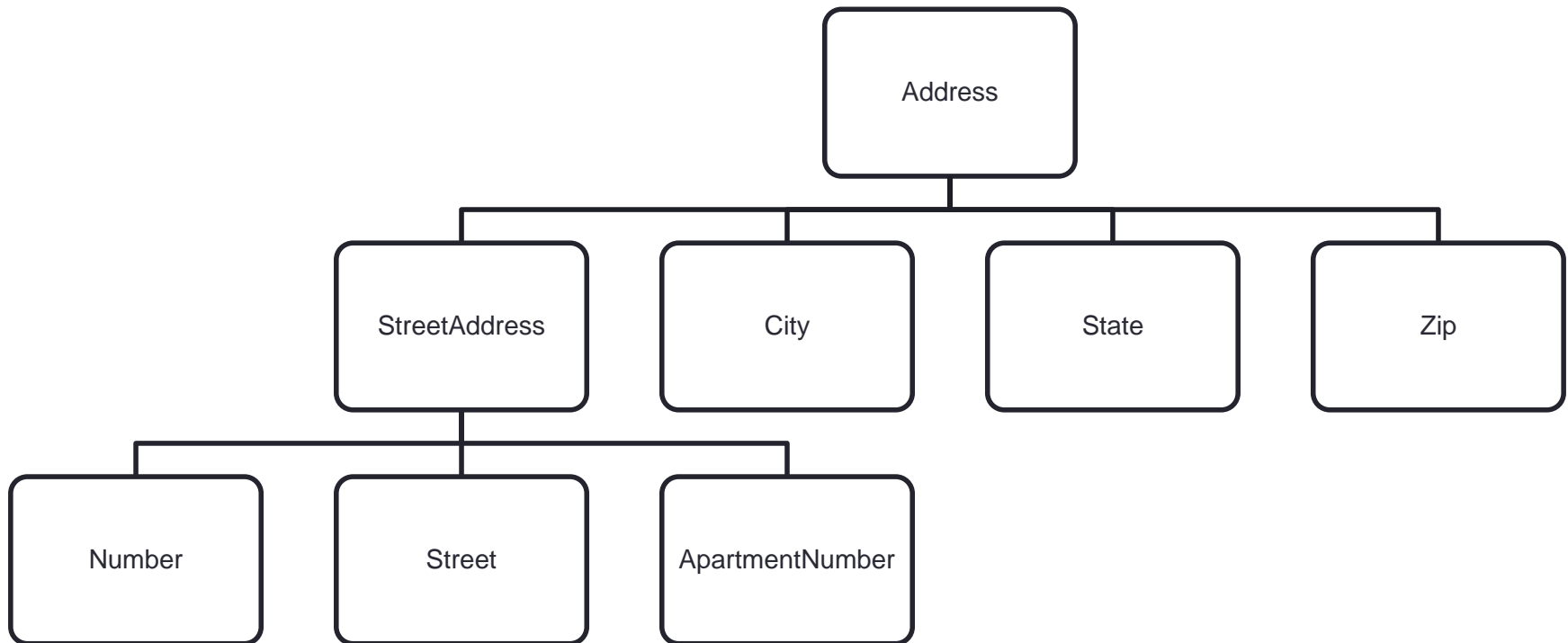
- Simple attributes are atomic values, which are not divisible further.
- E.g. a student's phone number is an atomic value of 10 digits.

Composite Attributes

- Attributes are further divided in a tree like structure.
- Every node is then connected to its attribute.
- That is, composite attributes are represented by ellipses that are connected with an ellipse.



Hierarchy of Composite Attribute: Address



Single-valued and Multivalued Attribute

- If an attribute can have more than one value it is called an multivalued attribute.
 - E.g. person can have more than one phone number, email_address, a teacher entity can have multiple subject values.
 - It is important to note that this is different to an attribute having its own attributes (composite attribute).
 - There may be lower and upper bounds to constrain the number of values allowed for each entity



- Single-valued attributes contain single value.
 - E.g. NIC_Number of a person, Age of a person

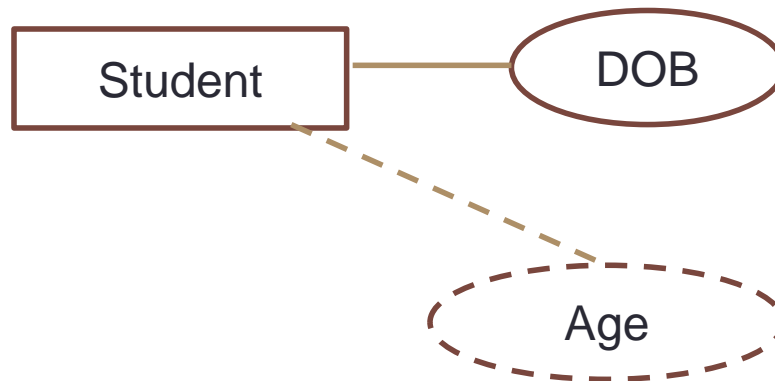
Stored Attribute and Derived Attribute

- Derived attribute is an attribute based on another attribute.
 - They do not exist in the physical database, but their values are derived from other attributes present in the database.
 - This is found rarely in ER diagrams.
 - E.g.
 - Circle Area can be derived from the Radius.
 - AverageSalary in a department should not be saved directly in the database, instead it can be derived.
 - Age can be derived from DOB (data of birth).
 - Some attribute values can be derived from related entities
 - E.g. NumberOfEmployees of a department entity can be derived from counting the number of employees working for that department
- DOB (Date of Birth) is a stored attribute
 - Derived attributes are derivable from stored attributes

Stored Attribute and Derived Attribute

DOB is a Stored Attribute

Age is a Derived Attribute



Null Values

- A particular entity may not have an applicable value for an attribute
- E.g.
 - ApartmentNumber of an address only applicable to apartments and not to other types of residences like single-family homes
 - CollegeDegree attribute is only applicable to a person with a degree
- For such situations a special value called null is created.
 - For a person with no college degree will have null value for CollegeDegree attribute
- Also null value could be used if the value of an attribute of a particular entity
 - E.g.
 - It is known that the attribute value exists but is missing: If the height attribute is missing we can use null value
 - It is not known whether the attribute value exists: If the telephone number of a specific person is not known we can use null value

Identifier Attributes

- Identifiers (keys)
 - Key will identify specific instance in the entity class
 - E.g.

- NationalIdentityCardNumber,
- StudentId,
- EmployeeId,
- EmailAddress,
- DepartmentId

Sample unique key

Sample non-unique key

e.g. Employee entity

All the employees in a department will have the same DepartmentID. That is, there will be more than 1 employee with the same DepartmentID.

So it not unique to specific instance of the Employee.

Types of Keys

Varies based on:

- Uniqueness
 - Keys may be unique or non-unique
 - If the key is unique, the data value for the key must be unique among all instances of the entity
- Composite
 - A composite key consists of two or more attributes
 - It gains uniqueness by combining two or more attributes
 - E.g. Flight Number and Flight Date

Candidate Keys

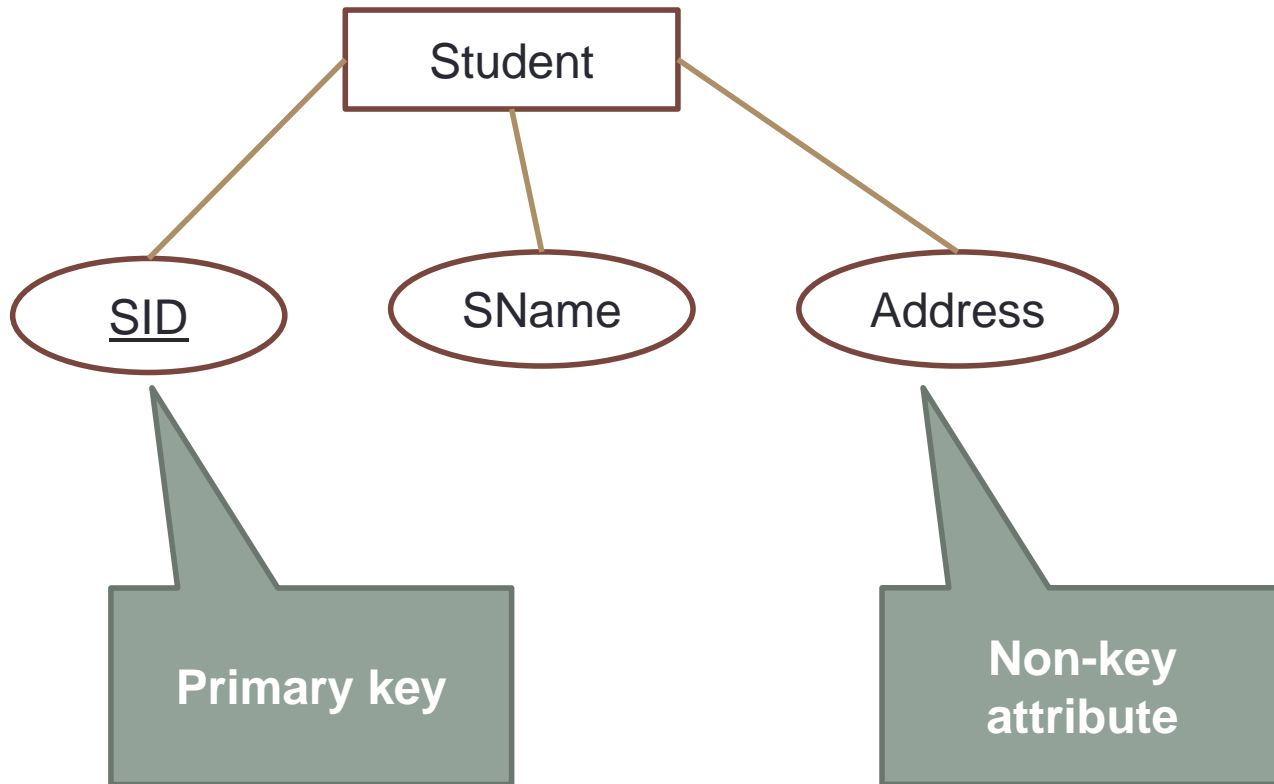
- Candidate keys are defined as the set of attributes from which primary key can be selected.
- It is an attribute or set of attribute that can act as a primary key for an entity (or table) to uniquely identify each instance in that entity.
- **Primary key** is a candidate key that is most appropriate to become main key of the entity .
- It is a key that uniquely identify each instance in an entity .

Non-key Attributes

- They are attributes other than **candidate key** attributes in a table.
- E.g. in Student entity
 - DateOfBirth
 - Location
 - GPA

Attributes

- Example



Example

- A company has several departments. Each department has a supervisor and at least one employee. Employees must be assigned to at least one, but possibly more departments. At least one employee is assigned to a project, but an employee may be on vacation and not assigned to any projects. The important data fields are the names of the departments, projects, supervisors and employees, as well as the supervisor and employee number and a unique project number.
- **Find Attributes**

Example – Identify attributes

- The only attributes indicated are
 - the names of the departments, projects, supervisors and employees,
 - the supervisor and employee NUMBER
 - a unique project number.

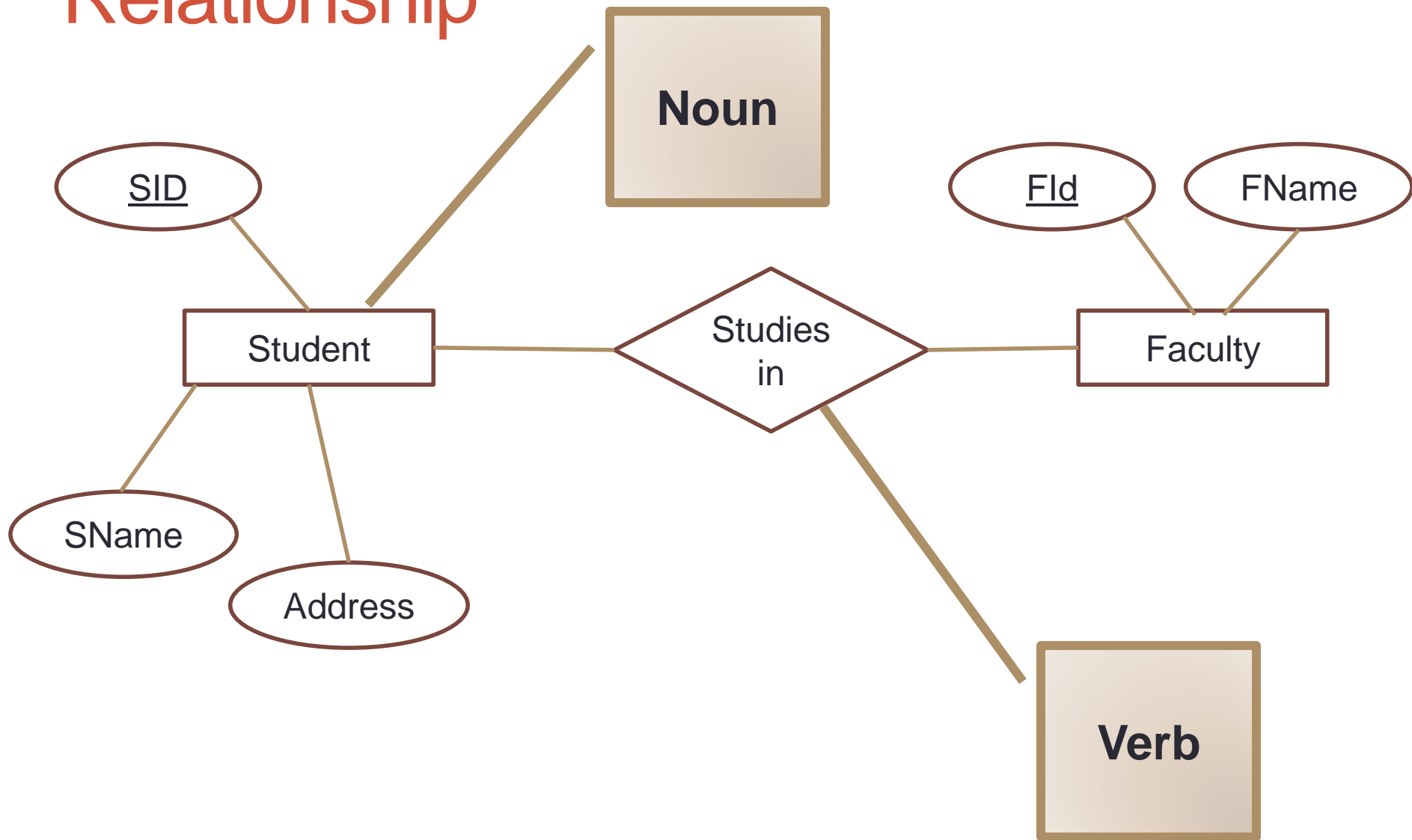
Relationship

- It is the logical association among entities
- Relationships are mapped with entities in various ways.
- Mapping cardinalities define the number of association between two entities. E.g. one to one, one to many, etc.
- E.g.
 - Each **Student** takes several **Modules**
 - Each **Module** is taught by a **Lecturer**
 - Each **Employee** works for a single **Department**

Relationship

- Relationships have
 - A name
 - A set of entities that participate in them
 - A degree - the number of entities that participate (most have degree 2)
 - A cardinality ratio

Relationship



Naming Relationships

- A relationship represents an action being taken. Therefore, a relationship name is a verb phrase in present tense
- Relationship in ERD should be read from left to right or from up to down. Use passive voice if the action is not carried out by the entity on the left (or top).
 - E.g. Subject (left) – **Is_studied_by** – Student (right)
- A relationship name states the action taken, not the result of the action.
 - E.g. Use **Works_in** not *Staff_of*, Use **Is_assigned** not *Assignment*, Use **Orders** not *Business*, Use **Participates** not *A_participant_of*

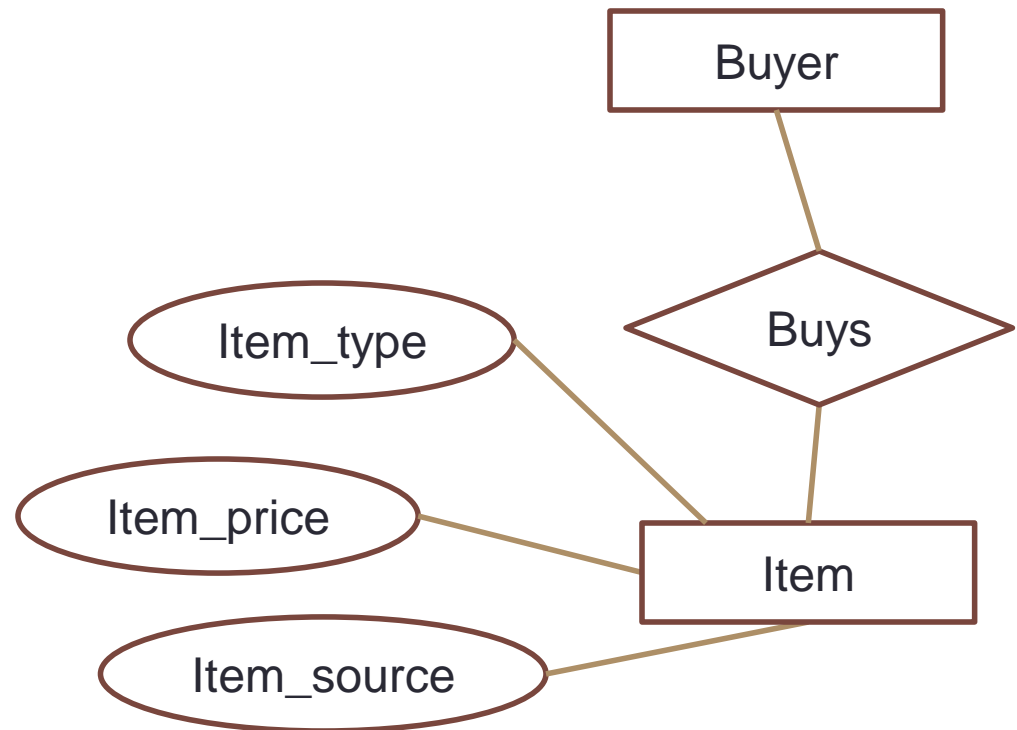
Naming Relationships

- Avoid using vague terms
 - E.g. *Is_related_to*
- Ignore calculating processes
 - E.g. Calculate average marks
- Ignore business activities like transfer, send, summarize that are not relevant to design of the database system
 - E.g. sending report card to parents should be ignored

Example 1

- An inventory software used in a retail shop will have a database that monitors elements such as item type, item source and item price for each item and demographic details of Buyers.

- Identify entities
 - Buyer, item
- Identify Attributes
- Identify relationships



Example 2

- A company has several departments. Each department has a supervisor and at least one employee. Employees must be assigned to at least one, but possibly more departments. At least one employee is assigned to a project, but an employee may be on vacation and not assigned to any projects. The important data fields are the names of the departments, projects, supervisors and employees, as well as the supervisor and employee number and a unique project number.
- **Find Relationships**

Example 2: Find Relationships

Entity Relationship Matrix:

	Department	Employee	Supervisor	Project
Department		is assigned	run by	
Employee	belongs to			works on
Supervisor	runs			
Project		uses		

Example 2: ER Diagram

