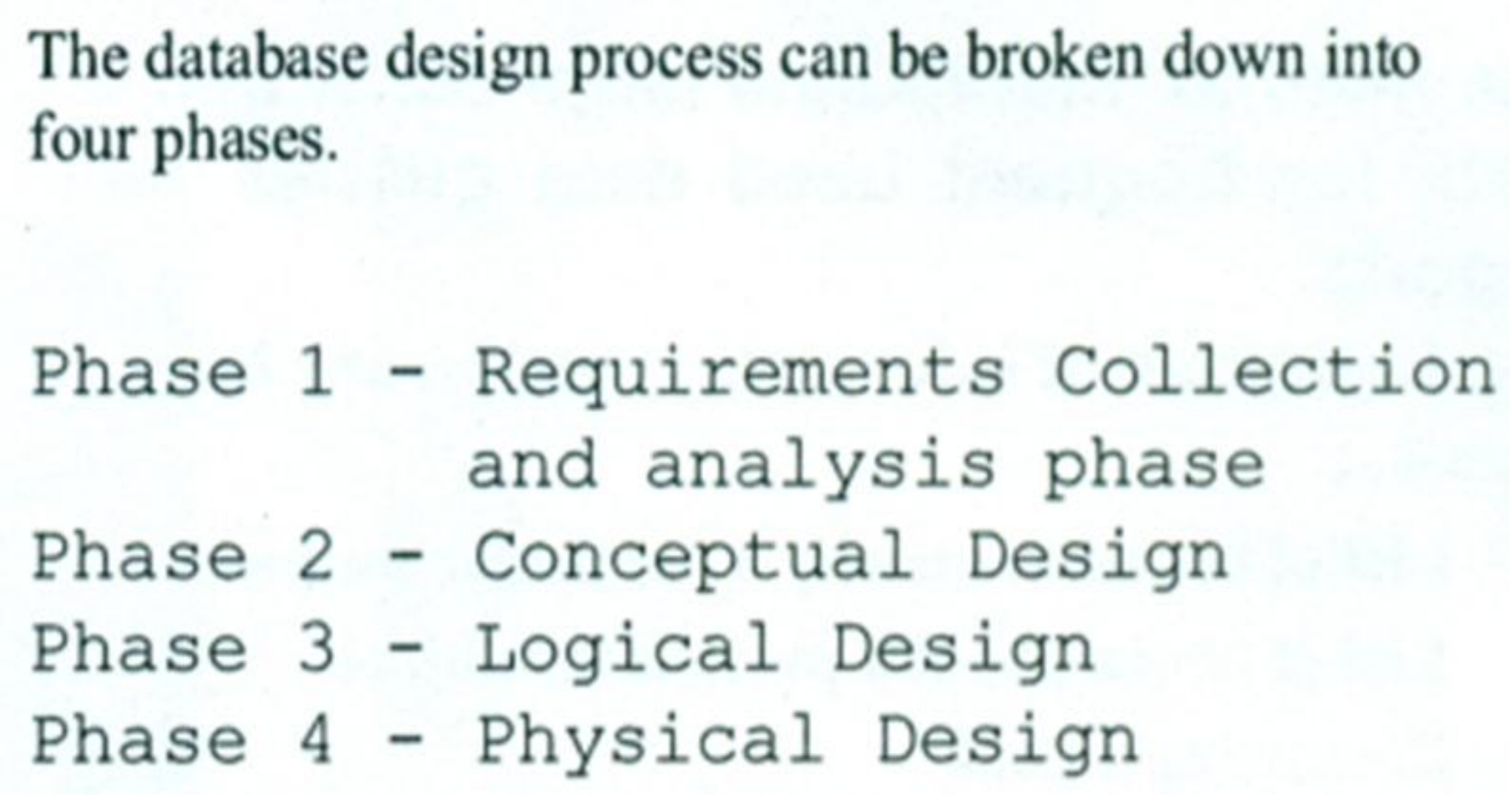
Conceptual data modelling

(Handout 2)

Database Design



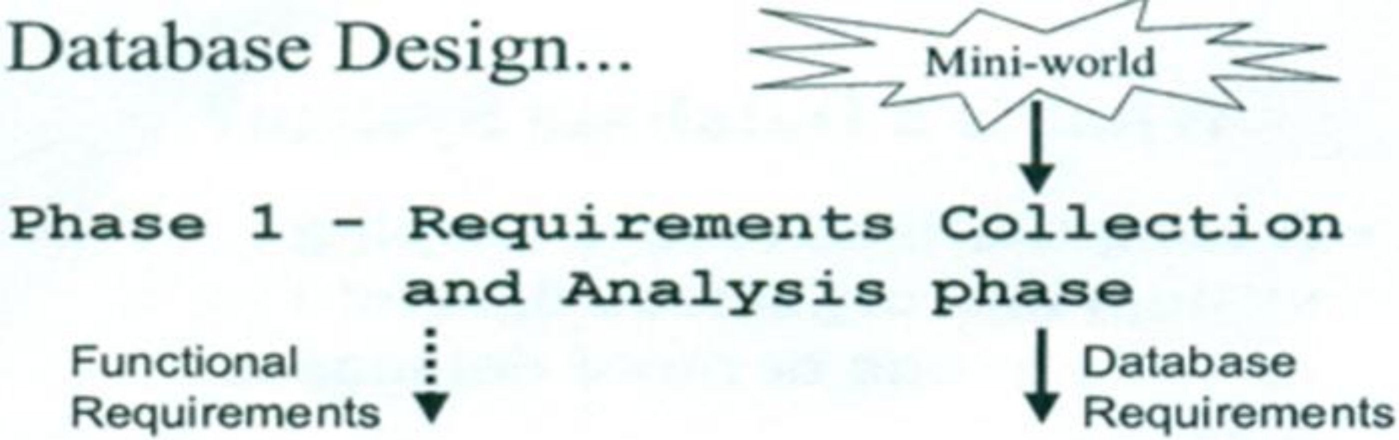
The database design process can be broken down into four phases.

Phase 1 - Requirements Collection
and analysis phase

Phase 2 - Conceptual Design

Phase 3 - Logical Design

Phase 4 - Physical Design



Database Design...

Mini-world

Phase 1 - Requirements Collection and Analysis phase

Functional Requirements

Database Requirements

Functional requirements capture the intended behavior of the system (Function or task, service)

- Calculate EPF
- Calculate salary
- Update employee record
- Print pay slip
- Online students registration (service)

Prospective database uses are interviewed to understand and document their data requirements.

Data requirements:

Employee no, name, address

Department no, name

Project no, name, locations

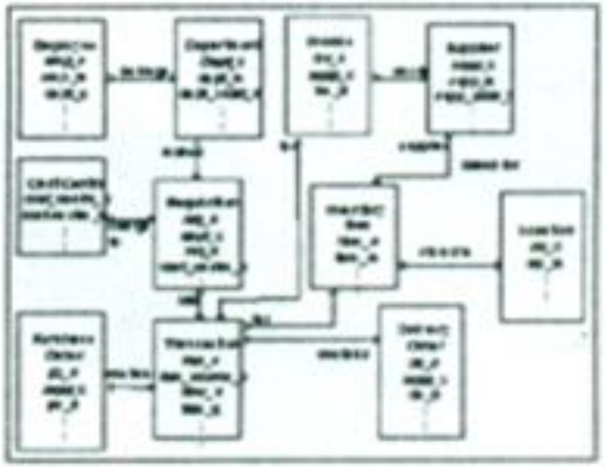
Database Design

Database Requirements

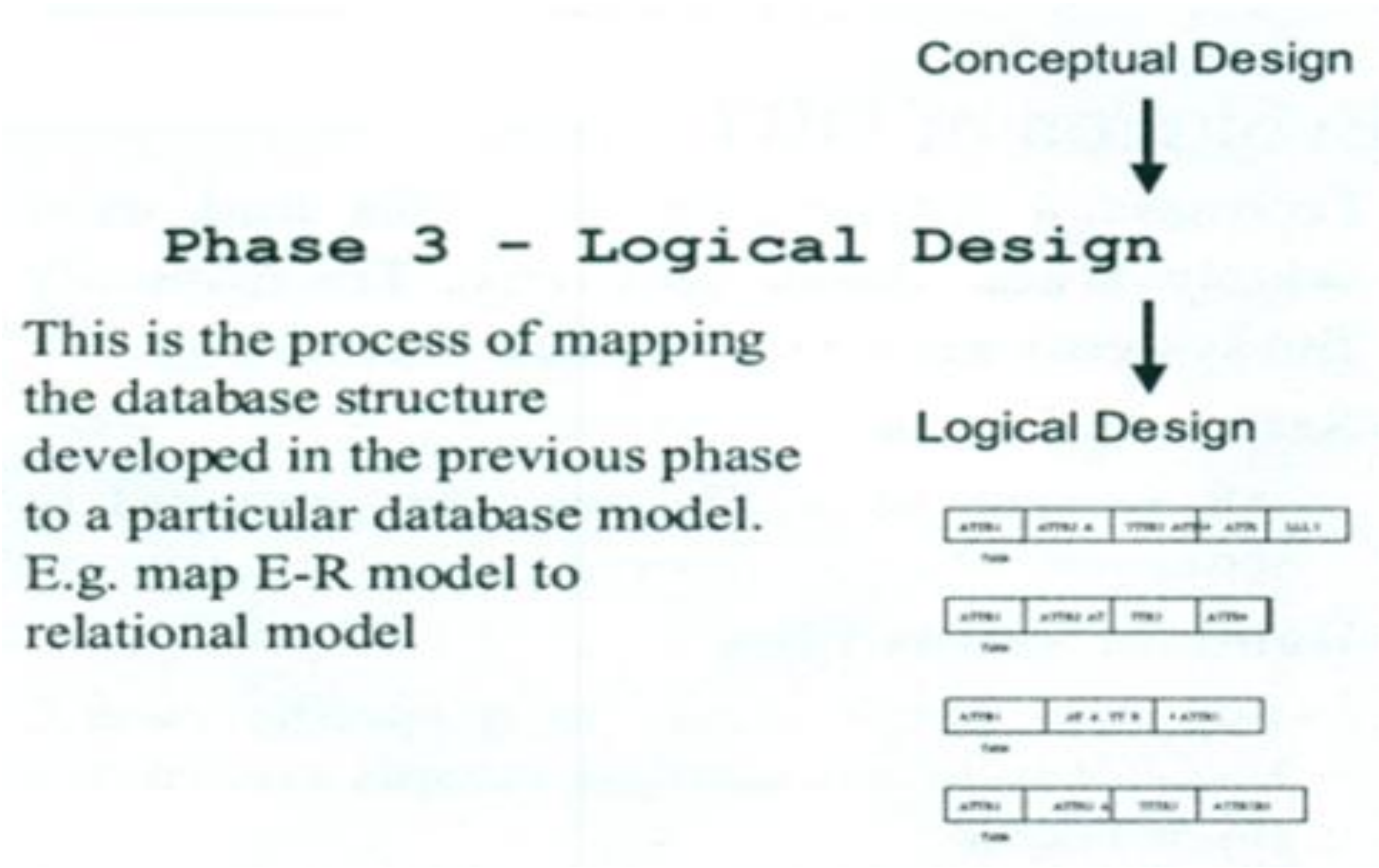
Phase 2 - Conceptual Design

This is high level description of the structure of a database. E.g. E-R diagram

Conceptual Design



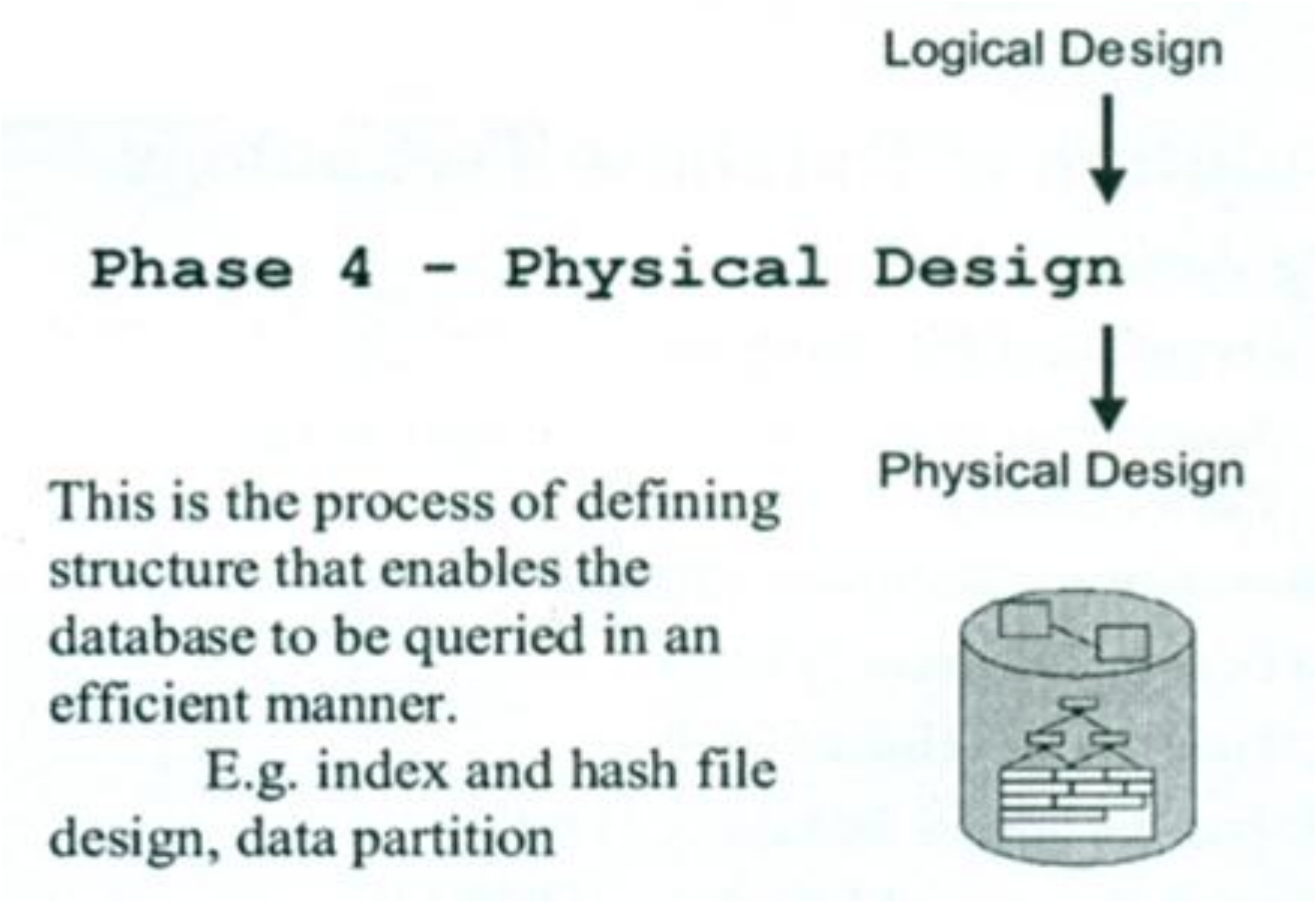
Database Design



Phase 3 - Logical Design

This is the process of mapping the database structure developed in the previous phase to a particular database model. E.g. map E-R model to relational model

Database Design



Logical Design

↓

Phase 4 - Physical Design

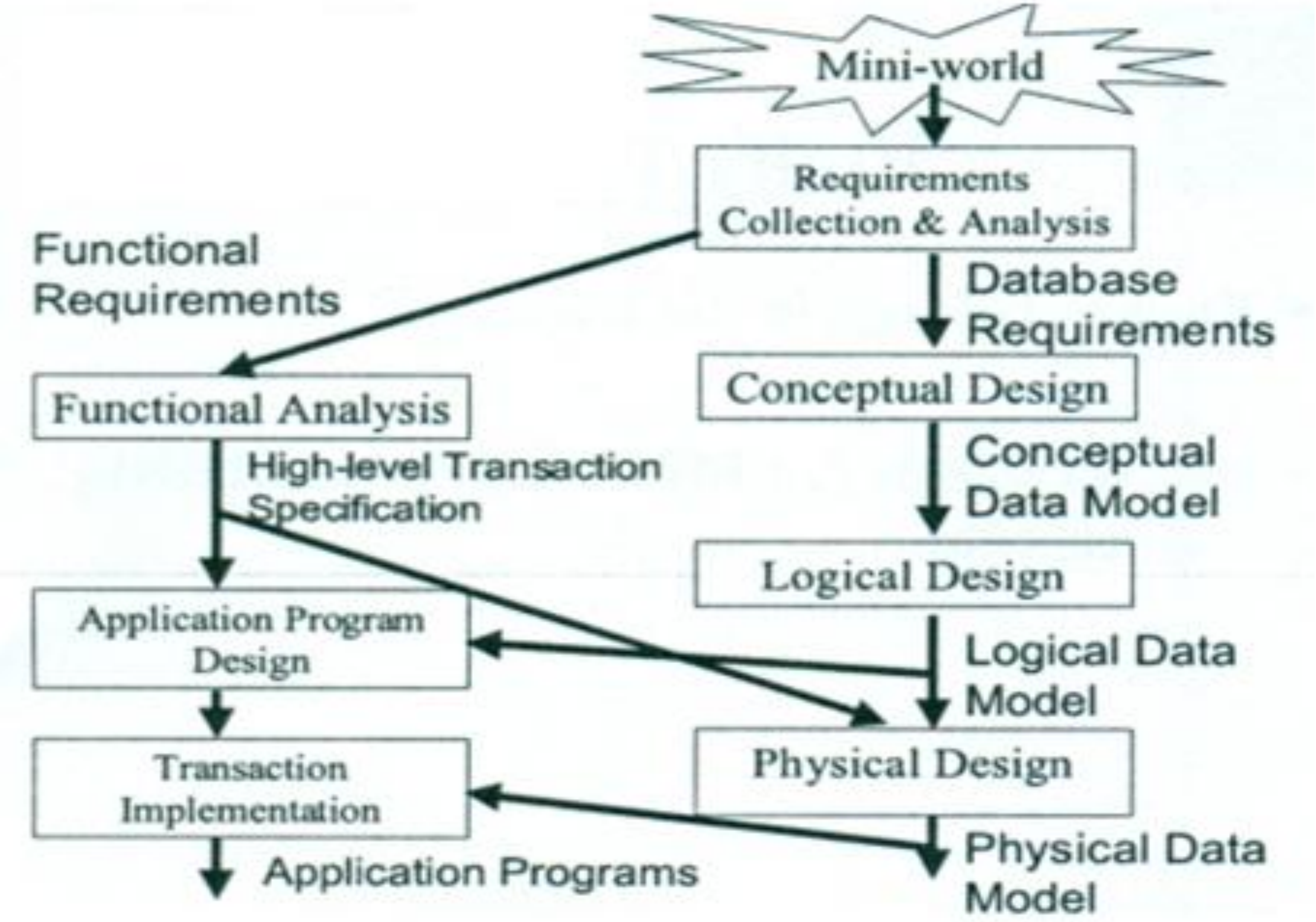
↓

Physical Design

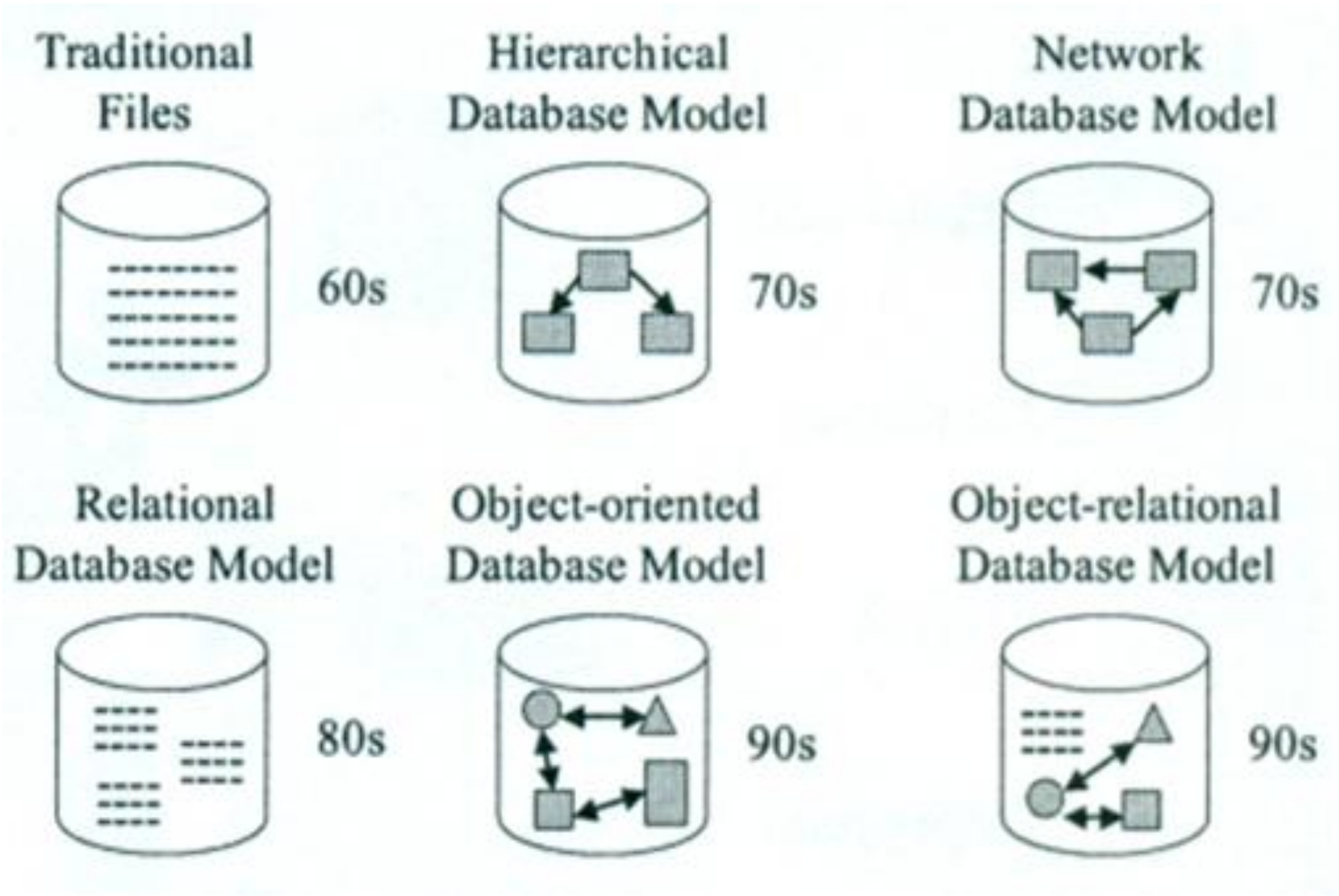
This is the process of defining structure that enables the database to be queried in an efficient manner.

E.g. index and hash file design, data partition

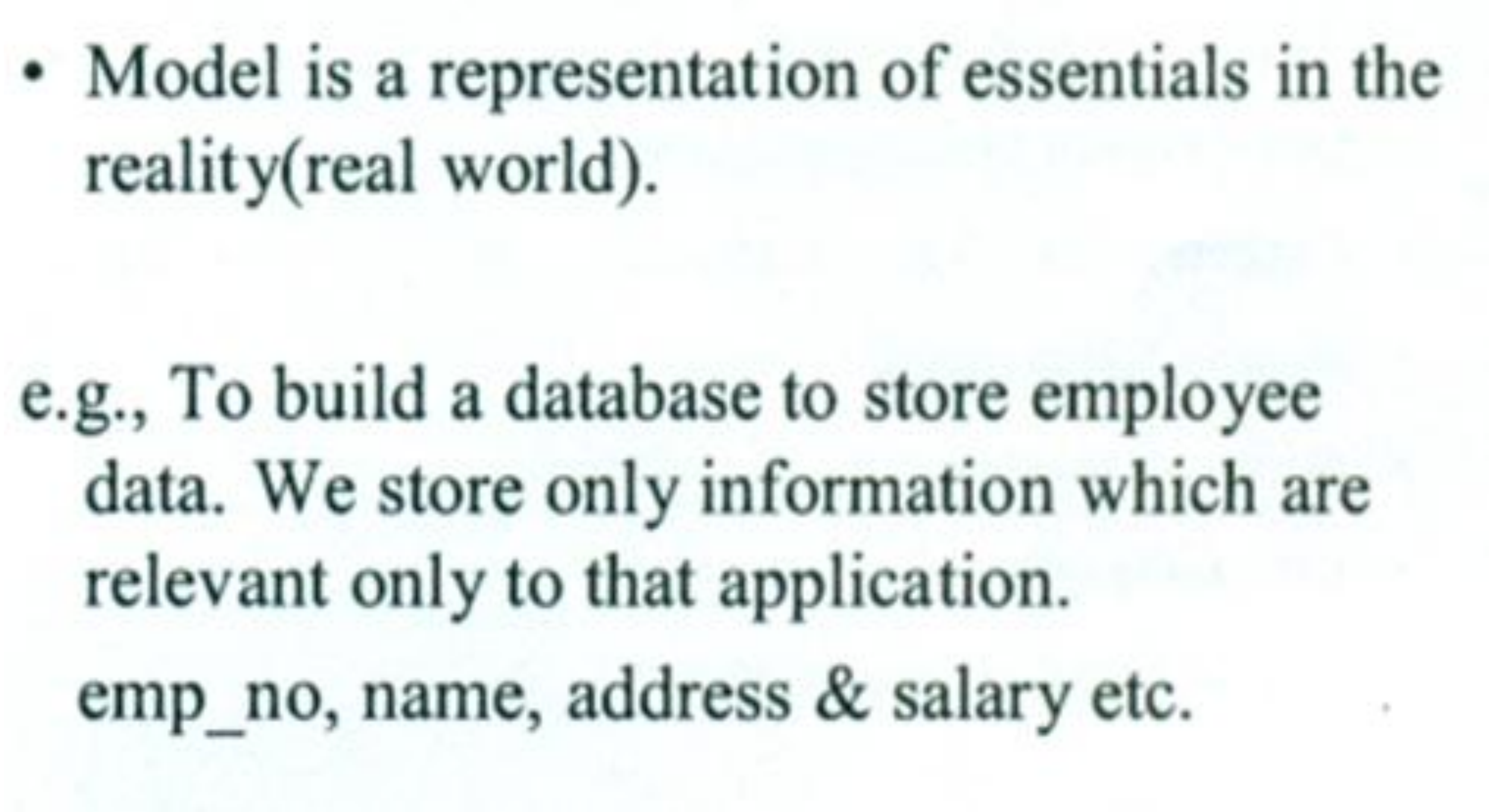
Phases of Database Design



Types of Database Models


Traditional Files
60s
Hierarchical Database Model
70s
Network Database Model
70s
Relational Database Model
80s
Object-oriented Database Model
90s
Object-relational Database Model
90s

Model

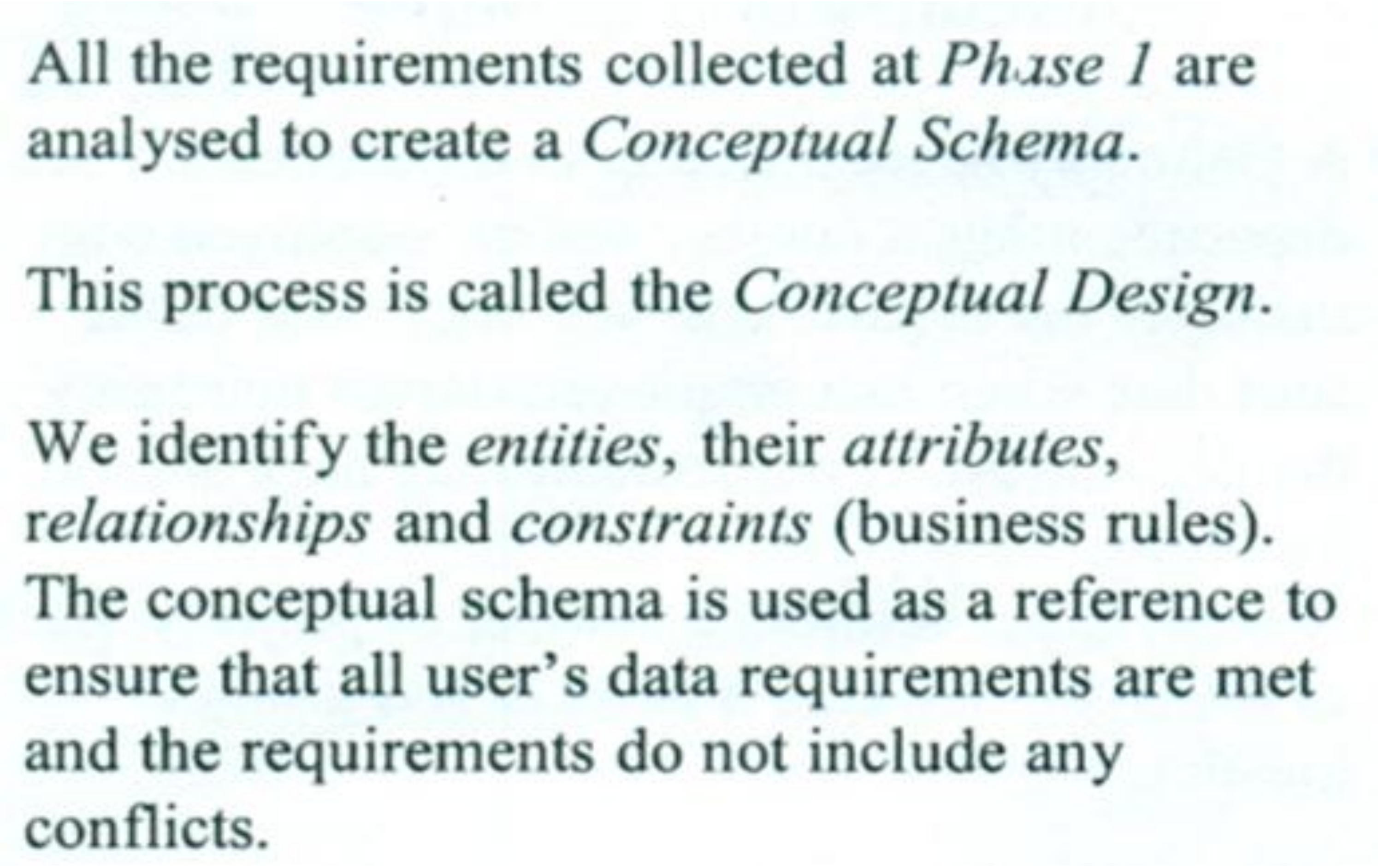


- Model is a representation of essentials in the reality(real world).

e.g., To build a database to store employee data. We store only information which are relevant only to that application.

emp_no, name, address & salary etc.

Conceptual Design

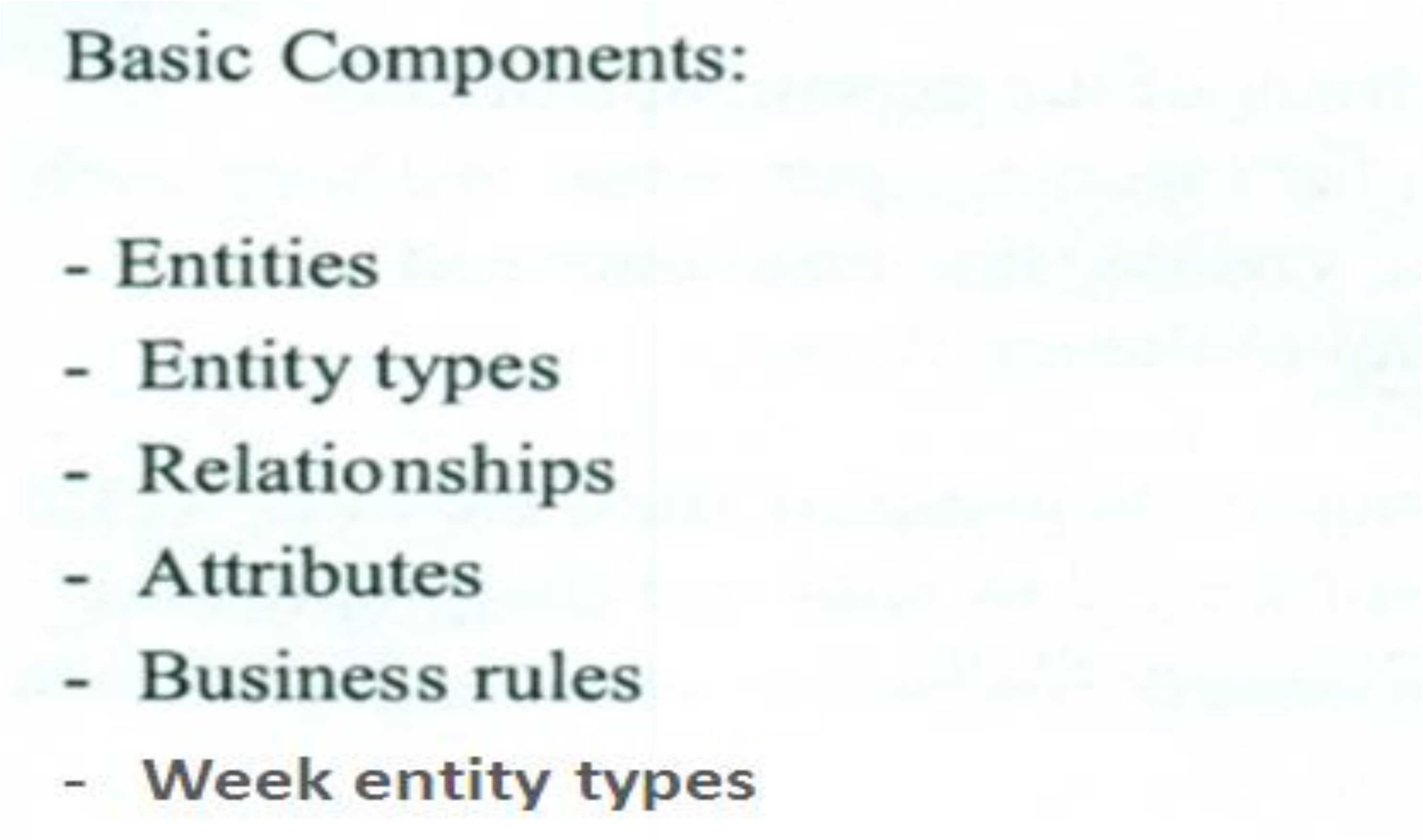


All the requirements collected at *Phase 1* are analysed to create a *Conceptual Schema*.

This process is called the *Conceptual Design*.

We identify the *entities*, their *attributes*, *relationships* and *constraints* (business rules). The conceptual schema is used as a reference to ensure that all user's data requirements are met and the requirements do not include any conflicts.

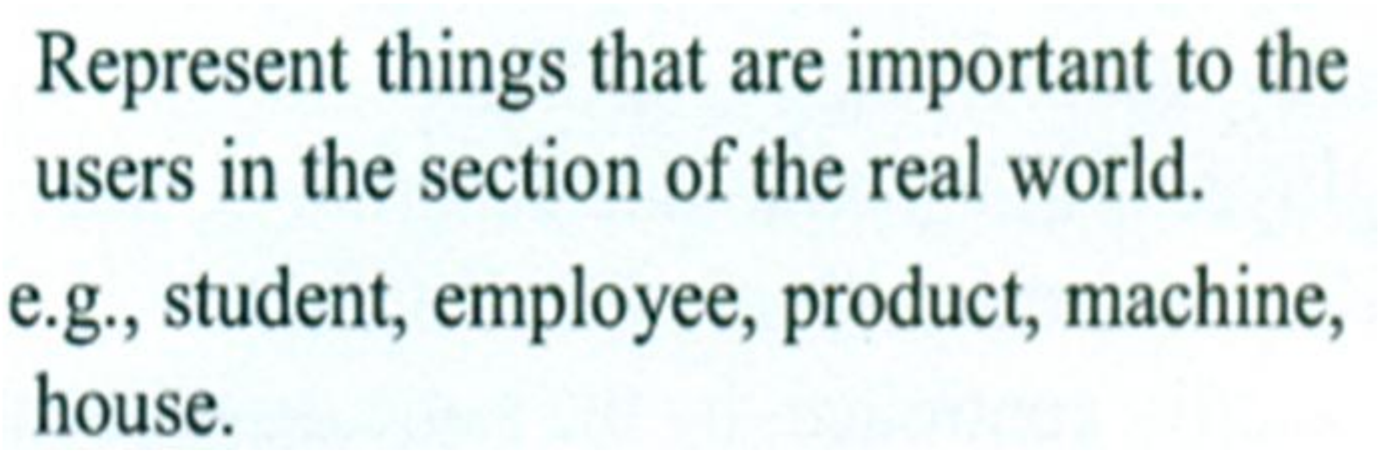
Conceptual Data Modelling



Basic Components:

- Entities
- Entity types
- Relationships
- Attributes
- Business rules
- Week entity types

Entity



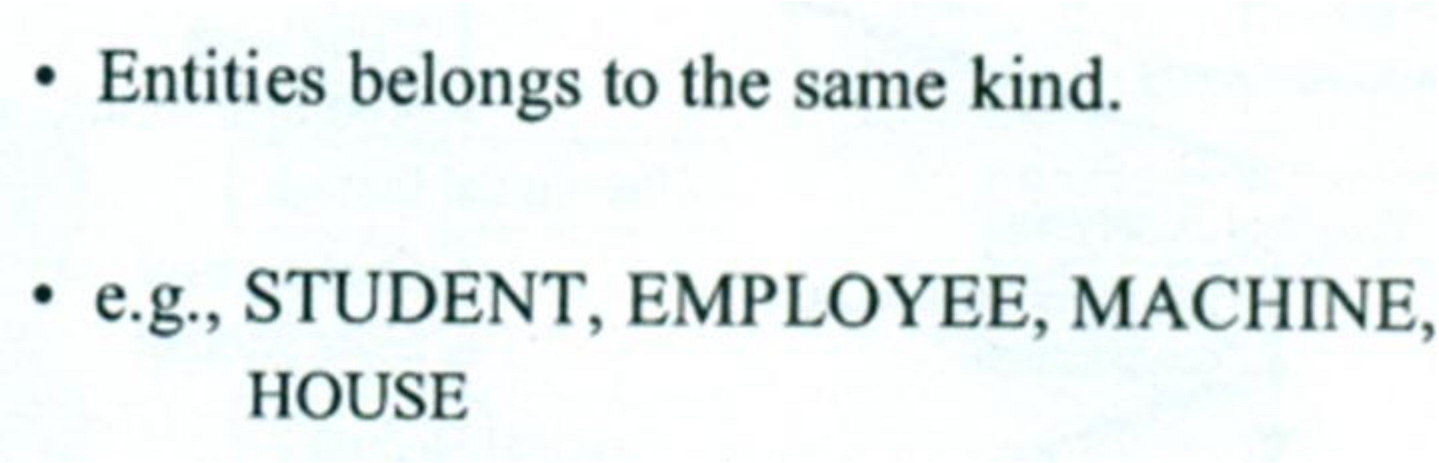
Represent things that are important to the users in the section of the real world.

e.g., student, employee, product, machine, house.

Students: Malan, Peter

Employee: Bandula, Chandrasiri

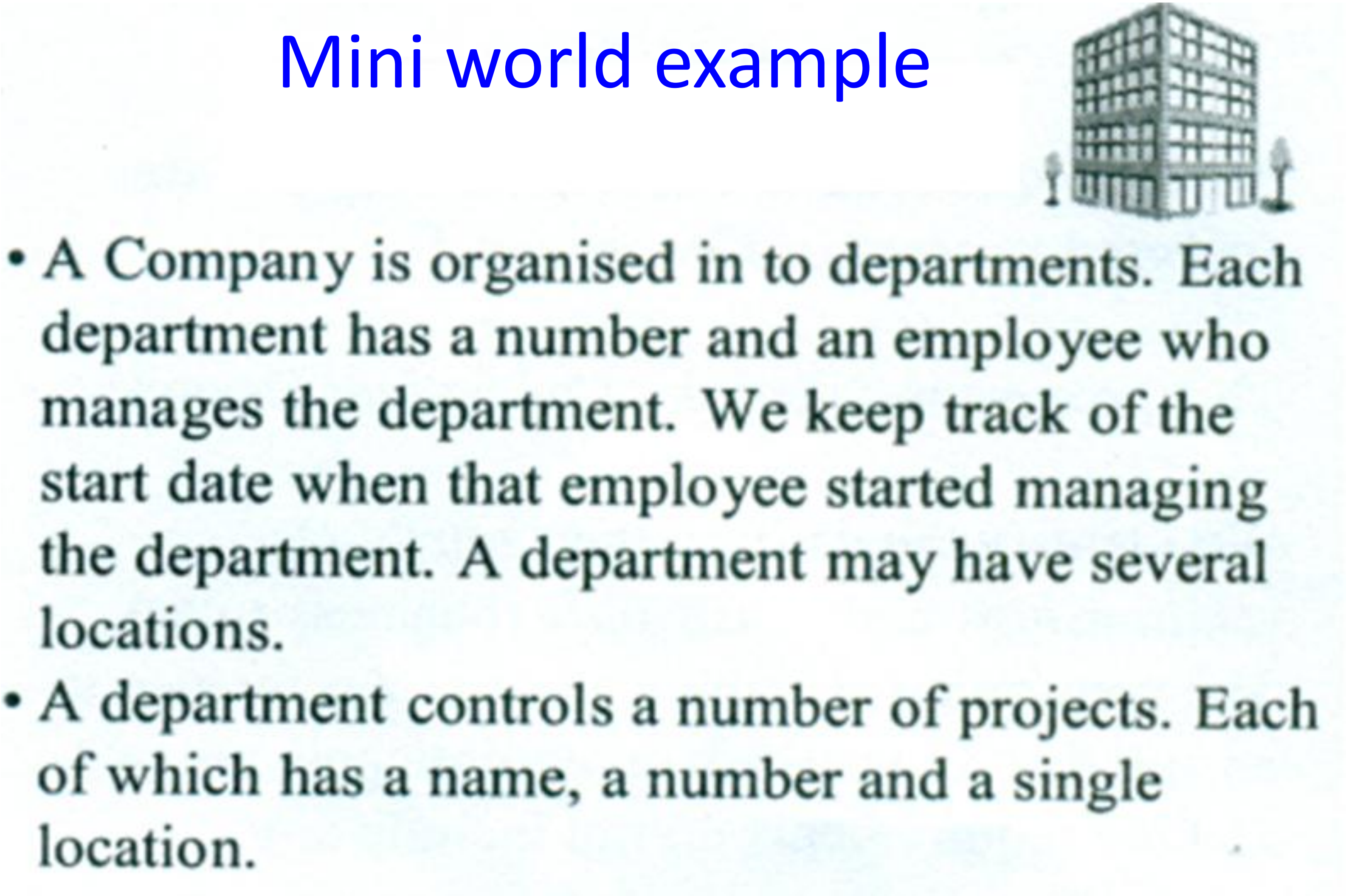
Entity Types



- Entities belongs to the same kind.

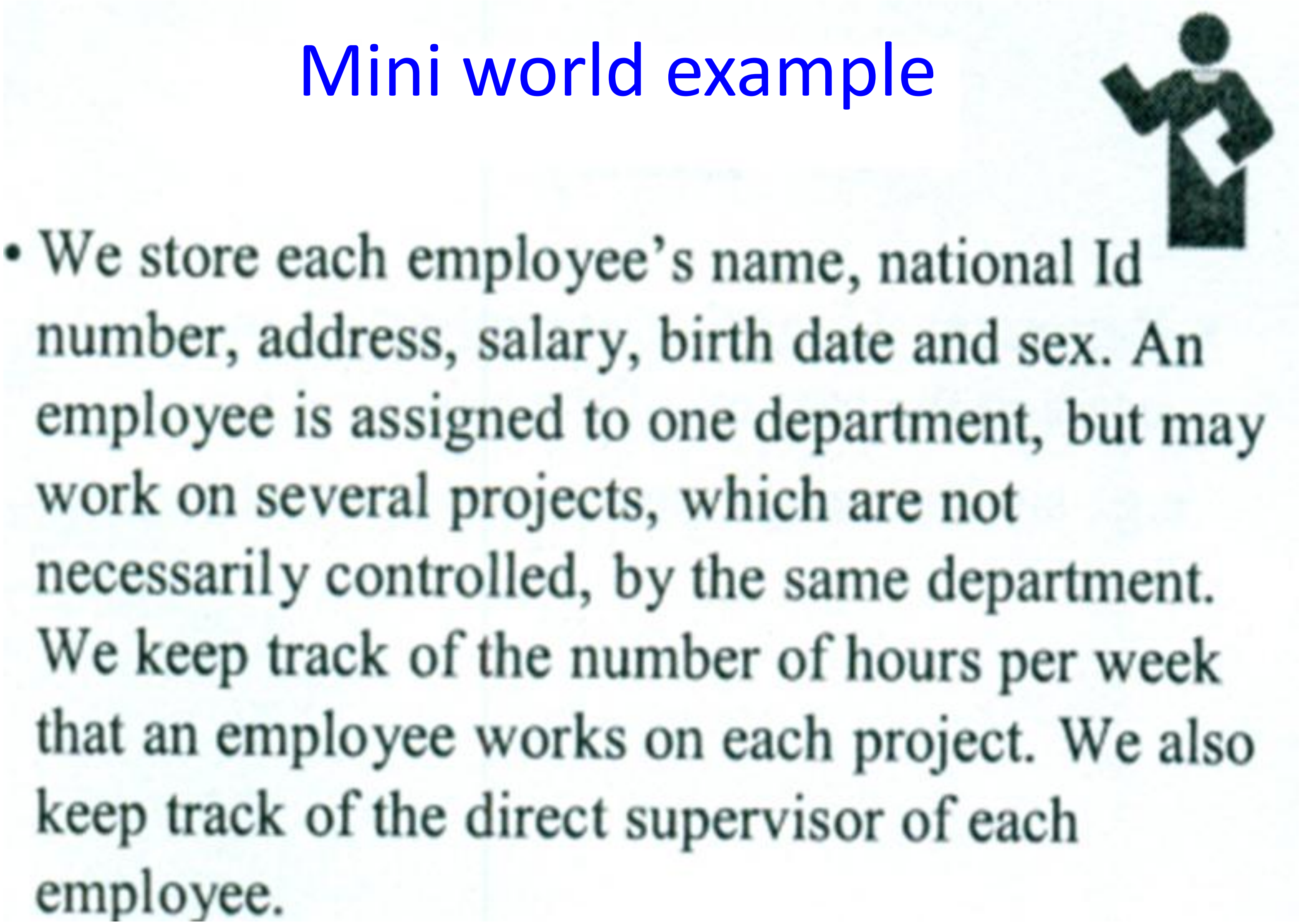
- e.g., STUDENT, EMPLOYEE, MACHINE, HOUSE

Mini world example



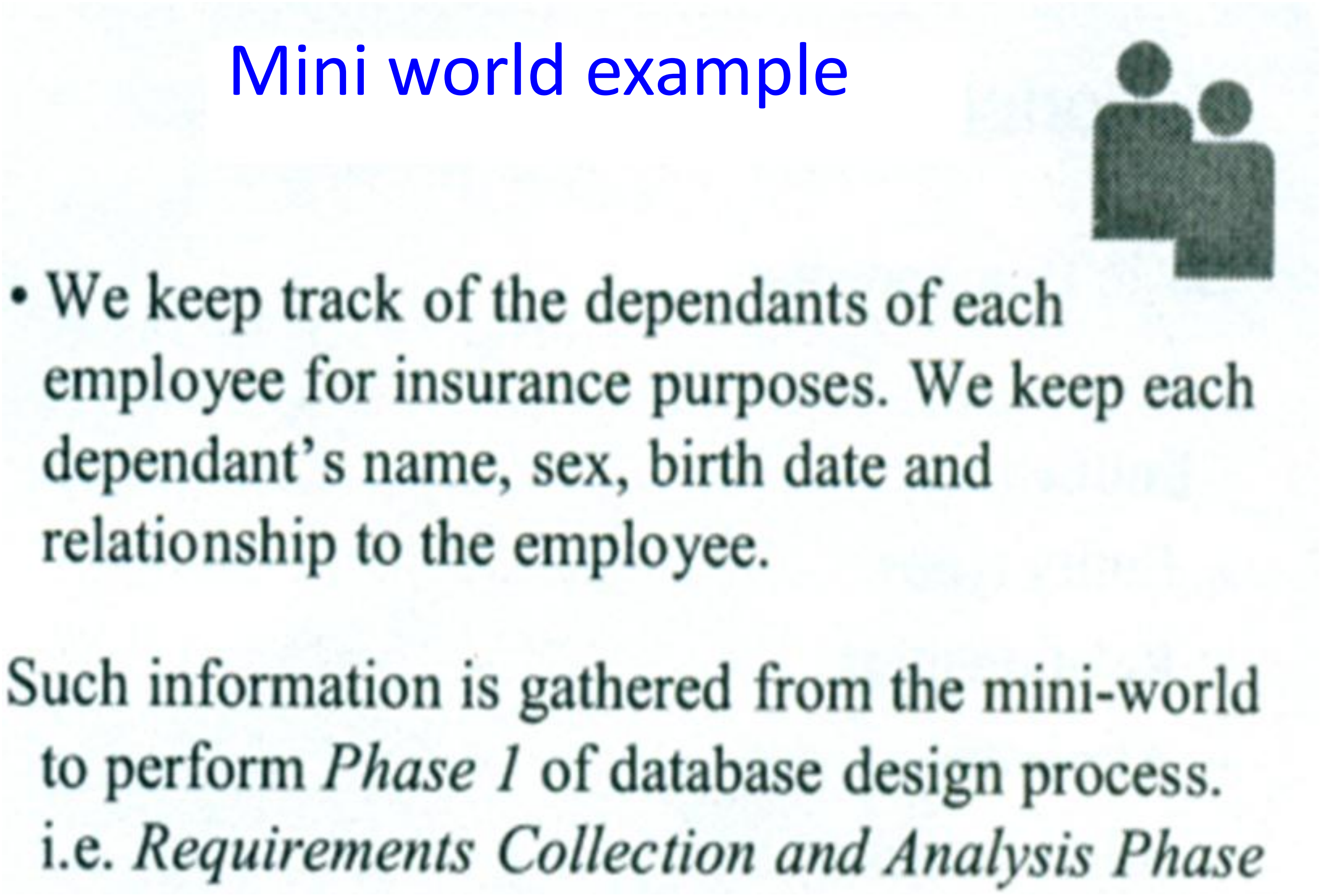
- A Company is organised in to departments. Each department has a number and an employee who manages the department. We keep track of the start date when that employee started managing the department. A department may have several locations.
- A department controls a number of projects. Each of which has a name, a number and a single location.

Mini world example



- We store each employee's name, national Id number, address, salary, birth date and sex. An employee is assigned to one department, but may work on several projects, which are not necessarily controlled, by the same department. We keep track of the number of hours per week that an employee works on each project. We also keep track of the direct supervisor of each employee.

Mini world example



- We keep track of the dependants of each employee for insurance purposes. We keep each dependant's name, sex, birth date and relationship to the employee.

Such information is gathered from the mini-world to perform *Phase 1* of database design process. i.e. *Requirements Collection and Analysis Phase*

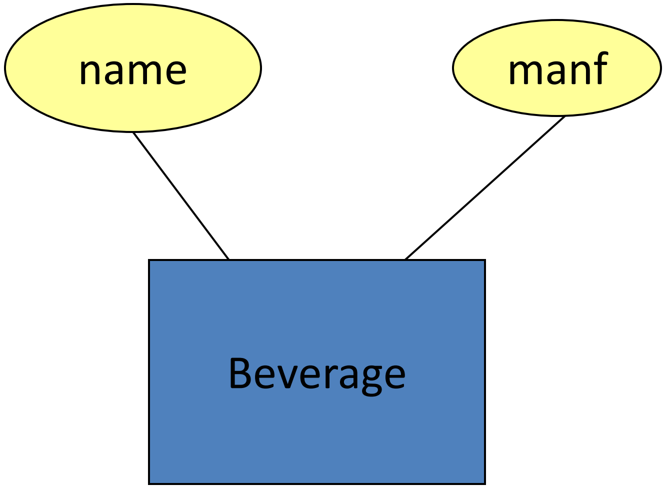
Attribute

Attribute *is a* property of (the entities) an entity set.

Student attribute: Name, Student Id, Address, Gender

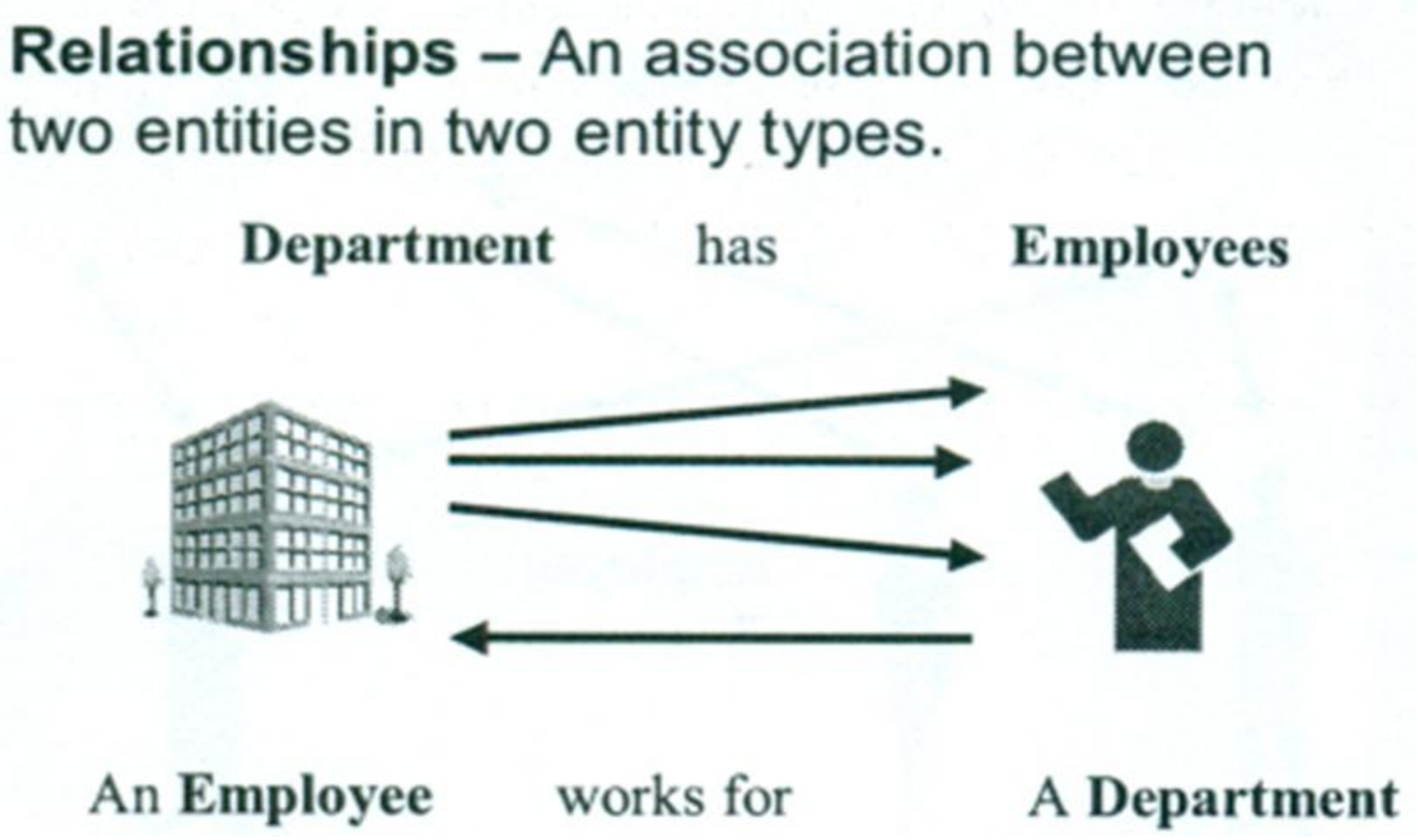
Attributes are simple values, e.g. integers or character strings.

Example



- Entity set Beverages has two attributes, name and manf (manufacturer).
- Each Beverages entity has values for these two attributes, e.g. ("Ginger beer", "Elephant house")

Conceptual Design



Relationships – An association between two entities in two entity types.

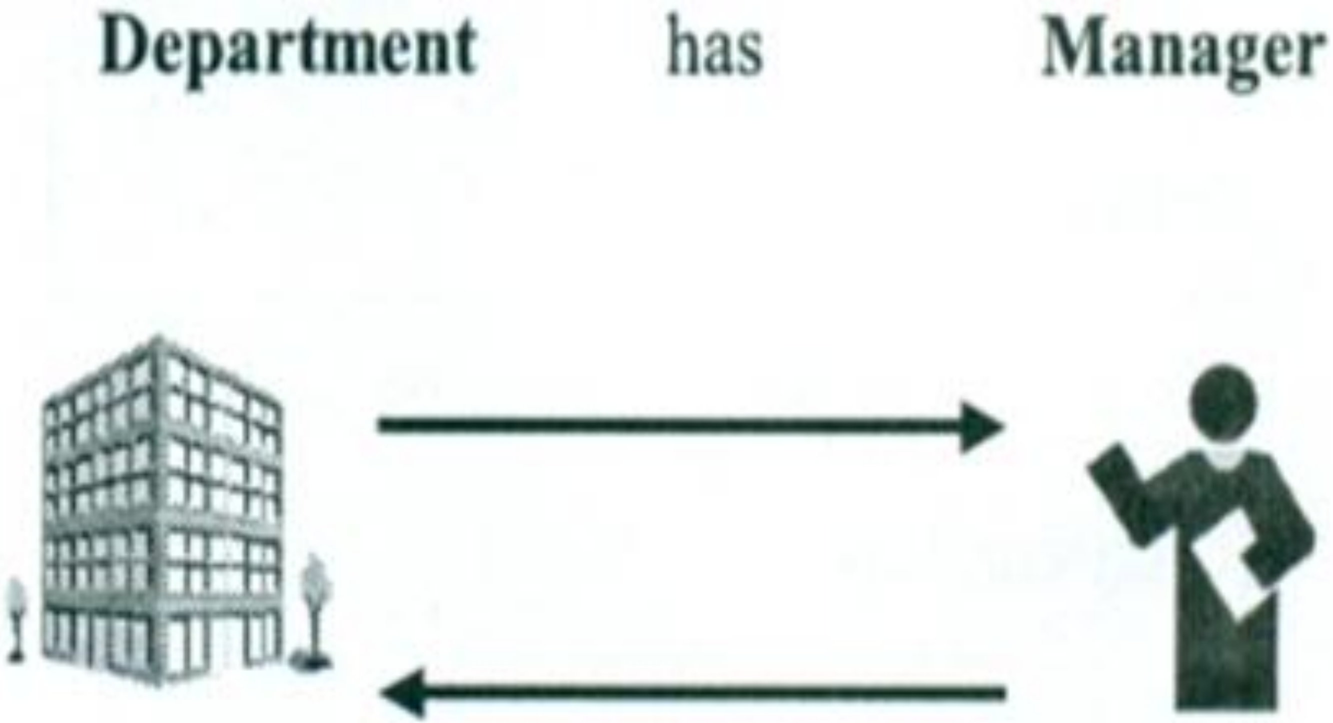
Department has **Employees**

An **Employee** works for A **Department**

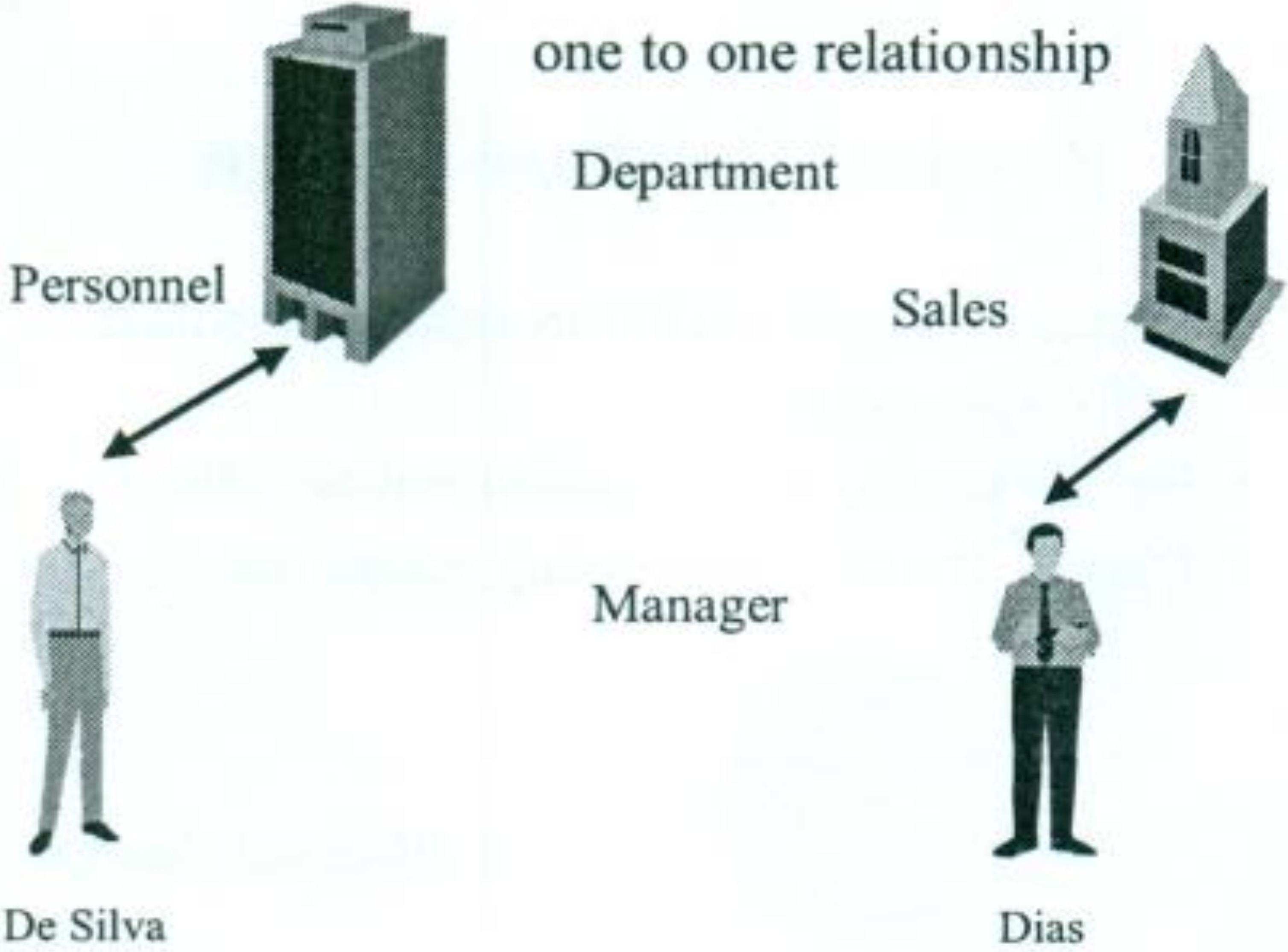
One-One Relationships

- In a *one-one* relationship, each entity of either entity set is related to at most one entity of the other set.
- Example: Relationship has between entity sets Department and Manager
  - A department can be lead by only one manager.

One – One relationships



one to one relationship



Department

Personnel

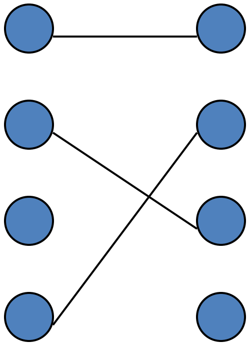
Sales

Manager

De Silva

Dias

In Pictures:

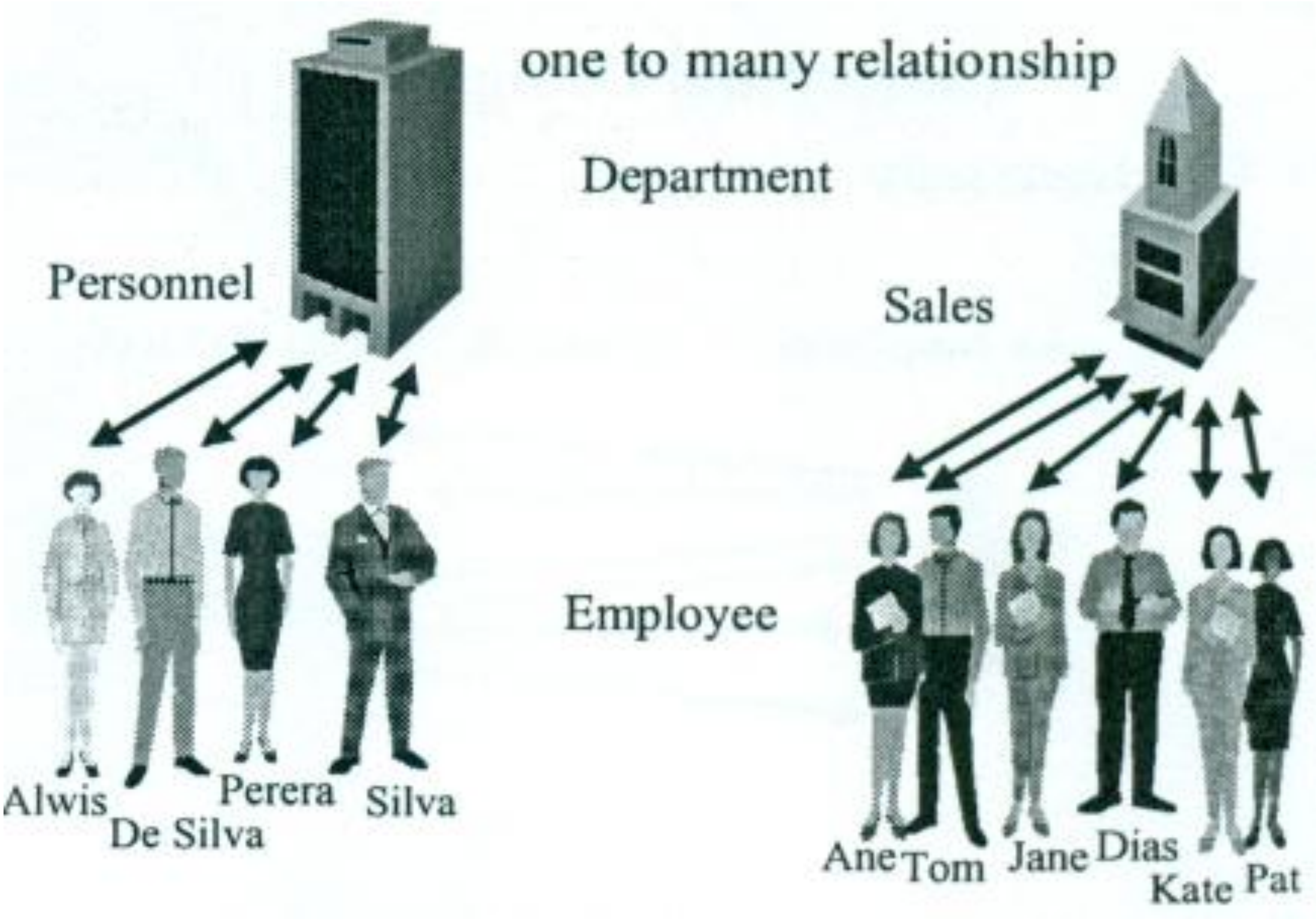


one-one

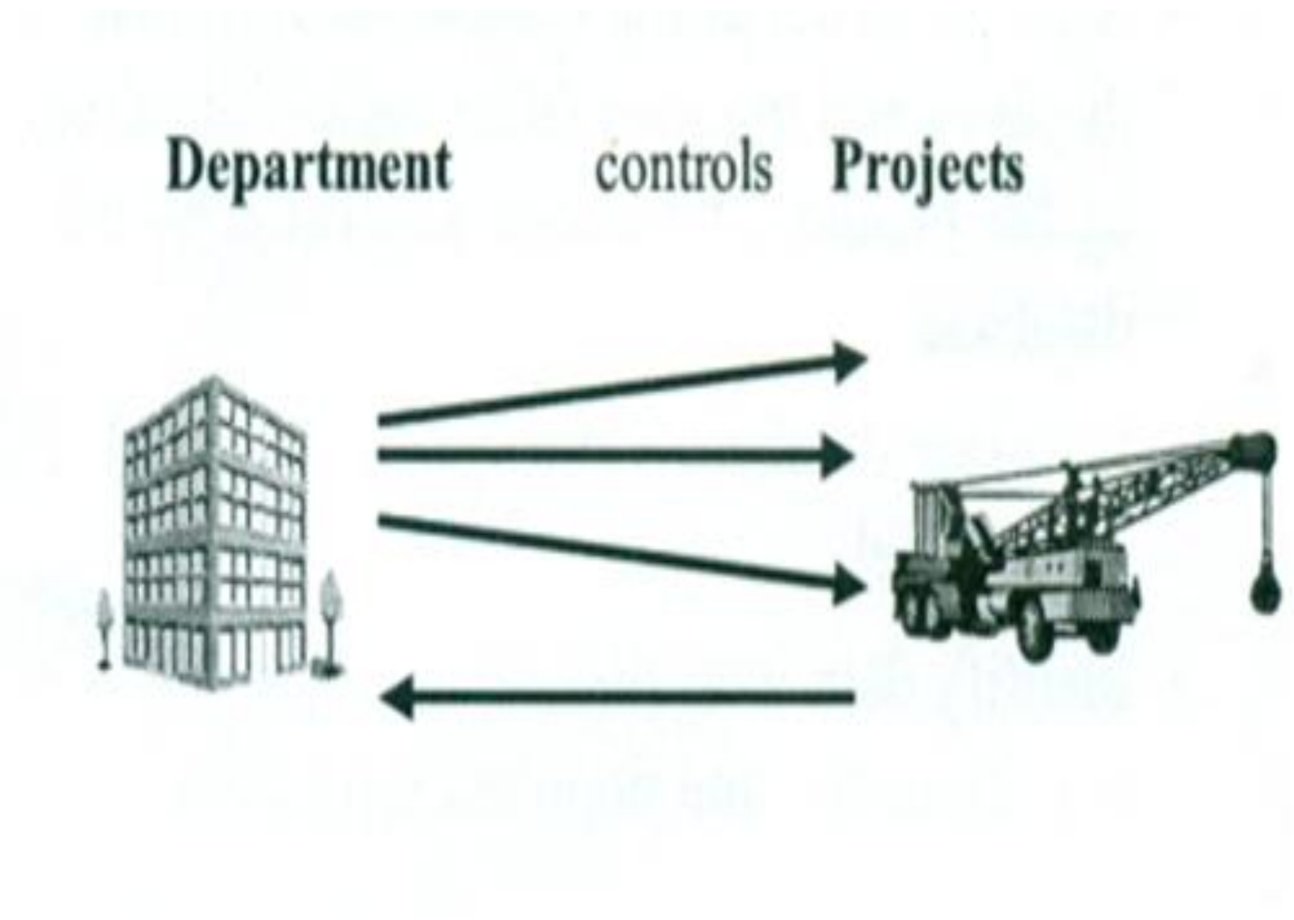
One - Many Relationships

- Each entity of the first set is connected to more than one entity of the second set.
- entity of the first set can be connected to zero, one, or many entities of the second set.

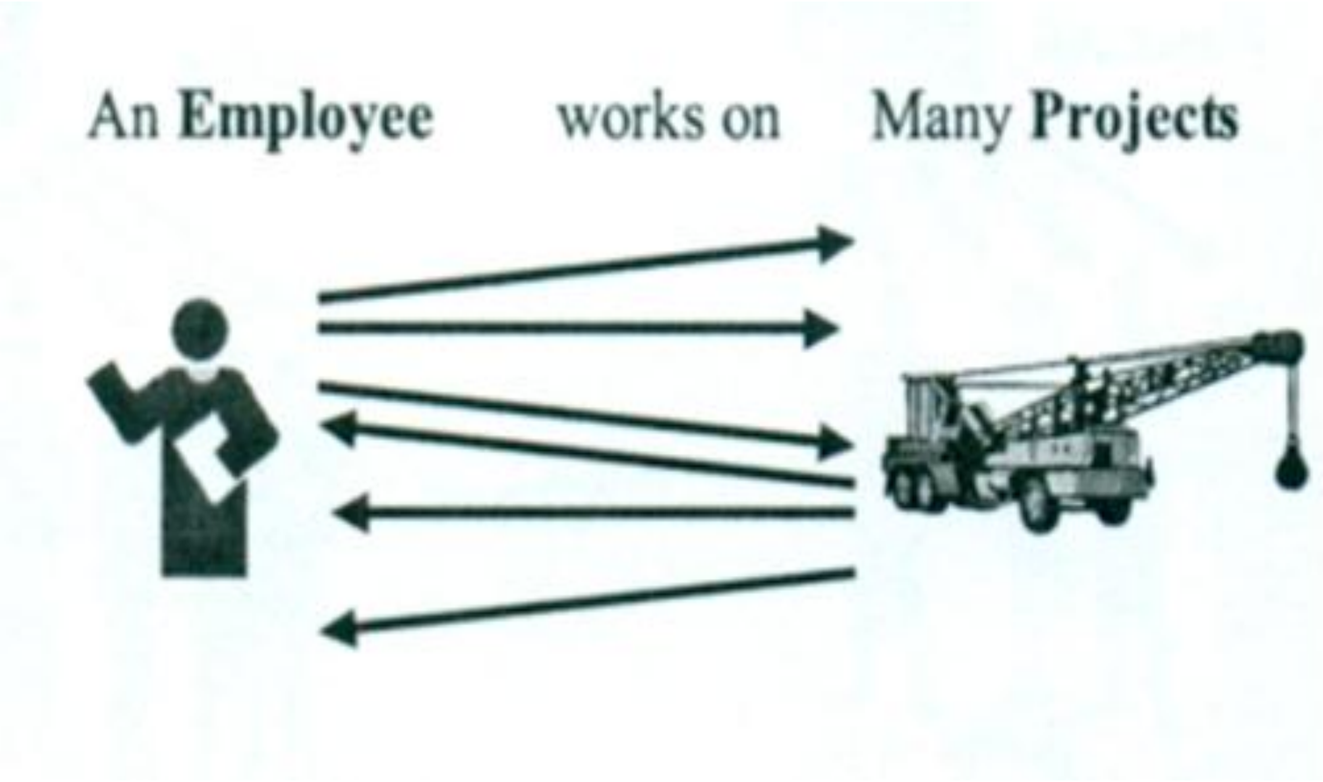
Relationships



One – Many Relationships

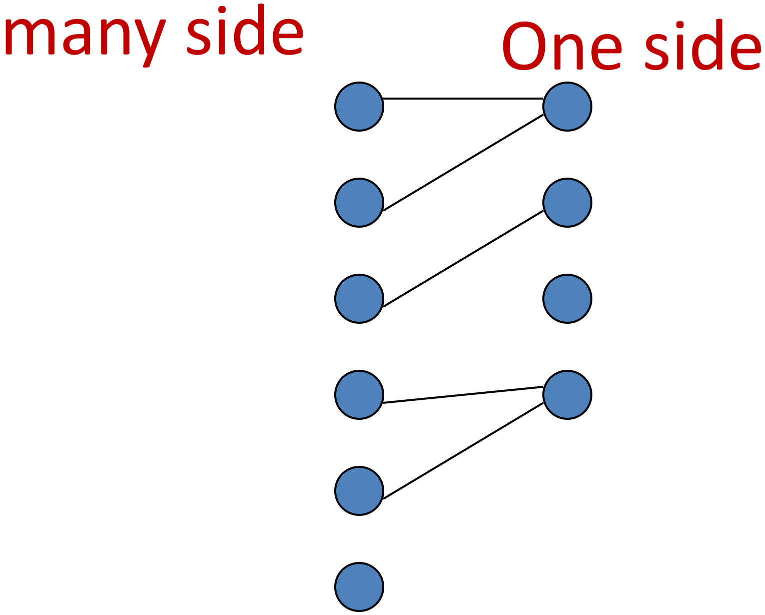


One – Many Relationships



An **Employee** works on Many **Projects**

In Pictures:

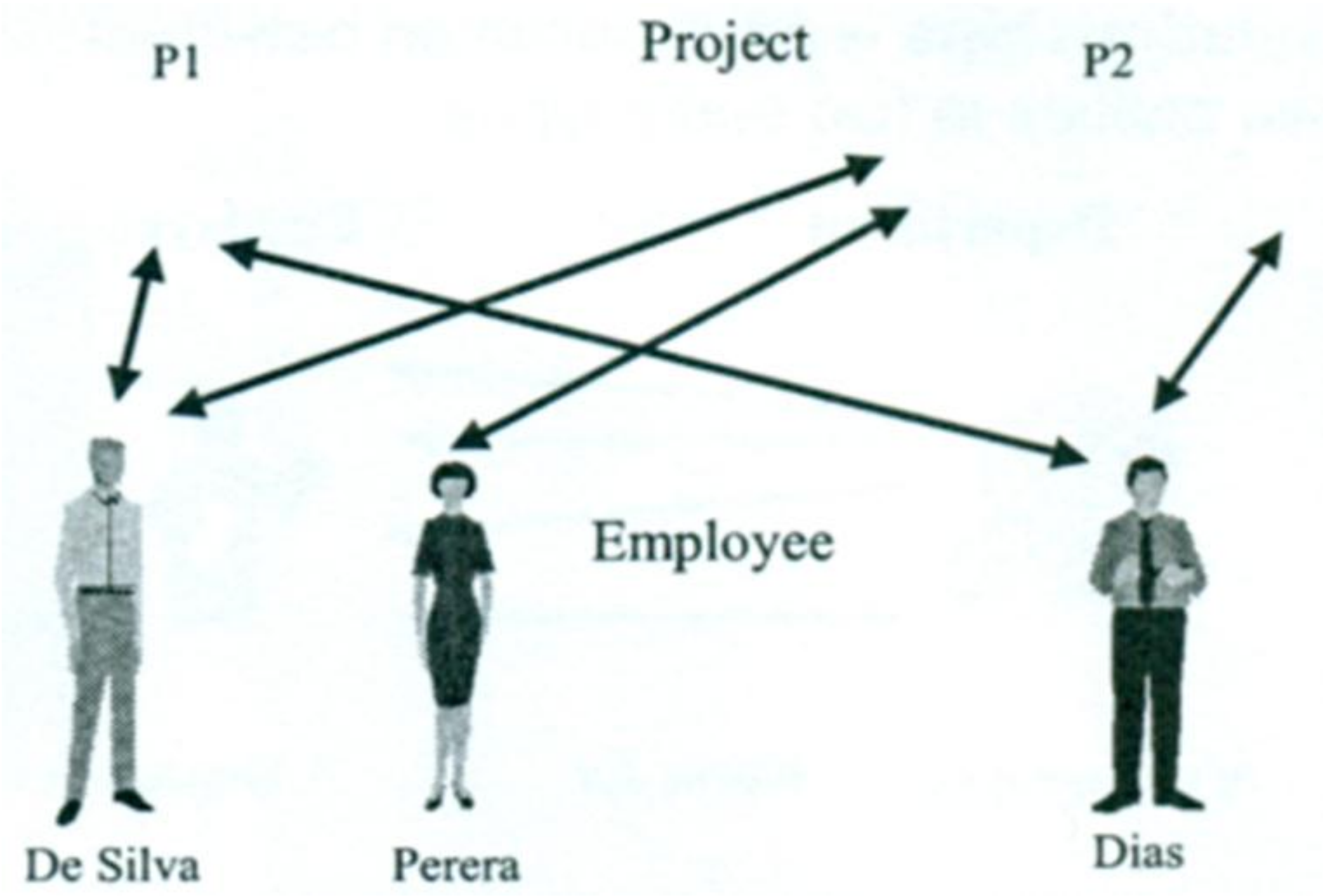


one - many

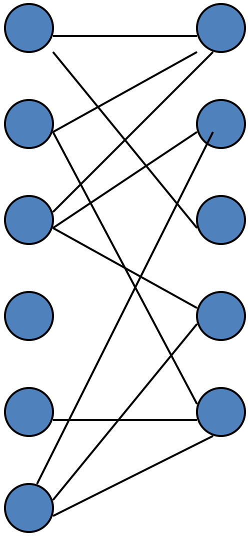
Many-Many Relationships

- In a *many-many* relationship, an entity of either set can be connected to many entities of the other set.
  - E.g., a shop sells many beverages; a beverage is sold by many shops.

Many-Many Relationships

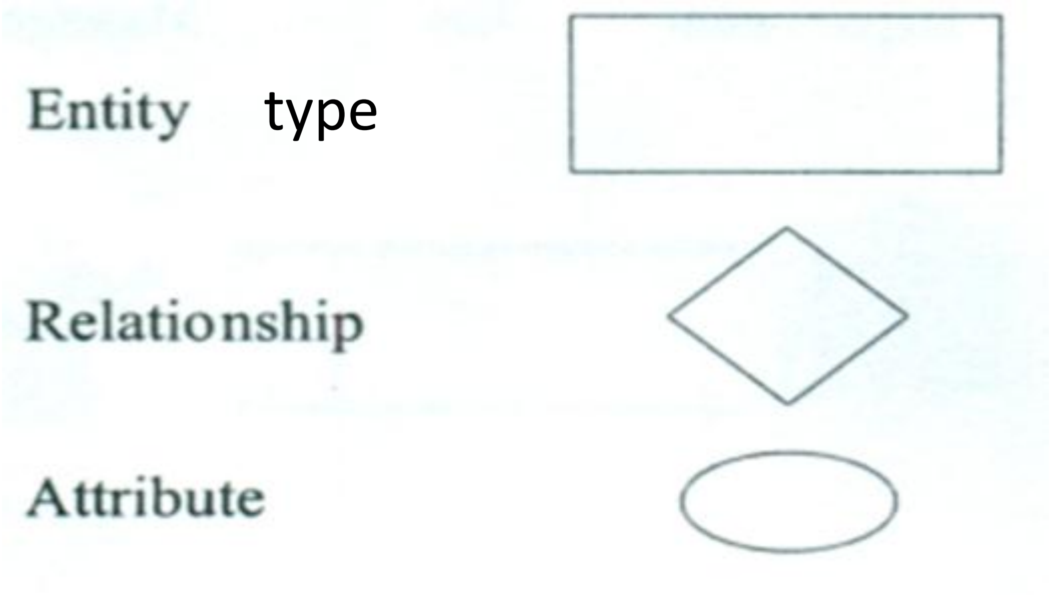


In Pictures:



many-many

Notations



Entity type

Relationship

Attribute

- In an entity-relationship diagram:
  - Entity set = rectangle.
  - Attribute = oval, with a line to the rectangle.
  - Relationship is a diamond.

Exercise

Supplier supply products.

Customer has a savings account.

University offers degrees.

Employees work on projects.

A teacher teaches courses.

Multiplicity/Cardinality

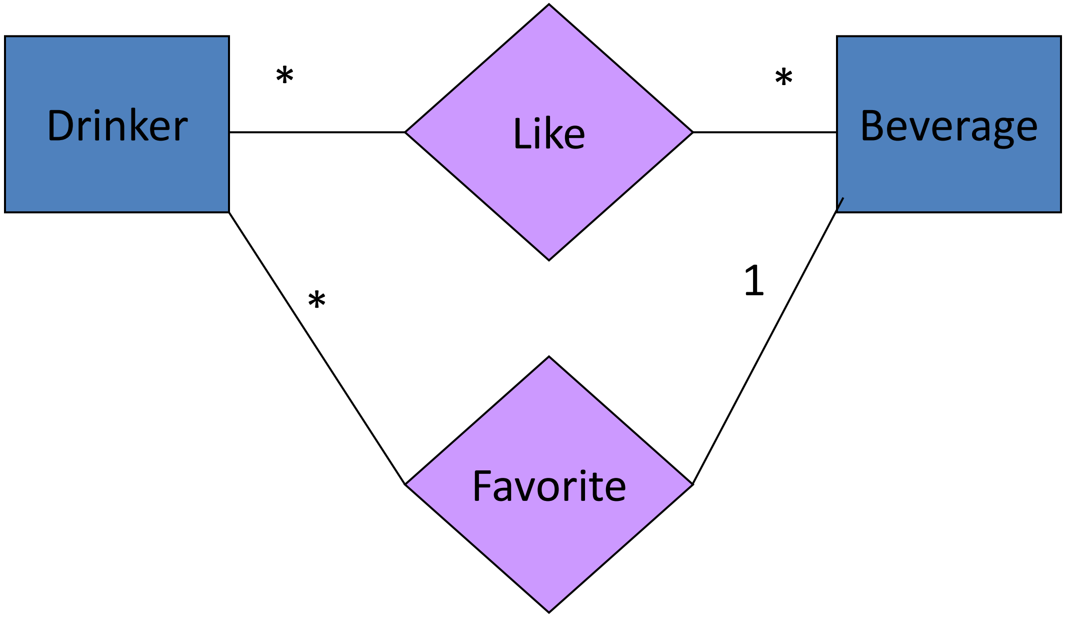
Maximum cardinality

The maximum cardinality of a relationship refers to the maximum number of instances in one entity set that are relating to a single instance in the other entity set.

Minimum cardinality

The minimum cardinality of a relationship refers to the minimum number of instances in one entity set that are relating to a single instance in the other entity set.

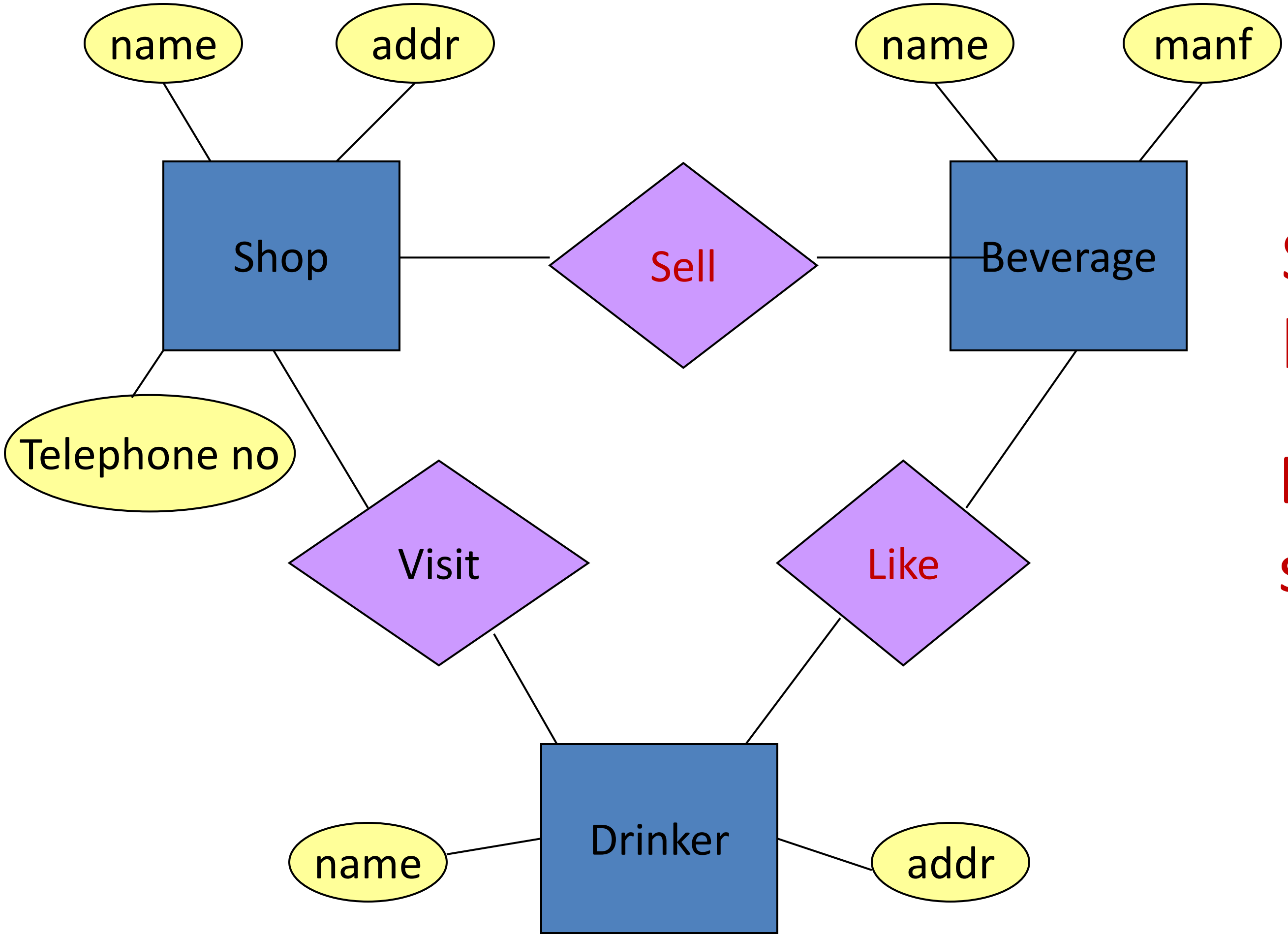
Example



- A drinker likes many beverages and beverage may like by many drinkers

- A drinker has one only favorite and beverage may favorite of many drinkers

Example

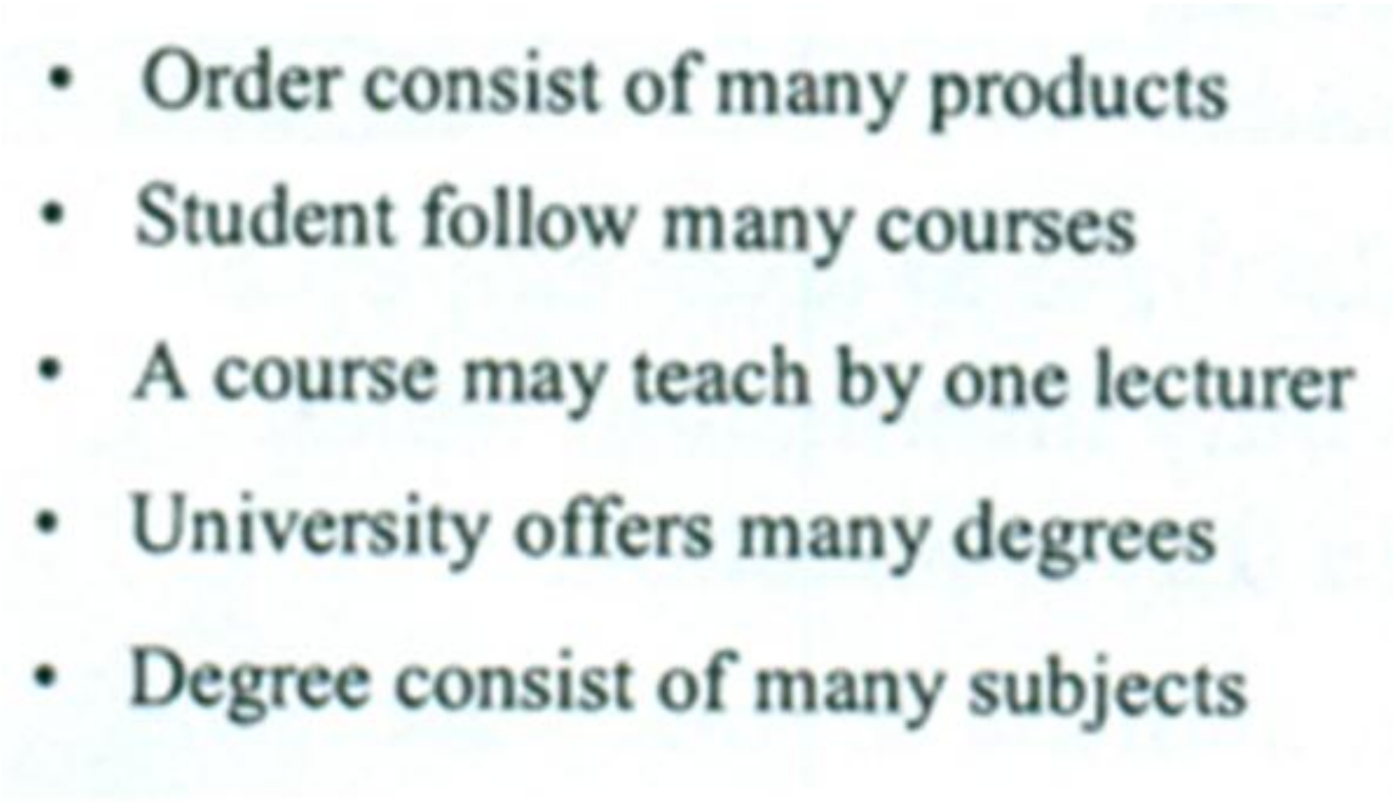


Shops sell some beverages

Drinkers like some beverages

Drinkers visit some shops

Exercise



- Order consist of many products
- Student follow many courses
- A course may teach by one lecturer
- University offers many degrees
- Degree consist of many subjects

Relationship Set

- The relationship set is a set of all related entities, one from each of the related entity sets.

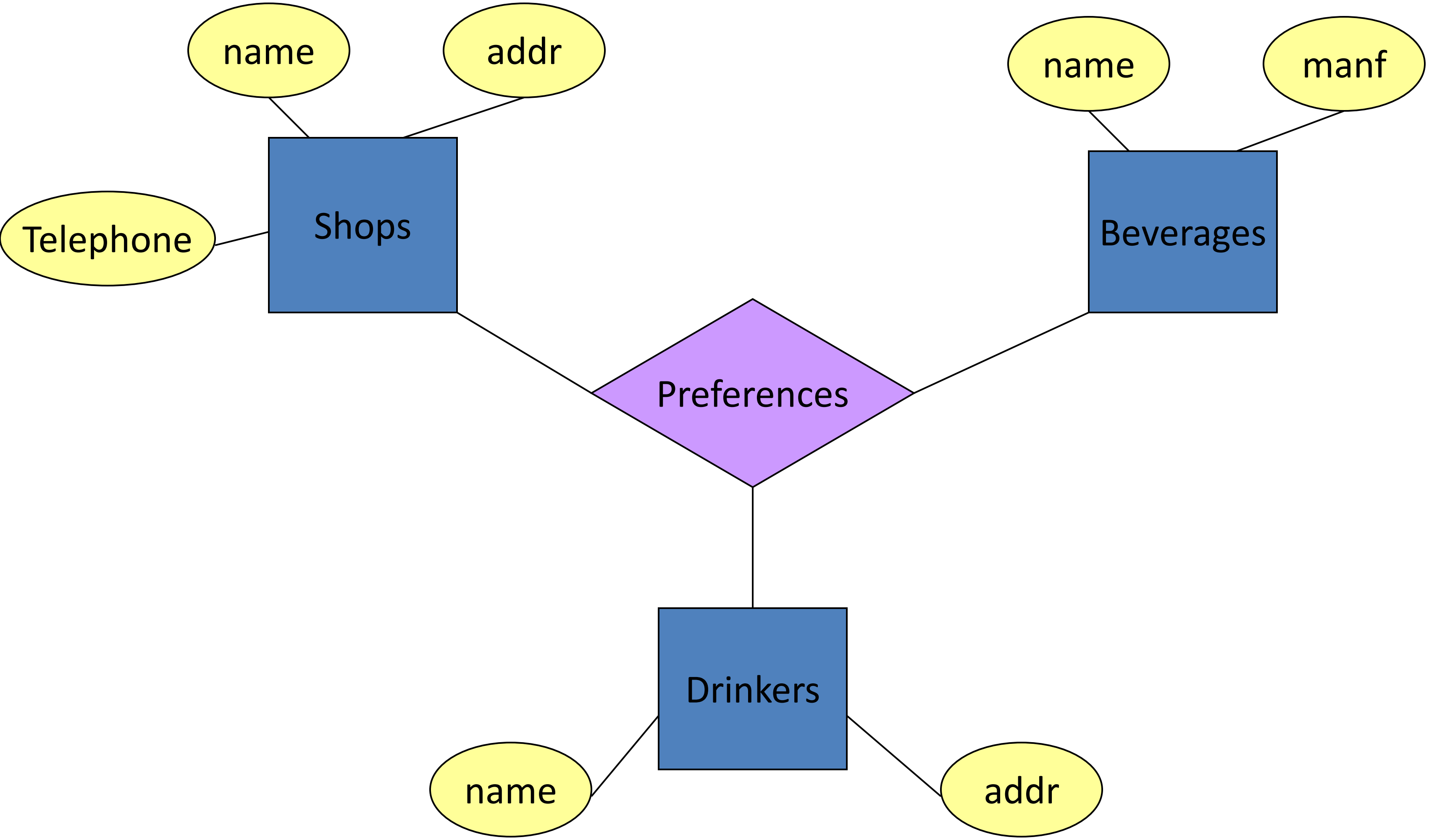
- For the relationship Sells, we might have a relationship set like:

Shop	Beverage
Joe’s shop	Coca cola
Joe’s shop	Ginger beer
Sue’s shop	Fanta
Sue’s shop	Coca cola

Multi-way Relationships

- Sometimes, we need a relationship that connects more than two entity sets.
- Suppose that drinkers will only drink certain beverages at certain shops.
  - Our three binary relationships Likes, Sells, and Visits do not allow us to make this distinction.
  - But a 3-way relationship would.

Example

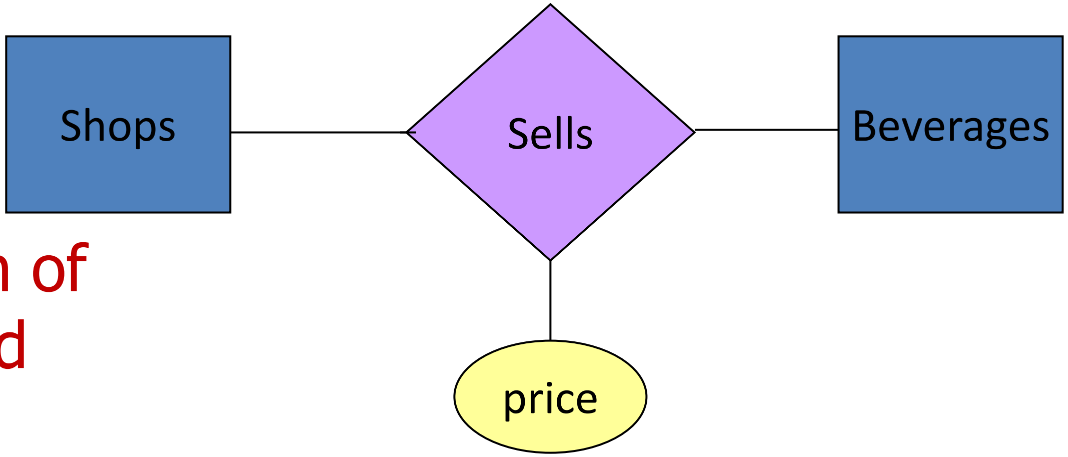

name
addr
Telephone
Shops
name
manf
Beverages
Preferences
Drinkers
name
addr

A Typical Relationship Set

Shop	**Drinker**	**Beverage**
Joe’s shop	Ann	Coca cola
Sue’s shop	Ann	Fanta
Sue’s shop	Ann	Ginger beer
Joe’s shop	Bob	Coca cola
Joe’s shop	Bob	Ginger beer
Joe’s shop	Cal	Fanta
Sue’s shop	Cal	Coca cola

Attributes on Relationships

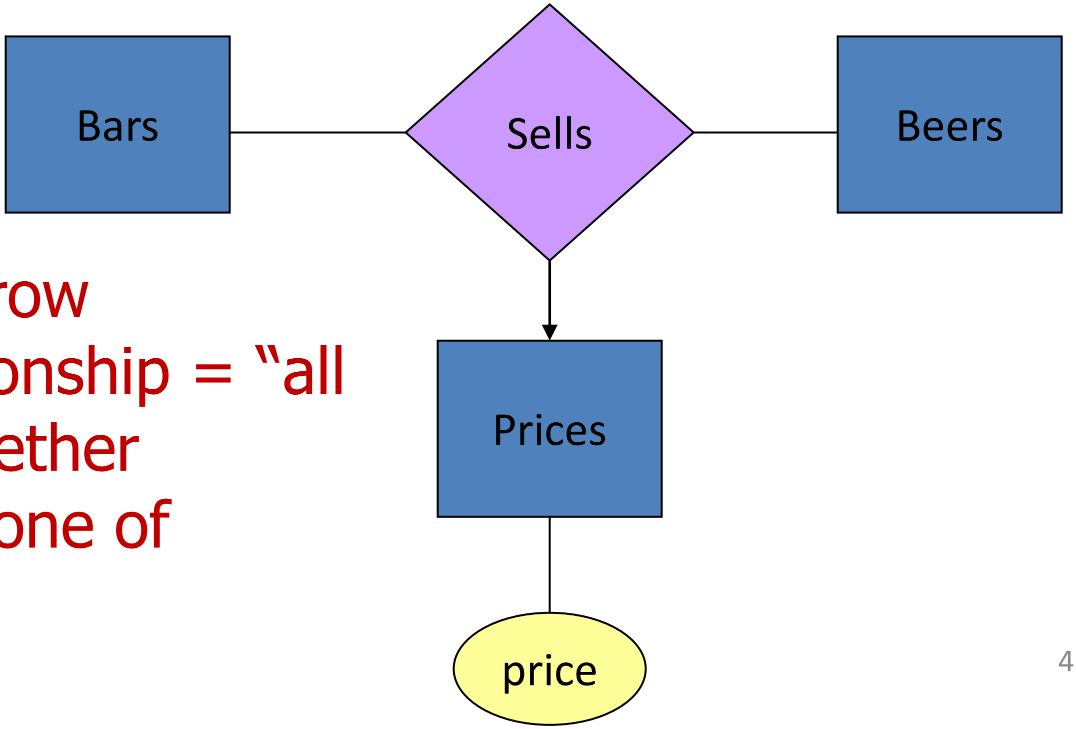
- Sometimes it is useful to attach an attribute to a relationship.
- Think of this attribute as a property of tuples in the relationship set.



Price is a function of both the shop and the beverages not of one alone.

Equivalent diagrams without attributes on Relationships

- Create an entity set representing values of the attribute.
- Make that entity set participate in the relationship.



Note convention: Arrow from multiway relationship = "all other entity sets together determine a unique one of these."

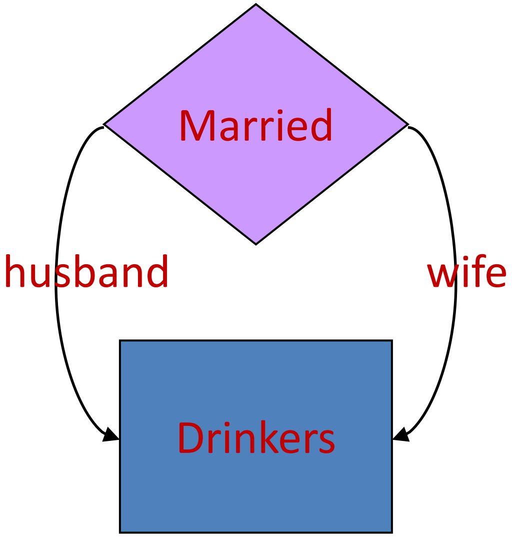
Roles

- Sometimes an entity set appears more than once in a relationship.
- Label the edges between the relationship and the entity set with names called *roles*.

Example

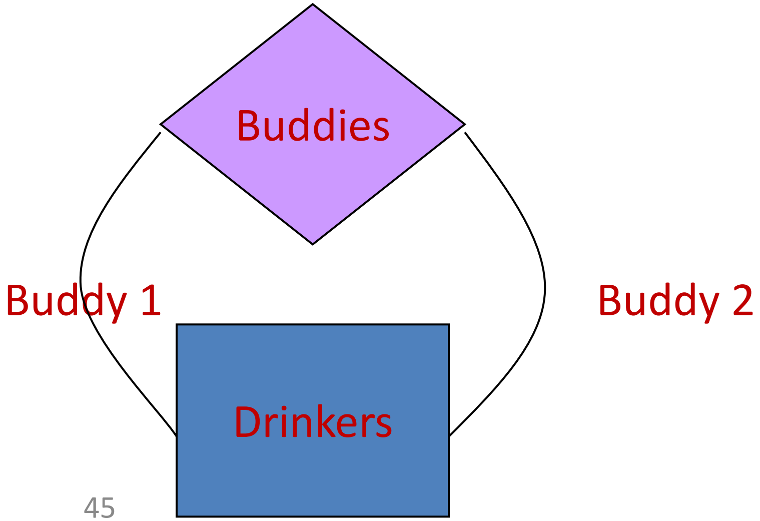
Relationship Set

Husband	Wife
Bob	Ann
Joe	Sue
…	…



Relationship Set

Buddy1	Buddy2
Bob	Ann
Joe	Sue
Ann	Bob
Joe	Moe
…	…



Keys

- A *key* for an entity set *E* is a set *K* of one or more attributes, such that given any two distinct entities *e1* and *e2* in *E*, *e1* and *e2* cannot have identical values for each of the attributes in the key *K*.

  - It is allowed for two entities to agree on some, but not all, of the key attributes.
- We must designate a key for every entity set.
- Underline the key attribute(s) in the ER diagram.

Keys

Employee

Emp_No	**Emp_Name**	**Department**
170	Silva	7
850	Perera	4
340	Dias	4
100	Silva	6

Primary key

Salary

Emp_No	**Eff-Date**	**Amt**
170	1/1/98	8000
850	3/7/99	9000
170	1/6/97	7000
100	1/6/97	7500

Primary key

Exercise: Find the primary key

Employee

Emp_No	Project_No	Project_Location
170	1	Kandy
850	2	Galle
170	2	Colombo
100	3	Kandy

Employee_Project

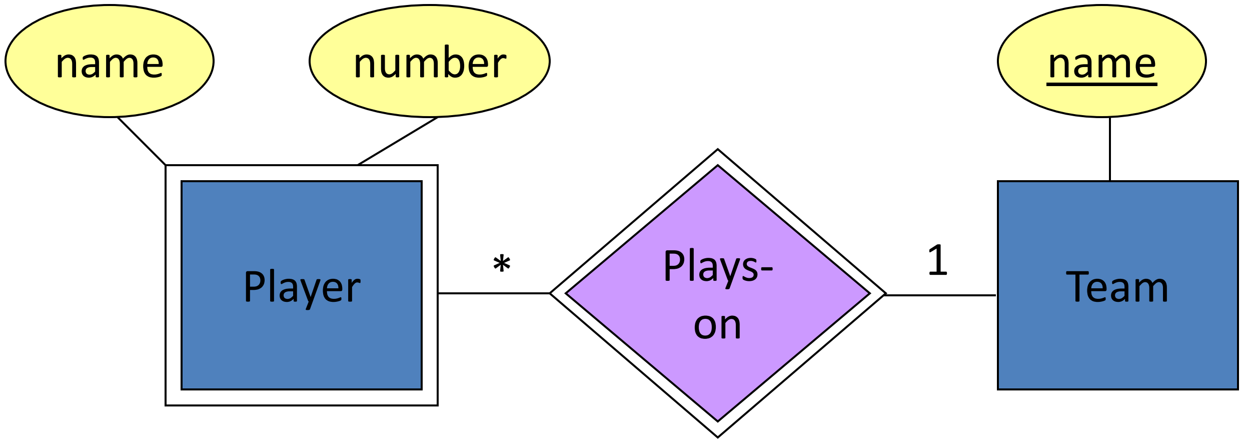
Emp_No	Project_No	Project_District	Project_Town	Hours
170	1	Kandy	Peradeniya	70
850	2	Galle	Ahangama	60
170	2	Colombo	Ratmalana	50
100	3	Kandy	Yatinuwara	50
170	1	Kandy	Katugasthota	80

Weak Entity Sets

- An entity that does not have a key attribute

- Occasionally, entities of an entity set need “help” to identify them uniquely.
- Entity set *W* is said to be ***weak,*** if in order to identify entities of *W* uniquely, we need to follow one or more ***one-many relationships*** from *W* and include the ***key*** of the related entitie(s) (E) from the connected entity set(s).

Weak Entity Sets



- Double diamond for *supporting* one-many relationship.
- Double rectangle for the weak entity set.

Example 1

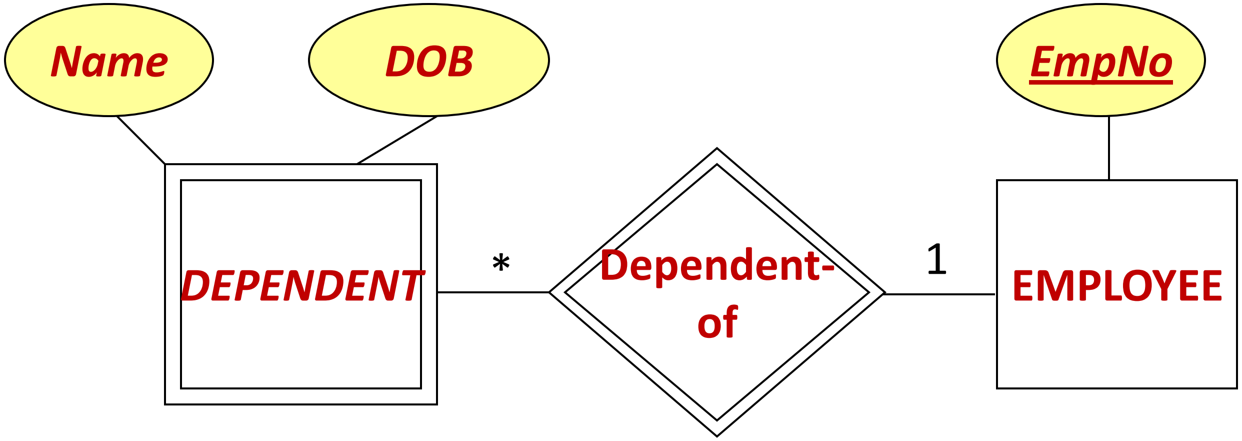
- name is almost a key for football players, but there might be two with the same name.
- number is certainly not a key, since players on two teams could have the same number.
- But number, together with the team name related to the player by Plays-on relationship should be unique.

Example 2

- Suppose that a DEPENDENT of an employee has the following relation schema,

  DEPENDENT(Name, DOB, Sex, Relationship)

  - ***Name*** cannot use as the primary key.
- An employee can have one or more dependent.



- DEPENDENT is a weak entity type with EMPLOYEE as its identifying entity type via the identifying relationship type DEPENDENT_OF

Rules for Weak Entity Sets

- A weak entity set has one or more one-many relationships to other (supporting) entity sets.
  - Not every one-many relationship from a weak entity set need be supporting.
- The *key* for a weak entity set is its own underlined attributes and the *keys* for the supporting entity sets.
  - E.g., (player) number and (team) name is a key for Players in the previous example.

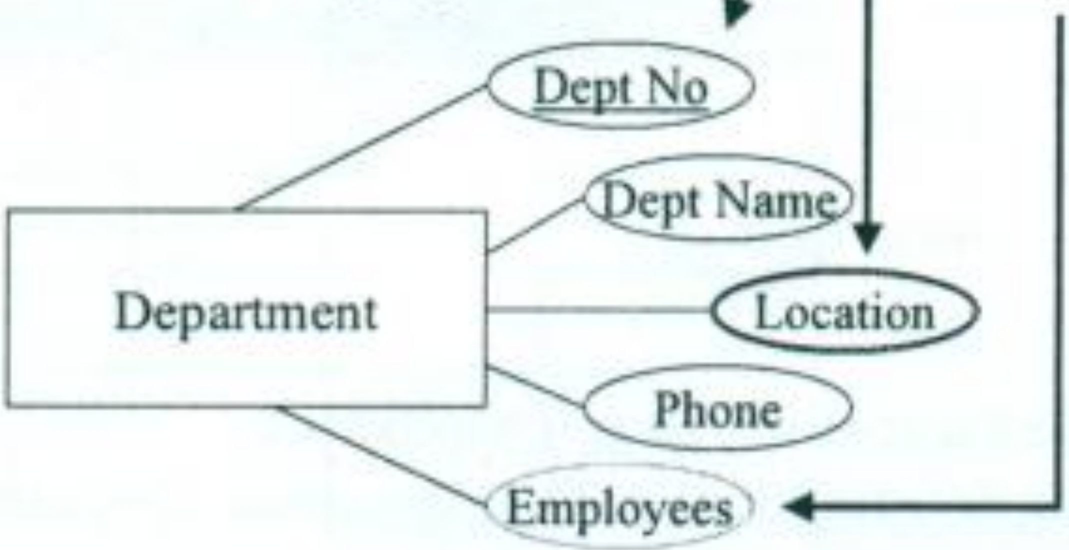
Types of Attributes

- ***Simple***: Each entity has a single atomic value for the attribute; for example SSN or Sex
- ***Composite***: The attribute may be composed of several components; for example, Address (Apt#, House#, Street, City, State, ZipCode, Country) or Name(FirstName, MiddleName, LastName). Composition may form a hierarchy where some components are themselves composite.
- ***Multi-valued***: An entity may have multiple values for that attribute; for example, Color of a CAR or PreviousDegrees of a STUDENT. Denoted as {Color} or {PreviousDegrees}.

- In general, composite and multi-valued attributes may be nested arbitrarily to any number of levels although this is rare. For example, PreviousDegrees of a STUDENT is a composite multi-valued attribute denoted by {PreviousDegrees(College, Year, Degree, Field)}.

Detailed Conceptual Design

Dept No	unique identifier of a dept.	Identifier
Dept Name	name of a department	Unique
Location	location of a department	Multi-valued
Phone	phone no. of a department	
Employees	no. of employees in a dept.	Derived



Detailed Conceptual Design

Employee

Emp No	unique identifier of an emp.	Identifier
Emp Name	name of an employee	Composite
First Name	first name of an employee	
Mid Initials	middle initials of an employee	
Last Name	last name of an employee	
NID	national id of an employee	Unique
Address	address of an employee	
Salary	salary of an employee	
Gender	sex of an employee	
DOB	birth date of an employee	

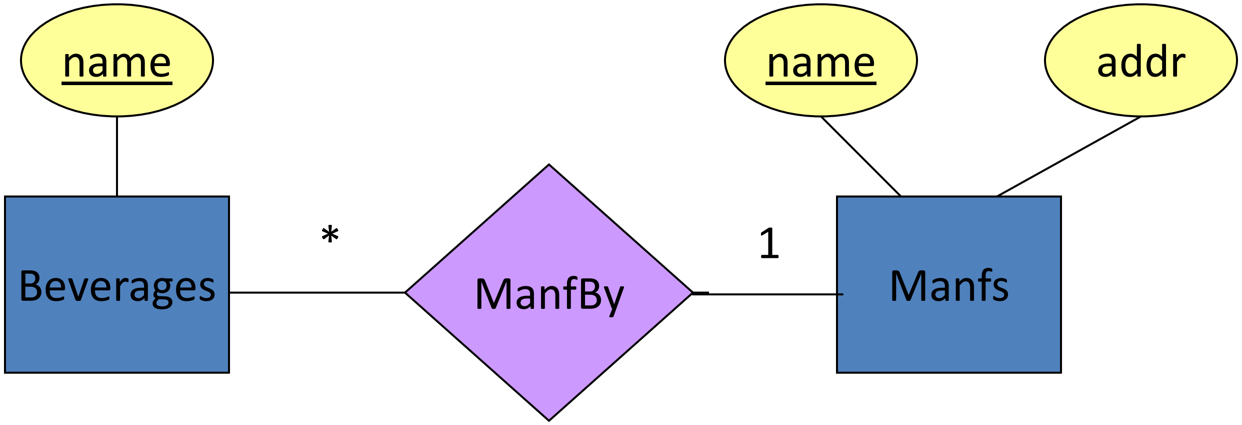
Design Techniques

1. Avoid redundancy.
2. Limit the use of weak entity sets.
3. Don’t use an entity set when an attribute will do.

Avoiding Redundancy

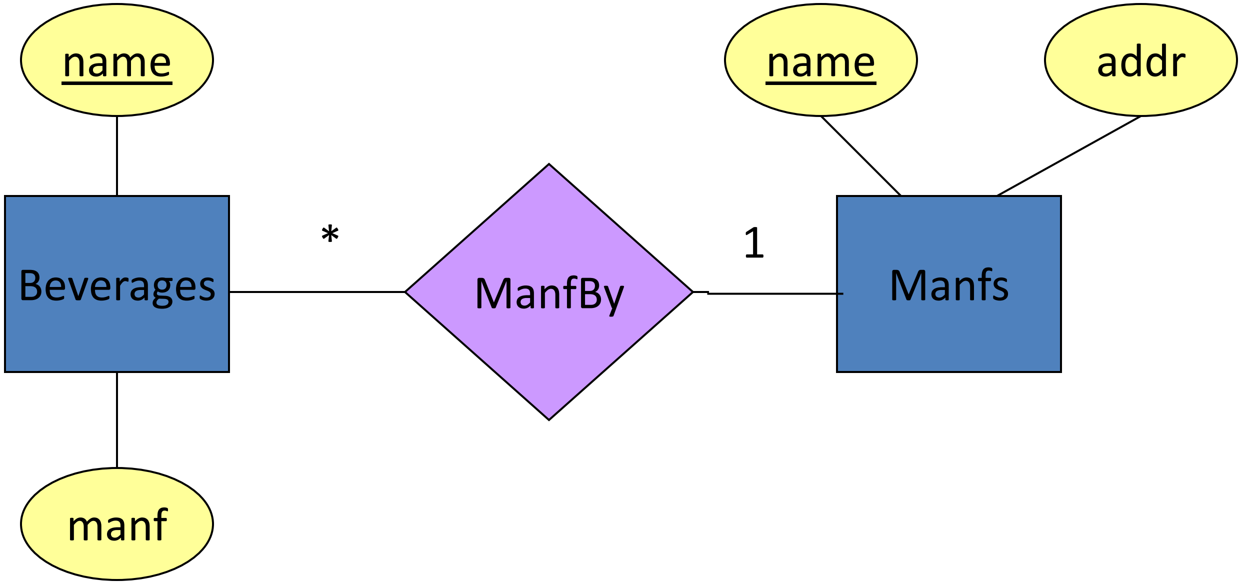
- *Redundancy* occurs when we say the same thing appear in two or more different places.
- Redundancy wastes space and (more importantly) encourages inconsistency.
  - The two instances of the same fact may become inconsistent if we change one and forget to change the other.

Example: Good



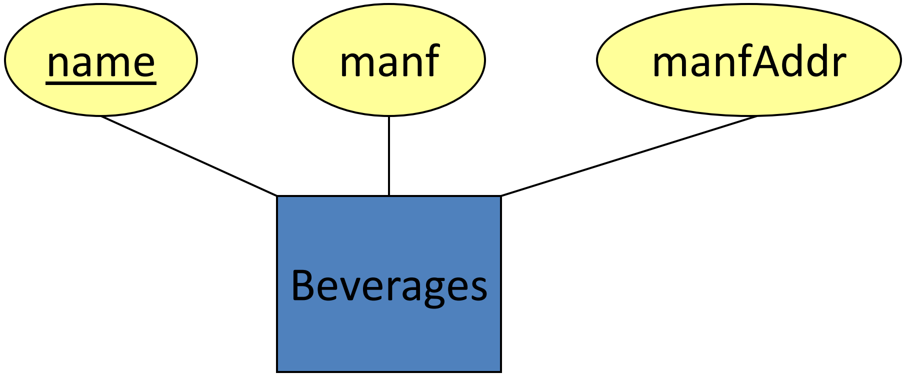
This design stores the address of each manufacturer exactly once.

Example: Bad



This design states the manufacturer of a beverage twice: as an attribute and as a related entity.

Example: Bad



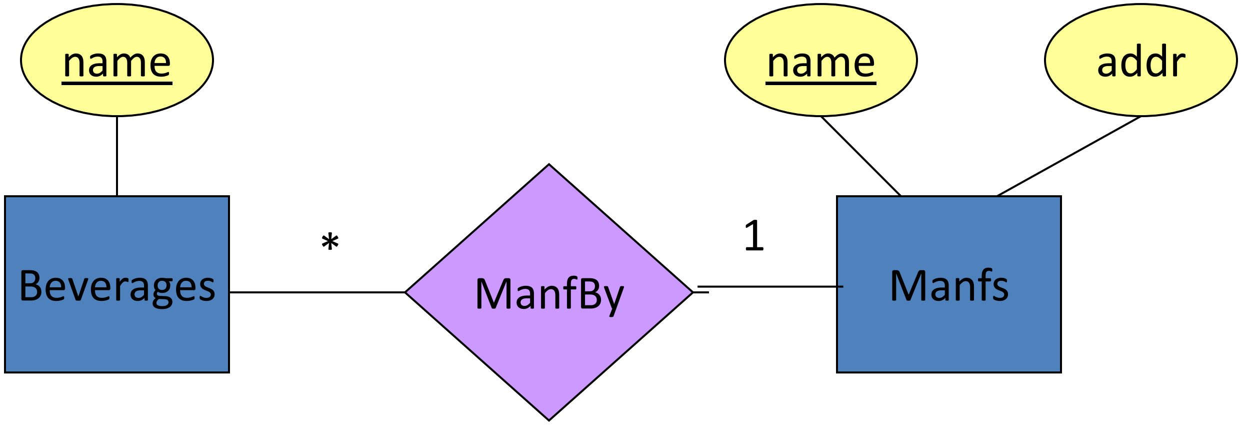
This design repeats the manufacturer’s address once for each beverage and loses the address if there are temporarily no beverages for a manufacturer.

Entity Sets Versus Attributes

- An entity set should satisfy at least one of the following conditions:
  - It is more than the name of something; it has at least one non-key attribute.

    or
  - It is the “many” in a one-many or many-many relationship.

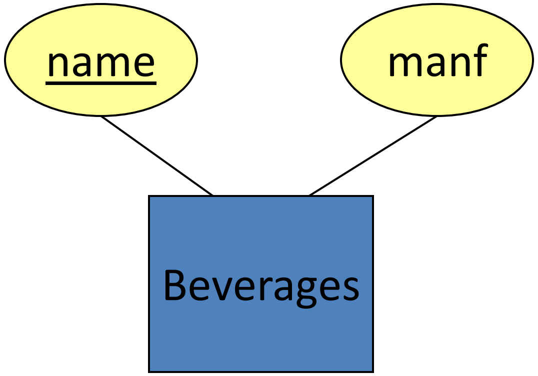
Example: Good



- Manfs deserves to be an entity set because of the non-key attribute addr.

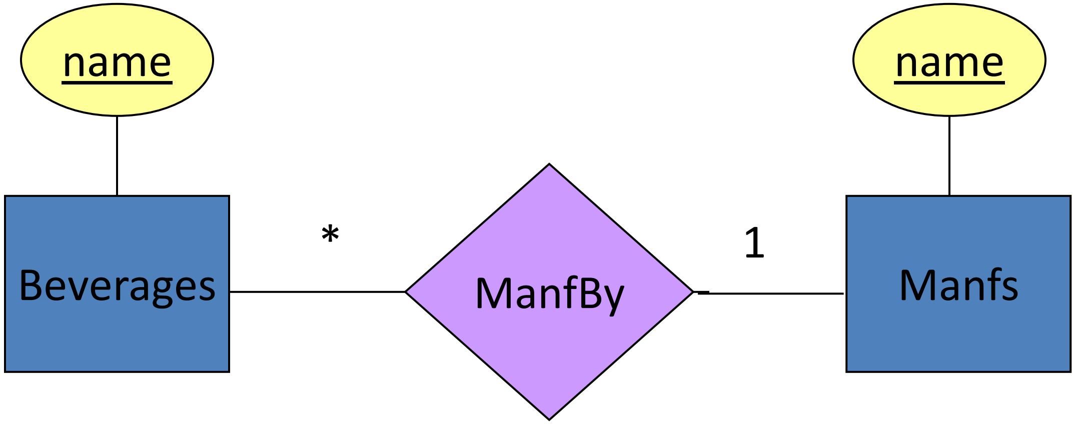
- Beverages deserves to be an entity set because it is the “many” of the one-many relationship ManfBy.

Example: Good



There is no need to make the manufacturer an entity set, because we record nothing about manufacturers besides their name.

Example: Bad



Since the manufacturer is nothing but a name, and is not at the “many” end of any relationship, it should not be an entity set.