Database fundamentals

Agenda

- Basics Definitions
- The database development process
- DBMS
- ERD Model
- Mapping Diagram

Basics Definitions

Database: A collection of related data.

Data: stored representations of meaningful objects and events.

- Structured: numbers, text, dates
- Unstructured: images, video, documents

Information: data processed to increase knowledge of the person using the data.

The Database Development Process

Data in context

Class Roster

Course: MGT 500 Business Policy

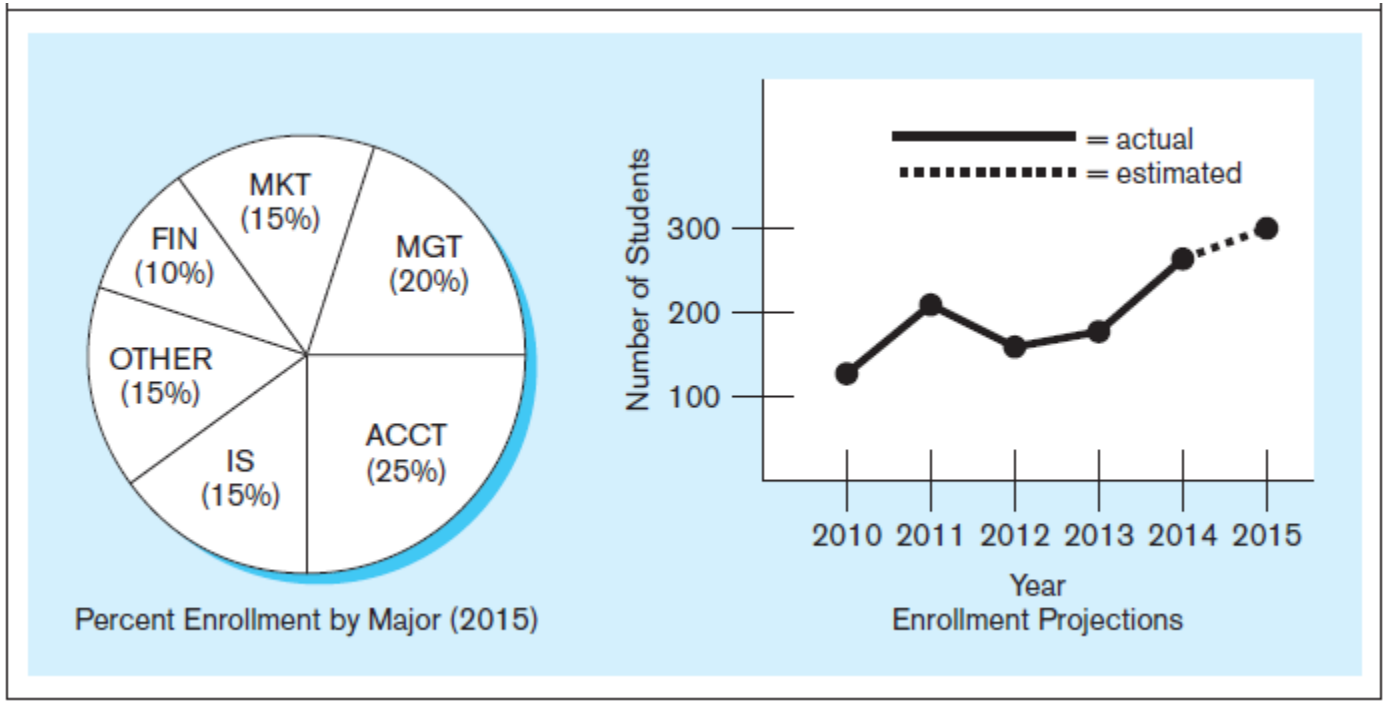
Semester: Spring 2015

Section: 2

Name	ID	Major	GPA
Baker, Kenneth D.	324917628	MGT	2.9
Doyle, Joan E.	476193248	MKT	3.4
Finkle, Clive R.	548429344	PRM	2.8
Lewis, John C.	551742186	MGT	3.7
McFerran, Debra R.	409723145	IS	2.9
Sisneros, Michael	392416582	ACCT	3.3

Context helps users understand data

Summarized data



Graphical displays turn data into useful information that managers can use for decision making and interpretation

Descripted data

TABLE 1-1 Example Metadata for Class Roster

Data Item	Metadata					
Name	**Type**	**Length**	**Min**	**Max**	**Description**	**Source**
Course	Alphanumeric	30			Course ID and name	Academic Unit
Section	Integer	1	1	9	Section number	Registrar
Semester	Alphanumeric	10			Semester and year	Registrar
Name	Alphanumeric	30			Student name	Student IS
ID	Integer	9			Student ID (SSN)	Student IS
Major	Alphanumeric	4			Student major	Student IS
GPA	Decimal	3	0.0	4.0	Student grade point average	Academic Unit

Descriptions of the properties or characteristics of the data, including data types, field sizes, allowable values, and data context

DISADVANTAGES OF FILE PROCESSING

- **Program-Data Dependence**

  All programs maintain metadata for each file they use.

- **Duplication of Data**

  Different systems/programs have separate copies of the same data.

- **Limited Data Sharing**

  No centralized control of data.

- **Lengthy Development Times**

  Programmers must design their own file formats.

PROBLEMS WITH DATA DEPENDENCY

- Each application programmer must maintain his/her own data.
- Each application program needs to include code for the metadata of each file.
- Each application program must have its own processing routines for reading, inserting, updating, and deleting data.
- Lack of coordination and central control.
- Non-standard file formats.

Duplicate Data

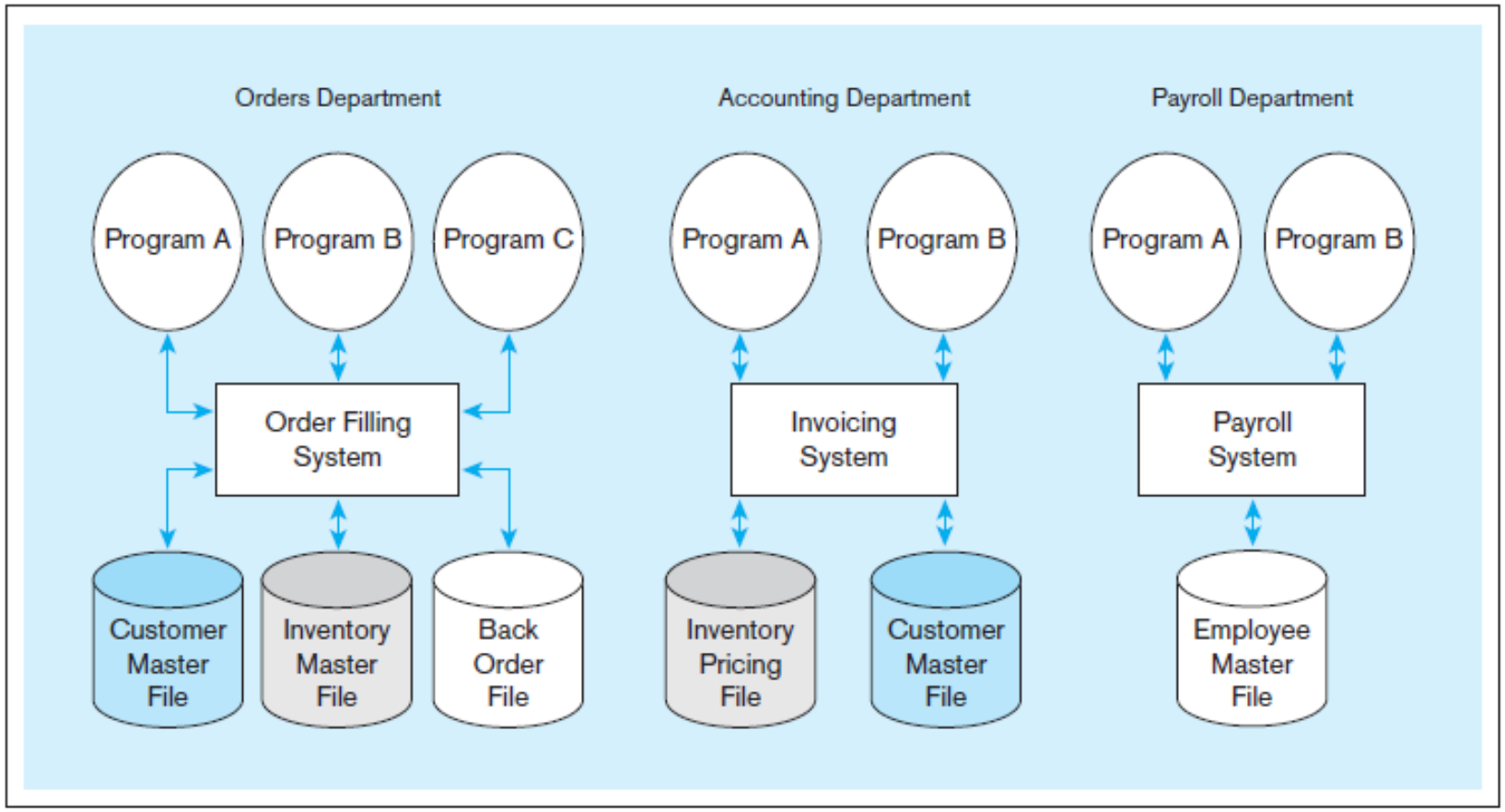


FIGURE 1-2 **Old file processing systems at Pine Valley Furniture Company**

SOLUTION: THE DATABASE APPROACH

- Central repository of shared data.
- Data is managed by a controlling agent.
- Stored in a standardized, convenient form.
- Requires a Database Management System (DBMS)

DATABASE MANAGEMENT SYSTEM

- **A software system that is used to create, maintain, and provide controlled access to user databases**
- **DBMS** manages **data resources** like **an operating system manages hardware** resources

Basics Definitions

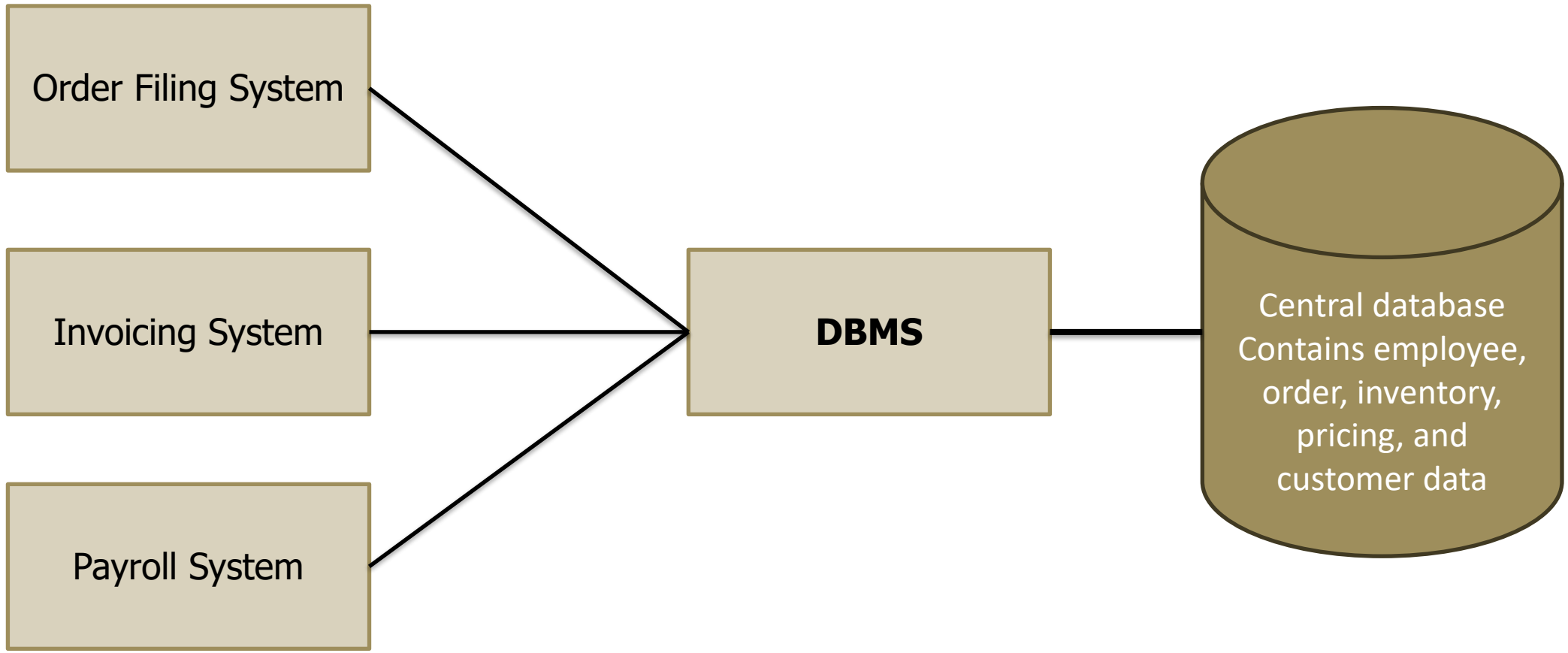
Database: A collection of related data.

Metadata: data that describes the properties and context of user data.

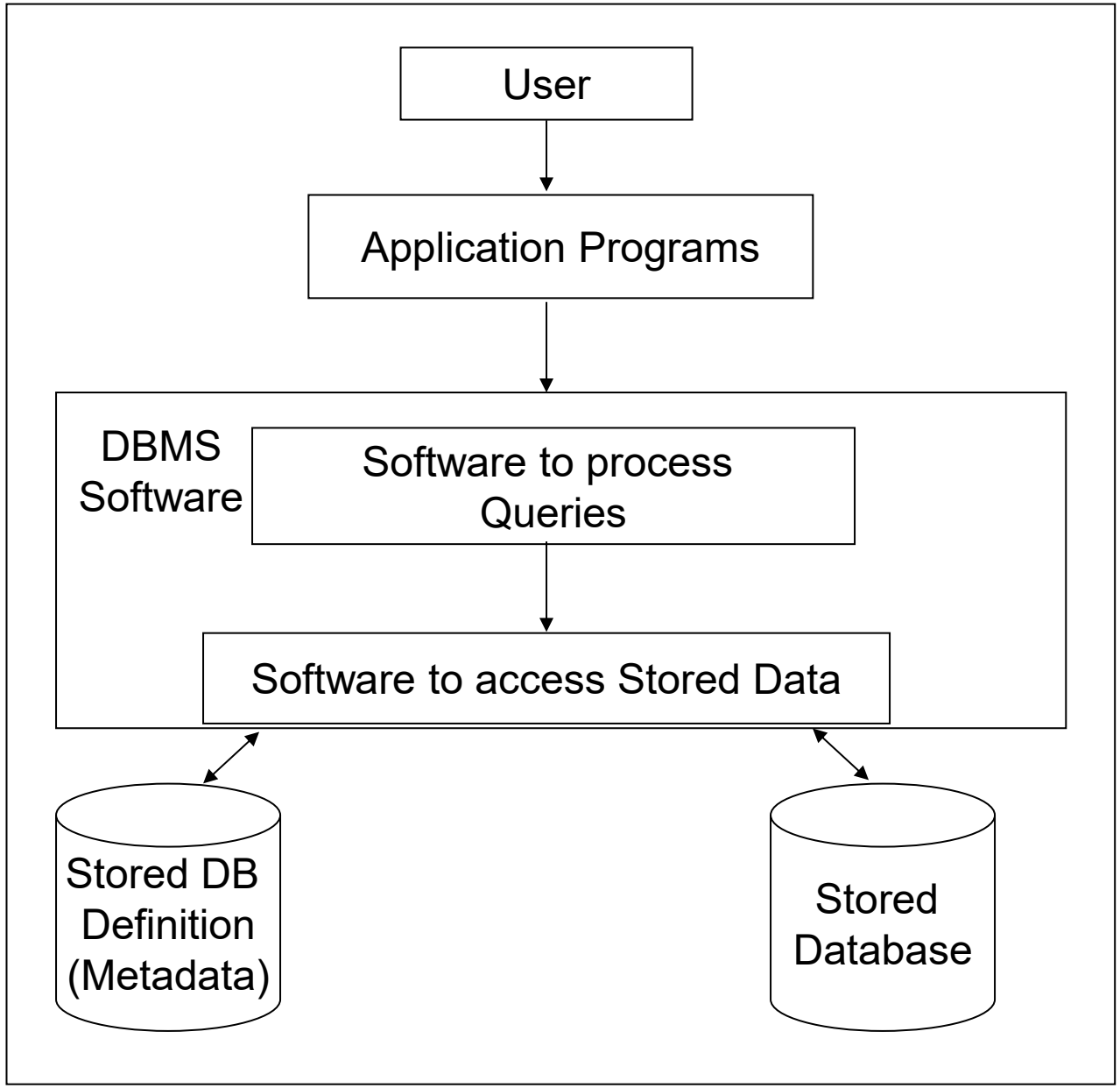
Database Management System (DBMS): A software package/ system to facilitate the creation and maintenance of a computerized database.

Database System: The DBMS software together with the data itself.
Sometimes, the applications are also included. (Software + Database)

DATABASE MANAGEMENT SYSTEM



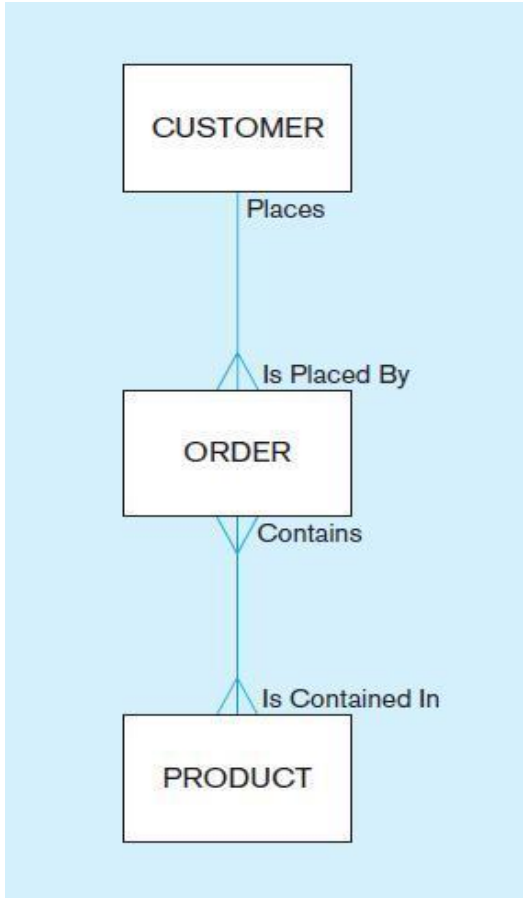
Database System



ELEMENTS OF THE DATABASE APPROACH

- **Data models:** Graphical system capturing nature and relationship of data
  1. **Enterprise Data Model:** high-level entities and relationships for the organization
  2. **Project Data Model:** more detailed view, matching data structure in database or data warehouse**.**
- **Entities:** Noun form describing a person, place, object, event, or concept, Composed of attributes.
- **Relationships:** Between entities.

Comparison of enterprise and project level data models

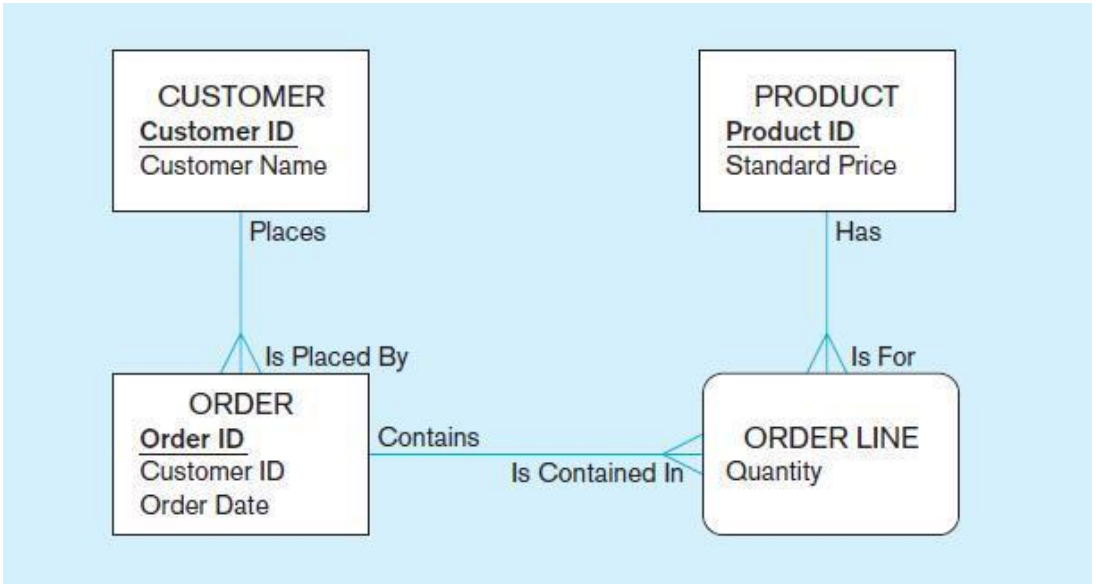


Segment of an **enterprise** data model

high-level entities and relationships

Segment of a **project-level** data model

detailed view, matching data structure



ERD Model

Entity Relationship Modeling

- **Entity-Relationship Diagram (ERD):** identifies information required by the business by displaying the relevant entities and the relationships between them.

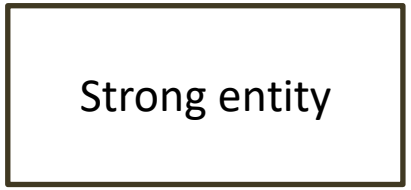
Entity Relationship Modeling (Cont'd)

In building a data model a number of questions must be addressed:

- What entities need to be described in the model?
- What characteristics or attributes of those entities need to be recorded?
- Can an attribute or a set of attributes be identified that will uniquely identify one specific occurrence of an entity?
- What associations or relationships exist between entities?

Definitions

Entity - An entity is a *thing* in the real world with an independent existence. Physical existence (for example, a particular person, car) or conceptual existence (for instance, a job, or a university course). **Types of entities: Weak Entity and Regular Entity(strong entity)**



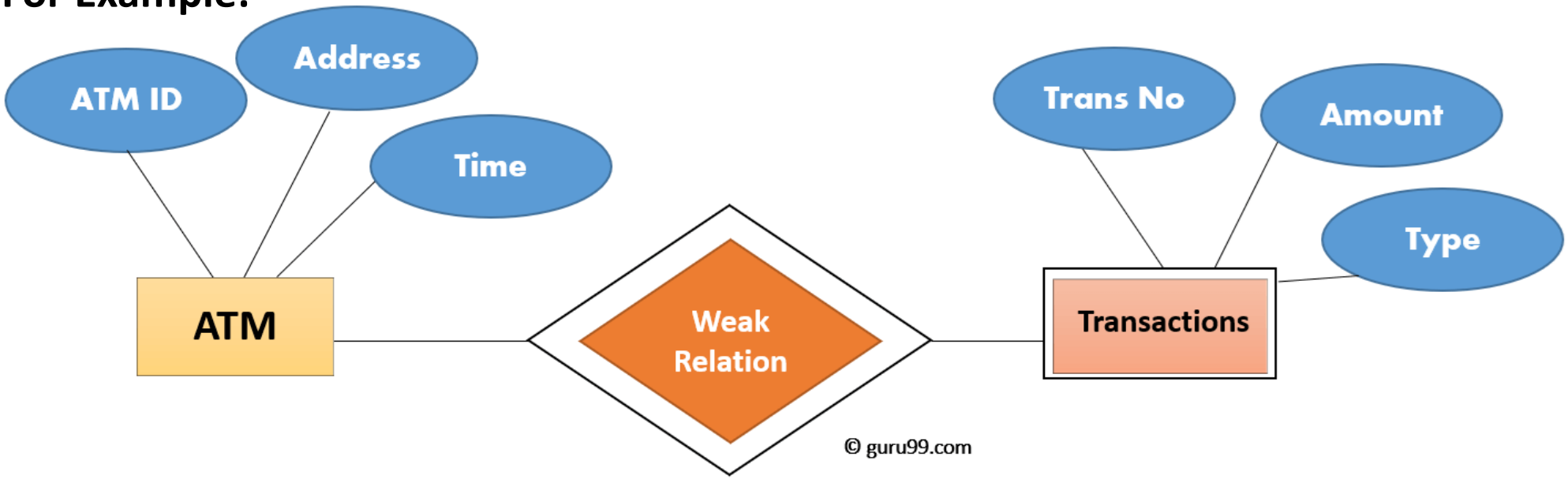
Entity Instance - An instance is a particular occurrence of an entity. For example, each person is an instance of an entity, each car is an instance of an entity, etc.

Attribute - The particular properties that describe the entity. An EMPLOYEE entity may be described by the employee's name, age, address and salary attributes.

Weak Entity

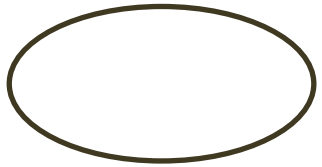
A weak entity is a type of entity which doesn't have its key attribute. It can be identified uniquely by considering the primary key of another entity. For that, weak entity sets need to have participation.

For Example:



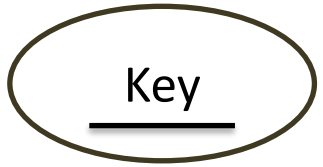
Attributes

- **An entity** has a set of attributes
- **Attribute** defines property of an entity

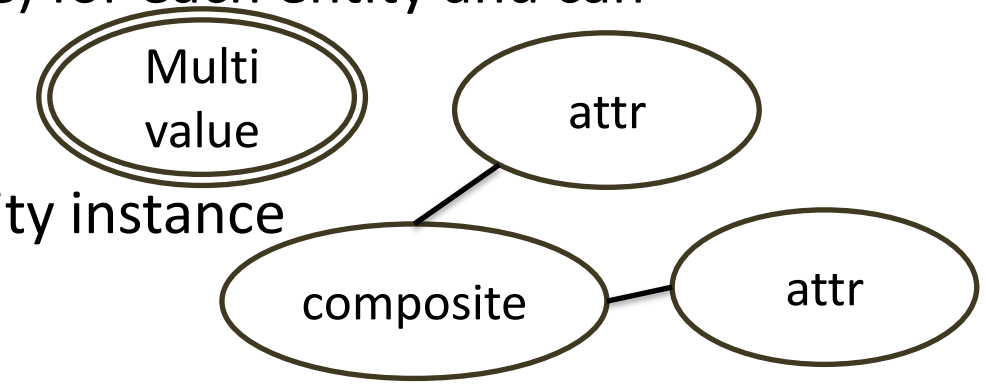


- **It is** given a name
- **Attribute** has value for each entity and value may change over time
- **Example : BOOK entity** has the following attributes (TITLE, ISBN, AUTHOR, PUBLISHER, YEAR, PRICE)
- **An attribute** may be **multi-valued,** i.e., it has more than one value for a given entity; e.g., a book may have many authors

Types of Attributes



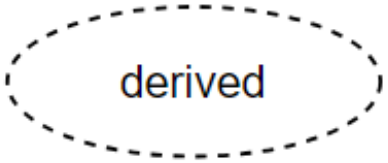
Key: an attribute whose values are distinct (unique) for each entity and can be used to uniquely identify the record



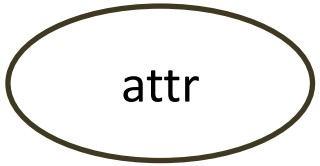
Multi-valued: has a set of values for the same entity instance

Composite: can be divided into smaller subparts

Derived: can be calculated from another attribute or entity



Single/Simple: Attributes that are not divisible and have a single value for a particular entity instance

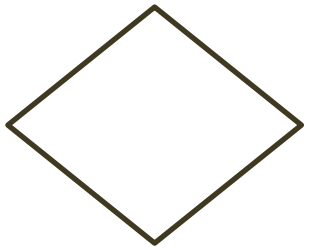


Key Attribute

- **Single Key**: For example, SSN of EMPLOYEE
- **Composite Key**: the *combination* of the attribute values that together form a key and must be distinct for each entity. For example, ID and Application_no
- **Candidate Key**: when an entity type has more than one key, those are candidate keys.
- **Foreign Key (referential attributes):** Attributes that define relationships between entities. A foreign key is a column or set of columns in a table(entity) that refers to the primary key of another table(entity). ; e.g., Department ID is the primary key of Department and Department ID is a foreign key of Employee defining the relationship "Employee works for Department.

Relationships

Relationships - A relationship is a connection between entity classes.



1. **Degree of a Relationship**: is the number of participating entity.

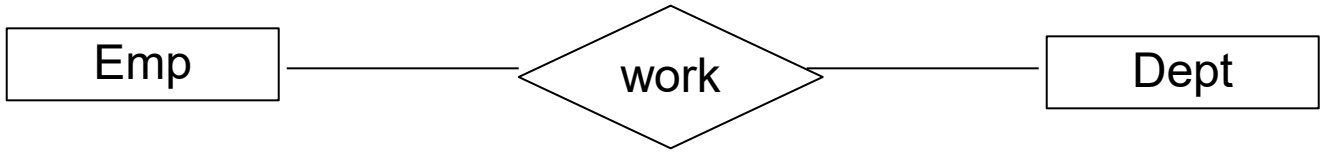
1. **Cardinality Ratio**: specifies the maximum number of relationship

The cardinality of a relationship indicates the number of instances in entity class E1 that can or must be associated with instances in entity class E2

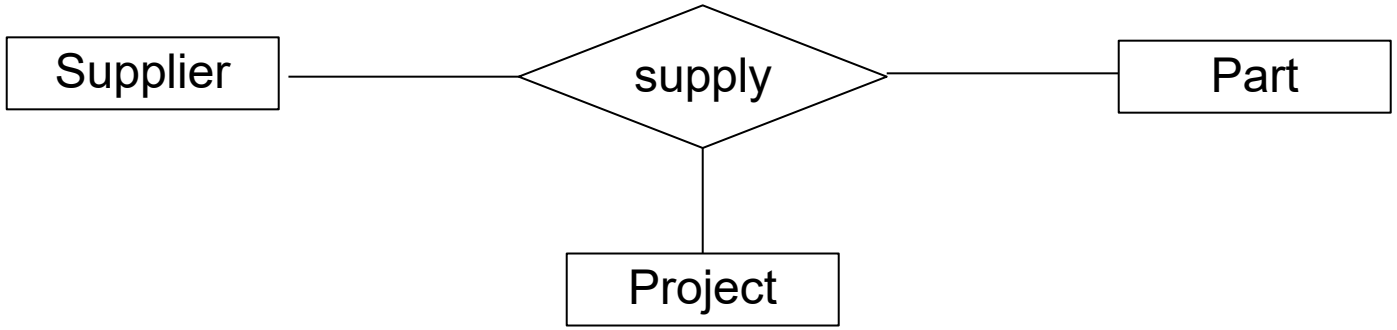
Degree of relationships

a. Unary/ Recursive

b. Binary



c. Ternary

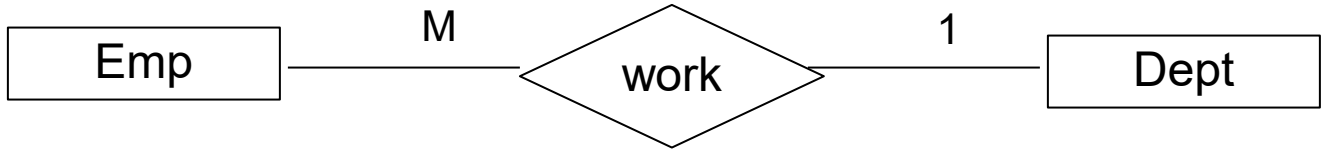


Relationships

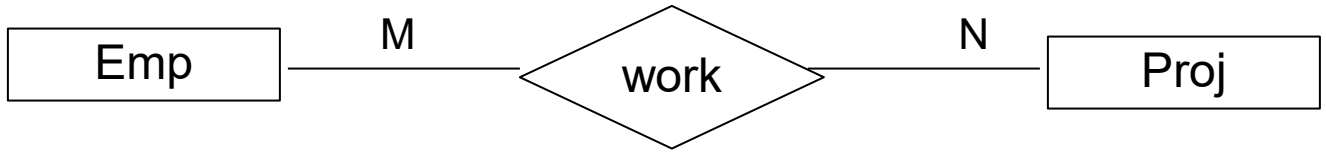
2. Cardinality Ratio

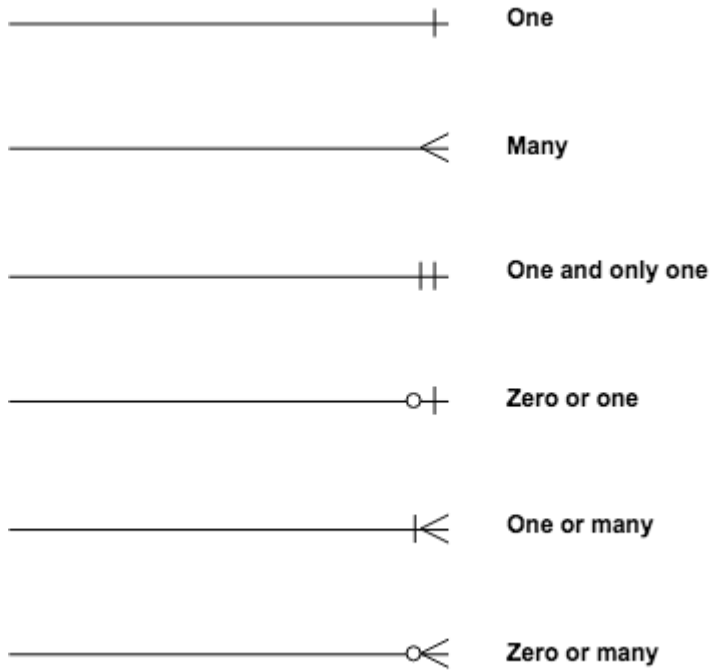
a. One to one

b. One to many



a. Many to many





example

- A company is organized into departments. Each department has a unique name, a unique number, and a particular employee who manages the department. A department may have several locations.

- A department may control a number of projects, each of which has a unique name, a unique number, and a single location. A project must be controlled by department

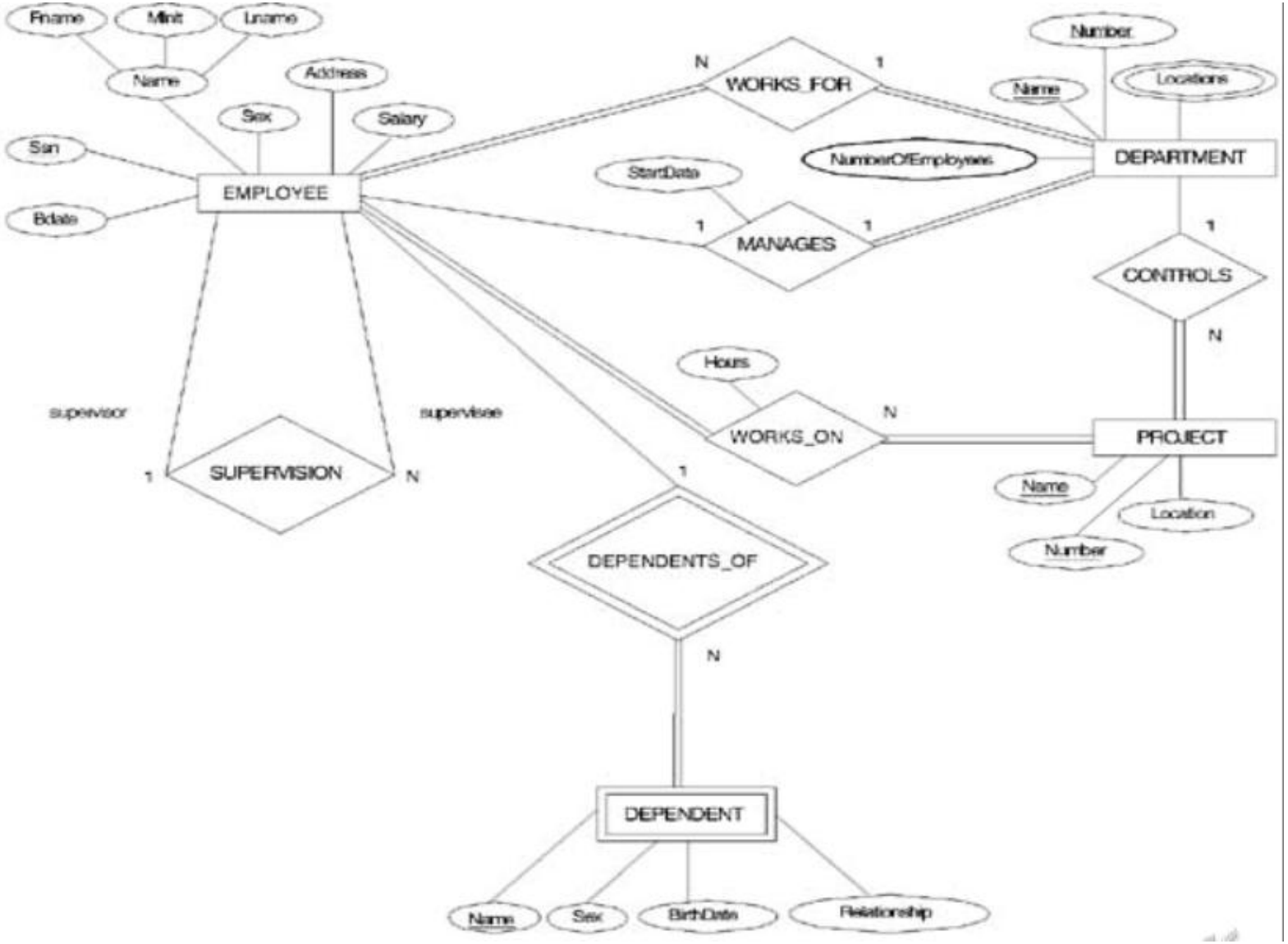
- We store employee's name, social security number, address, salary, gender and birth date. An employee must be assigned to one department and must work on one or more projects, which are not necessarily controlled by the same department. We keep track of the number of hours per week that an employee works on each project. We also keep track of the direct supervisor of each employee.

- We want to keep track of the dependents of each employee for insurance purposes. We keep each dependent's first name, gender, birth date and relationship to that employee.

How to Create ERD

Following are the steps to create an ER Diagram:

example


Fname
Minit
Lname
Name
Address
Sex
Salary
Ssn
Bdate
EMPLOYEE
N
WORKS_FOR
1
Number
Name
Locations
DEPARTMENT
NumberOfEmployees
StartDate
1
MANAGES
1
1
CONTROLS
N
supervisor
supervisee
1
SUPERVISION
N
Hours
N
WORKS_ON
PROJECT
1
Name
Location
Number
DEPENDENTS_OF
N
DEPENDENT
Name
Sex
BirthDate
Relationship

Mapping Diagram

Mapping Diagram

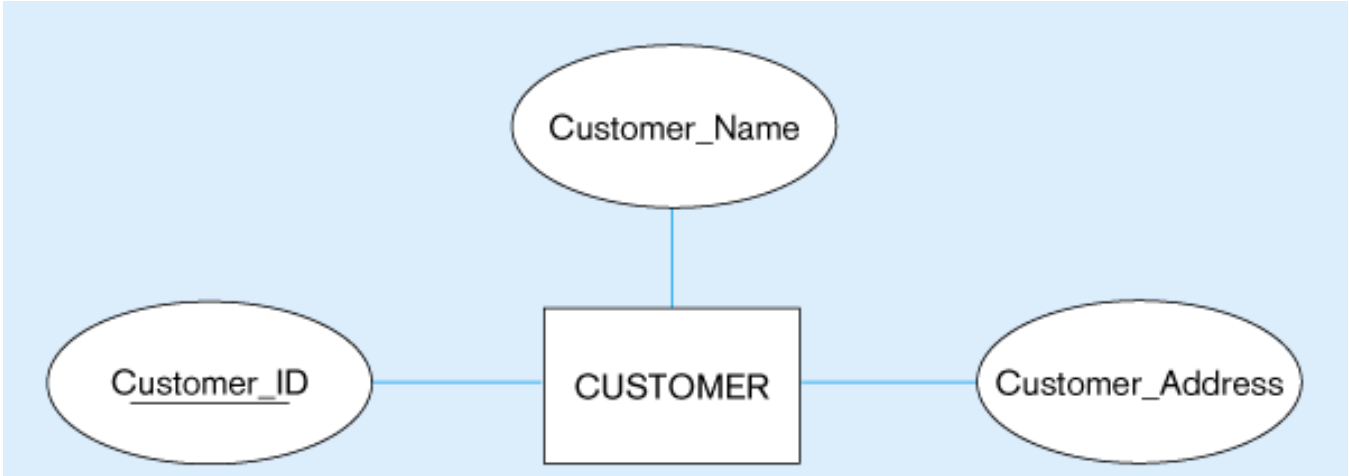
- we map the Entity Relationship Diagram (ERD) to a relational database and implement it as a **physical database** through following map rules and creating **mapping diagrams**.
- To complete the mapping from an Entity Relationship Diagram (ERD) to relations, **we must consider** the entity types, relationship types, and attributes that are specified for the model.

Mapping Rules

- Mapping Strong Entities to Relations:
  - **Simple attributes:** E-R attributes map directly onto the relation.
  - **Composite attributes:** Use only their simple, component attributes.
  - **Multi-valued Attribute:** Becomes a separate relation(table) with a foreign key taken from the superior entity.

Mapping a Strong entity

(a) CUSTOMER entity type with simple attributes



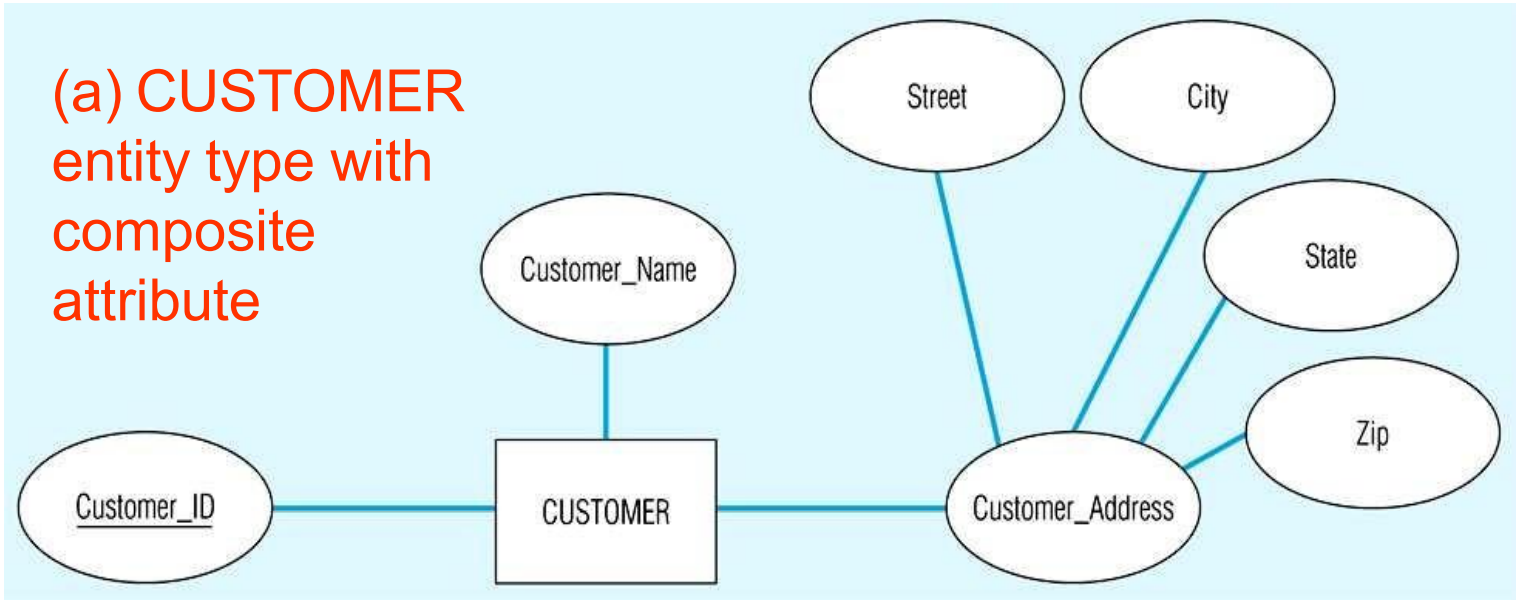
(b) CUSTOMER relation

CUSTOMER

Customer_ID	Customer_Name	Customer_Address

Mapping a composite attribute

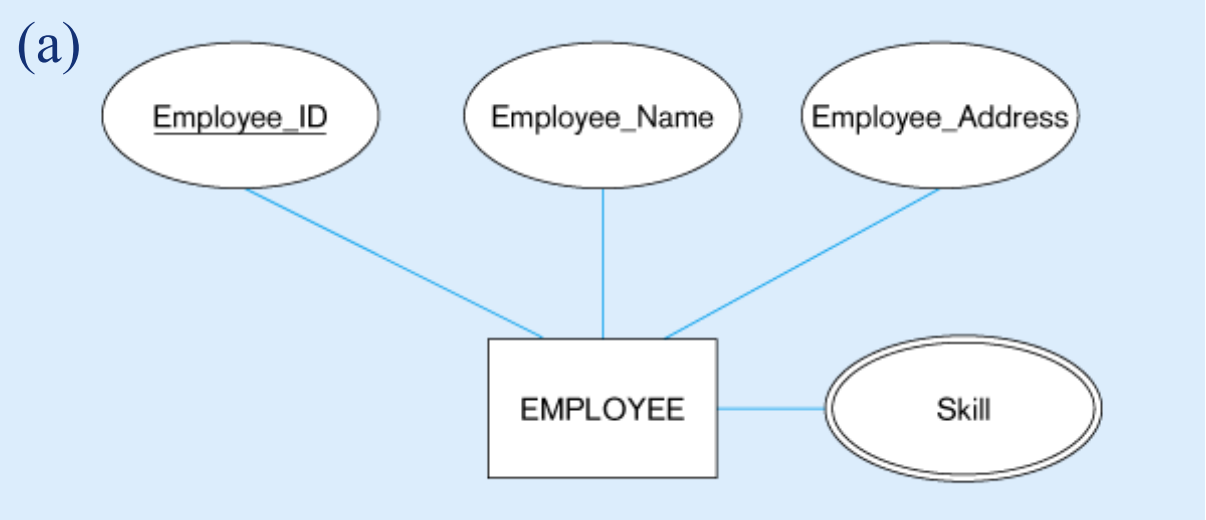
(a) CUSTOMER entity type with composite attribute



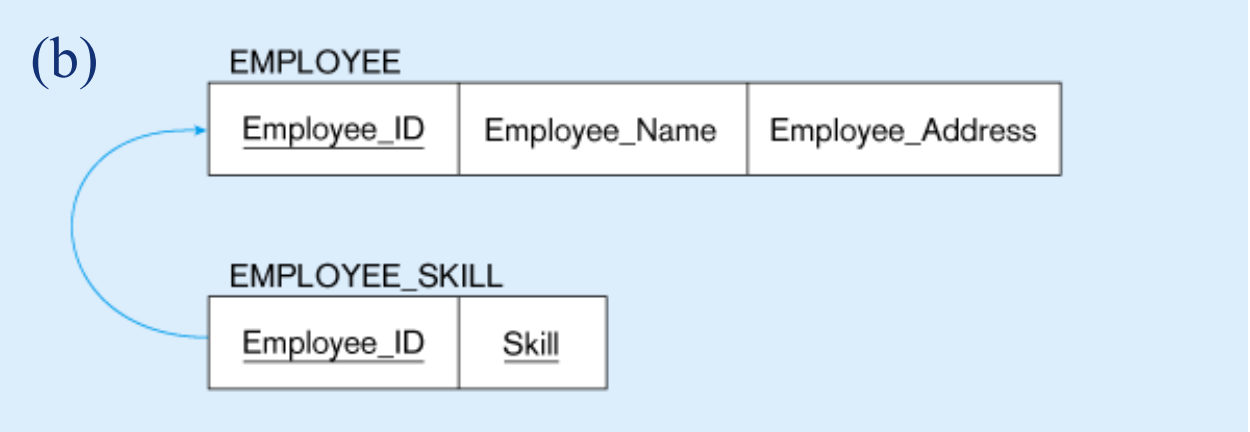
CUSTOMER (b) CUSTOMER relation with address detail

Customer_ID	Customer_Name	Street	City	State	Zip

Mapping a multivalued attribute



Multivalued attribute becomes a separate relation with foreign key



1 – to – many relationship between original entity and new relation

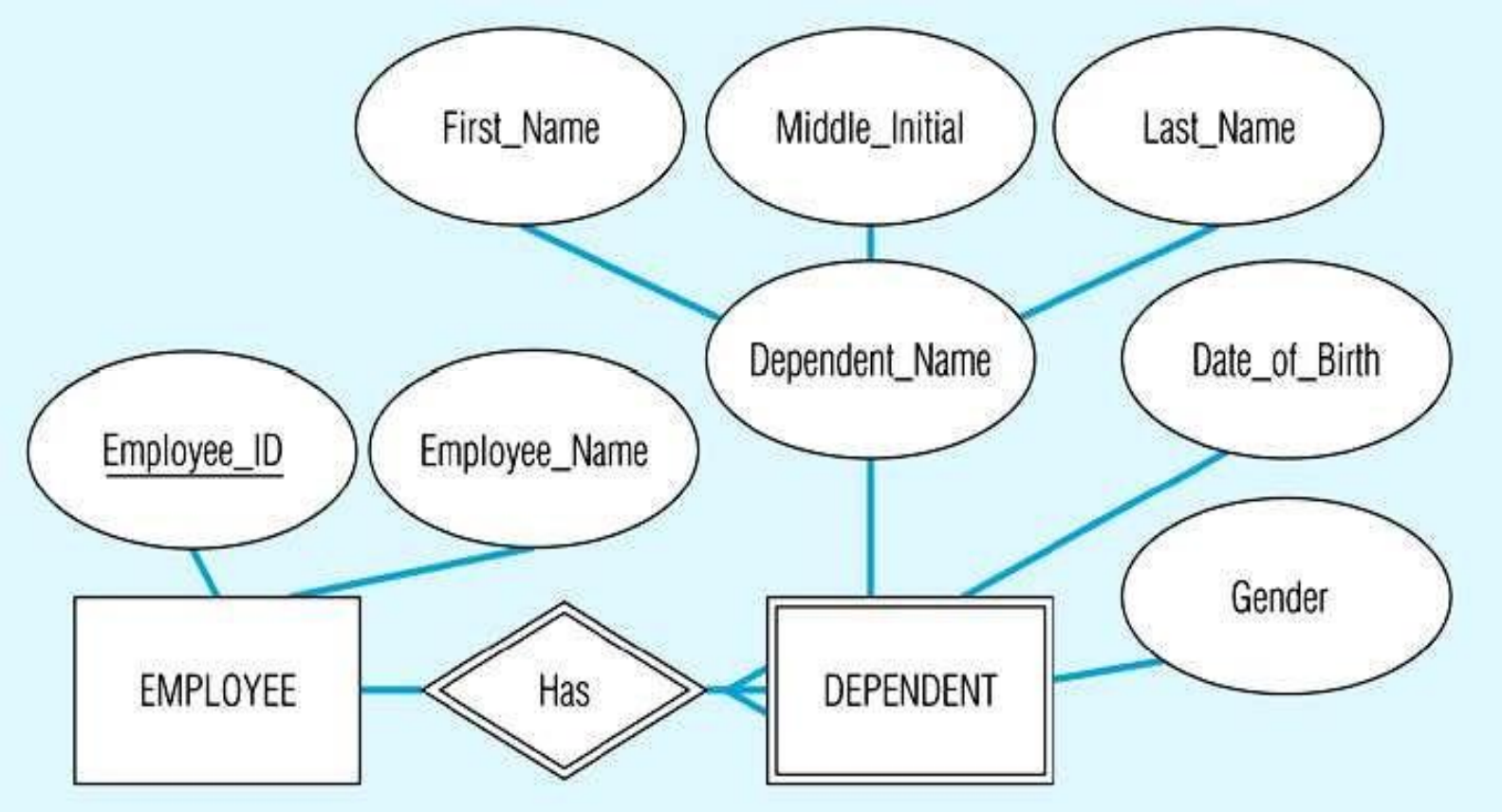
Transforming ERD into Relations

Mapping Weak Entities

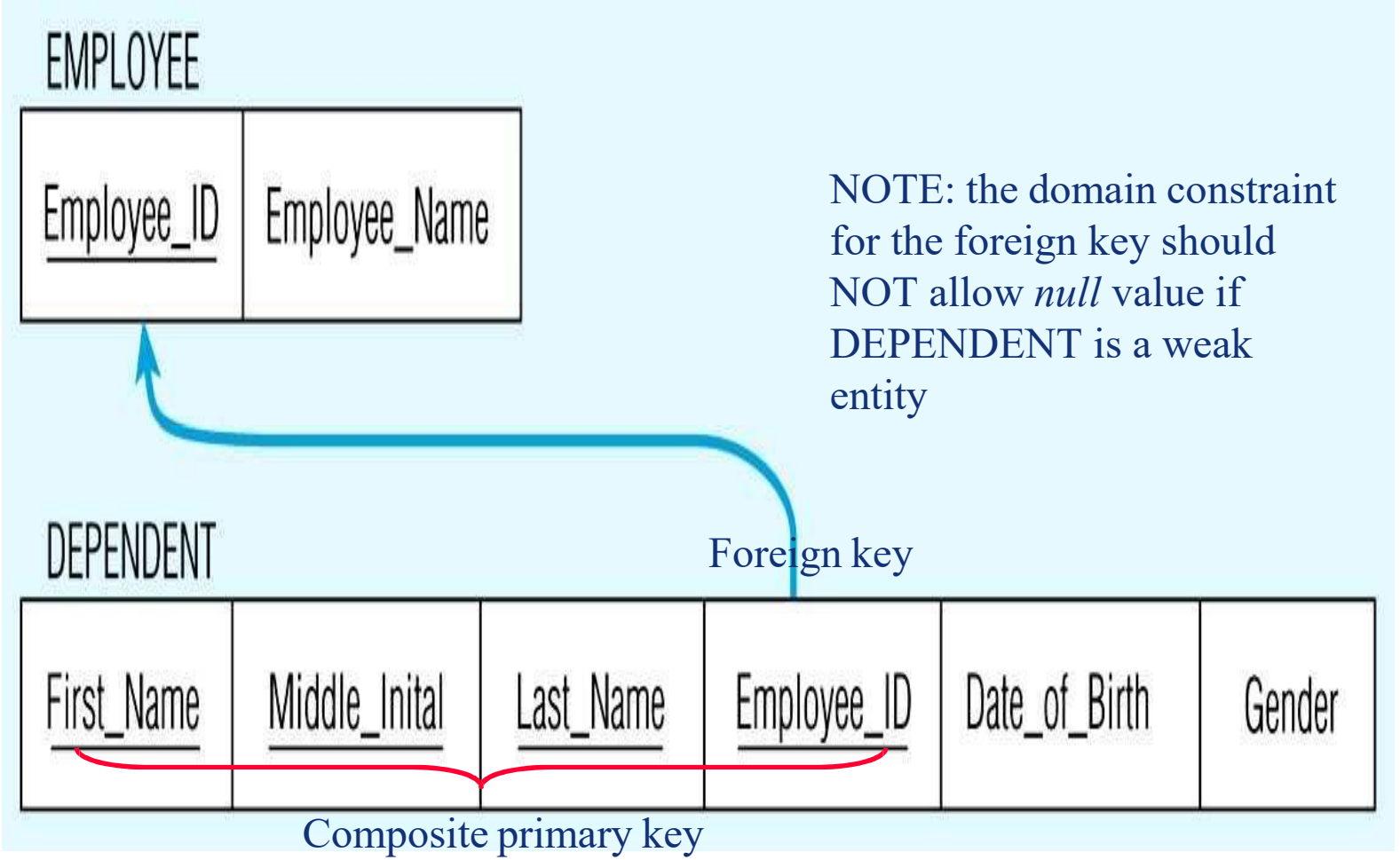
- Becomes a separate relation with a foreign key taken from the superior entity
- Primary key composed of:
  - Partial identifier of weak entity
  - Primary key of identifying relation (strong entity)

Example of mapping a weak entity

(a) Weak entity DEPENDENT



Relations resulting from weak entity



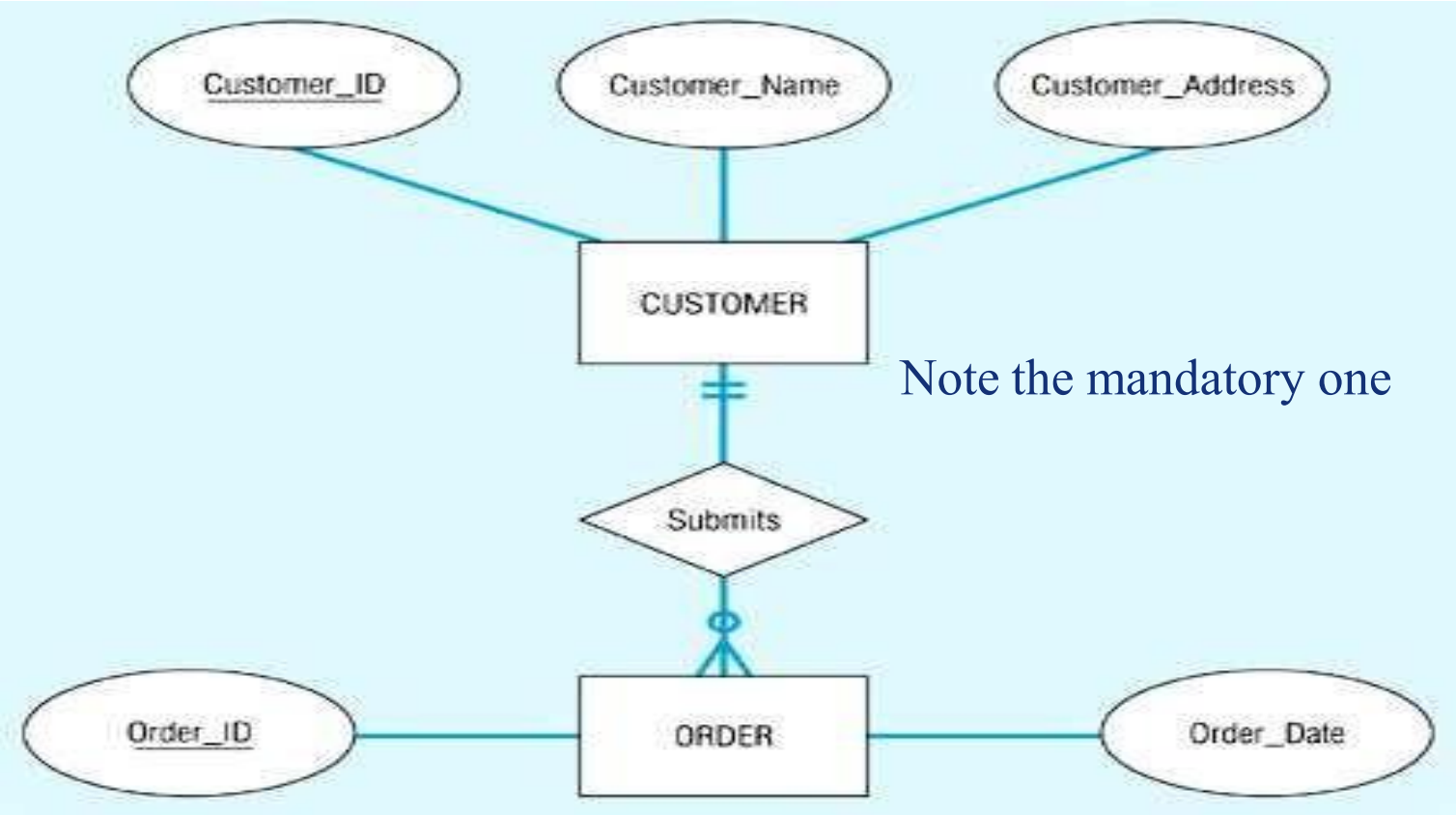
Transforming ERD into Relations

Mapping Binary Relationships

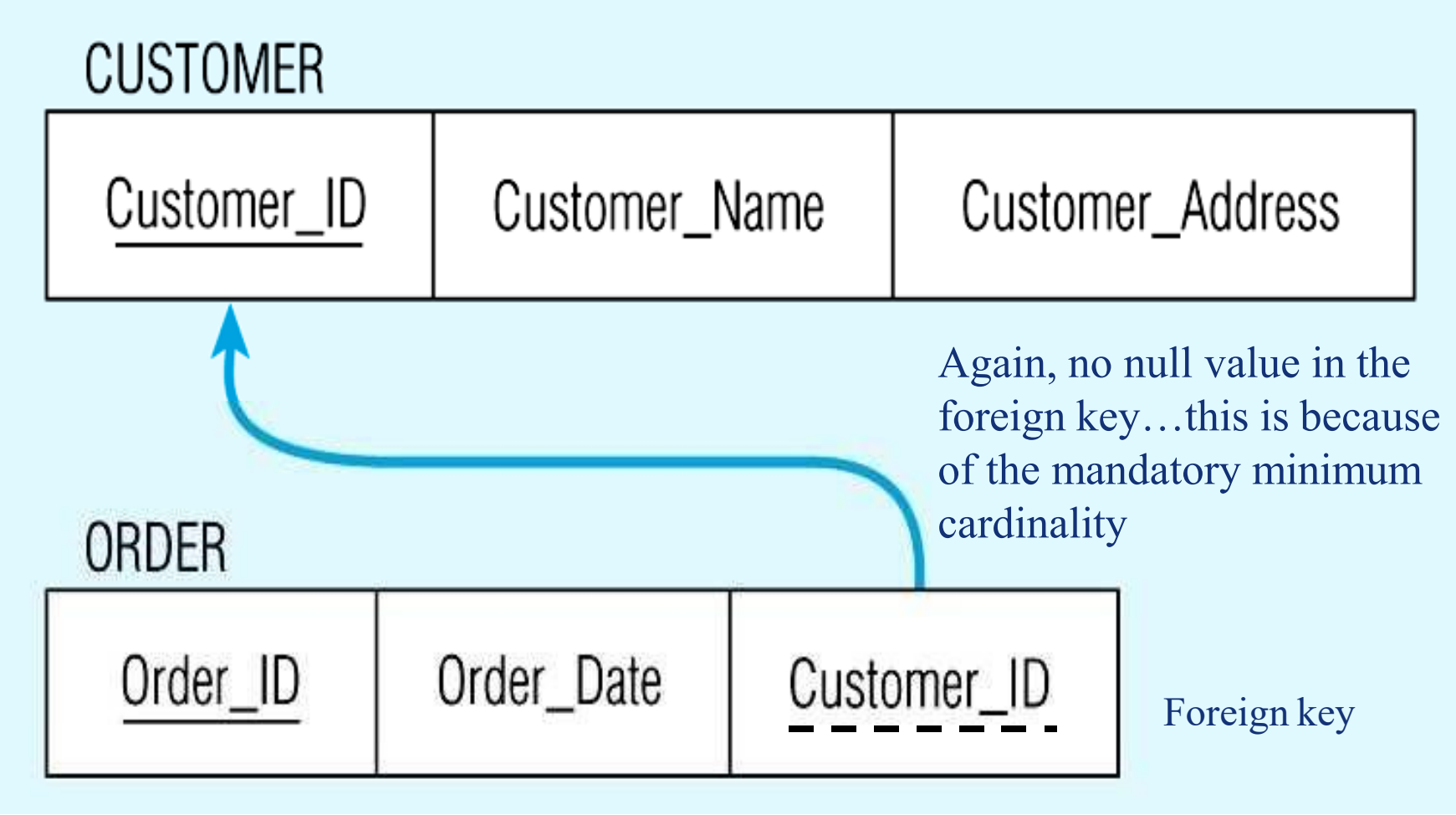
- One-to-Many becomes a foreign key on the ma- Primary key on the one side ny side
- Many-to-Many - Create a ***new relation*** with the primary keys of the two entities as its primary key
- One-to-One - Primary key on the mandatory side becomes a foreign key on the optional side

Example of mapping a 1:M relationship

(a) Relationship between customers and orders

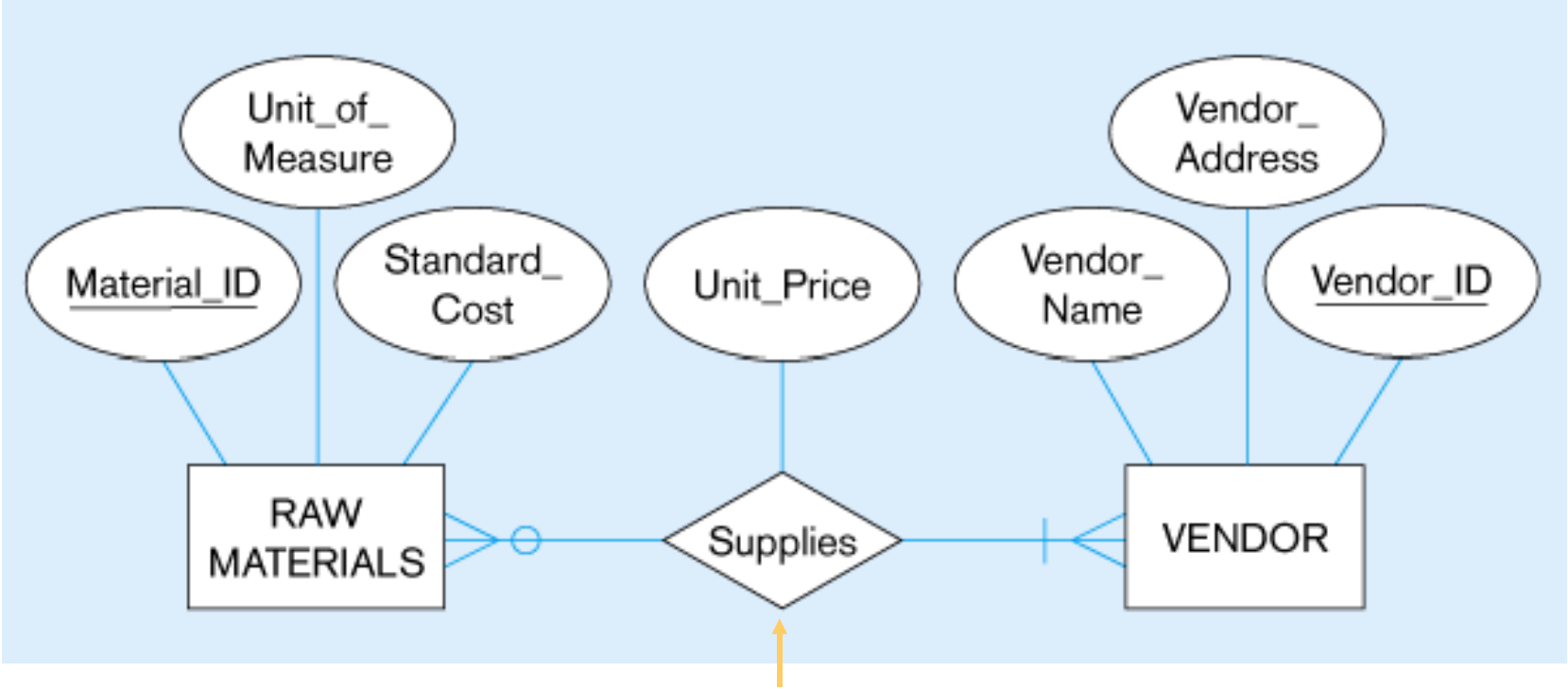


Mapping the relationship



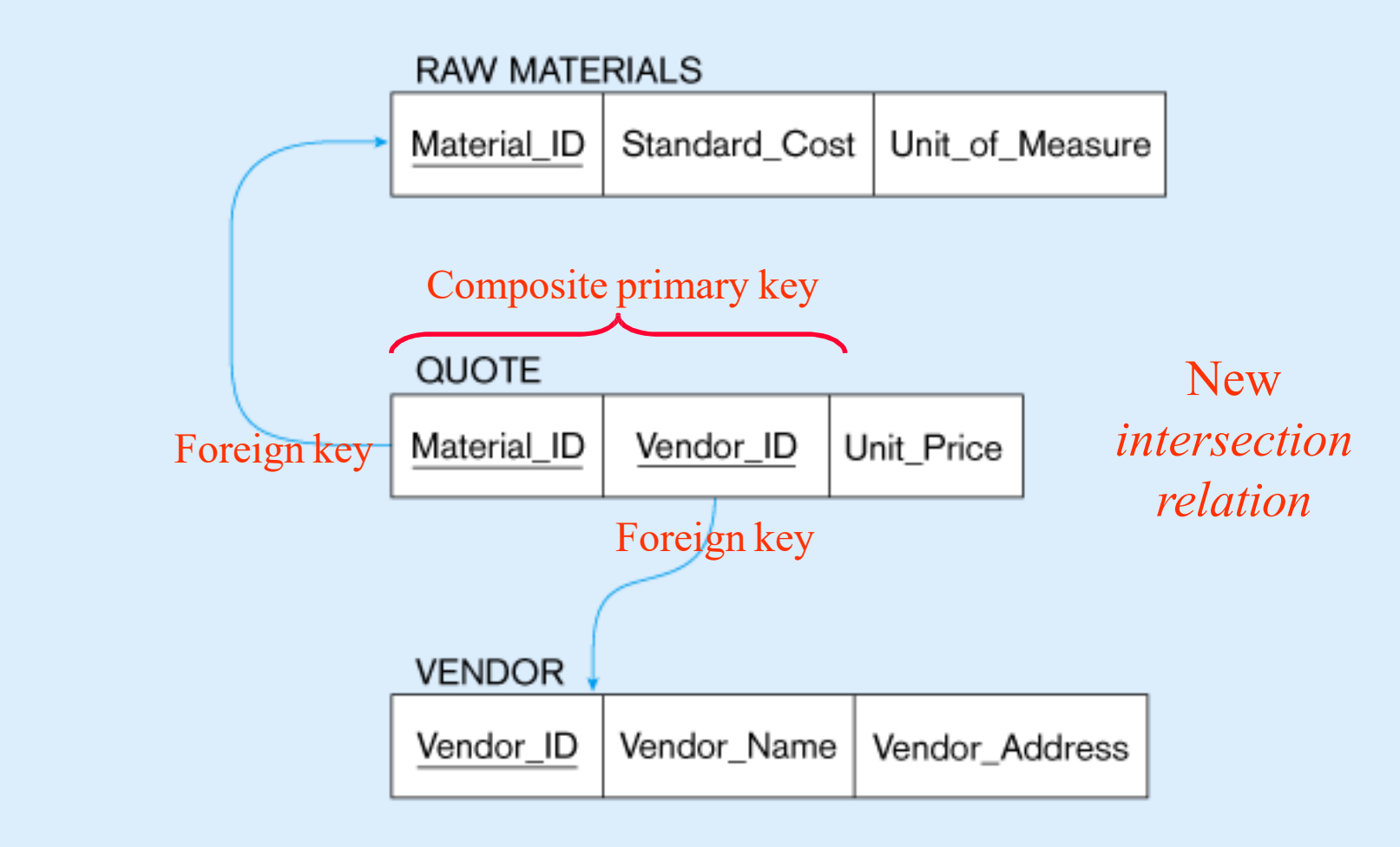
Example of mapping an M:N relationship

(a) ER diagram (M:N)

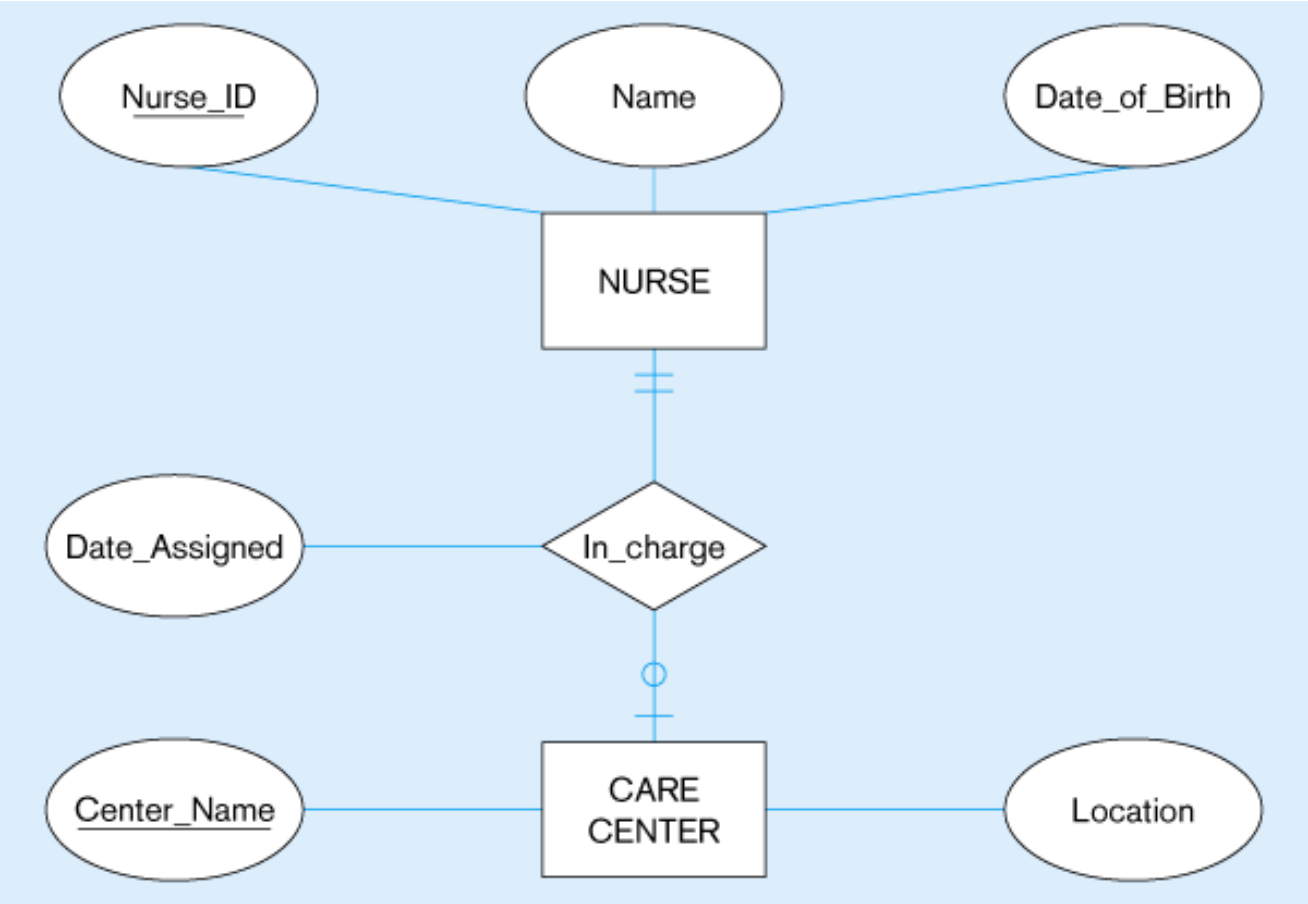


The *Supplies* relationship will need to become a separate relation

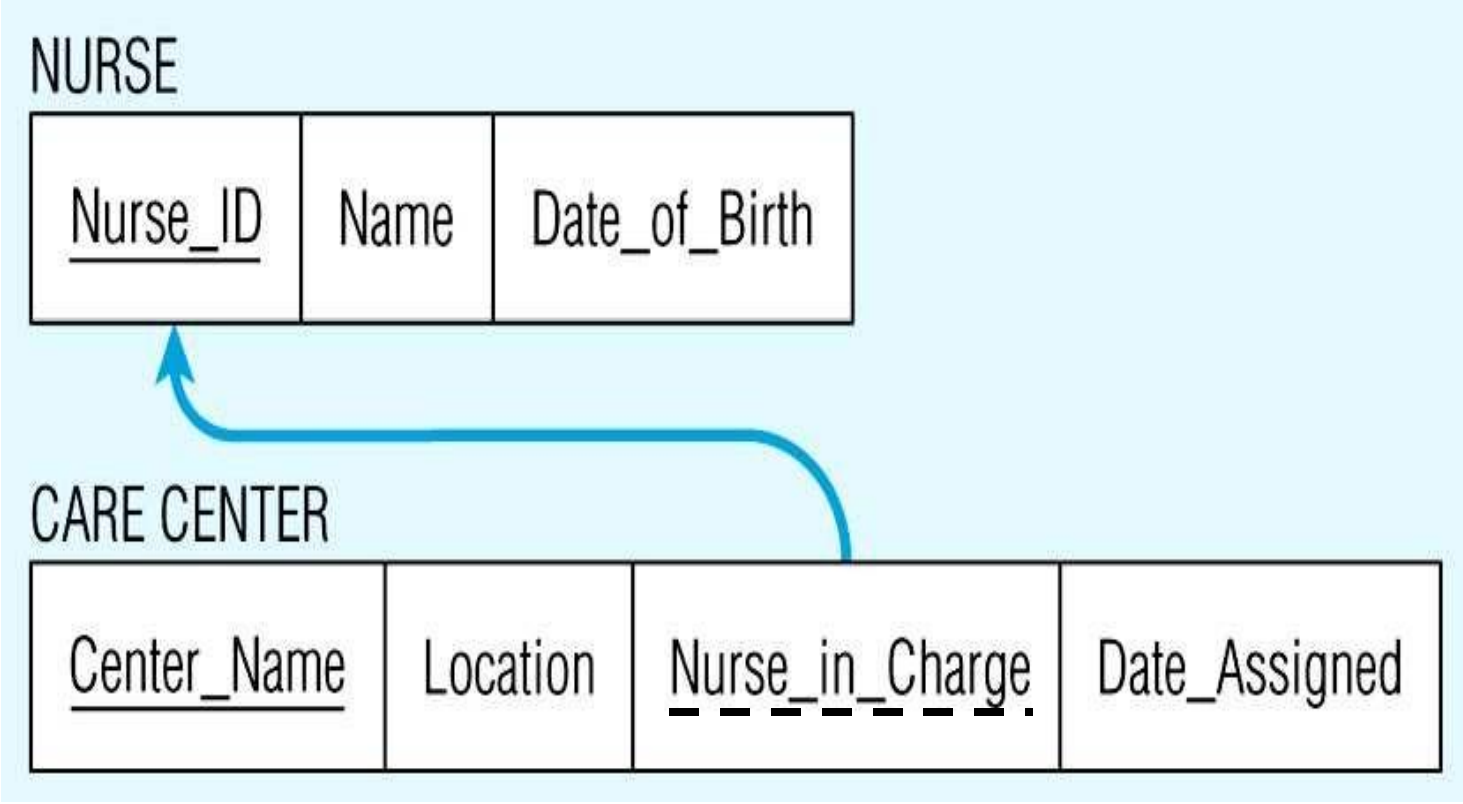
Three resulting relations



Mapping a binary 1:1 relationship



Resulting relations

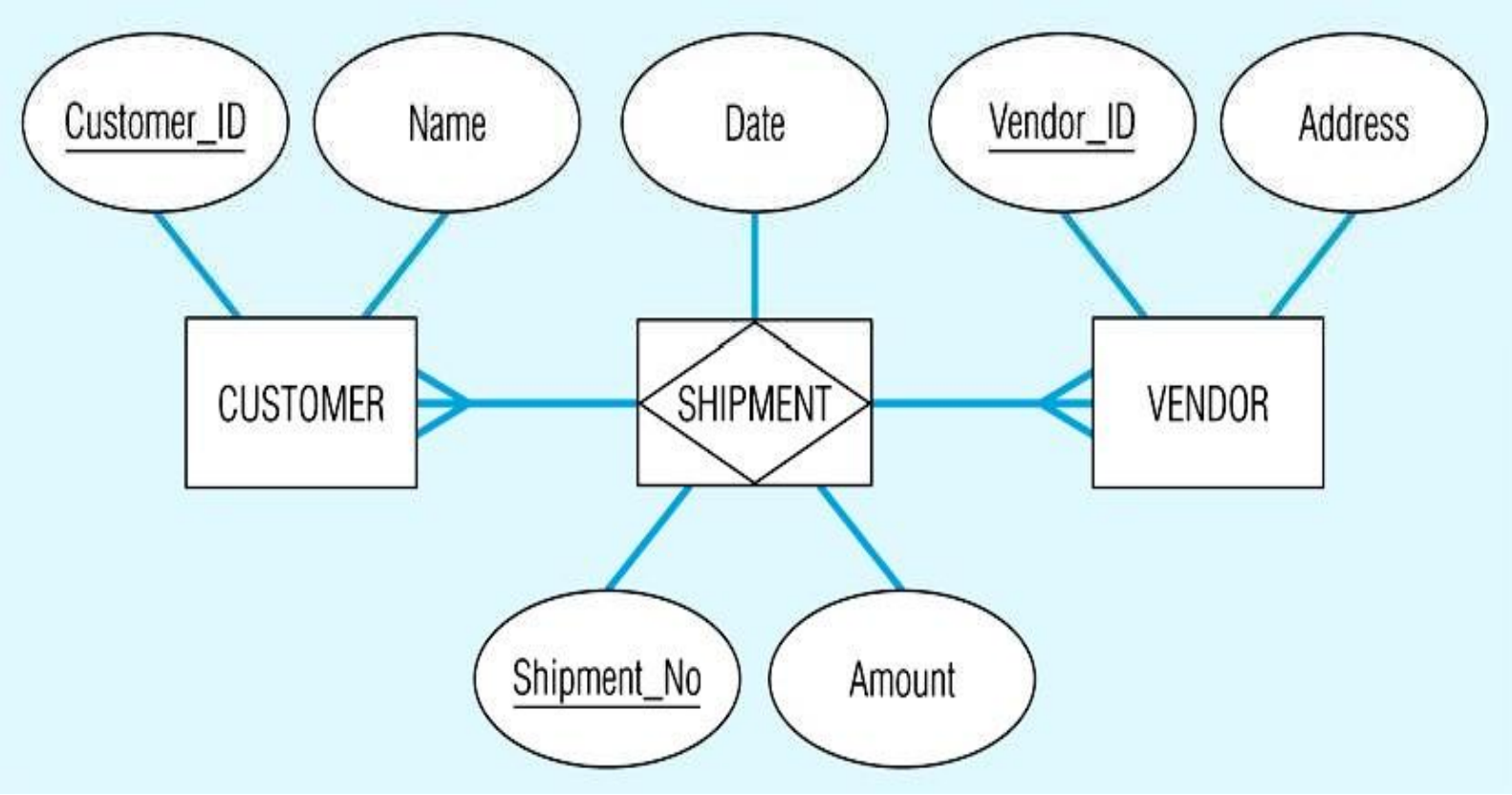


Transforming ERD into Relations

Mapping Associative Entities

- Identifier Not Assigned
  - Default primary key for the association relation is composed of the primary keys of the two entities (as in M:N relationship)
- Identifier Assigned
  - It is natural and familiar to end-users
  - Default identifier may not be unique

Mapping an associative entity

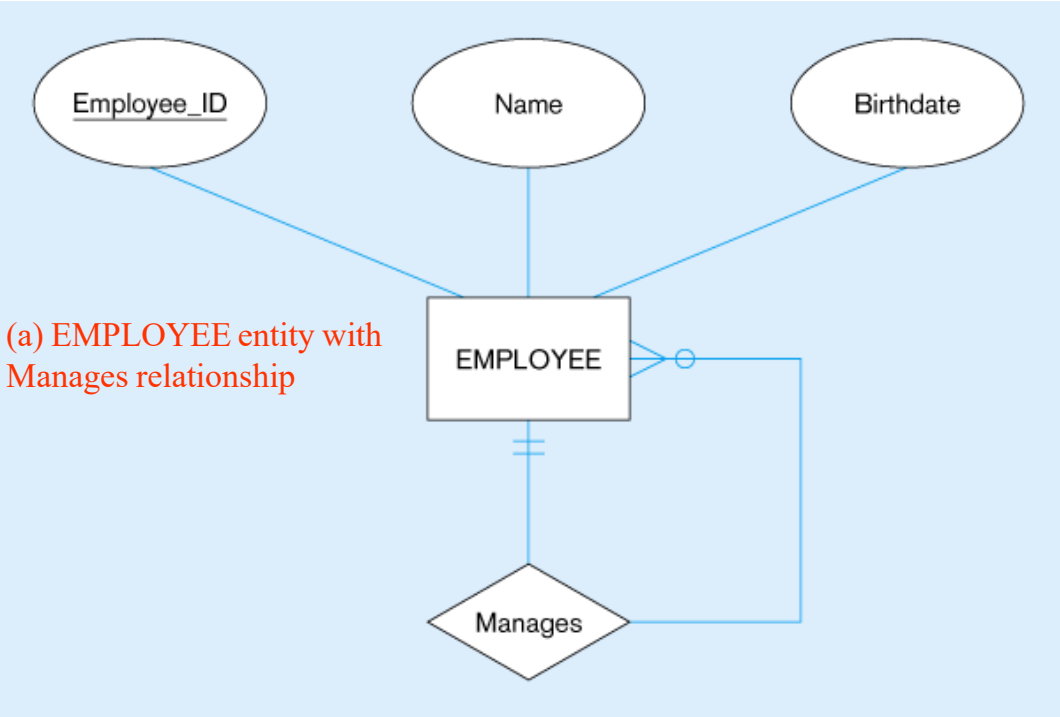


Transforming ERD into Relations

Mapping Unary Relationships

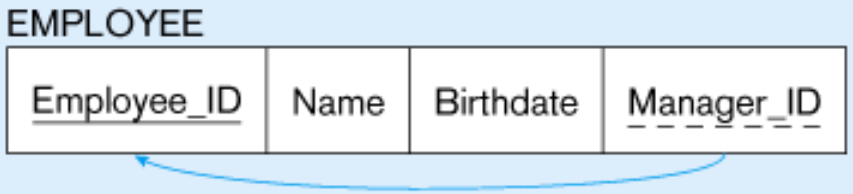
- One-to-Many - Recursive foreign key in the same relation
- Many-to-Many - Two relations:
  - One for the entity type
  - One for an associative relation in which the primary key has two attributes, both taken from the primary key of the entity

Mapping a unary 1:N relationship

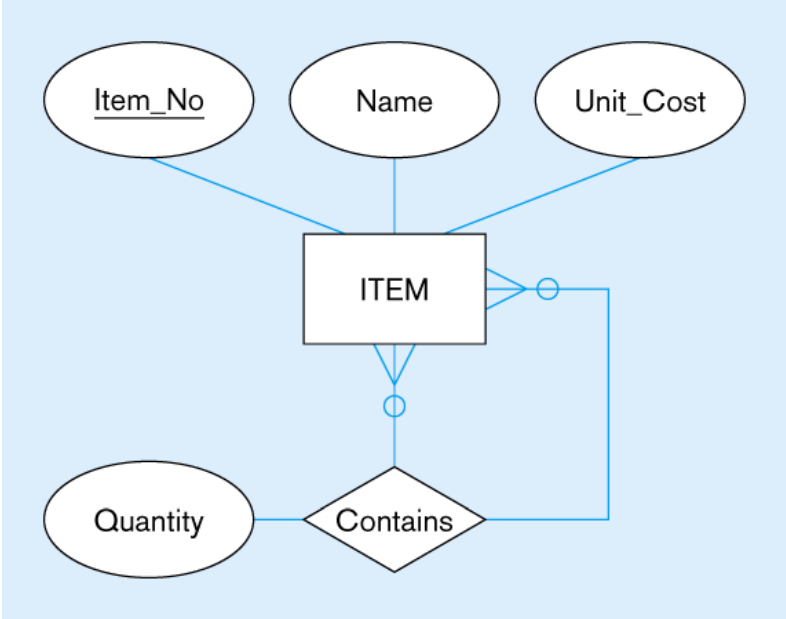


(a) EMPLOYEE entity with Manages relationship

(b) EMPLOYEE relation with recursive foreign key

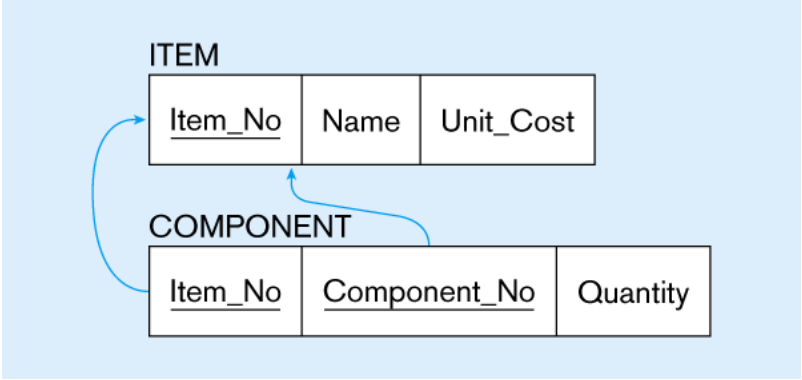


Mapping a unary M:N relationship



(a) Bill-of-materials relationships (M:N)

(b) ITEM and COMPONENT relations

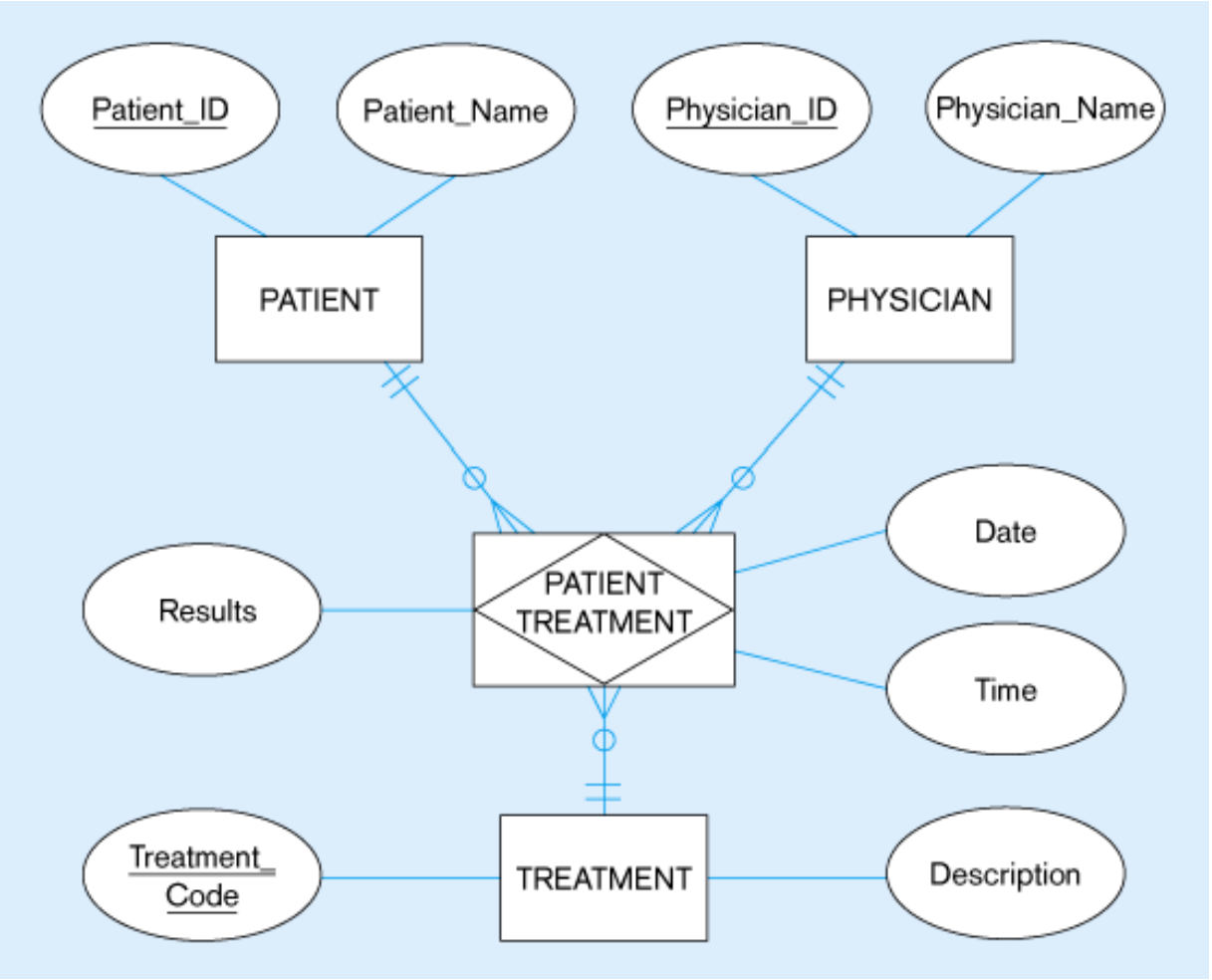


Transforming ERD into Relations

- Mapping Ternary (and n-ary) Relationships
  - One relation for each entity and one for the associative entity
  - Associative entity has foreign keys to each entity in the relationship

Mapping a ternary relationship

(a) Ternary relationship with associative entity

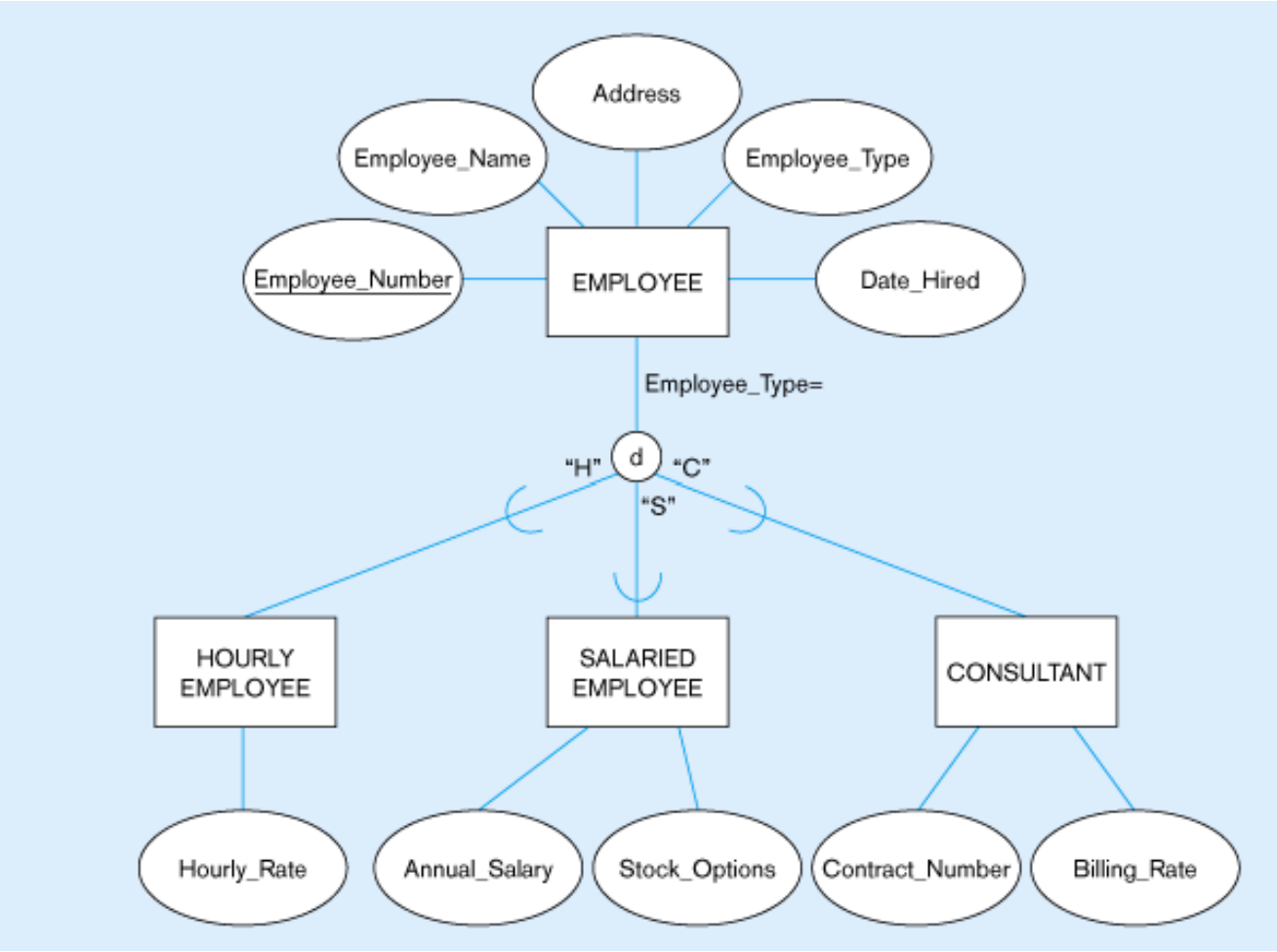


Transforming ERD into Relations

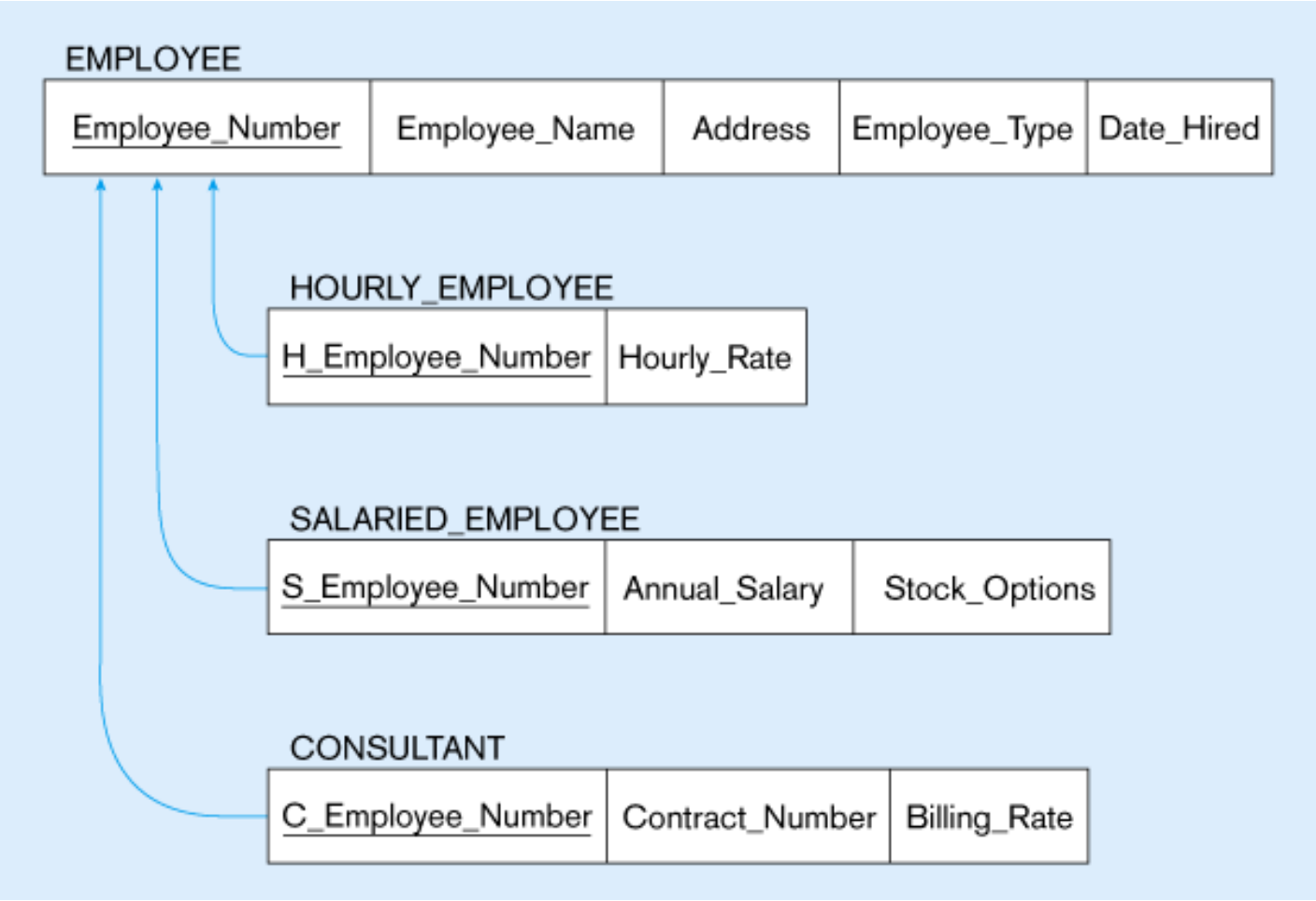
Mapping Supertype/Subtype Relationships

- One relation for supertype and for each subtype
- Supertype attributes (including identifier and subtype discriminator) go into supertype relation
- Subtype attributes go into each subtype; primary key of supertype relation also becomes primary key of subtype relation
- 1:1 relationship established between supertype and each subtype, with supertype as primary table

Supertype / subtype relationships



Mapping Supertype/subtype relationships to relations



Any Questions ?

For more details see this link…

example

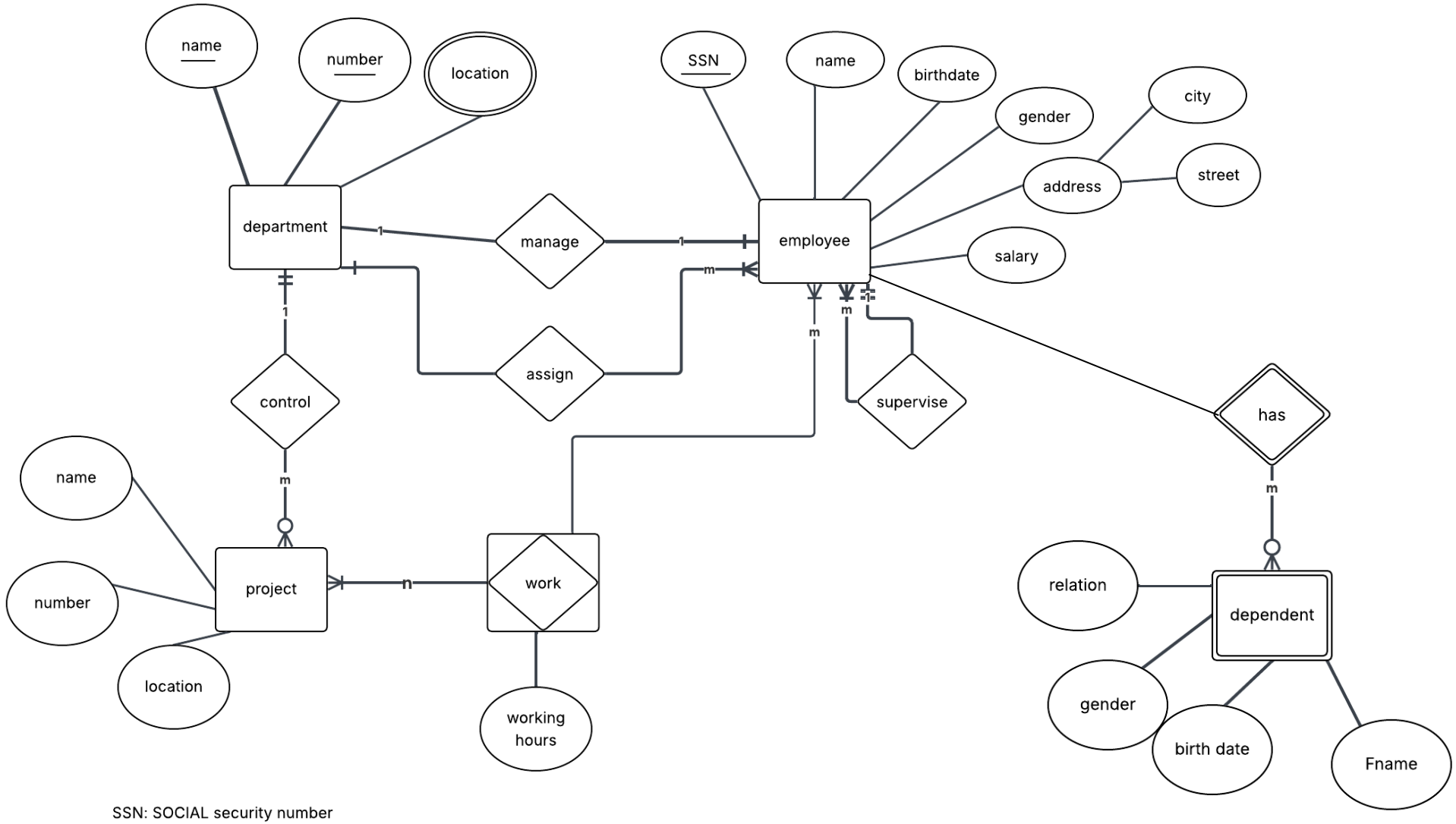
- A company is organized into departments. Each department has a unique name, a unique number, and a particular employee who manages the department. A department may have several locations.

- A department may control a number of projects, each of which has a unique name, a unique number, and a single location. A project must be controlled by department

- We store employee's name, social security number, address, salary, gender and birth date. An employee must be assigned to one department and must work on one or more 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.

- We want to keep track of the dependents of each employee for insurance purposes. We keep each dependent's first name, gender, birth date and relationship to that employee.

ERD solution



SSN: SOCIAL security number

Mapping diagram solution

example

Table: Employee Columns:

- employee_id (unique)
- employee_name
- social_security_number
- address
- salary
- gender
- birth_date
- department_number (foreign key referencing Department)
- supervisor_employee_id (foreign key referencing Employee)

example

Table: Department Columns:

- department_number (unique)
- department_name
- department_manager

Table: Location Columns:

- location_id (unique)
- location_name

Table: Project Columns:

- project_number (unique)
- project_name
- department_number (foreign key referencing Department)

example

Table: Employee_Project Columns:

- employee_id (foreign key referencing Employee)
- project_number (foreign key referencing Project)
- hours_per_week

Table: Dependent Columns:

- dependent_id (unique)
- employee_id (foreign key referencing Employee)
- dependent_name
- gender
- birth_date
- relationship

Programs

- Draw.io : https://www.drawio.com/
- Lucid Chart : https://lucid.app/
- edrawMax: https://www.edrawsoft.com/

Assignment

- **Draw ERD and mapping diagram for this case:-**

- A student in a university takes one or more courses according to his choice. However, in order to be a student, he must take at least one course with determining the semester of taking the course. The course may have attributes like course_id, course_name and course_fee.
- The student entity may have attributes like university_id, name and phone numbers and address(city and street and zip).
- A course is always thought by some course teacher. A course must be thought by at least one or more teacher. On the other hand, a teacher may teach at least one or more course.
- Each department has a chairman while chairman chairs only one department. Chairman is a week entity because its existence is dependent on corresponding department. The department has attributes like department_id and department_name.
- A teacher belongs to only one department. A department may have at least one or more teachers. The teacher can be visiting, part_time, or full_time.
- You should identify the primary key for each entity, if not exist, add one.

Thank you