Entity/Relationship Modelling

Database Design Process

1. Requirements Analysis → 2. Conceptual Design → 3. Logical, Physical, Security, etc.

Technical and non-technical people are involved

1. Requirements analysis

- What is going to be stored?
- How is it going to be used?
- What are we going to do with the data?
- Who should access the data?

1. Requirements Analysis | **2. Conceptual Design** | 3. Logical, Physical, Security, etc.

2. Conceptual Design

- A <u>high-level description</u> of the database
- Sufficiently <u>precise</u> that technical people can understand it
- But, <u>not so precise that non-technical people can't participate</u>

This is where E/R fits in.

1. Requirements Analysis | 2. Conceptual Design | 3. Logical, Physical, Security, etc.

3. More:

- Logical Database Design
- Physical Database Design
- Security Design

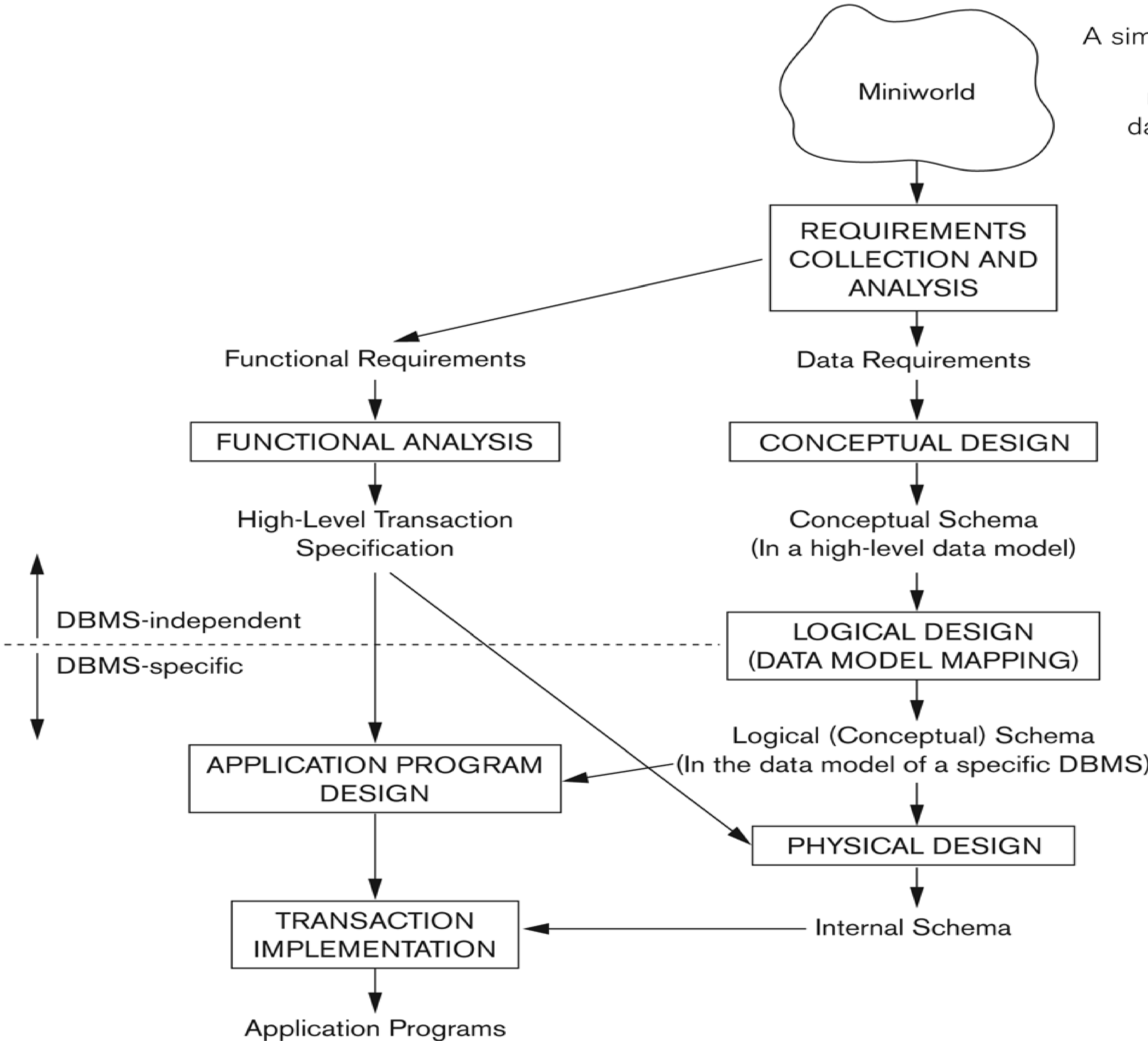


Figure 3.1
A simplified diagram to illustrate the main phases of database design.

Database Design

- Conceptual Design (cont'd)
  - The Entity-Relationship (ER) Model, UML
  - High-level, close to human thinking
  - Semantic model, intuitive, rich constructs
  - Not directly implementable
- Logical Design (data model mapping)
  - The relational data model
  - Logical design translates ER into relational model (SQL)
  - Map conceptual schema to an implementation data model (e.g., relational database model)
- Physical Design
  - Specify internal storage structures, indices, access paths and file organizations

Conceptual Design – ER Model

- What are the *entities* and *relationships* in a typical application?
  - What information about these entities and relationships should we store in the database?

- What are the *integrity constraints* or *business rules*
  - Key constraints
  - Participation constraints

- Representation through *ER diagrams*
  - ER diagrams are then mapped into relational schemas
  - Conversion is fairly mechanical in simple cases, but can be tricky

Components of ERD

1. Entity.

A data entity is anything real or abstract about which we want to store data.

Entity types fall into five classes: roles, events, locations, tangible things or concepts.

E.g. employee, payment, campus, book.

2. Relationship.

A data relationship is a natural association that exists between one or more entities.

E.g. Employees process payments.

3. Cardinality.

Defines the number of occurrences of one entity for a single occurrence of the related entity.

E.g. an employee may process many payments but might not process any payments depending on the nature of her job.

4. Attribute.

A data attribute is a characteristic common to all or most instances of a particular entity.

Synonyms include property, data element, field.

E.g. Name, address, Employee Number, pay rate are all attributes of the entity employee.

Entities and Entity Sets

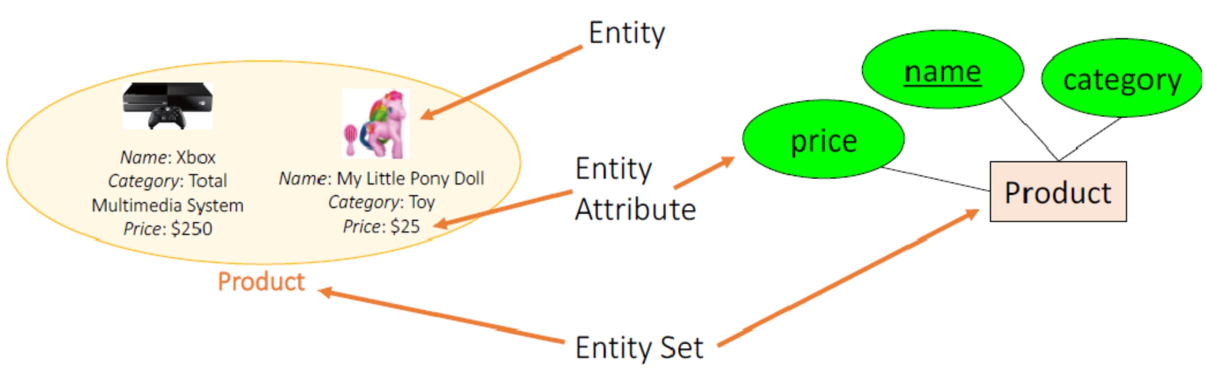
- ***Entity:*** represents a real-world object
  - Characterized using set of ***attributes***
  - Each attribute has a ***domain*** – similar to variable types

- ***Entity Set:*** represents collection of similar entities
  - E.g., all employees in an organization
  - All entities in an entity set share same set of attributes

Entities vs. Entity Sets

Example:

Entities are **not** explicitly represented in E/R diagrams!



Keys

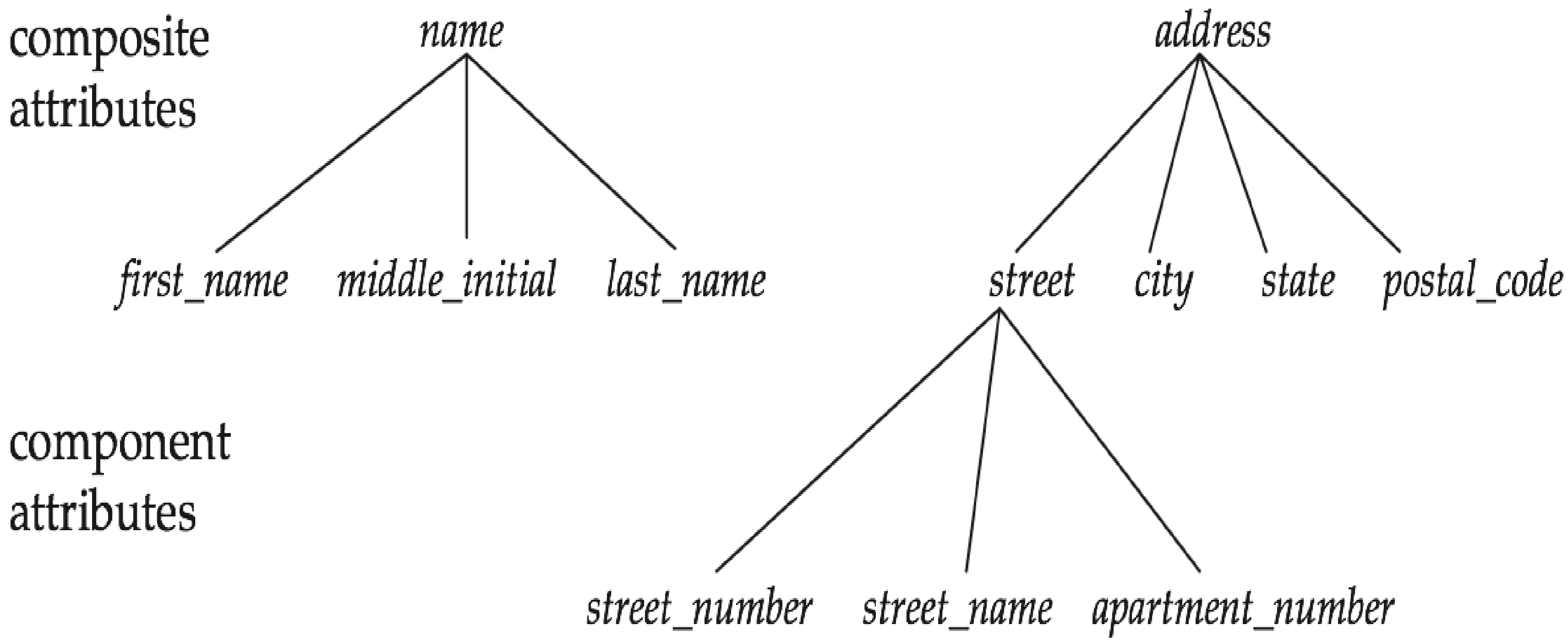
- Each entity set has a key
- Set of attributes that uniquely identify an entity (one entity in the entity set)
- Multiple candidate keys may exist
- Primary key selected among them

Attributes

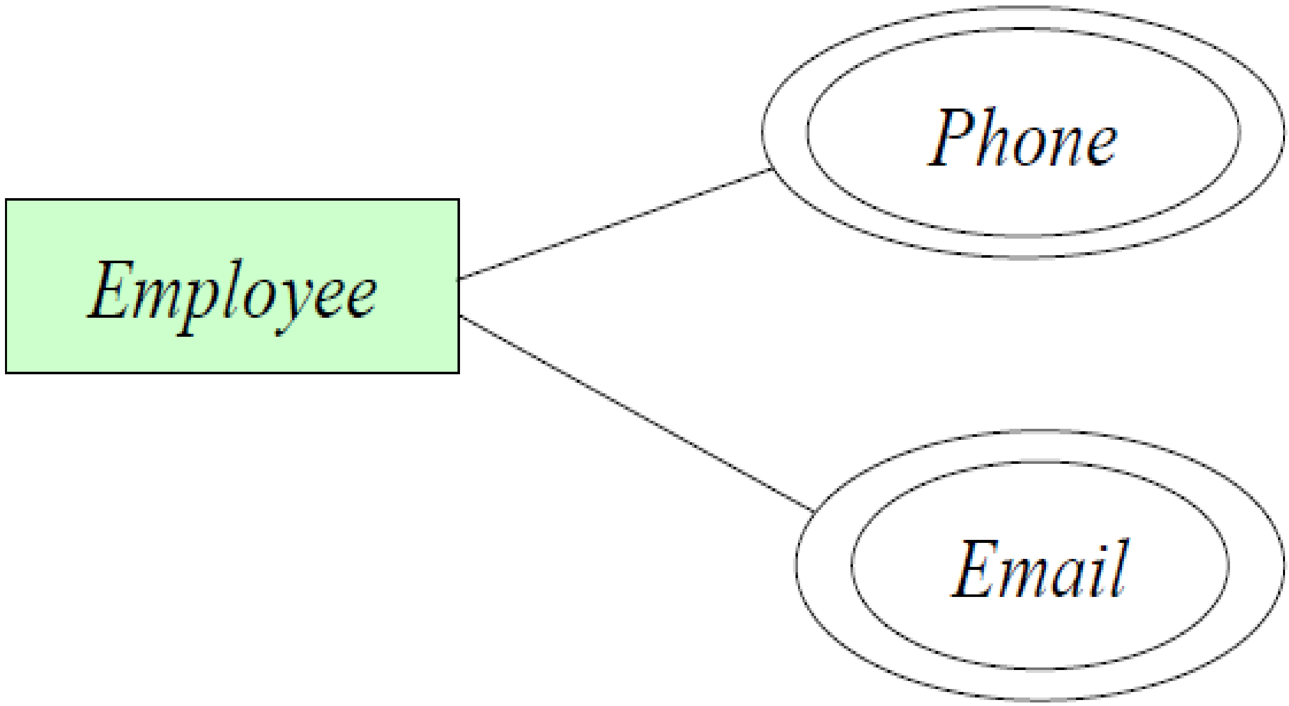
- An entity is represented by a set of attributes, that is descriptive properties possessed by all members of an entity set.
  - Example:

    instructor = (*ID*, *name*, *street*, *city*, *salary*)
    course= (*course_id*, *title*, *credits*)
- **Domain** – the set of permitted values for each attribute
- Attribute types:
  - **Simple** and **composite** attributes.
  - **Single-valued** and **multivalued** attributes
    - Example: multivalued attribute: *phone_numbers*
  - **Derived** attributes
    - Can be computed from other attributes
    - Example: age, given date_of_birth

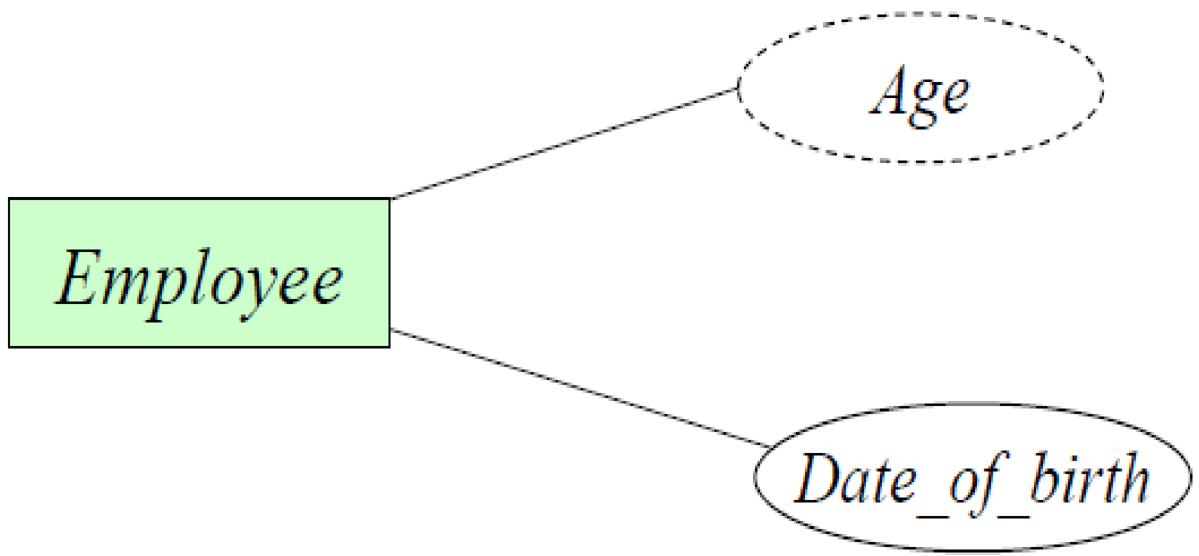
Composite Attributes



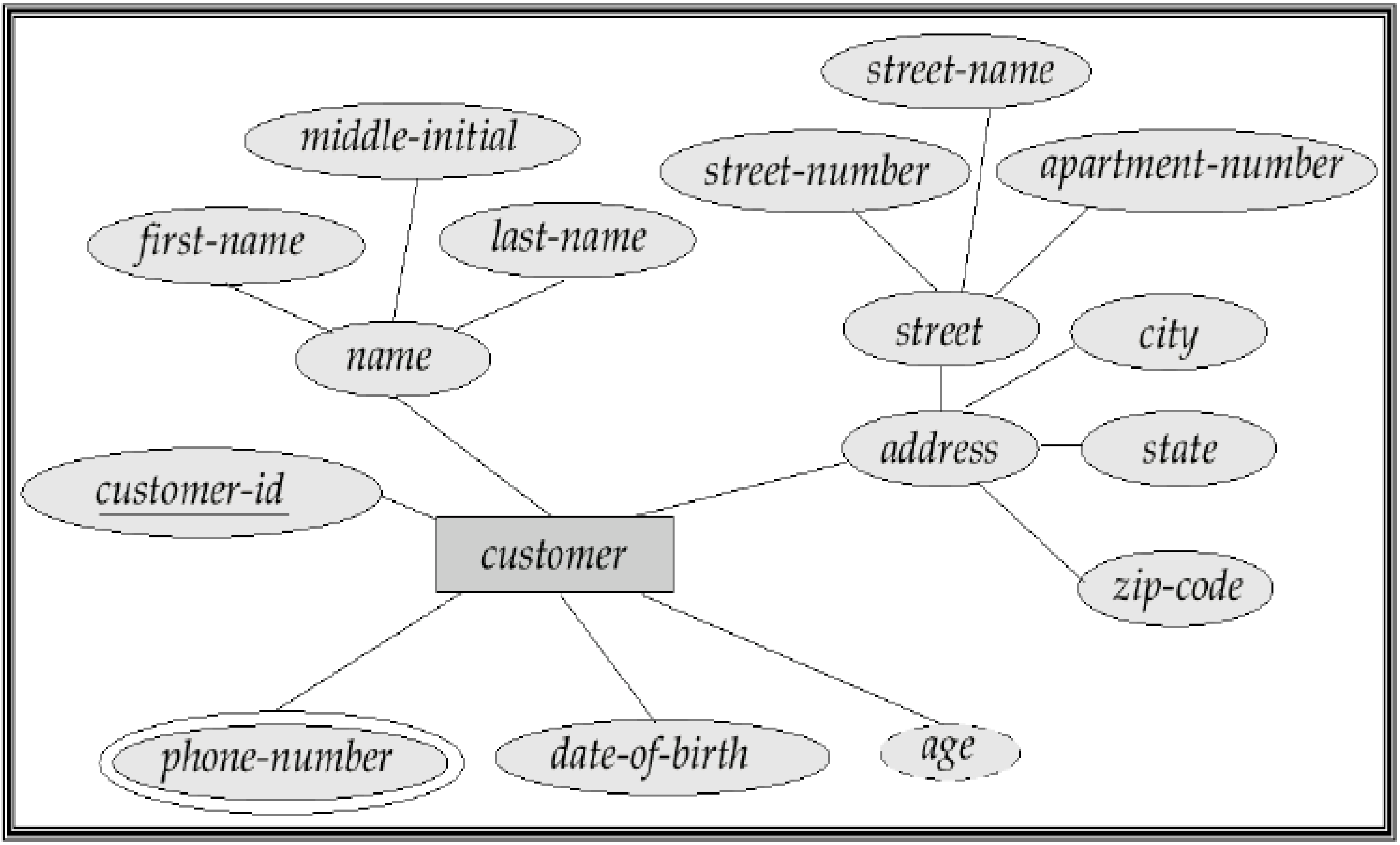
- Multivalued attribute: contains more than one value



- Derived attribute: computed from other attributes (e.g., age can be computed from the date of birth and the current date)



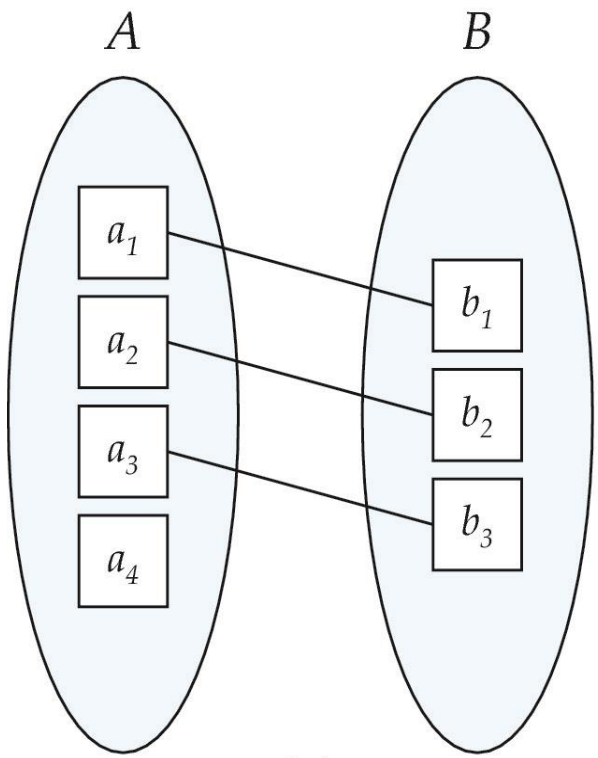
E-R diagram for entity *customer*



Mapping Cardinality Constraints

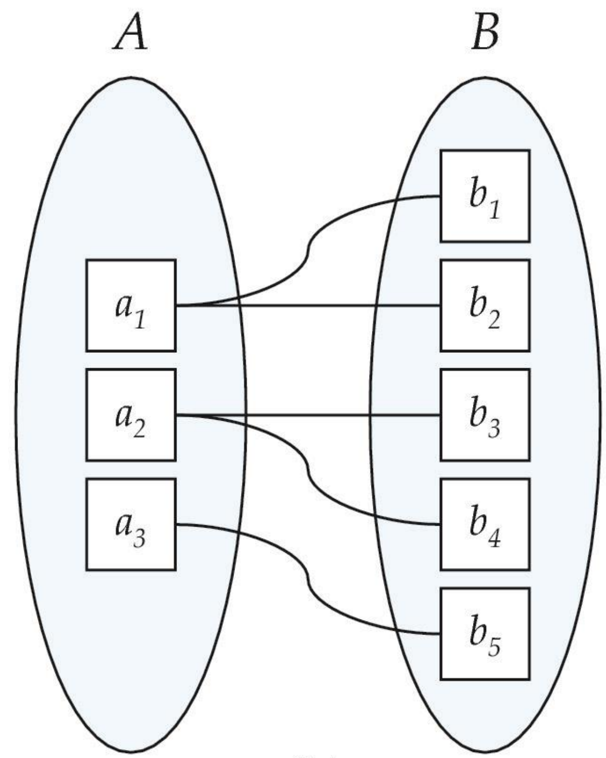
- Express the number of entities to which another entity can be associated via a relationship set.
- Most useful in describing binary relationship sets.
- For a binary relationship set the mapping cardinality must be one of the following types:
  - One to one
  - One to many
  - Many to one
  - Many to many

Mapping Cardinalities



(a)

One to one

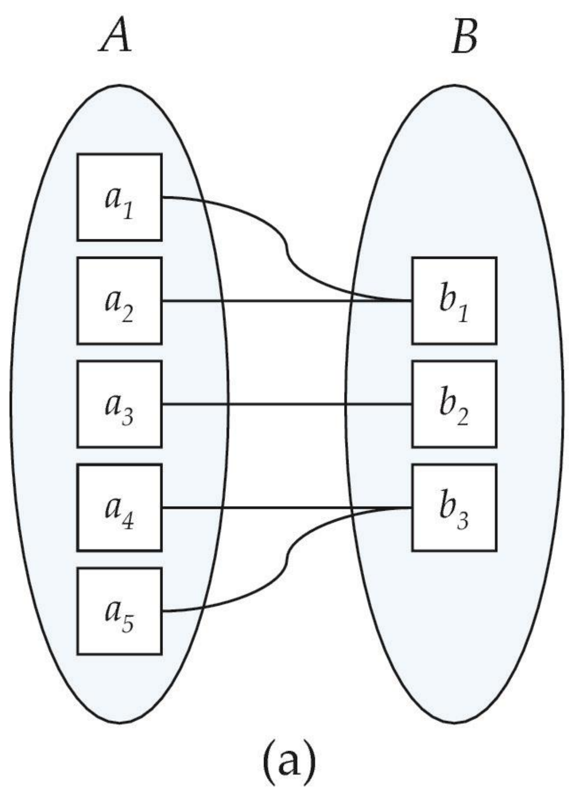


(b)

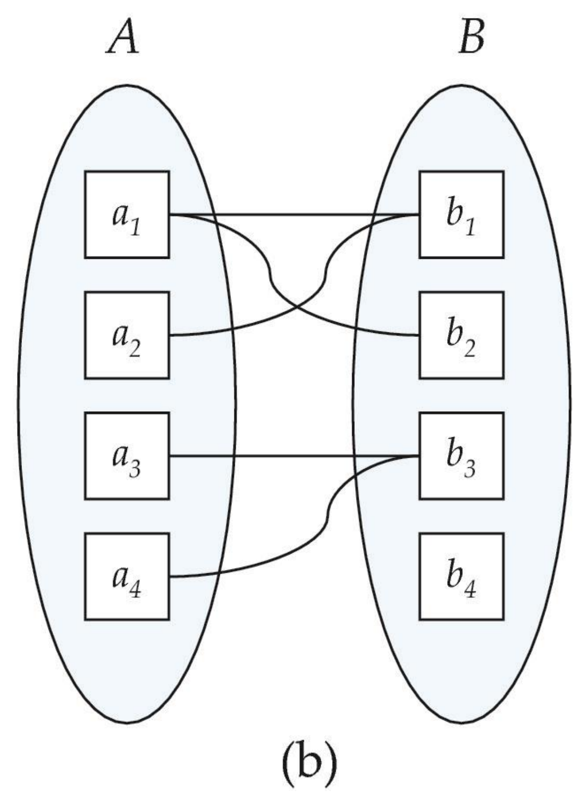
One to many

Note: Some elements in *A* and *B* may not be mapped to any elements in the other set

Mapping Cardinalities



Many to one

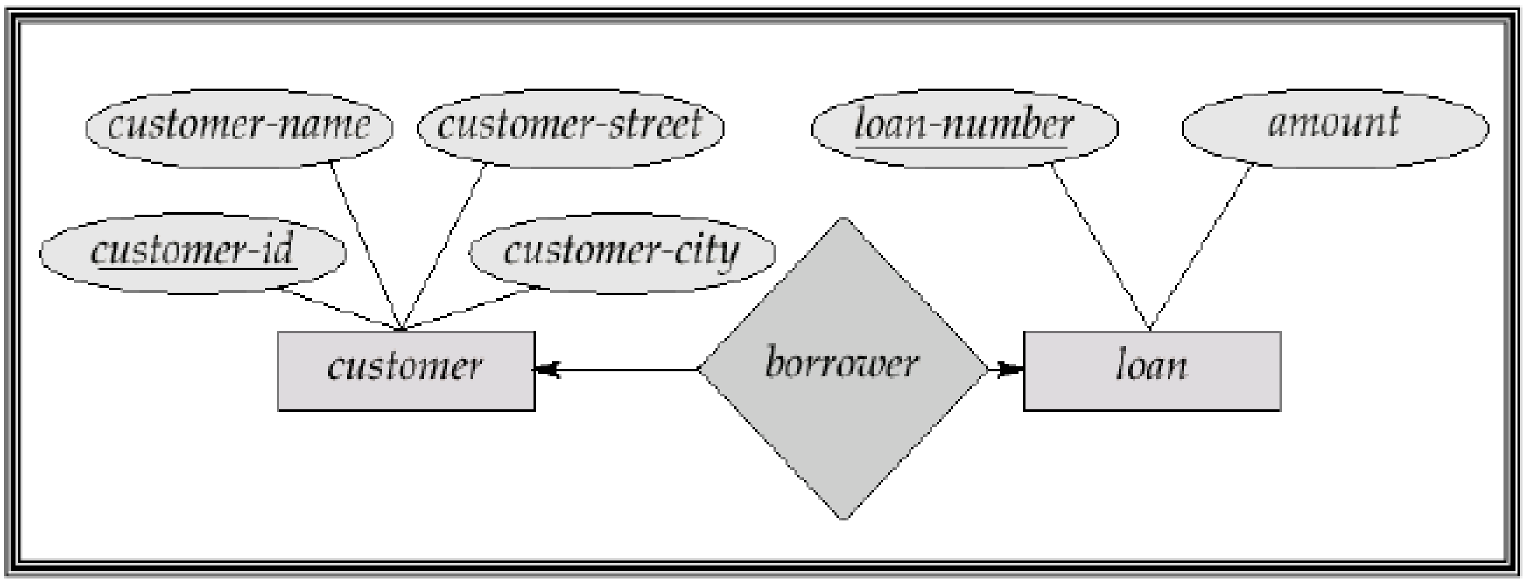


Many to many

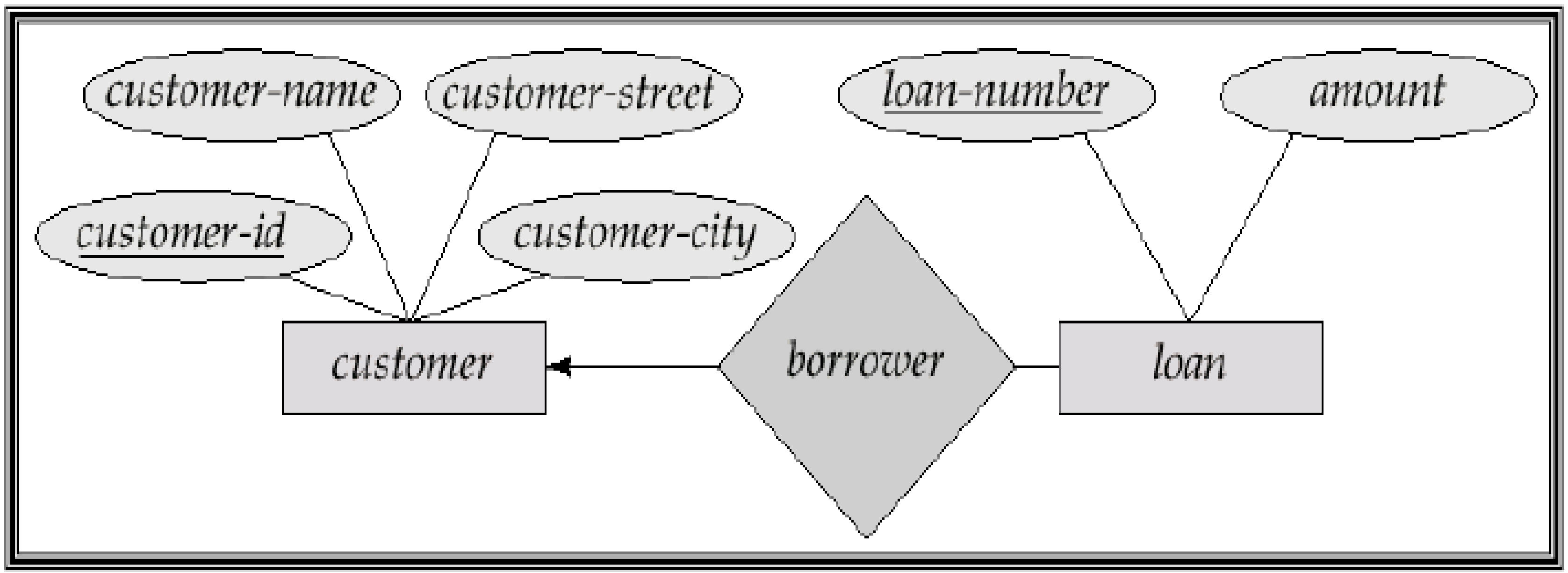
Note: Some elements in A and B may not be mapped to any elements in the other set

Cardinality Constraints

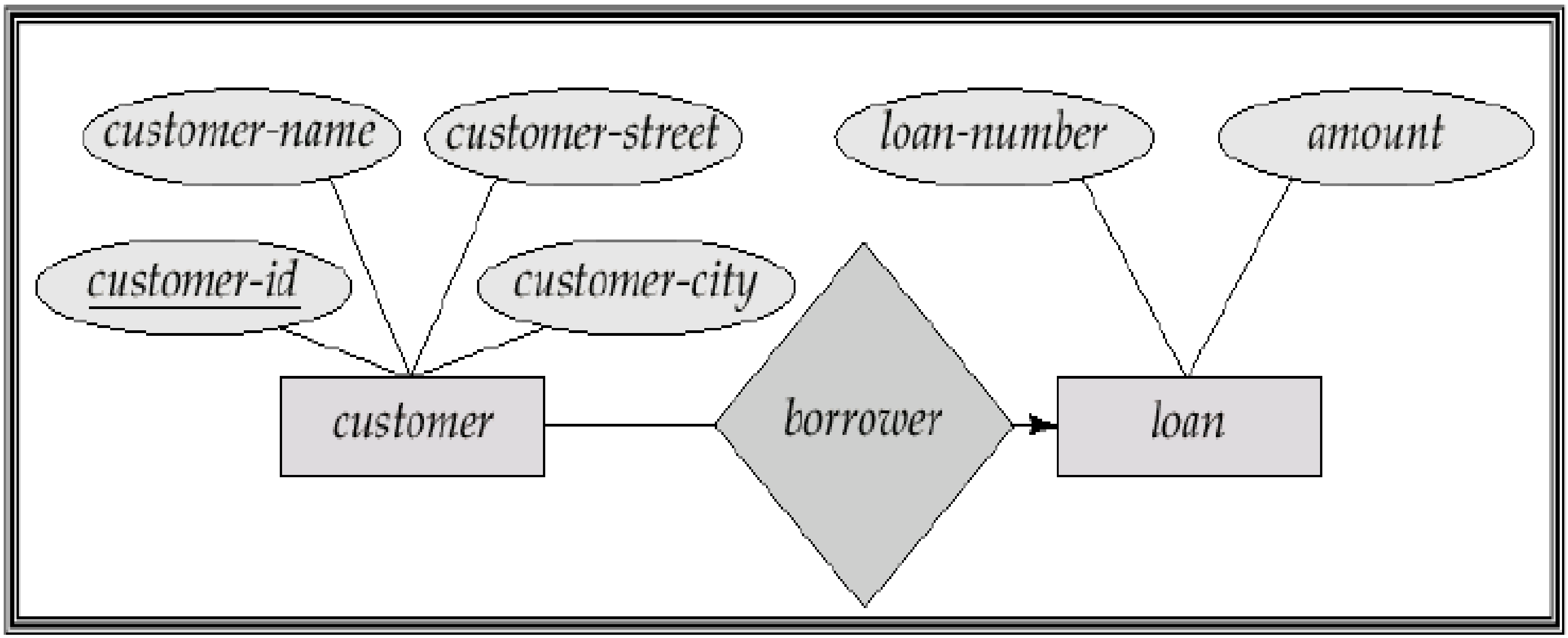
- We express cardinality constraints by drawing either a directed line (→), signifying "one", or an undirected line (—), signifying "many", between the relationship set and the entity set
- One-to-one relationship
  - A customer is associated with at most one loan via the relationship *borrower*
  - A loan is associated with at most one customer via *borrower*



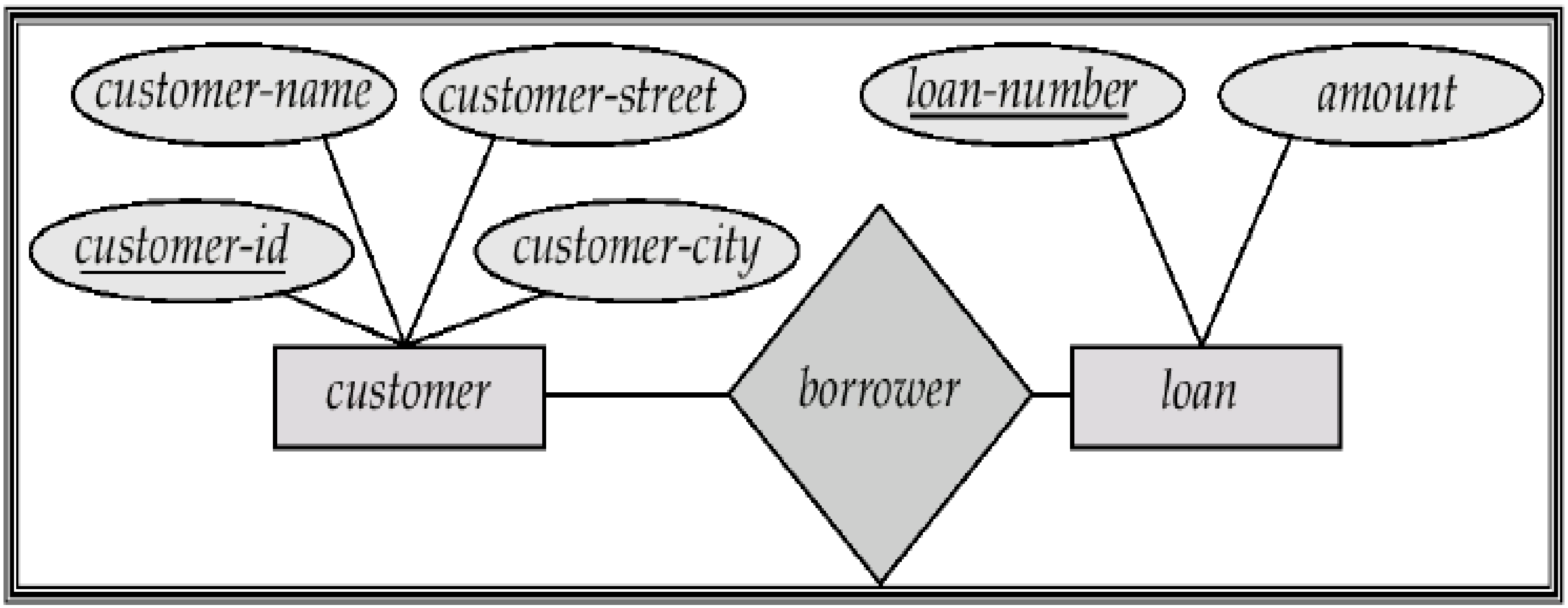
- One-to-many relationship
  - a loan is associated with at most one customer via *borrower*
  - a customer is associated with several (including 0) loans via *borrower*



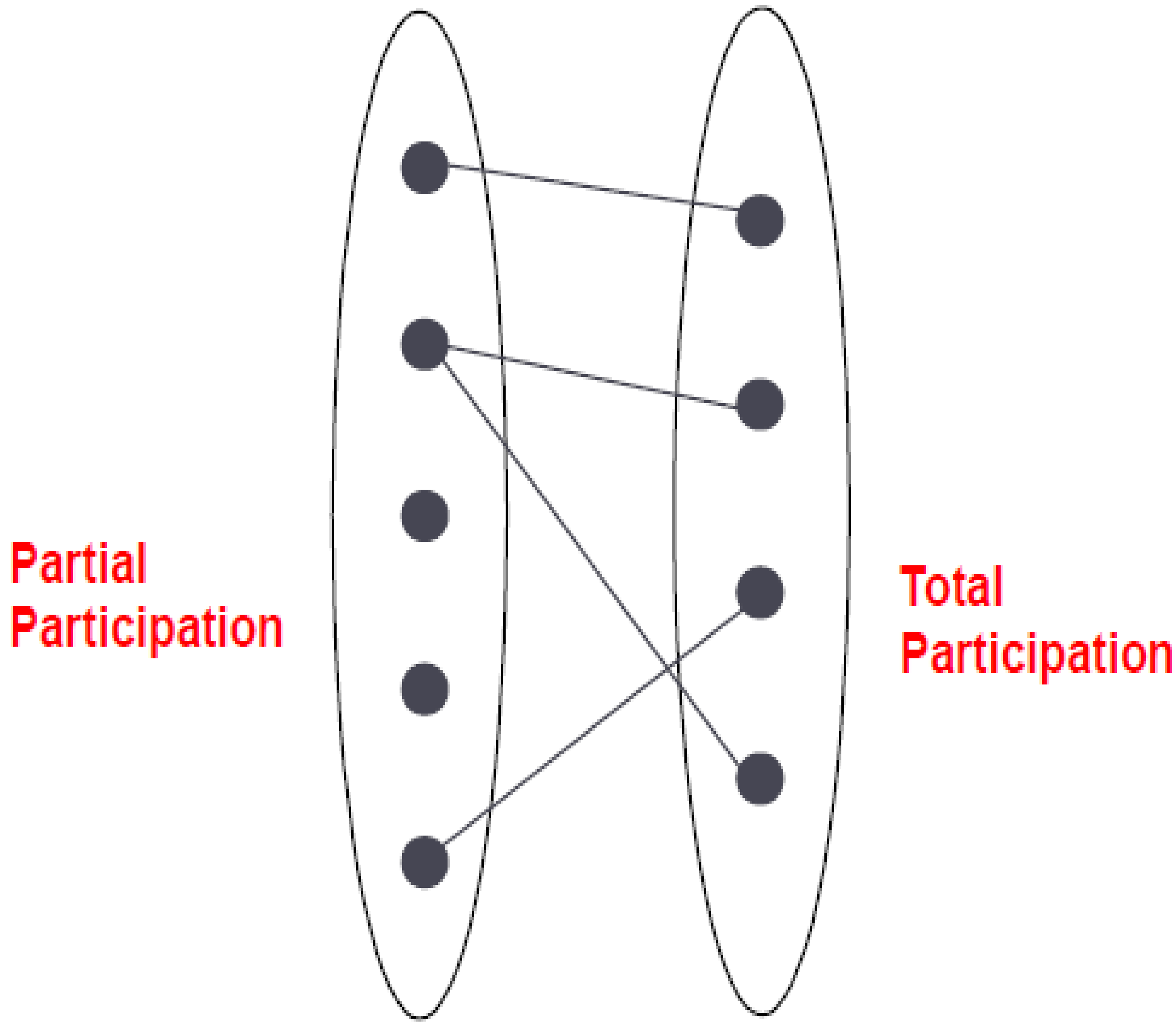
- Many-to-one relationship
  - a loan is associated with several (including 0) customers via *borrower*
  - a customer is associated with at most one loan via *borrower*



- Many-to-many relationship
  - A customer is associated with several (possibly 0) loans via *borrower*
  - A loan is associated with several (possibly 0) customers via *borrower*



Participation Constraints

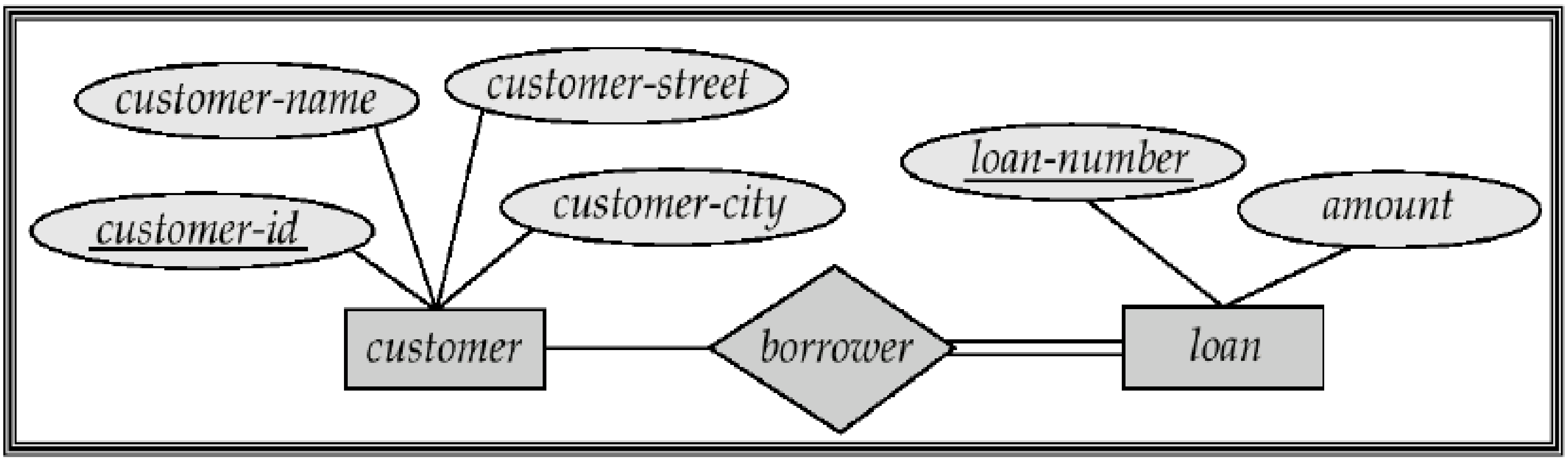


Participation of an Entity Set in a Relationship Set

- Total participation (indicated by double line): every entity in the entity set participates in at least one relationship in the relationship set
  - E.g., participation of *loan* in *borrower* is total

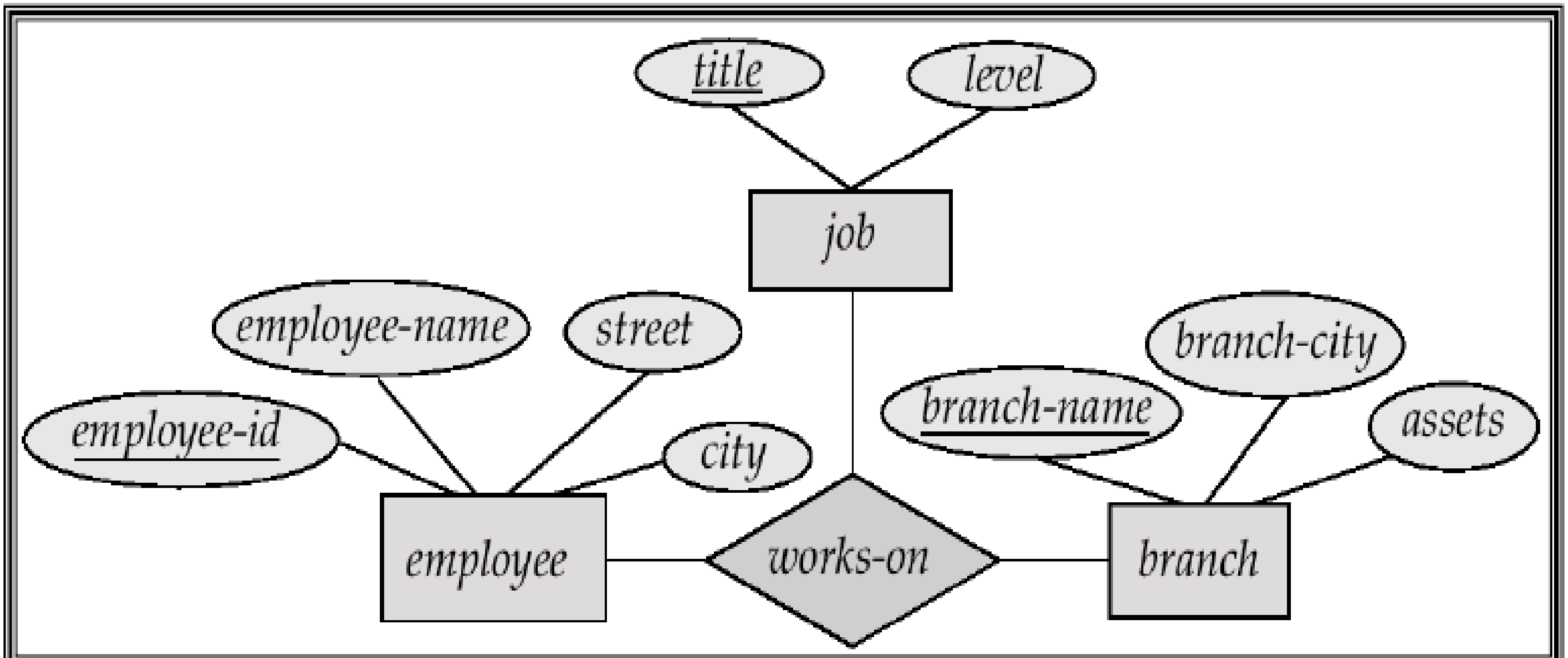
    every loan must have at least a customer associated to it via *borrower*
- Partial participation: some entities may not participate in any relationship in the relationship set
  - E.g., participation of *customer* in *borrower* is partial

    some customers may not have any loans



Ternary Relationship Sets

- Example:
  - Suppose that employees of a bank have jobs (responsibilities) at multiple branches, with different jobs at different branches
  - Then there is a ternary relationship set between entity sets *employee*, *job* and *branch*



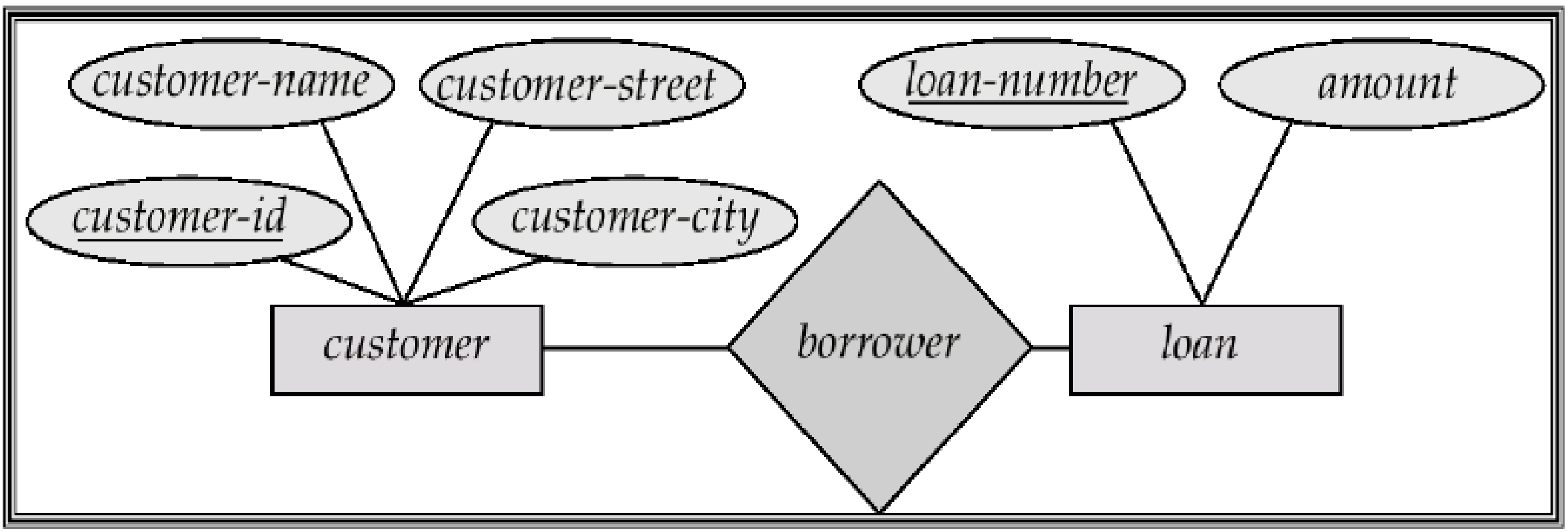
Relationship

A relationship is an association among several entities

- The degree refers to the number of entity sets that participate in the relationship set
  - Relationship sets that involve two entity sets are called binary (or of degree two)
  - Relationship sets among more than two entity sets are called ternary

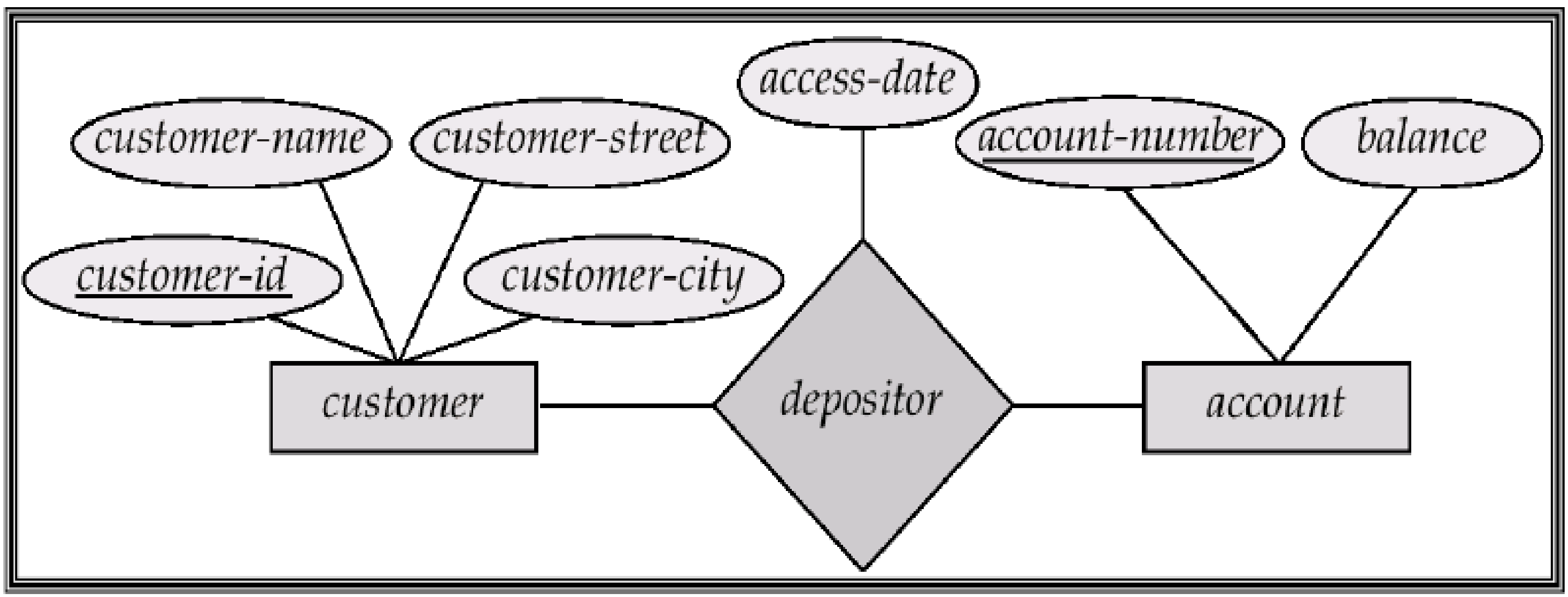
Example of (Binary) Relationship

- *borrower* is a relationship between *customer* and *loan*
  - it means that a customer can be associated with one or more loans and vice versa



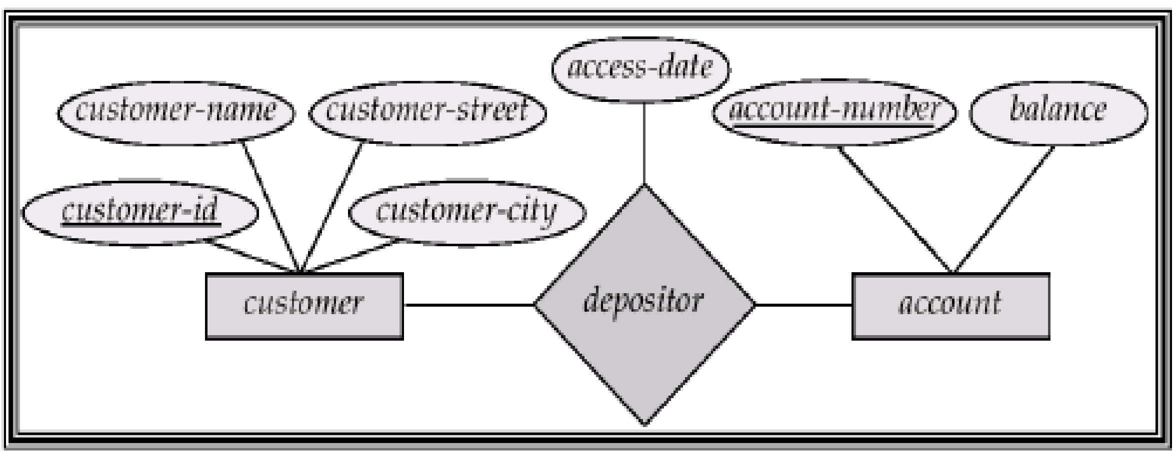
Attributes of Relationships

- *depositor* is a relationship between *customer* and *accounts*
- *access-date* is an attribute of *depositor*

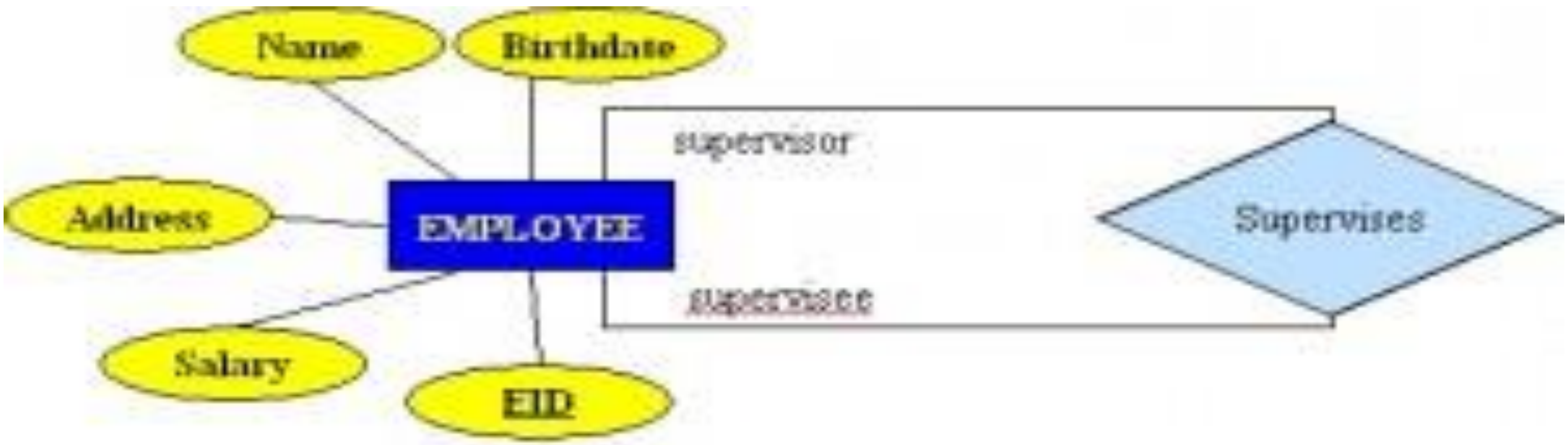


Keys for Relationship Sets

- The combination of primary keys of the participating entity sets forms a super key of a relationship set
  - (*customer-id*, *account-number*) is the super key of *depositor*
  - This means that a pair of entities can have at most one relationship in a particular relationship set
    - Problem: if we wish to track all access dates to each account by each customer, we cannot assume a relationship for each access
    - Solution: use a multivalued attribute for access dates
- Must consider the mapping cardinality of the relationship set when deciding the candidate keys



Unary Relationship



Relational Schema

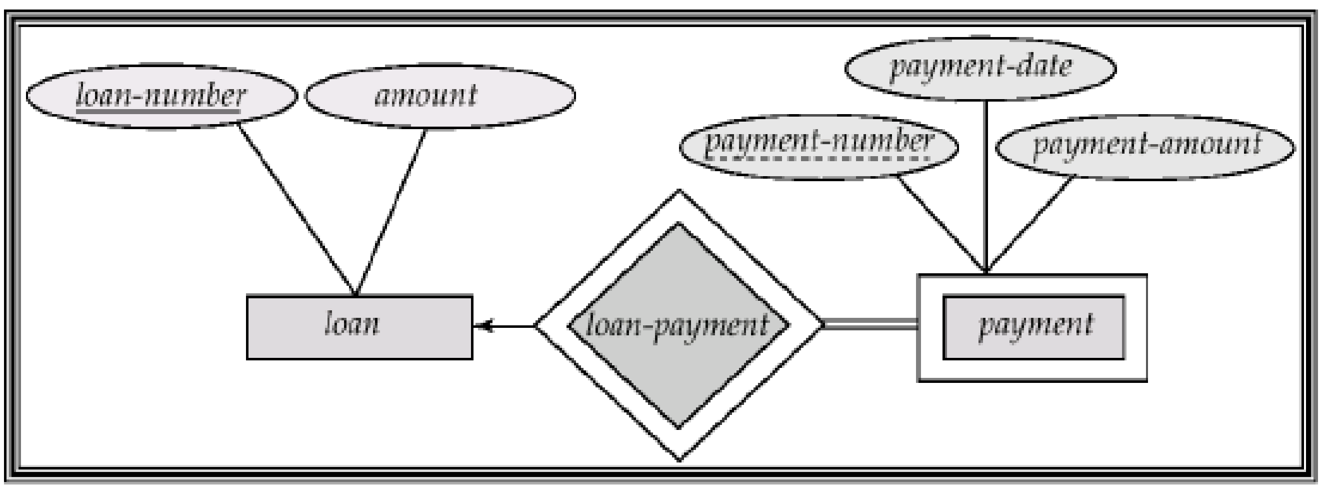
EMPLOYEE (EID, Name, Address, Birthdate, Salary, *Super-EID*)

Weak Entity Sets

- An entity set that does not have a primary key is referred to as a weak entity set
- The existence of a weak entity set depends on the existence of an identifying entity set
  - it must relate to the identifying entity set via a total, one-to-many relationship set from the identifying to the weak entity set
  - this relationship is called identifying relationship
- The discriminator (or partial key) of a weak entity set is the set of attributes that distinguishes the entities in the weak entity set that depend on a particular strong entity set
- The primary key of a weak entity set is formed by
  - the primary key of the (strong) identifying entity set
  - the weak entity set's discriminator

Weak Entity Sets

- A weak entity set is depicted by a double rectangle
- An identifying relationship is depicted by a double diamond
- We underline the discriminator of a weak entity set with a dashed line
- Example:
  - discriminator of *payment*: *payment-number*
  - Primary key for *payment*: (*loan-number*, *payment-number*)



- Another example:
  - A child may not be old enough to have a HKID number
  - Even if he/she has a HKID number, the company may not be interested in keeping it in the database.

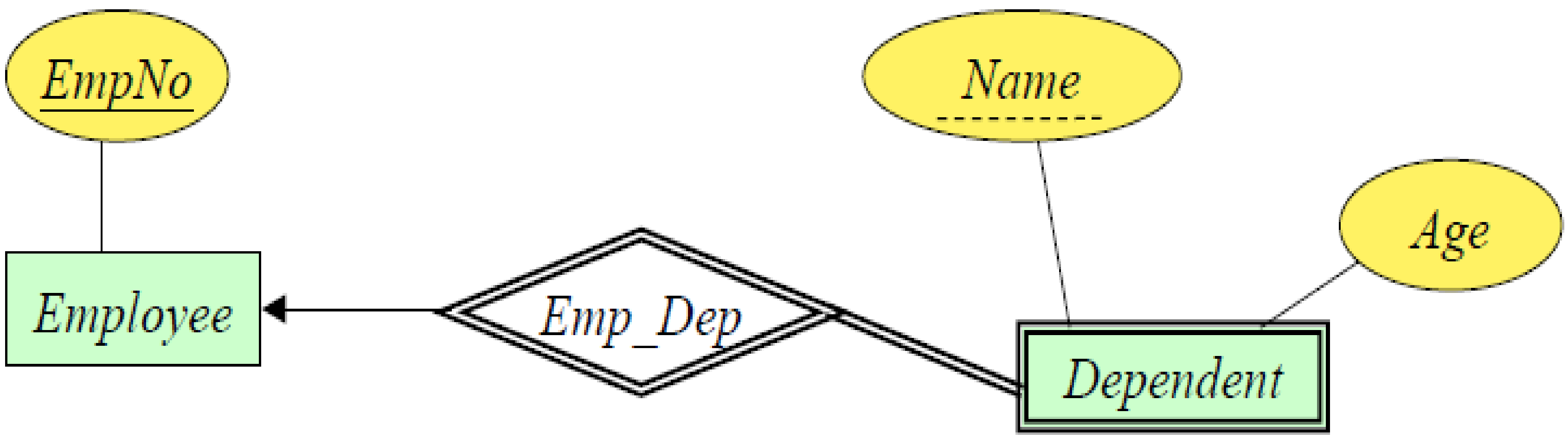
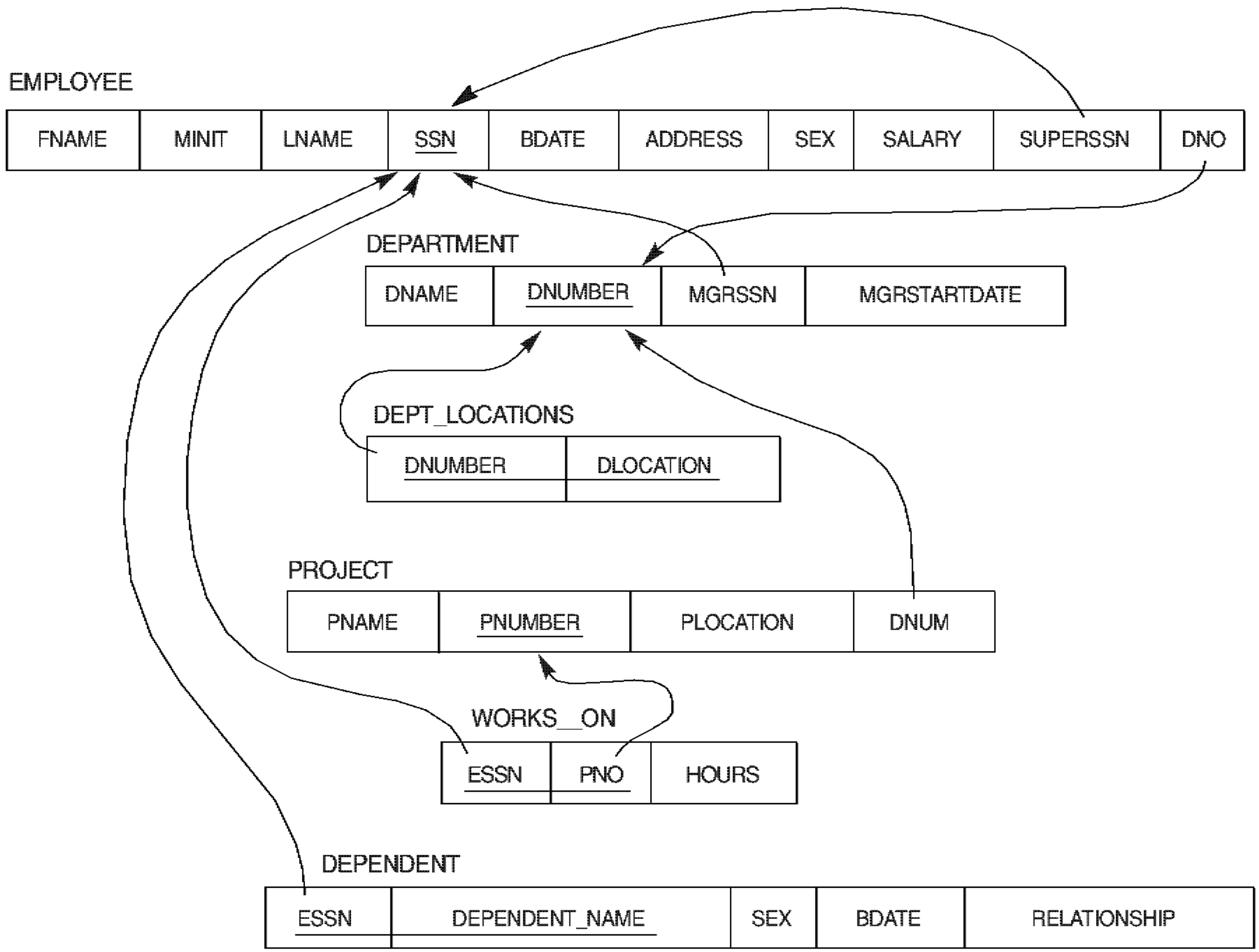
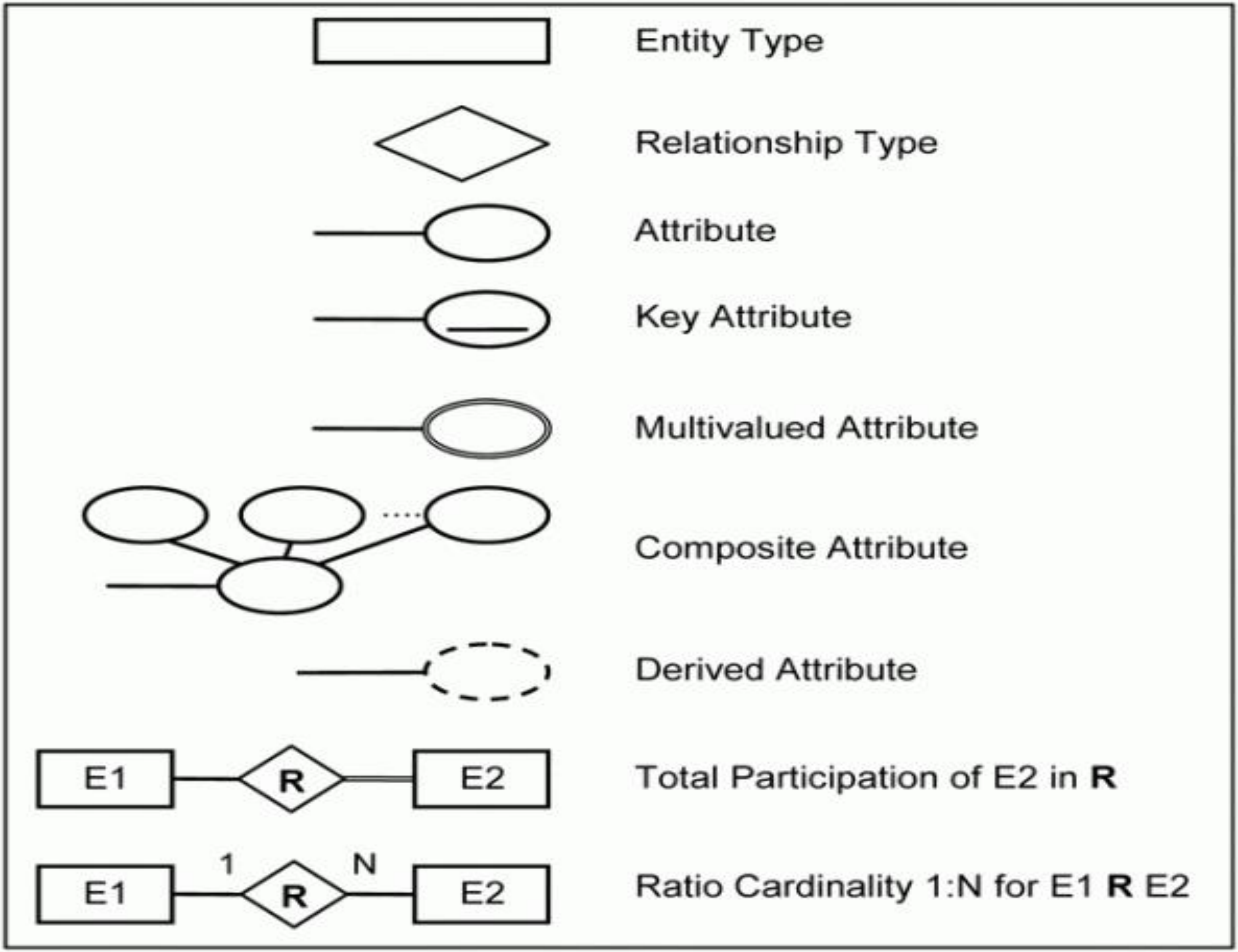


Figure 7.7 Referential integrity constraints displayed on the COMPANY relational database schema diagram.





Entity Type
Relationship Type
Attribute
Key Attribute
Multivalued Attribute
Composite Attribute
Derived Attribute
E1
R
E2
Total Participation of E2 in **R**
1
N
E1
R
E2
Ratio Cardinality 1:N for E1 **R** E2

A banking scenario

- *Banks have customers.*
- *Customers are identified by name, custid, phone number and address.*
- *Customers can have one or more accounts*
- *Accounts are identified by an account number, account type (savings, current) and a balance.*
- *Customers can avail loans.*
- *Loans are identified by loan id, loan type (car, home, personal)*
- *and an amount.*
- *Banks are identified by a name, code and the address of the main office.*
- *Banks have branches.*
- *Branches are identified by a branch number, branch name and an address.*
- *Accounts and loans are related to the banks' branches.*
- *Create an ER diagram for a database to represent this application*

Solution Step 1: Identify the entities

- Bank
- Branch
- Customer
- Account
- Loan

Solution Step 2: Identify attributes of entities

- Bank
  - Name
  - Code
  - Address

Branch
- Branch#
- Branch name
- Address

Customer
- Name
- Custid
- Phone
- Address

Account
- Account number
- Account type
- Balance

Loan
- Loan id
- Loan type
- Amount

Solution Step 3: Identify relationships between entities

- Bank has Branch
- Branch maintains accounts
- Branch offers loans
- Account is held by customer
- Loan is availed by customer

Solution Step 4: Analyze cardinality of relationships

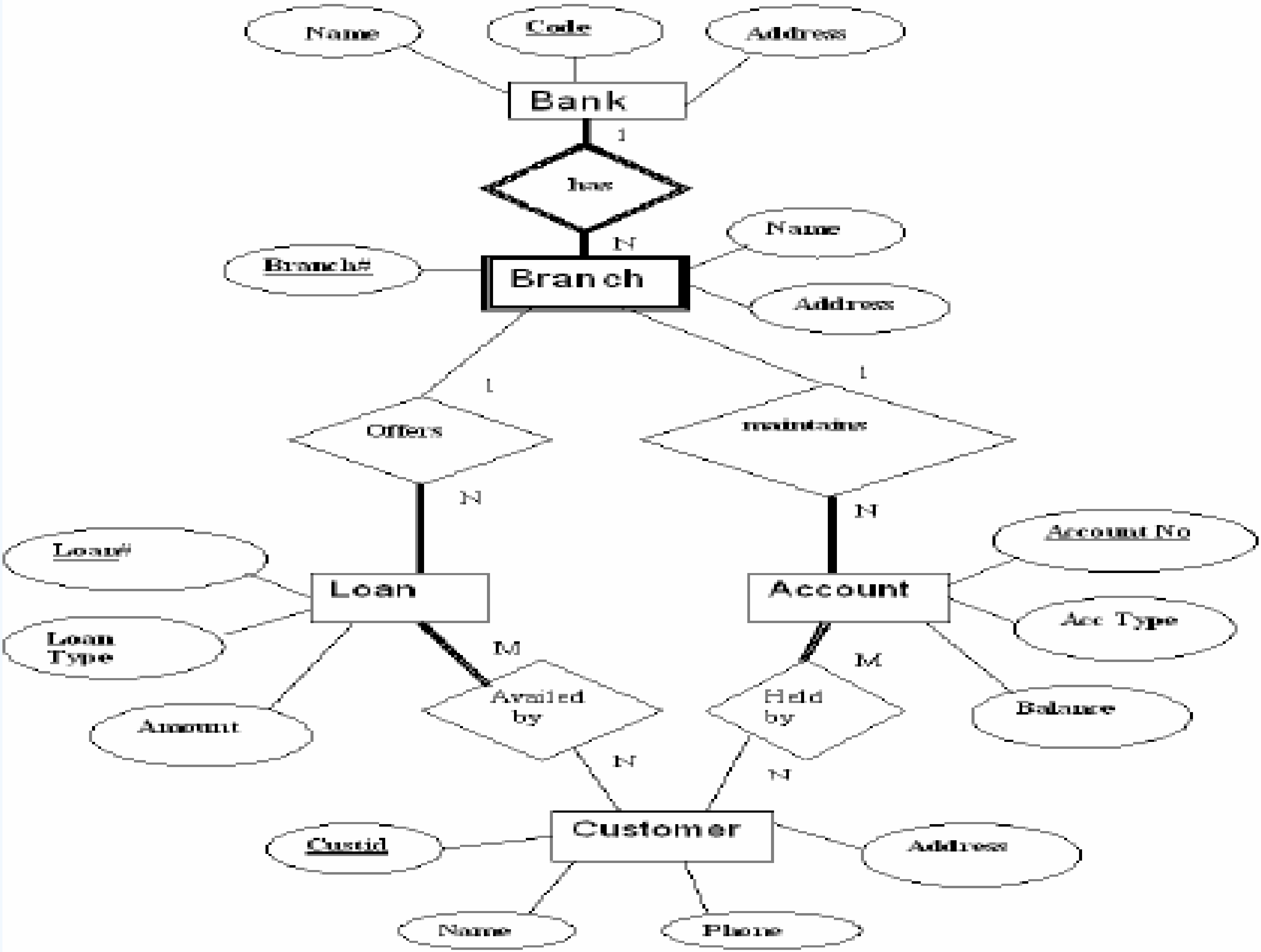
- Bank **has** Branch : A bank has many branches->**1:N**
- Branch **maintains** accounts: One branch maintains many accounts-> **1:N**
- Branch **offers** loans : One branch offers many loans -> **1:N**
- Account is **held by** customer -> **M:N**
- Loan is **availed by** customer ->**M:N**

Solution Step 5: Identify weak entities if any

Branch: Depends on strong entity Bank

Solution Step 6: Identify participation types

- Bank has Branch -> both **total**
- Branch maintains accounts-> Branch :partial Account: Total
- Branch offers loans -> Branch: partial Loan: Total
- Account is held by customer-> both Total
- Loan is availed by customer-> Loan : Total Customer: Partial



Name
Code
Address
Bank
1
has
N
Name
Branch#
Branch
Address
1
1
Offers
maintains
N
N
Account No
Loan#
Loan
Account
Loan Type
Acc Type
M
M
Availed by
Held by
Balance
Amount
N
N
Customer
Custid
Address
Name
Phone

Example

A university consists of a number of departments. Each department offers several courses. A number of modules make up each course. Students enrol in a particular course and take modules towards the completion of that course. Each module is taught by a lecturer from the appropriate department, and each lecturer tutors a group of students

Example - Entities

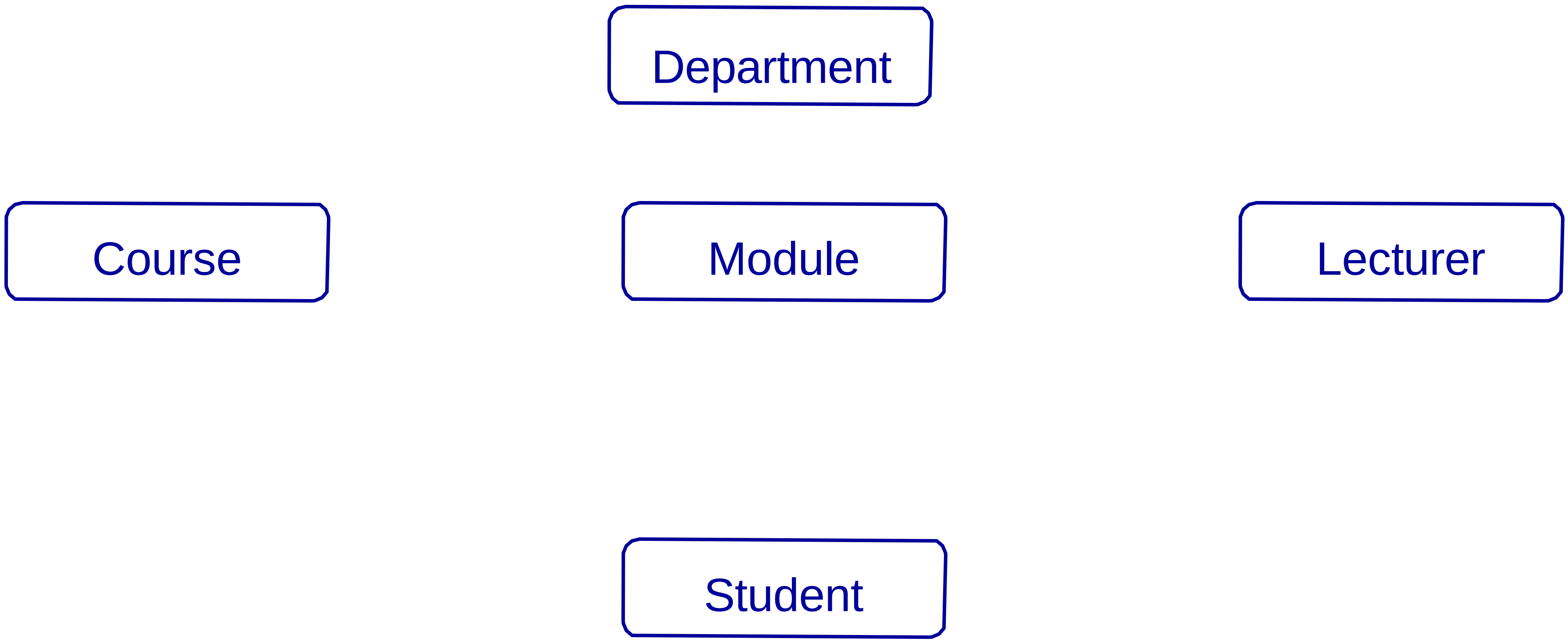
A university consists of a number of **departments**. Each department offers several **courses**. A number of **modules** make up each course. **Students** enrol in a particular course and take modules towards the completion of that course. Each module is taught by a **lecturer** from the appropriate department, and each lecturer tutors a group of students

Example - Relationships

A university consists of a number of departments. Each department **offers** several courses. A number of modules **make up** each course. Students **enrol in** a particular course and **take** modules towards the completion of that course. Each module is **taught by** a lecturer **from the** appropriate department, and each lecturer **tutors** a group of students

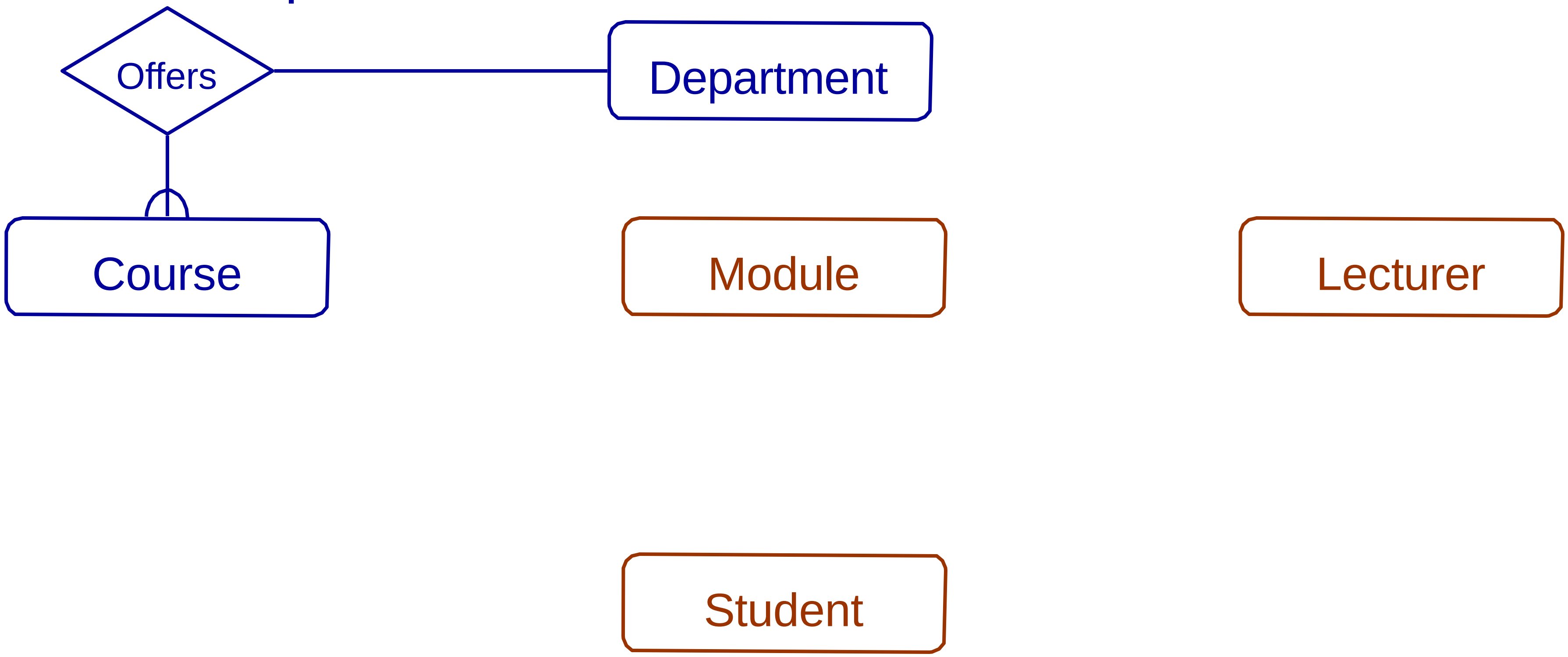
Example - E/R Diagram

Entities: Department, Course, Module, Lecturer, Student



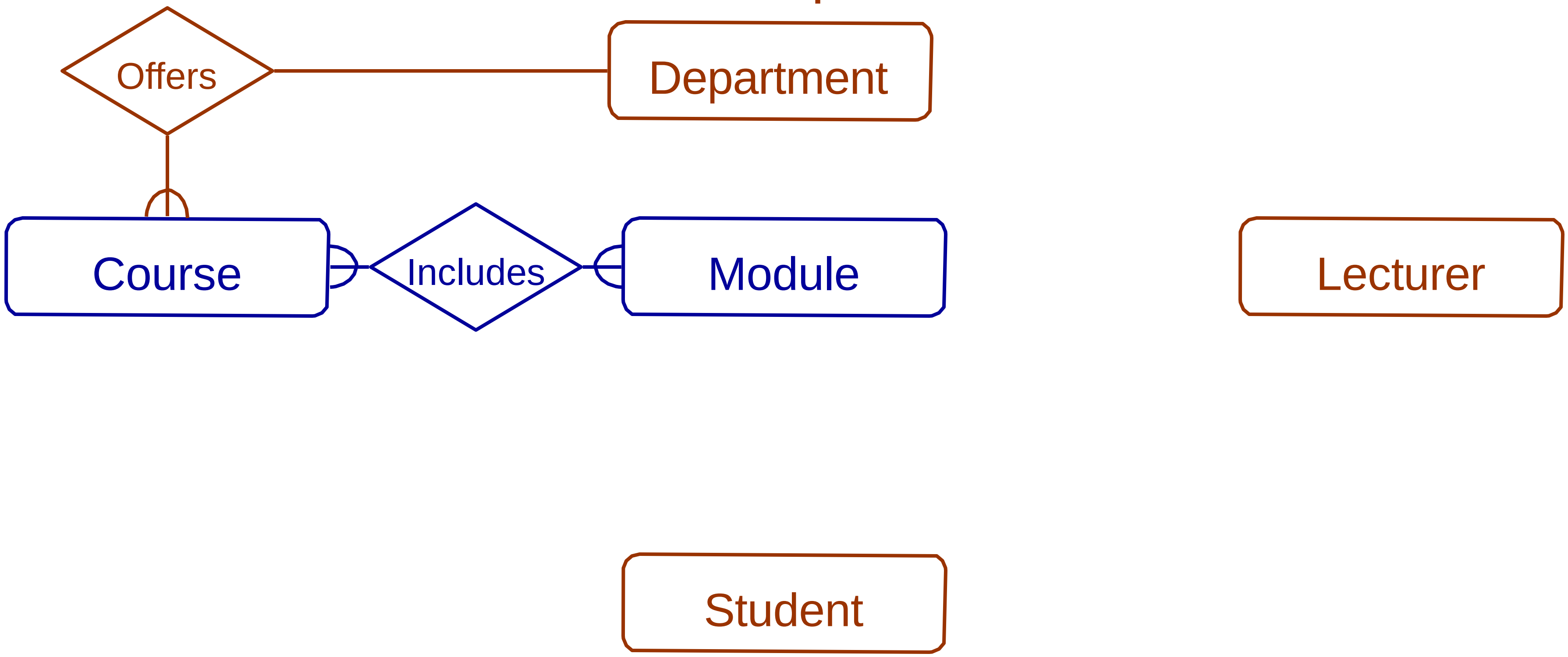
Example - E/R Diagram

Each department offers several courses



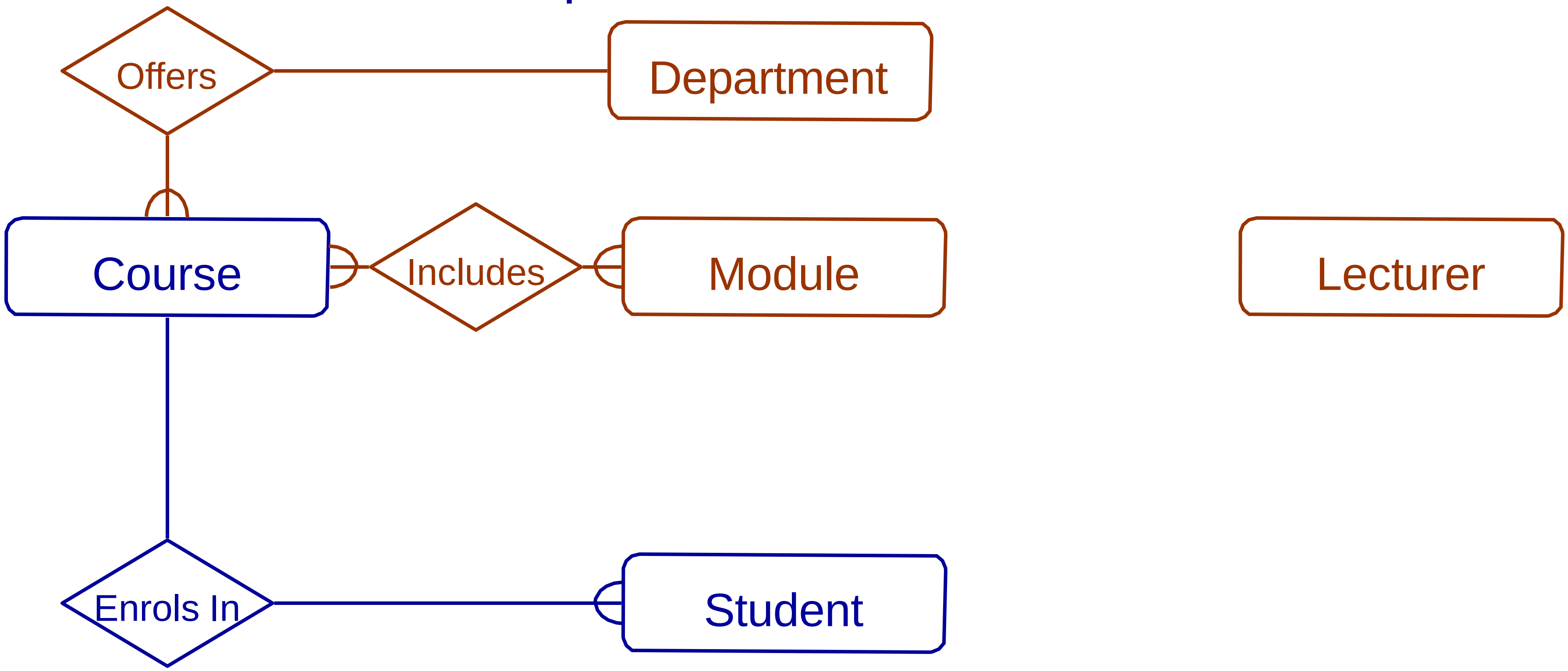
Example - E/R Diagram

A number of modules make up each courses



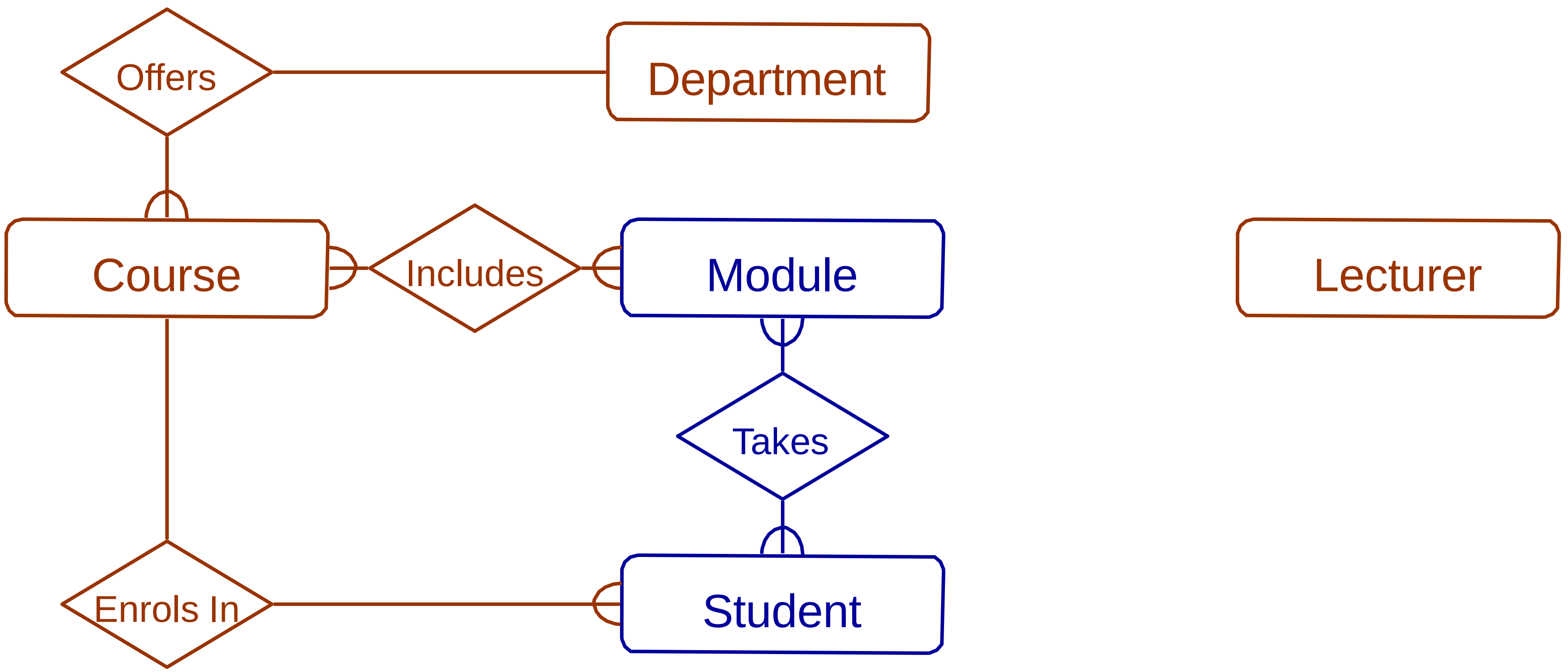
Example - E/R Diagram

Students enrol in a particular course



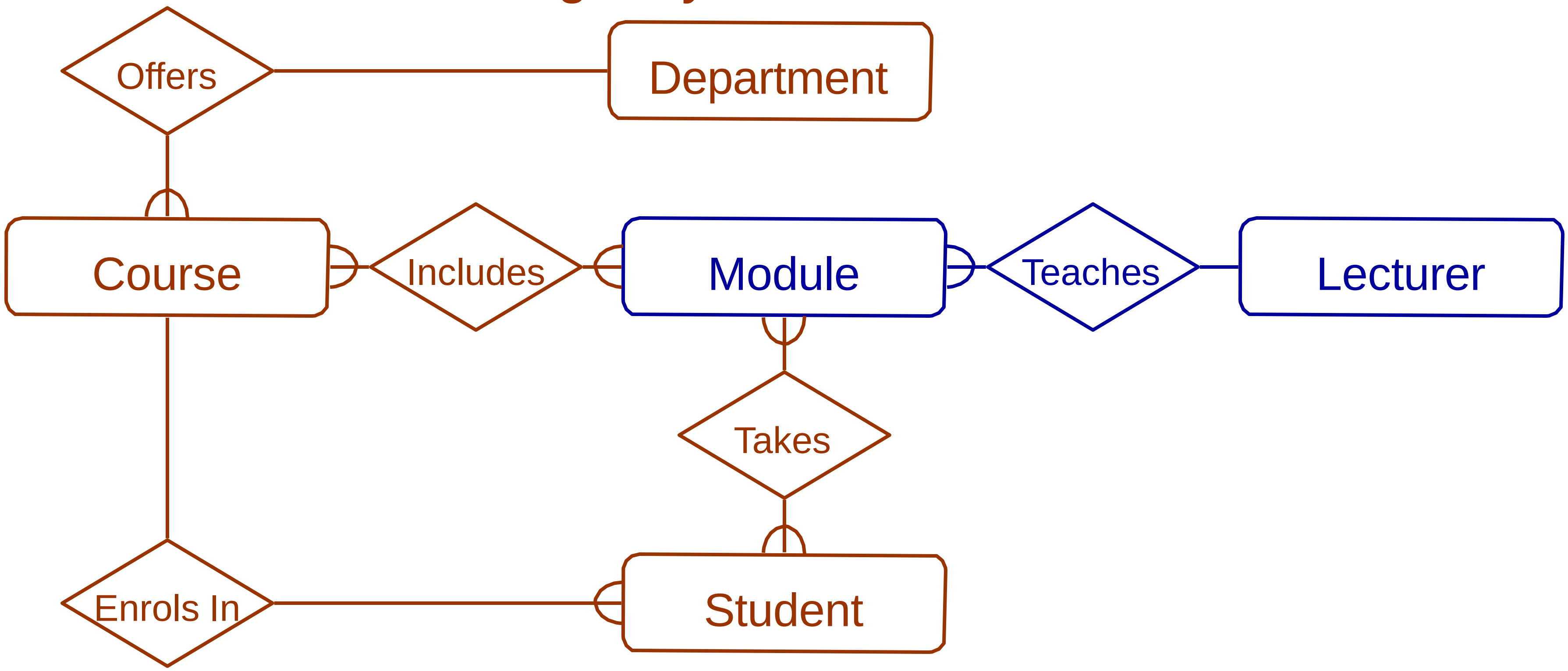
Example - E/R Diagram

Students … take modules



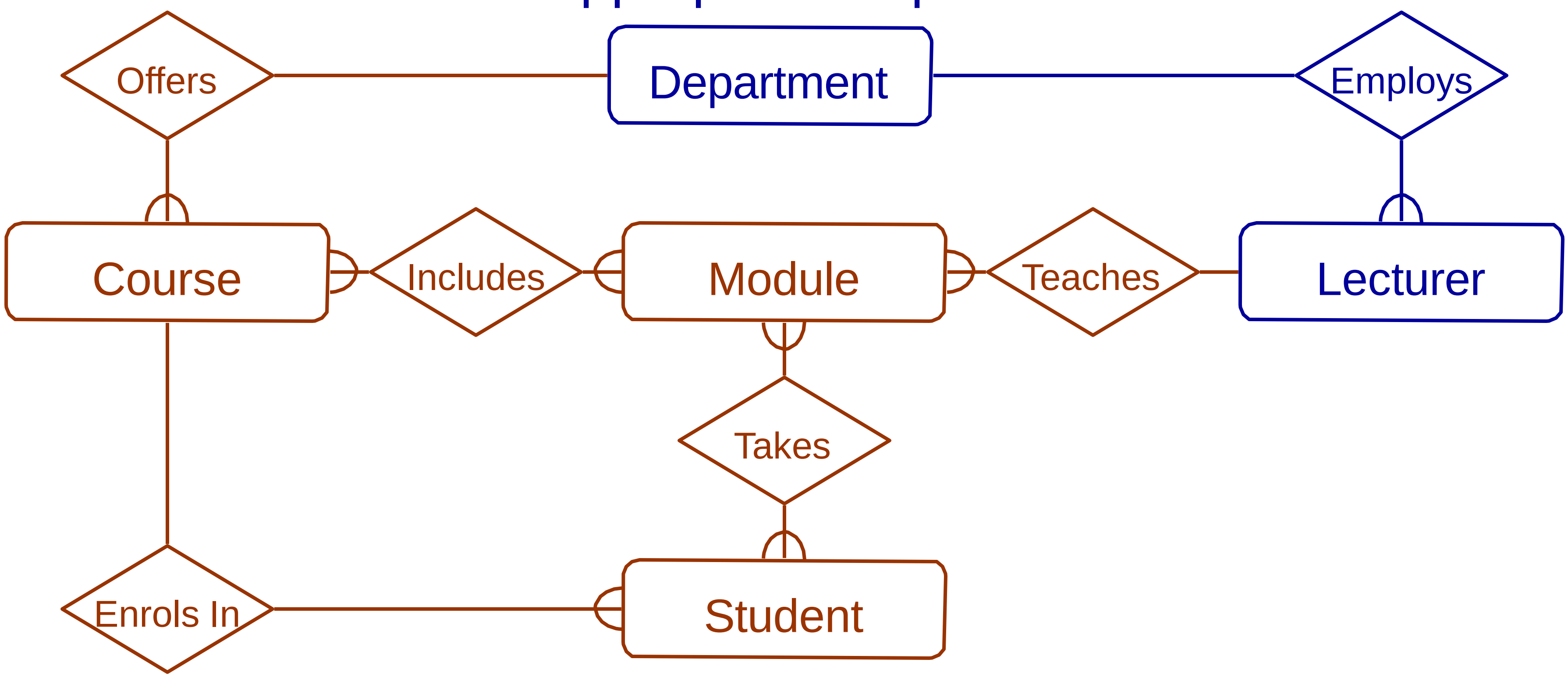
Example - E/R Diagram

Each module is taught by a lecturer



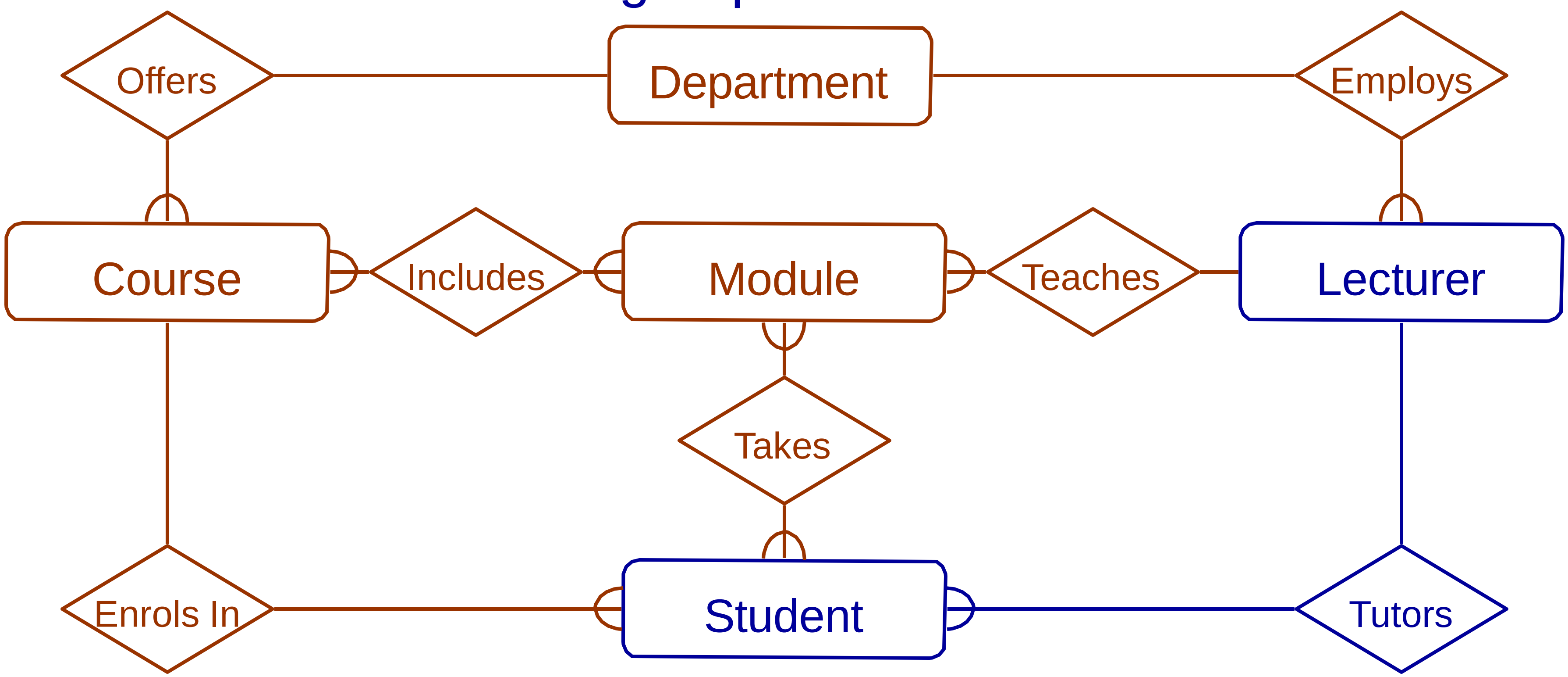
Example - E/R Diagram

a lecturer from the appropriate department

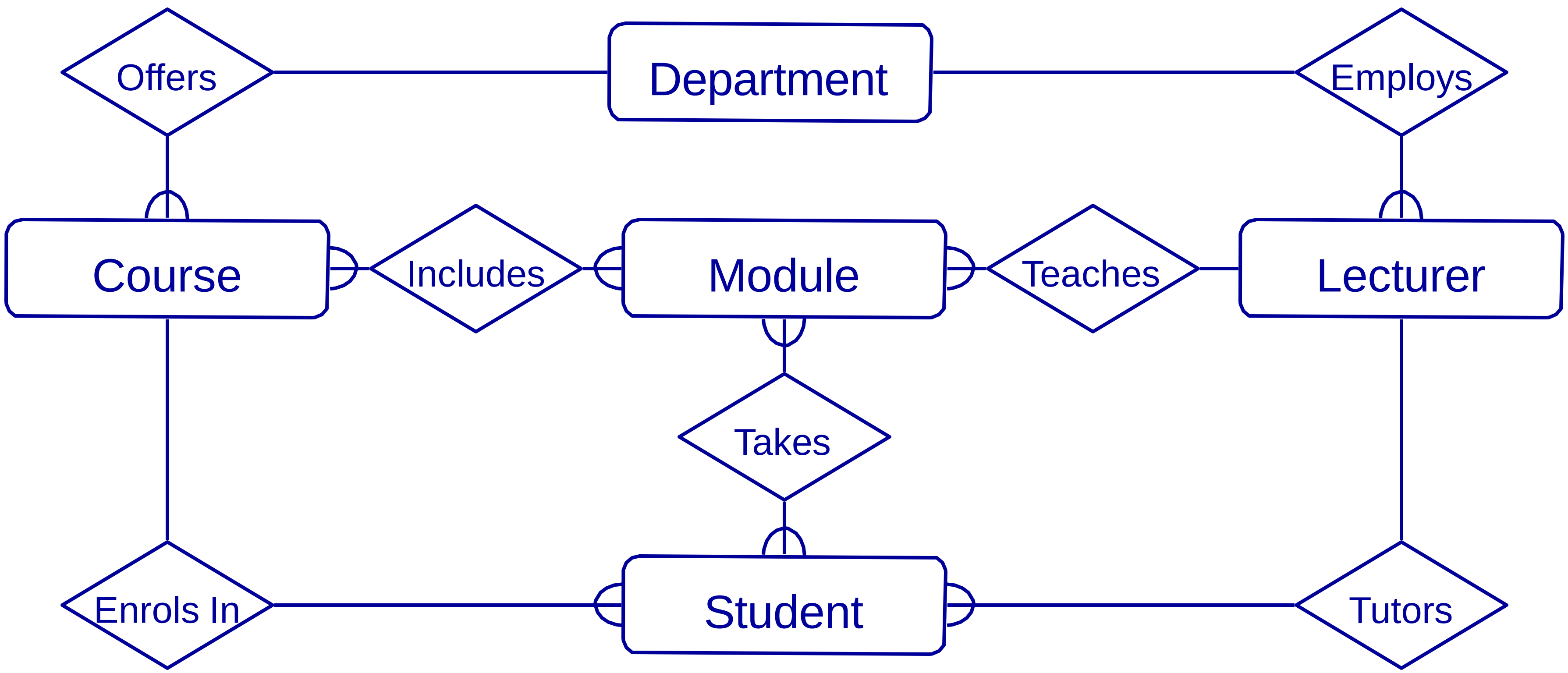


Example - E/R Diagram

each lecturer tutors a group of students



Example - E/R Diagram



Example - 2

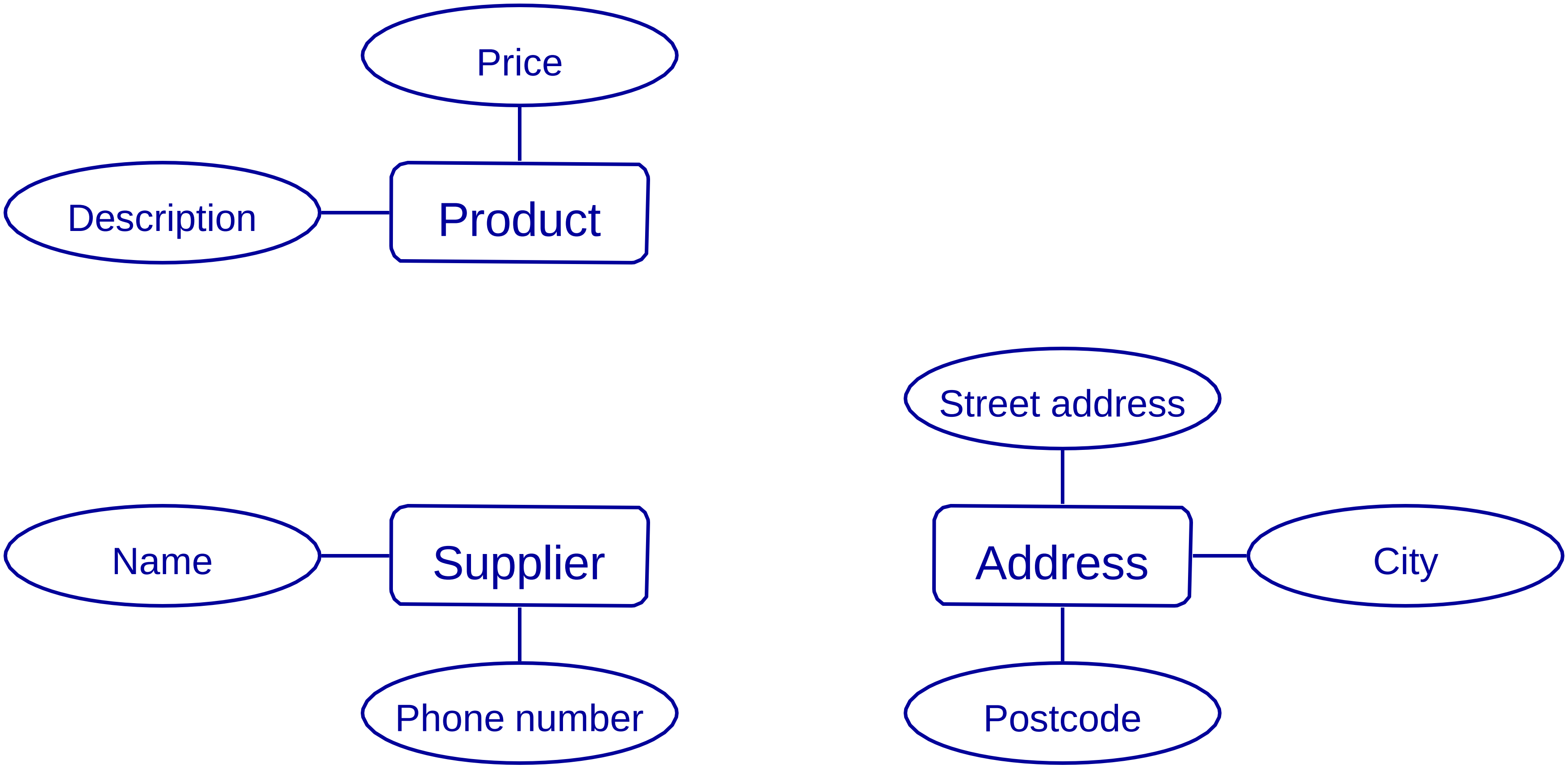
We want to represent information about products in a database. Each product has a description, a price and a supplier. Suppliers have addresses, phone numbers, and names. Each address is made up of a street address, a city, and a postcode.

Example - Entities/Attributes

- Entities or attributes:
    - product
    - description
    - price
    - supplier
    - address
    - phone number
    - name
    - street address
    - city
    - postcode

- Products, suppliers, and addresses all have smaller parts so we can make them entities
- The others have no smaller parts and belong to a single entity

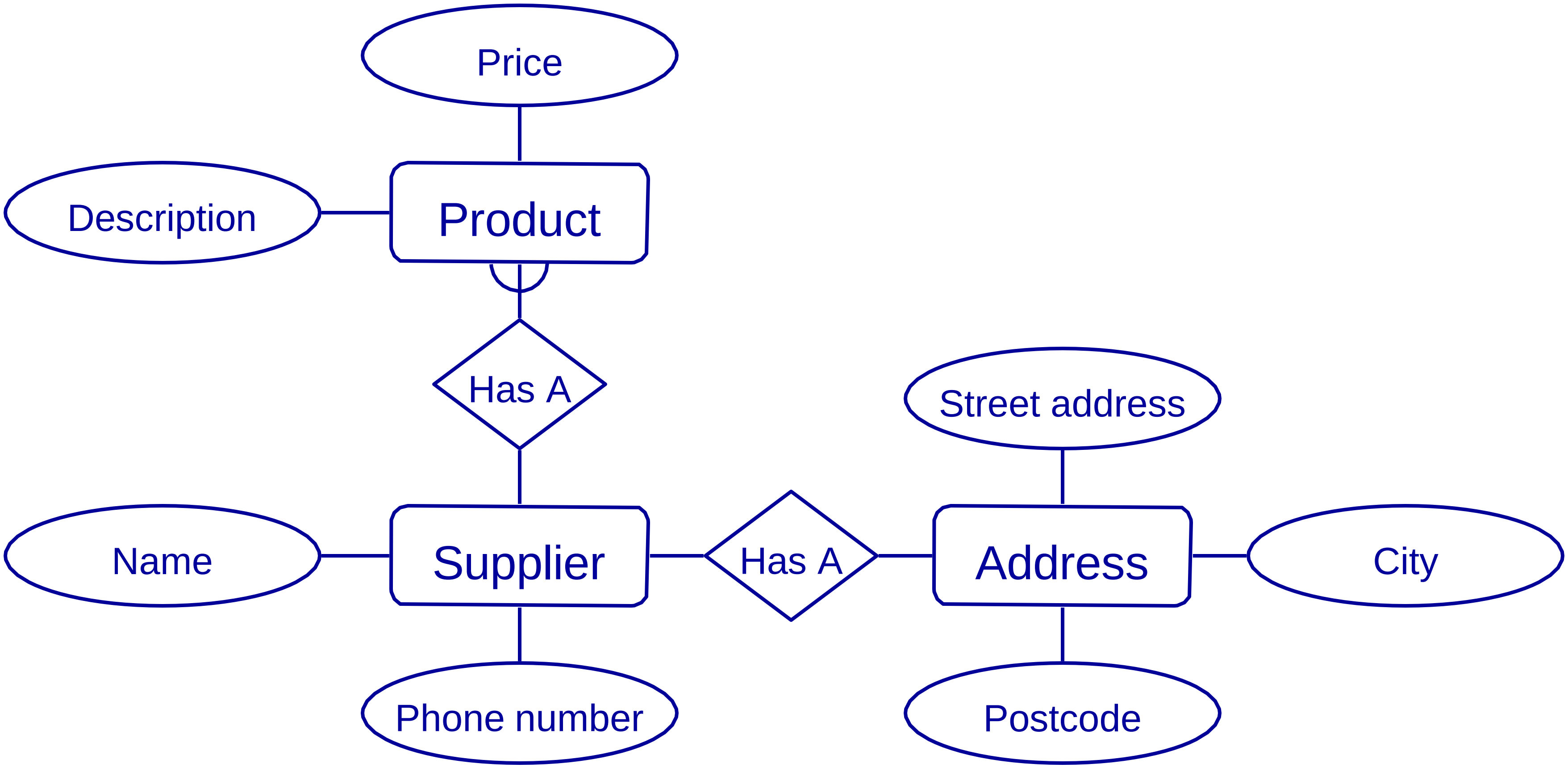
Example - E/R Diagram



Example - Relationships

- Each product has a supplier
  - Each product has a single supplier but there is nothing to stop a supplier supplying many products
  - A many to one relationship
- Each supplier has an address
  - A supplier has a single address
  - It does not seem sensible for two different suppliers to have the same address
  - A one to one relationship

Example - E/R Diagram

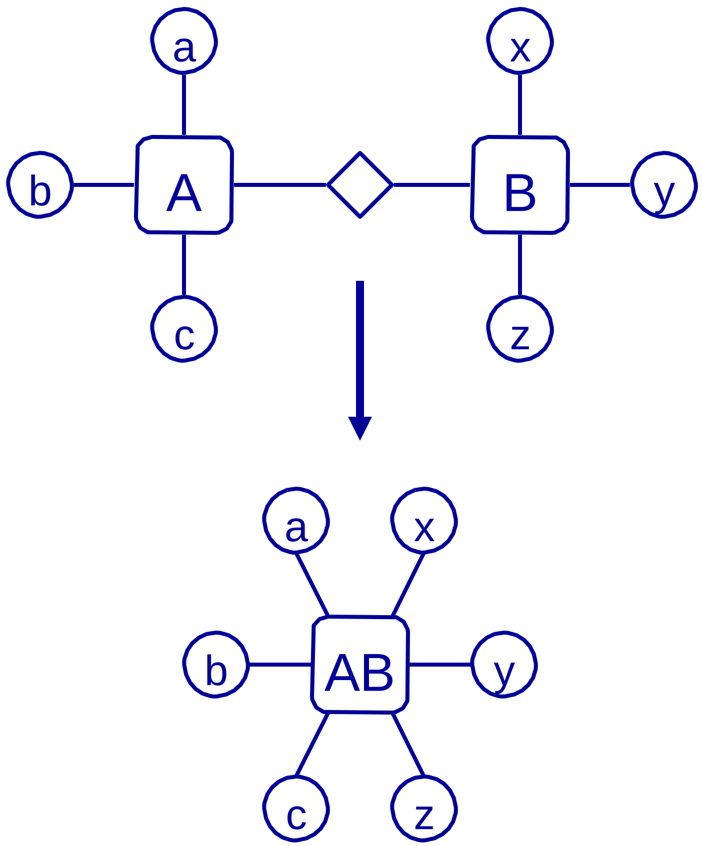


One to One Relationships

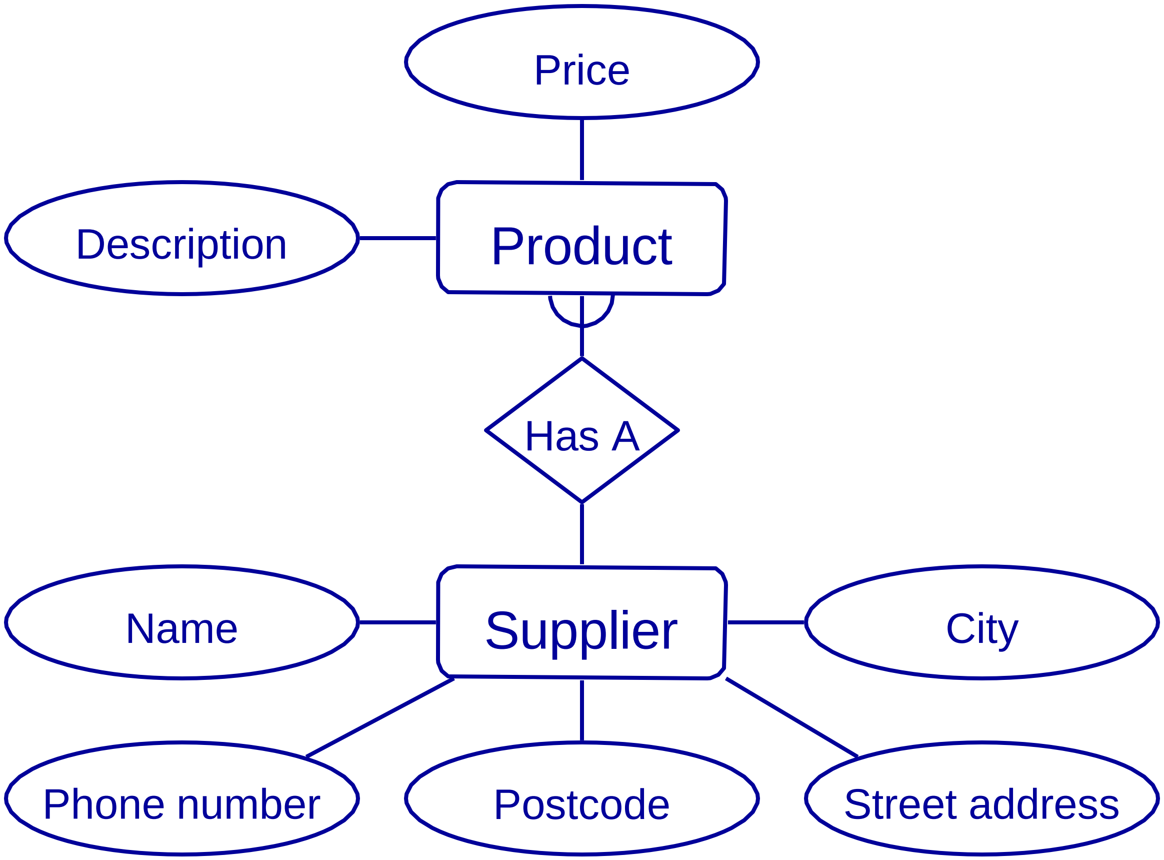
- **Some** relationships between entities, A and B, **might** be redundant if
  - It is a 1:1 relationship between A and B
  - Every A is related to a B and every B is related to an A
- Example - the supplier-address relationship
  - Is one to one
  - Every supplier has an address
  - We don’t need addresses that are not related to a supplier

Redundant Relationships

- We can merge the two entities that take part in a redundant relationship together
  - They become a single entity
  - The new entity has all the attributes of the old one



Example - E/R Diagram

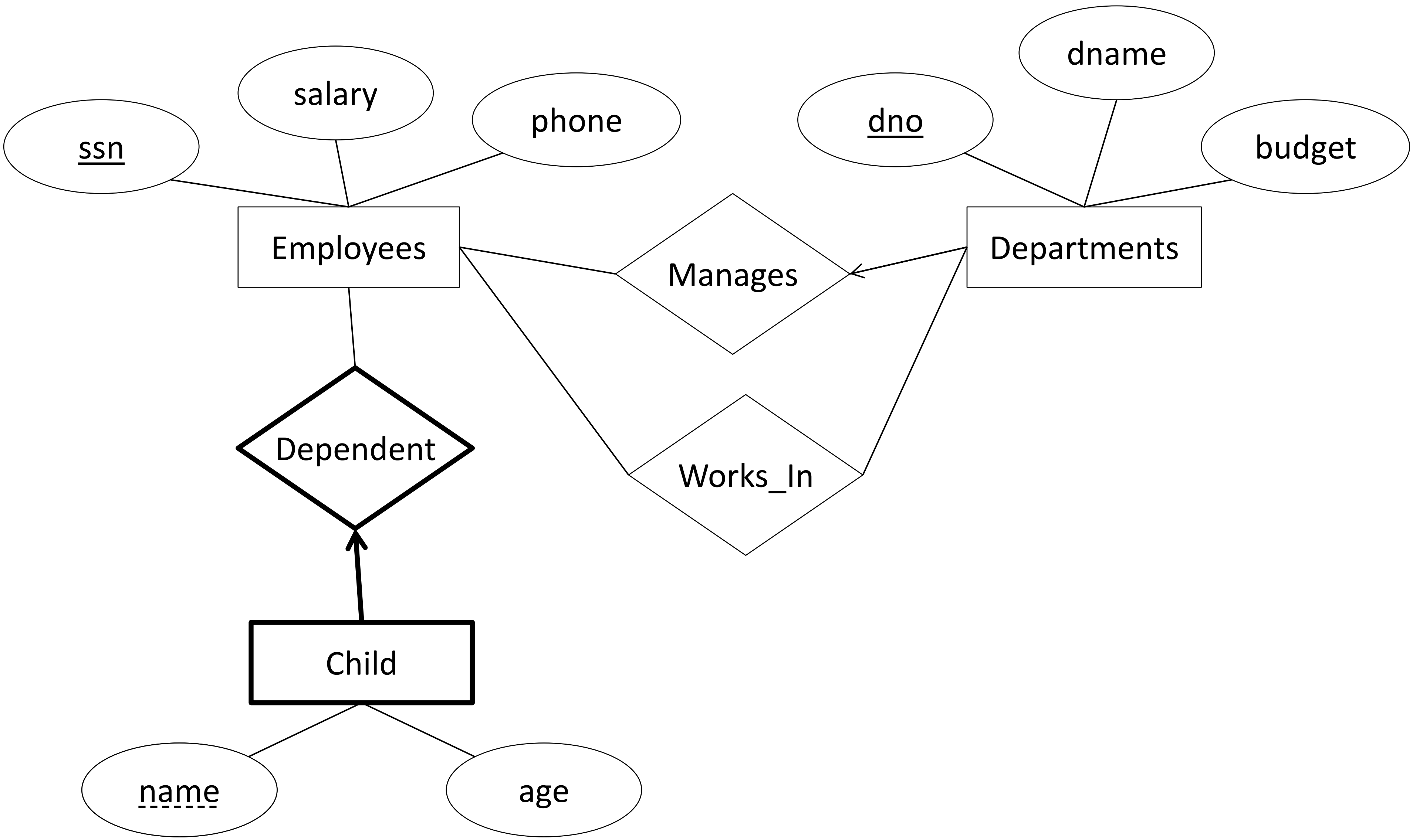


Example -3

Employees work in departments; each department is managed by an employee; a child must be identified uniquely by name when the parent (who is an employee; assume that only one parent works for the company) is known. We are not interested in information about a child once the parent leaves the company.

Draw an ER diagram that captures this information.

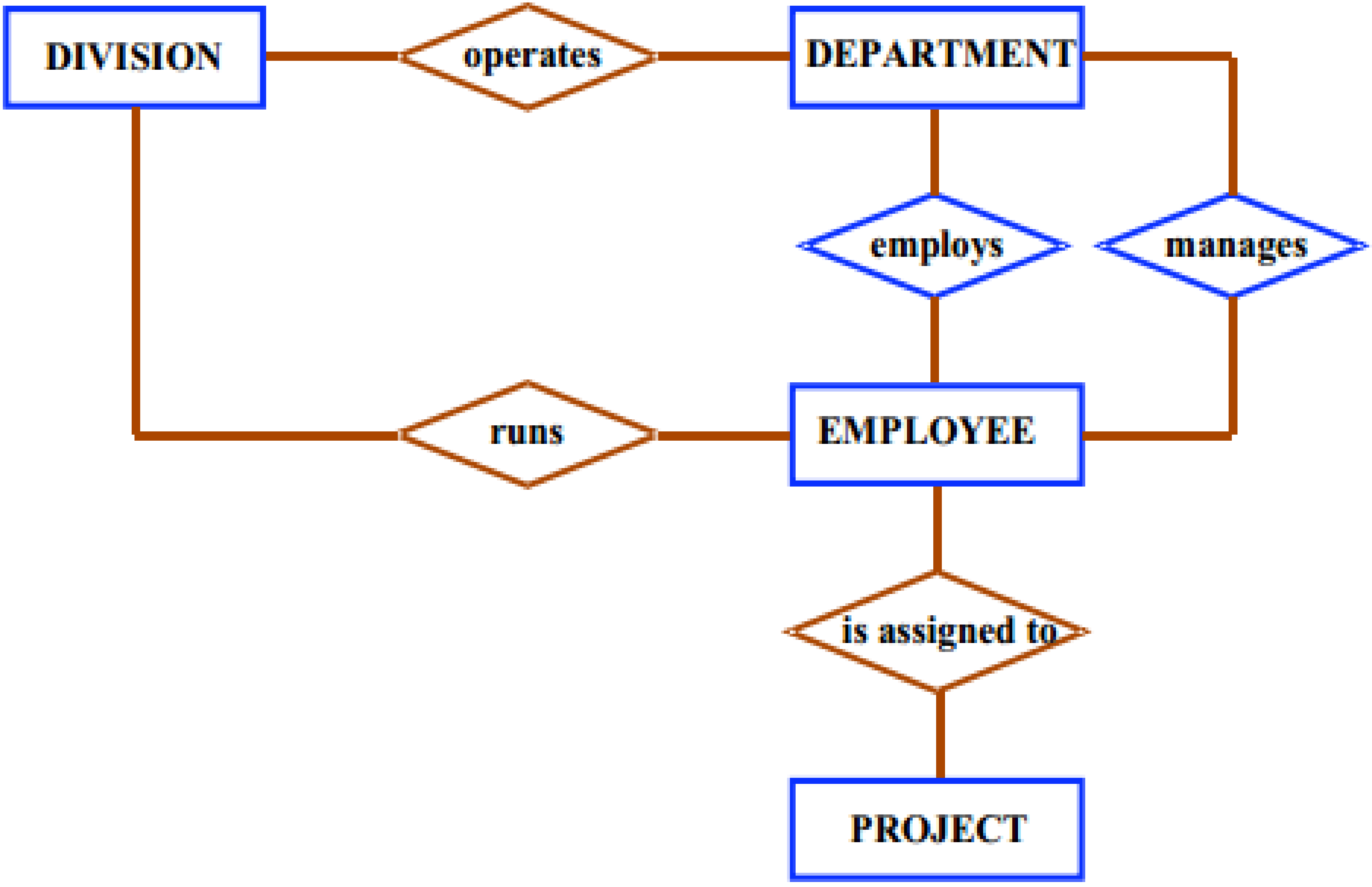
Solution



ssn
salary
phone
dno
dname
budget
Employees
Manages
Departments
Dependent
Works_In
Child
name
age

Example -4

A department employs many employees, but each employee is employed by one department.
A division operates many departments, but each department is operated by one division.
An employee may be assigned to many projects, and a project may have many employees assigned to it.
A project must have at least one employee assigned to it.
One of the employees manages each department, and each department is managed by only one employee.
One of the employees runs each division, and each division is run by one employee.


DIVISION
operates
DEPARTMENT
employs
manages
runs
EMPLOYEE
is assigned to
PROJECT