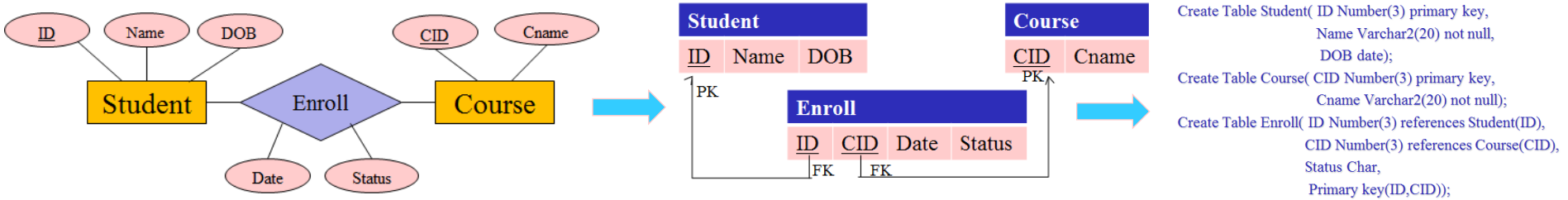
Module2 : Data Modeling using the Entity-Relationship(ER) model

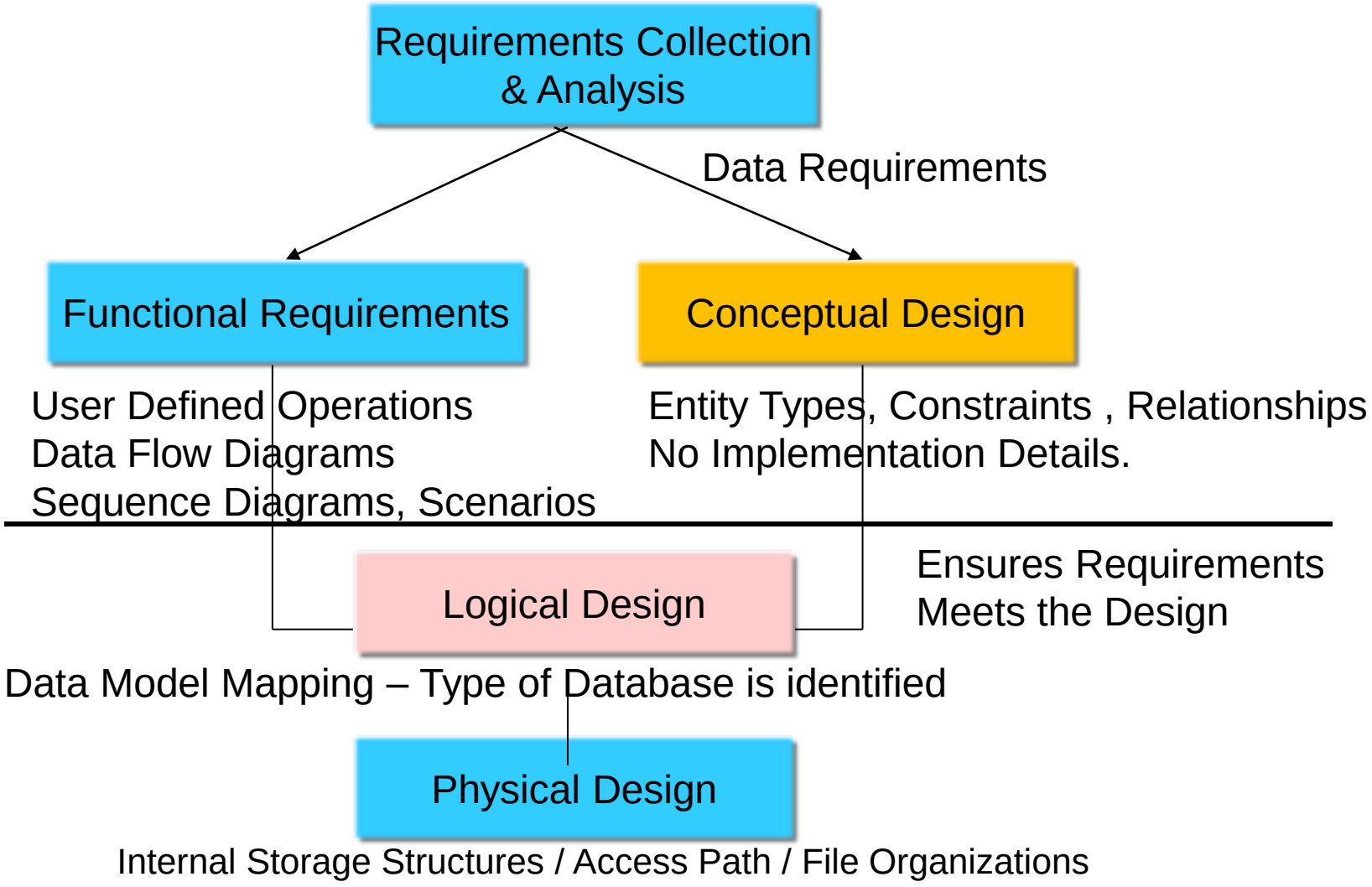


Dr. Aruna M G
Associate Professor
Dept. of AI&ML, DSCE

Contents

- **Data Modelling using the Entity-Relationship(ER) model:**
- Using High-Level Conceptual Data Models for Database Design, A sample Database Application, Entity types, Entity Sets, Attributes, and Keys, Relationship Types, Relationship Sets, Roles and Structural Constraints, Weak Entity types, Refining the ER Design, ER Diagrams, Naming Conventions and Design Issues, Relationship Types of Degree Higher than two, Relational Database Design using ER-to-Relational Mapping.

Database Design Phases…



Overview of Database Design Process

- Requirement Analysis
- Conceptual Design
- Logical Design
- Schema Refinement : (Normalization)
- Physical Database Design and Tuning

Database Design Steps

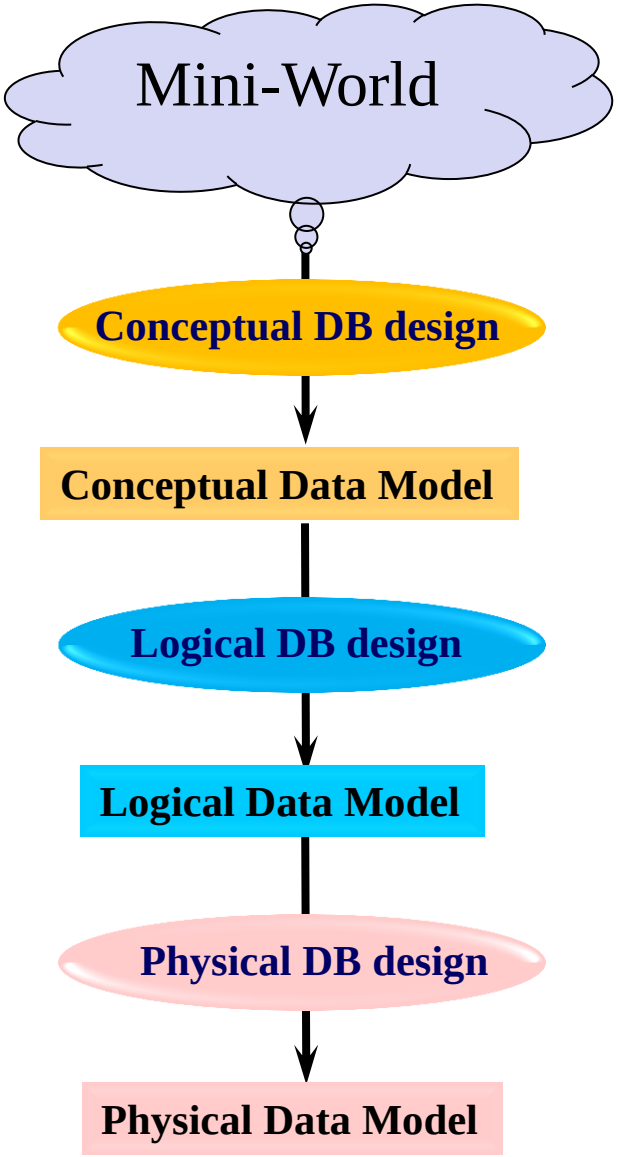
Entity-relationship Model

Typically used for conceptual database design

Three Levels of Modeling

Relational Model

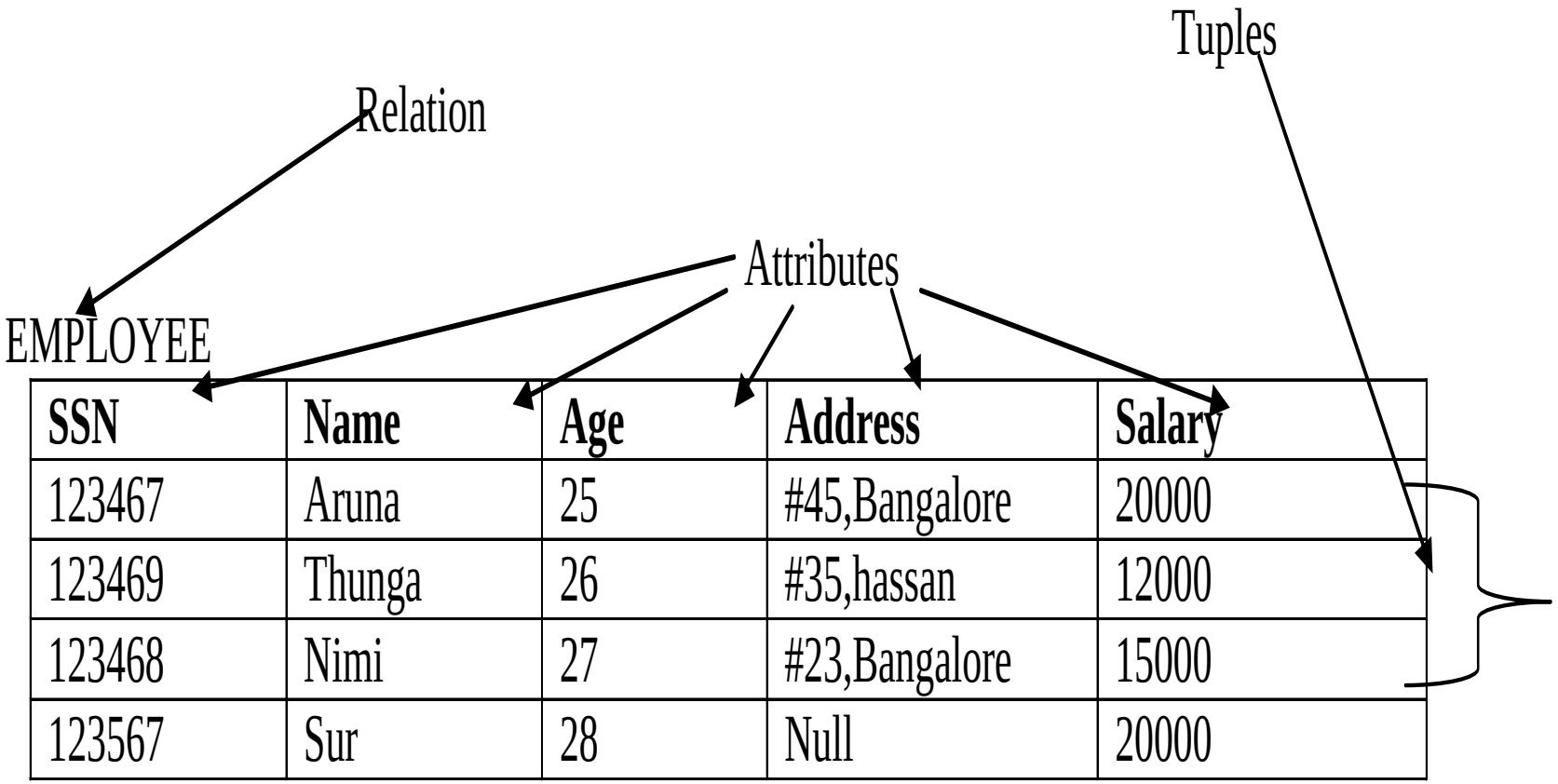
Typically used for logical database design



Relation

- A relational database is made up of a collection of tables or relation.

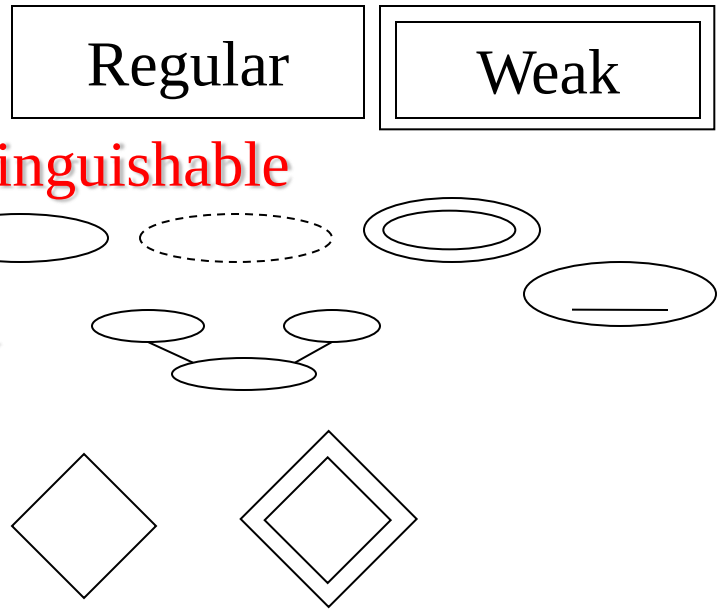
Common terms	DBMS	RDBMS
1. Database	Table	Database
2. Table	Table	Relation
3. Column	Field	Attribute
4. Row	record	tuple



SSN	Name	Age	Address	Salary
123467	Aruna	25	#45,Bangalore	20000
123469	Thunga	26	#35,hassan	12000
123468	Nimi	27	#23,Bangalore	15000
123567	Sur	28	Null	20000

Figure2.3Employee Relation

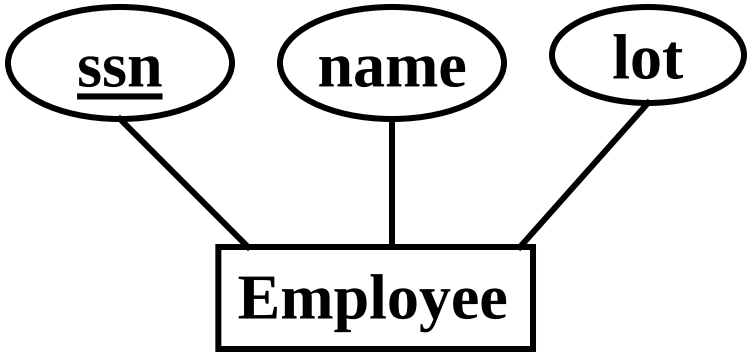
E-R Modeling Concepts



- **Entity**
  - is anything that exists and is distinguishable
- **Attribute**
  - properties that describe an entity
- **Relationship**
  - an association between entities

- **ER Model (Entity Relationship Model)**
- The ER model describes data as entities, relationships, and attributes

Entity, Entity Type, Entity Instance, Entity Set



Entity Type

Entity Instance

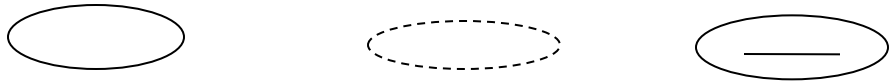
SSN	NAME	LOT
123- 22- 3666	Attishoo	48
231- 31- 5368	Smiley	22
131- 24- 3650	Smethurst	35

Entity Set

```
CREATE TABLE Employees
(ssn CHAR (11),
name CHAR (20),
lot INTEGER,
PRIMARY KEY (ssn))
```

Attributes

- Entity types have ***Attributes*** (or properties) which associate each entity with a value from a *domain* of values for that attribute



- **Attribute Type:**
- It is the property of entity type instance is called attribute type

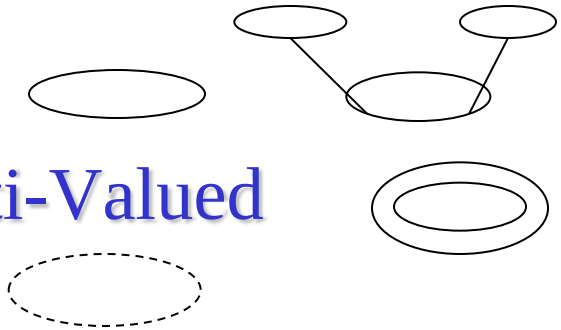
- **Attribute Instance:**
- It is the property of entity instance is called attribute instance.
- –A specific entity will have a value for each of its attributes.

For example a specific employee entity may have

- Name='John Smith', SSN='123456789', Address ='731, Fondren, Houston, TX', Sex='M', BirthDate='09-JAN-55‘

Types of Attributes

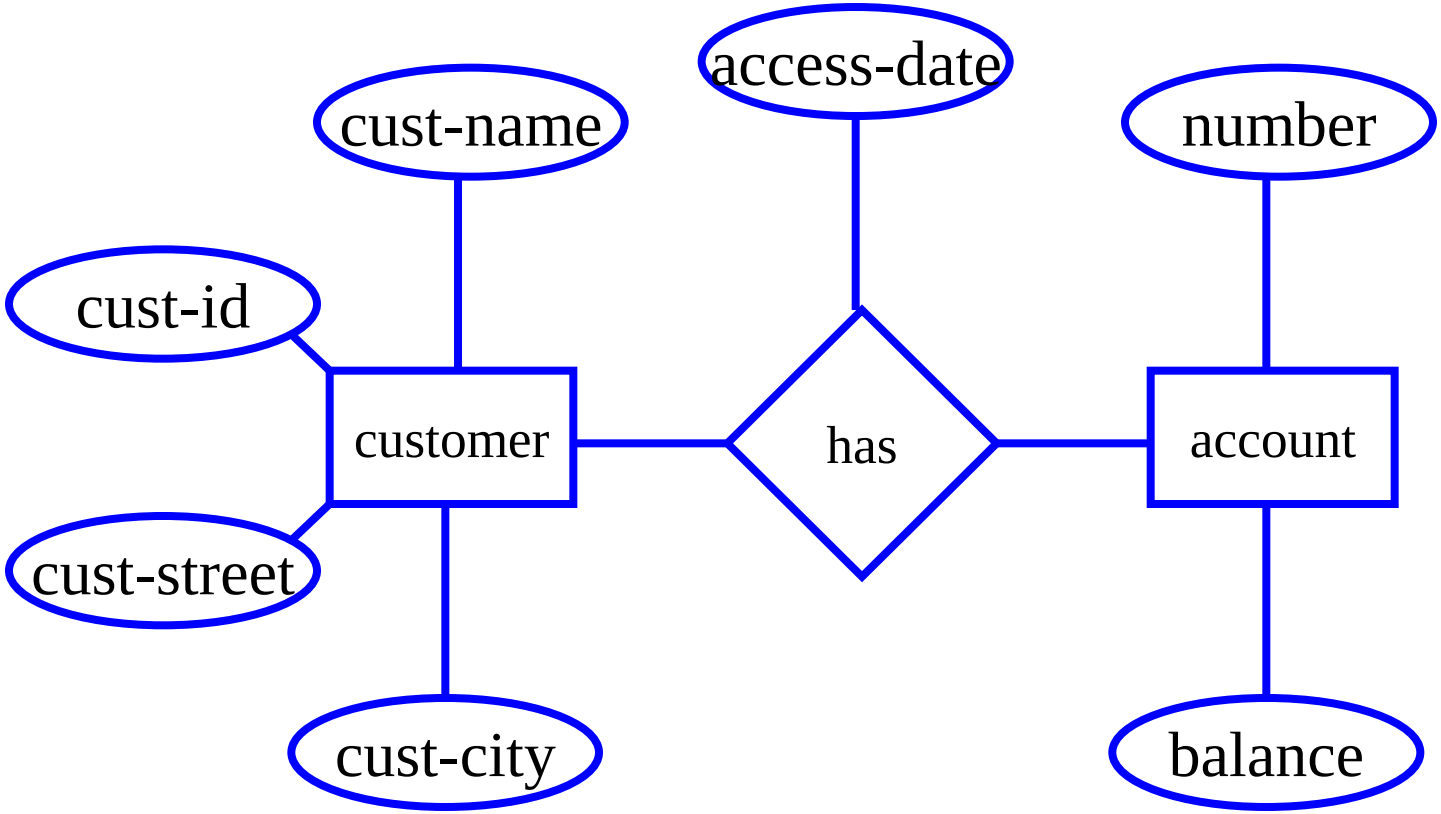
- Attributes can be
  - Simple v/s Composite
  - Single Valued v/s Multi-Valued
  - Stored v/s Derived
  - Key



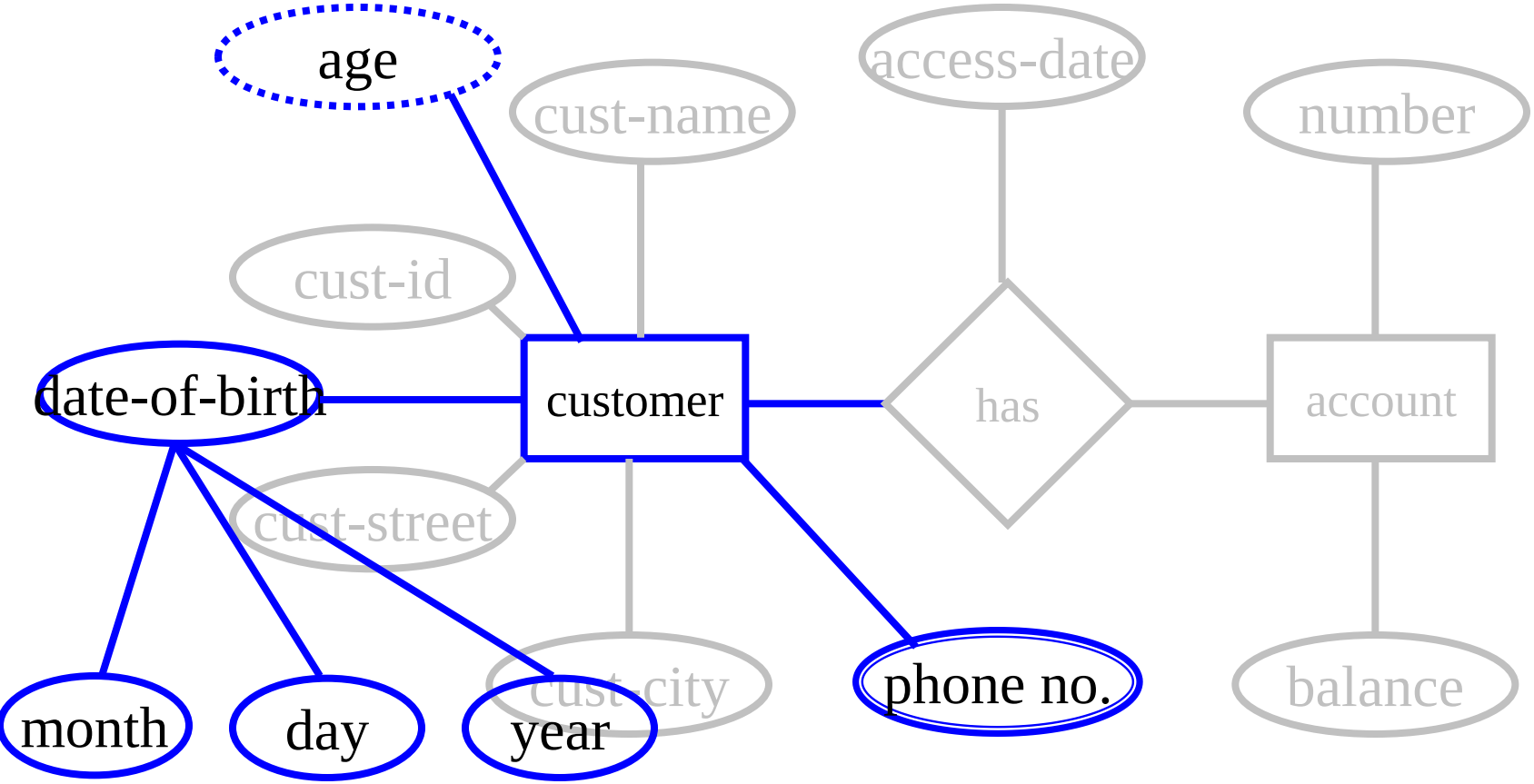
- Example:
  - simple (atomic) e.g. Surname; date of birth
  - composite e.g. address (street, town, postcode)
  - multi-valued e.g. phone number
  - complex nested , multi-valued and composite
  - Stored / derived e.g. D.O.B. ; age
  - Key e.g. SSN
- Relationship types can also have attributes!

Types of Attributes :

Simple, Stored, Single Valued



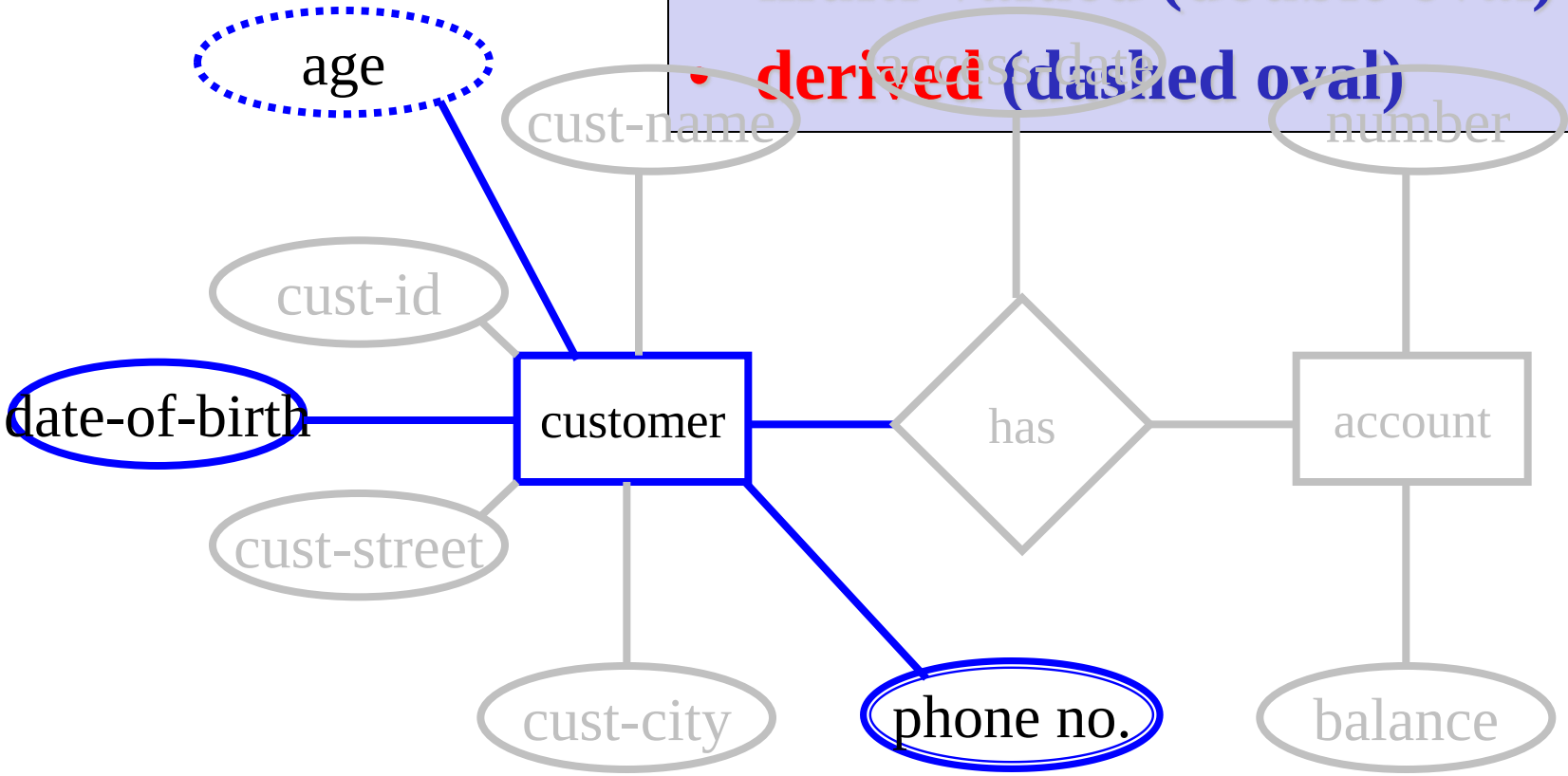
Types of Attributes



Composite Attribute

Types of Attributes

- **multi-valued (double oval)**
- **derived (dashed oval)**



Complex

- The **composite and multi-valued attributes** may be nested arbitrarily to any number of levels although this is rare.
- Or
- The combination of composite and multivalued attribute is also called complex attribute.
- Composite attribute between denoted by parentheses ( ) and multivalued attributes between braces { }.
- Example1: PreviousDegrees of a STUDENT is a composite multi-valued attribute denoted by {PreviousDegrees (College, Year, Degree, Field)}.
- Example2:

{AddressPhone({Phone(AreaCode,PhoneNumber)},
Address(StreetAddress(Number,Street,ApartmentNumber),
City,State,Zip)) }

Null Attirbute

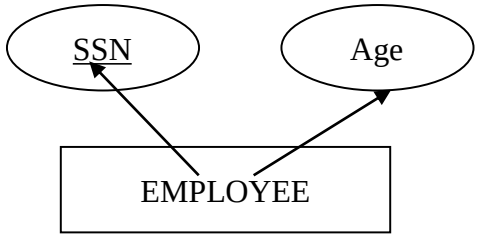
- An entity may not have any applicable value for an attribute. A special value is used in such situation called NULL.
- Example: PhoneNo or Degree attribute of EMPLOYEE entity.

The unknown category of Null is classified into two cases:

1. The first case arises when it is know that the attribute value exists but is missing.
2. The second case, when it is not known whether the attribute value exist or not.

Key Attributes

- Key = set of attributes identifying individual entities or relationships
- An entity type has an attribute whose values can be used to identify each entity uniquely is called a key attribute.
- Keys are one of the basic requirements of a relational database model. It is widely used to identify the tuples(rows) uniquely in the table. We also use keys to set up relations amongst various columns and tables of a relational database.



Different Types of Database Keys

- Candidate Key
- Primary Key
- Super Key
- Alternate Key
- Foreign Key
- Composite Key

Entity Keys

- ***Super Key***
  - any set of attributes that can distinguish entities
- ***Candidate Key***
  - a minimal Super Key
    - Can't remove any attribute and preserve key-ness
      - {cust-id, age} not a super key
      - {cust-name, cust-city, cust-street} is
        - » assuming cust-name is not unique
- ***Primary Key***
  - One of the Candidate Keys chosen as the **key** by DBA
  - Underlined in the ER Diagram

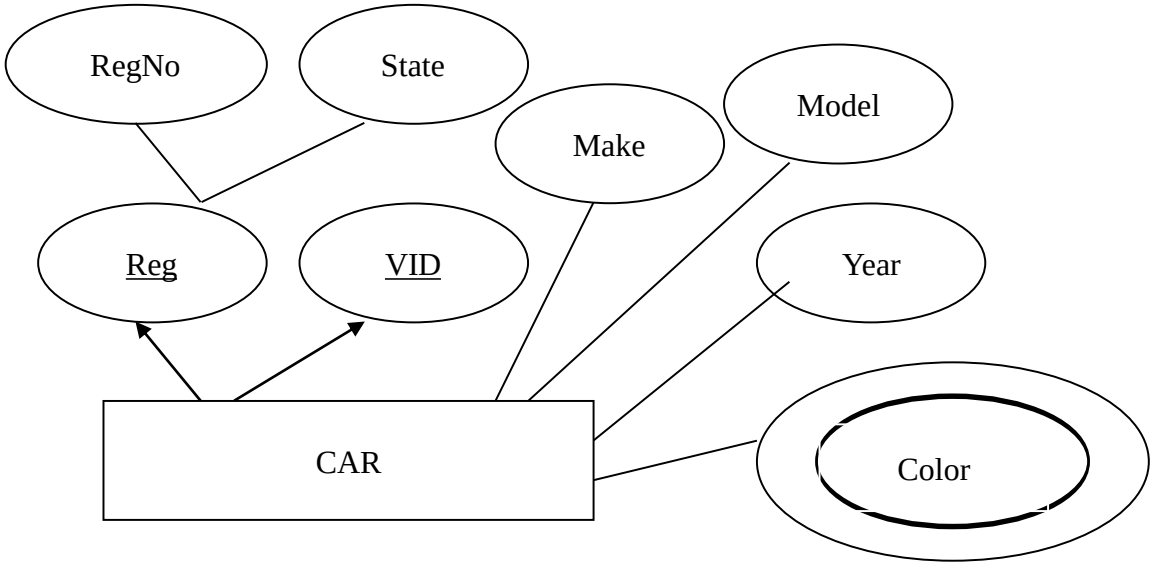
Super Key

- The set of attributes that can uniquely identify a tuple is known as Super Key. For Example, STUD_NO, (STUD_NO, STUD_NAME),
- A super key is a group of single or multiple keys that identifies rows in a table.
- It supports NULL values.
- Adding zero or more attributes to the candidate key generates the super key.
- A candidate key is a super key but vice versa is not true.
- Super Key values may also be NULL.

Candidate key

An entity type may have more than one key called candidate key.
Or

The minimal set of attributes that can uniquely identify a tuple is known as a candidate key.
For example, the CAR entity type may have two keys:
–VehicleIdentificationNumber (popularly called VIN) and
–VehicleTagNumber (Number, State), also known as license_plate number.



Candidate key

For Example, STUD_NO in STUDENT relation.

- It is a minimal super key.
- It is a super key with no repeated data is called a candidate key.
- The minimal set of attributes that can uniquely identify a record.
- It must contain unique values.
- It can contain NULL values.
- Every table must have at least a single candidate key.
- A table can have multiple candidate keys but only one primary key.
- The value of the Candidate Key is unique and may be null for a tuple.
- There can be more than one candidate key in a relationship.

Primary Key

- There can be more than one candidate key in relation out of which one can be chosen as the primary key.
- It is a unique key.
- It can identify only one tuple (a record) at a time.
- It has no duplicate values, it has unique values.
- It cannot be NULL.
- Primary keys are not necessarily to be a single column; more than one column can also be a primary key for a table.

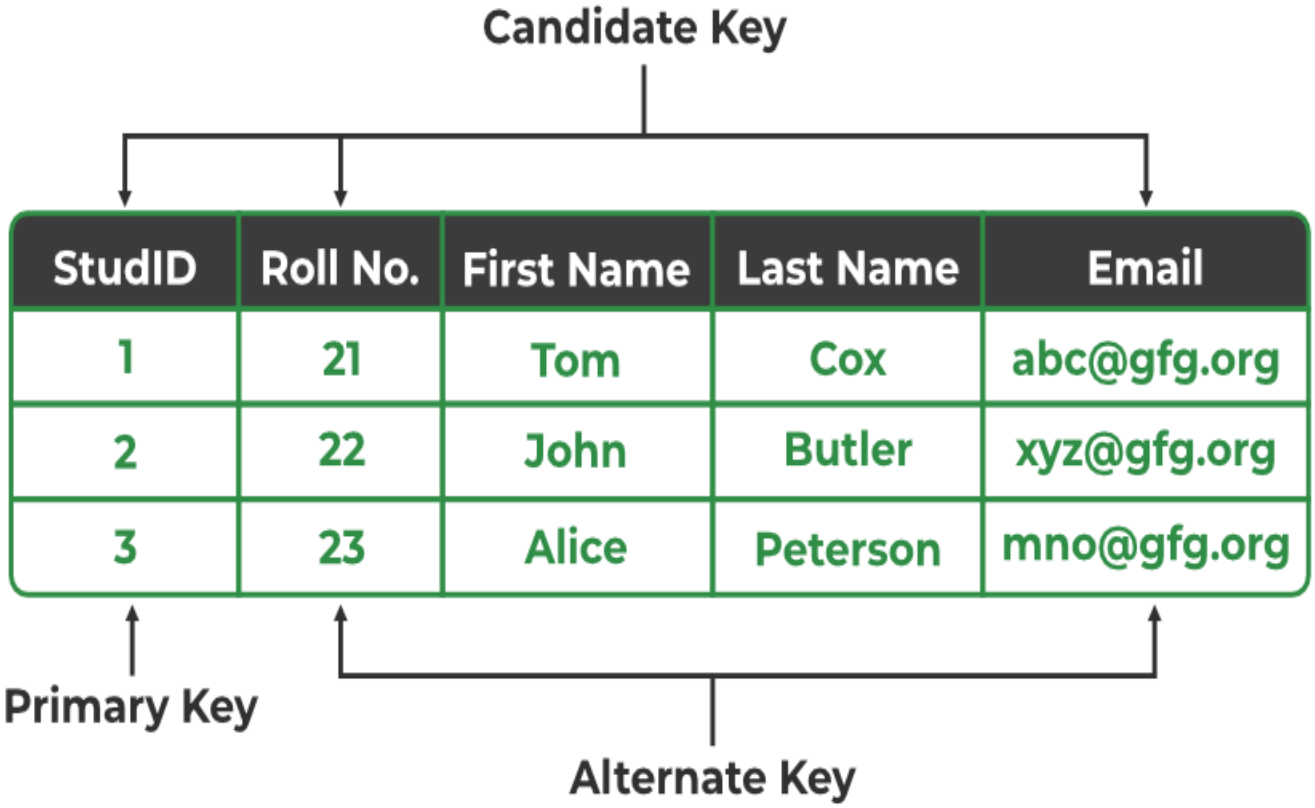
Example:

- STUD_NO, as well as STUD_PHONE, are candidate keys for relation STUDENT but STUD_NO can be chosen as the primary key (only one out of many candidate keys).
- STUDENT table -> Student(STUD_NO, SNAME, ADDRESS, PHONE) , STUD_NO is a primary key.

Table STUDENT

STUD_NO	SNAME	ADDRESS	PHONE
1	Shyam	Delhi	123456789
2	Rakesh	Kolkata	223365796
3	Suraj	Delhi	175468965

Example


Candidate Key
StudID
Roll No.
First Name
Last Name
Email
1
21
Tom
Cox
abc@gfg.org
2
22
John
Butler
xyz@gfg.org
3
23
Alice
Peterson
mno@gfg.org
Primary Key
Alternate Key

Composite Key (CK)

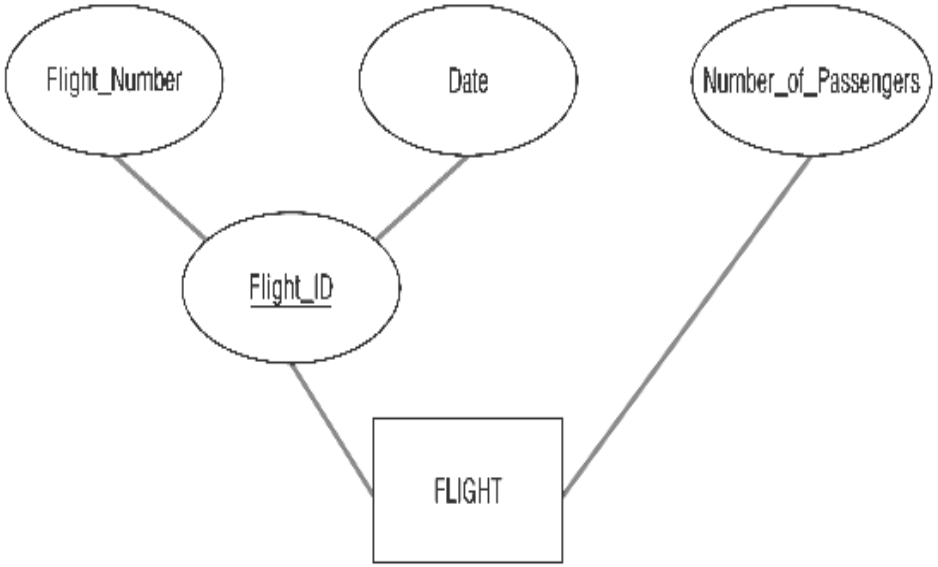
A key attribute may be composite. Such attribute is composite key.

Or

A table might not have a single column/attribute that uniquely identifies all the records of a table. To uniquely identify rows of a table, a combination of two or more columns/attributes can be used.

For example, VehicleTagNumber is a key of the CAR entity type with components (Number, State).

Example2: Flight table Flight_ID is composite key



Continued.

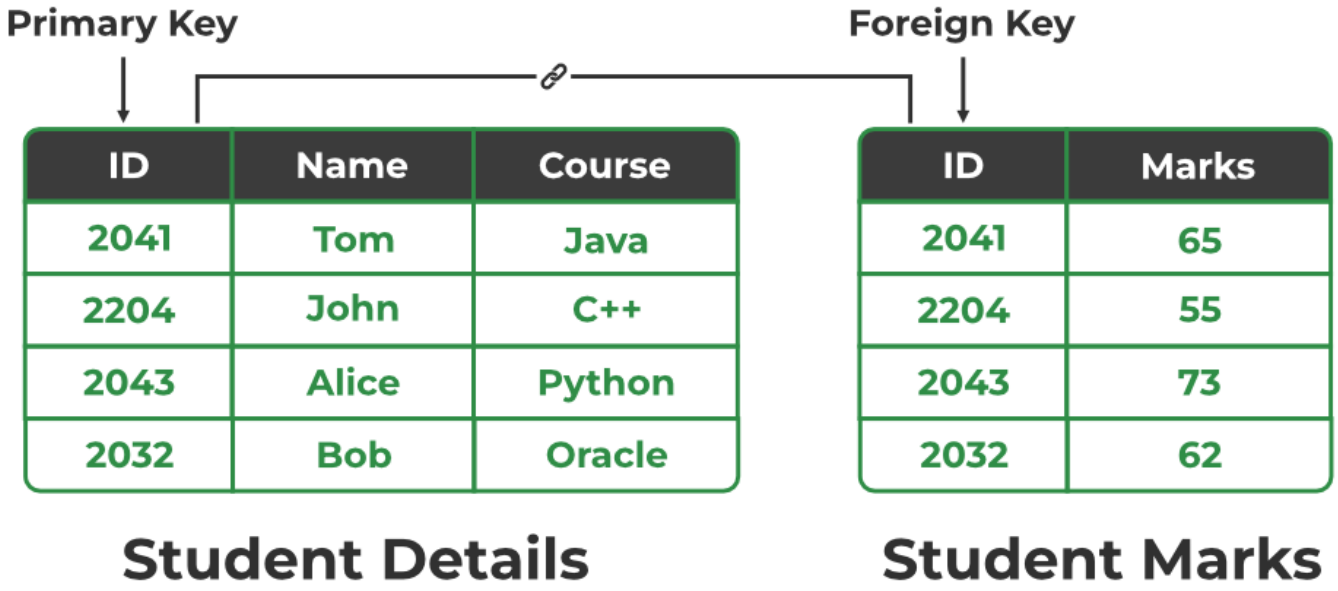
- It still can give duplicate values in rare cases. So, we need to find the optimal set of attributes that can uniquely identify rows in a table.
- It acts as a primary key if there is no primary key in a table
- Two or more attributes are used together to make a composite key.
- Different combinations of attributes may give different accuracy in terms of identifying the rows uniquely.

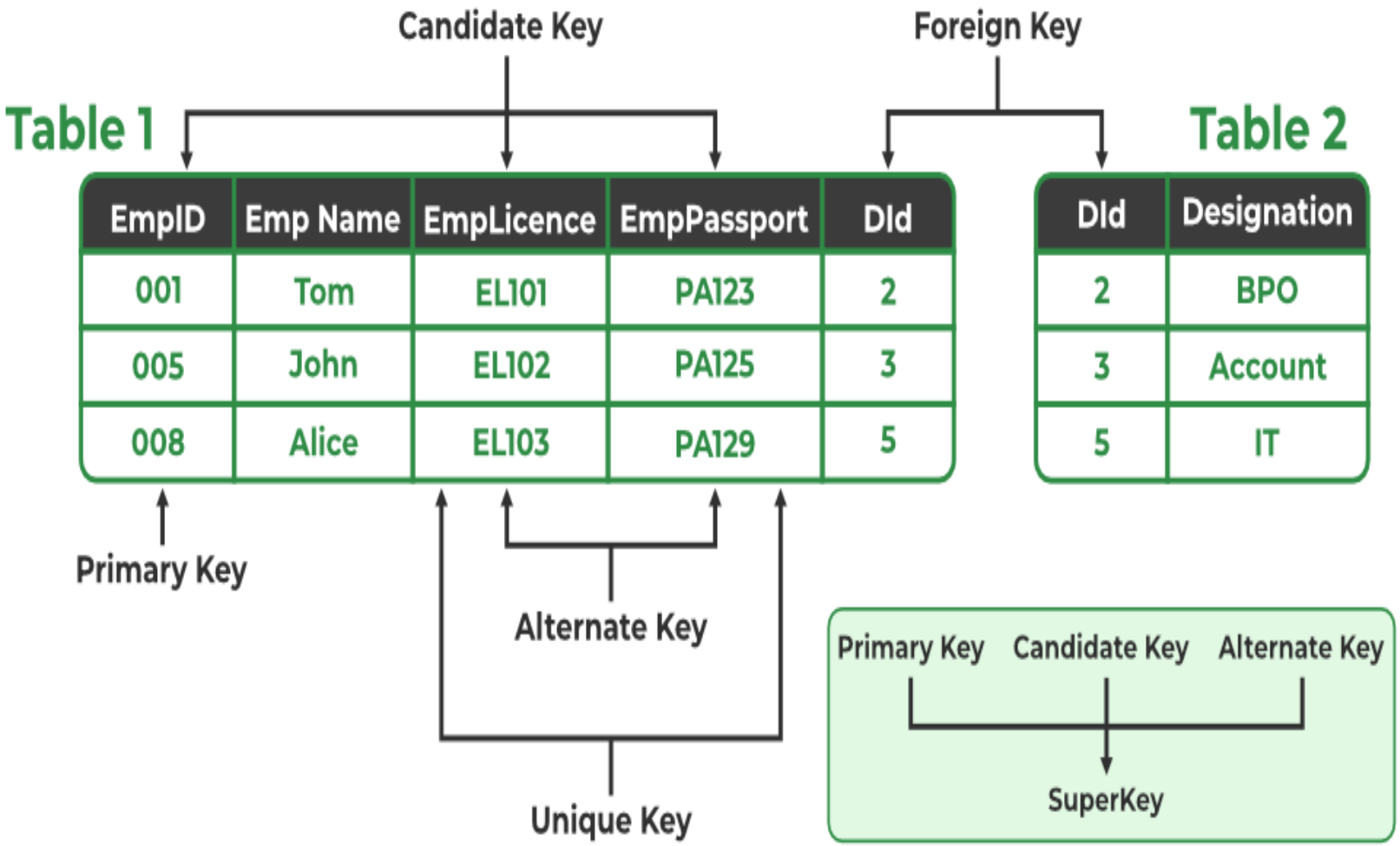
Foreign Key (FK)

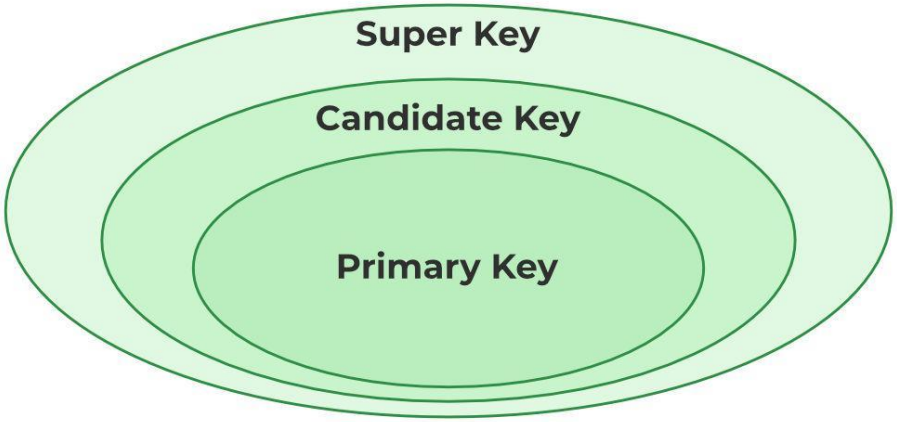
- If an attribute can only take the values which are present as values of some other attribute, it will be a foreign key to the attribute to which it refers.
- The relation which is being referenced is called **referenced relation** and the corresponding attribute is called referenced attribute the relation which refers to the referenced relation is called referencing relation and the corresponding attribute is called referencing attribute.
- The referenced attribute of the referenced relation should be the primary key to it.

FK

- It is a key it acts as a primary key in one table and it acts as secondary key in another table.
- It combines two or more relations (tables) at a time.
- They act as a cross-reference between the tables.

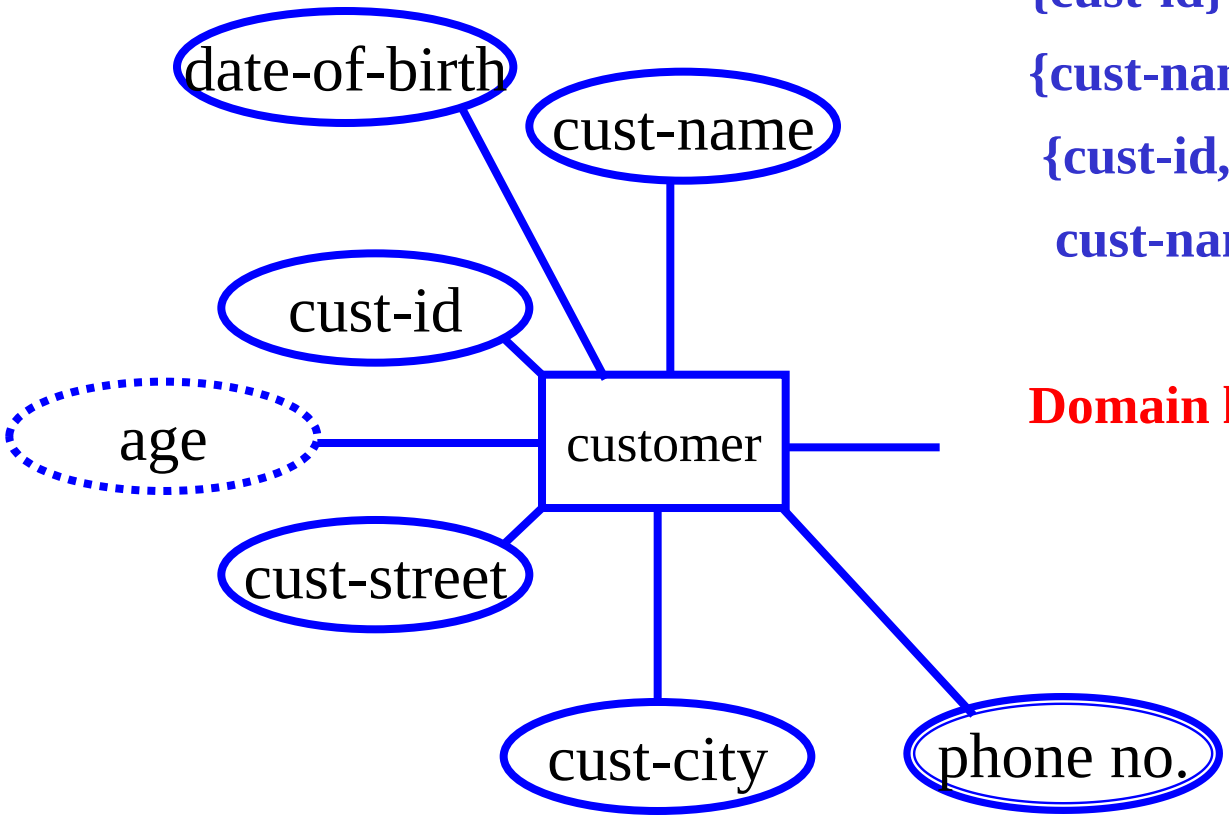



Candidate Key
Foreign Key
Table 1
Table 2
EmpID
Emp Name
EmpLicence
EmpPassport
DId
001
Tom
EL101
PA123
2
005
John
EL102
PA125
3
008
Alice
EL103
PA129
5
DId
Designation
2
BPO
3
Account
5
IT
Primary Key
Alternate Key
Unique Key
Primary Key
Candidate Key
Alternate Key
SuperKey



SCALER Topics

Entity Keys



Possible Keys:

{cust-id}

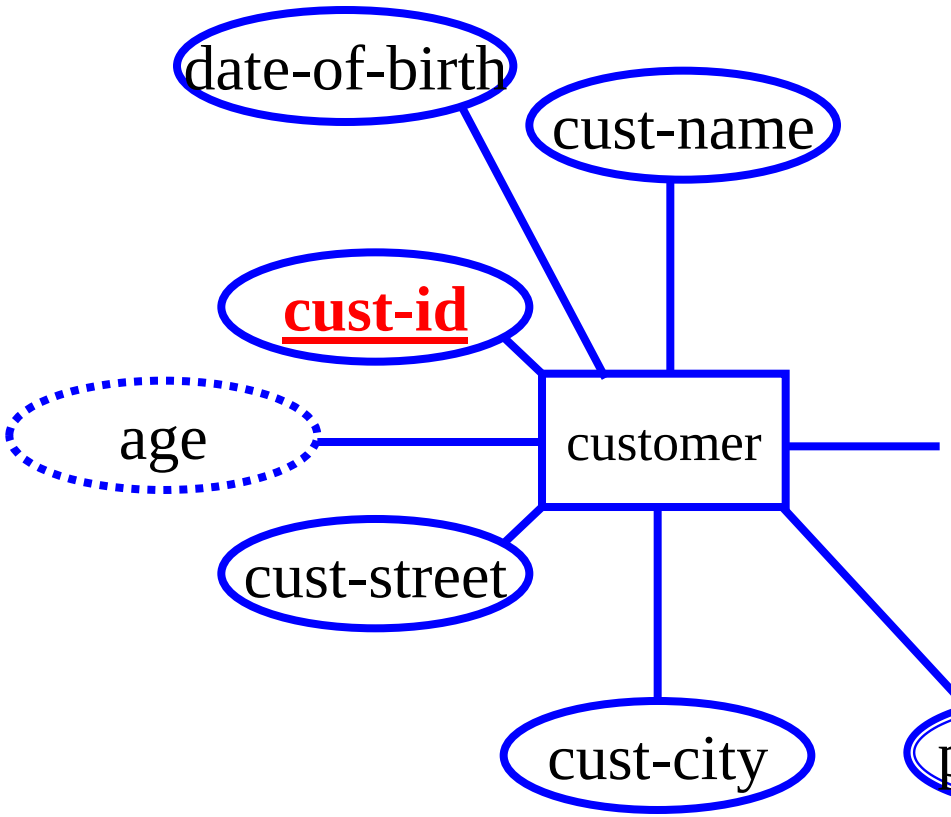
{cust-name, cust-city, cust-street}

{cust-id, age}

cust-name ?? Probably not.

Domain knowledge dependent !!

Entity Keys



- {*cust-id*} is a natural primary key
- Typically, SSN forms a good primary key
- Try to use a candidate key that rarely changes
  - e.g. something involving address not a great idea

Value set (or Domain) of attributes

- The domain of an attribute is the set of all possible values from which the attributes can take its values.
- Value sets are not displayed in ER diagrams. It is specified during creating entity type.

Example1: Season attribute have the possible values are Spring, Summer, Autumn, Winter.

Example2 :Gender attribute can have Female or Male.

Mathematically Definition of value sets of attribute:

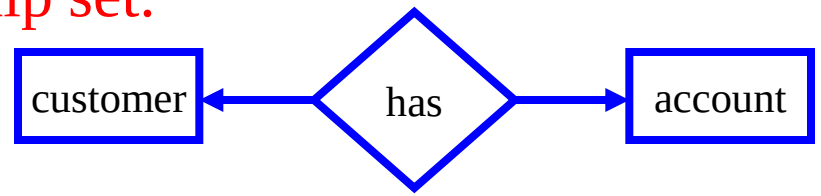
- An attribute A of entity type E whose vale set is V can be defined as a function from E to the power set P(V) of V;
- **A : E→ P(V)**

Relationships

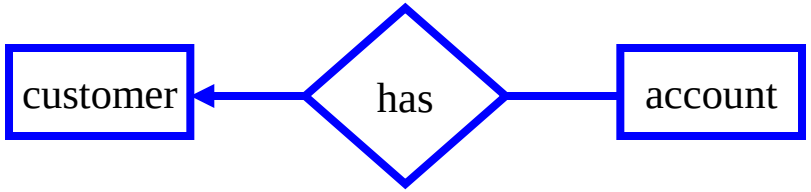
- A relationship is ".. An association among entities (the participants).." . Relationships link entities with each other

Express the number of entities to which another entity can be associated via a relationship set.

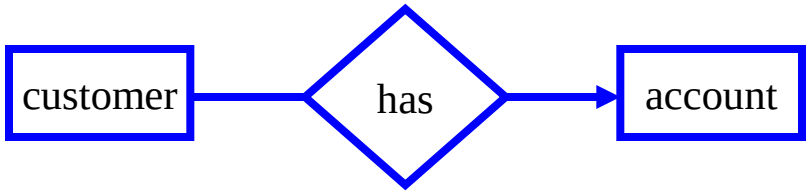
- One-to-One



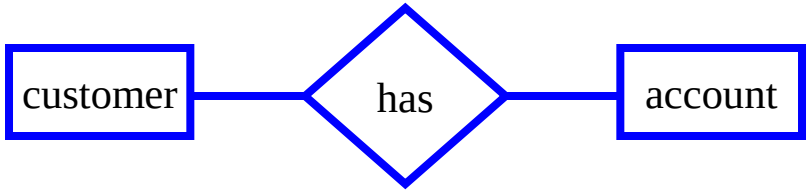
- One-to-Many



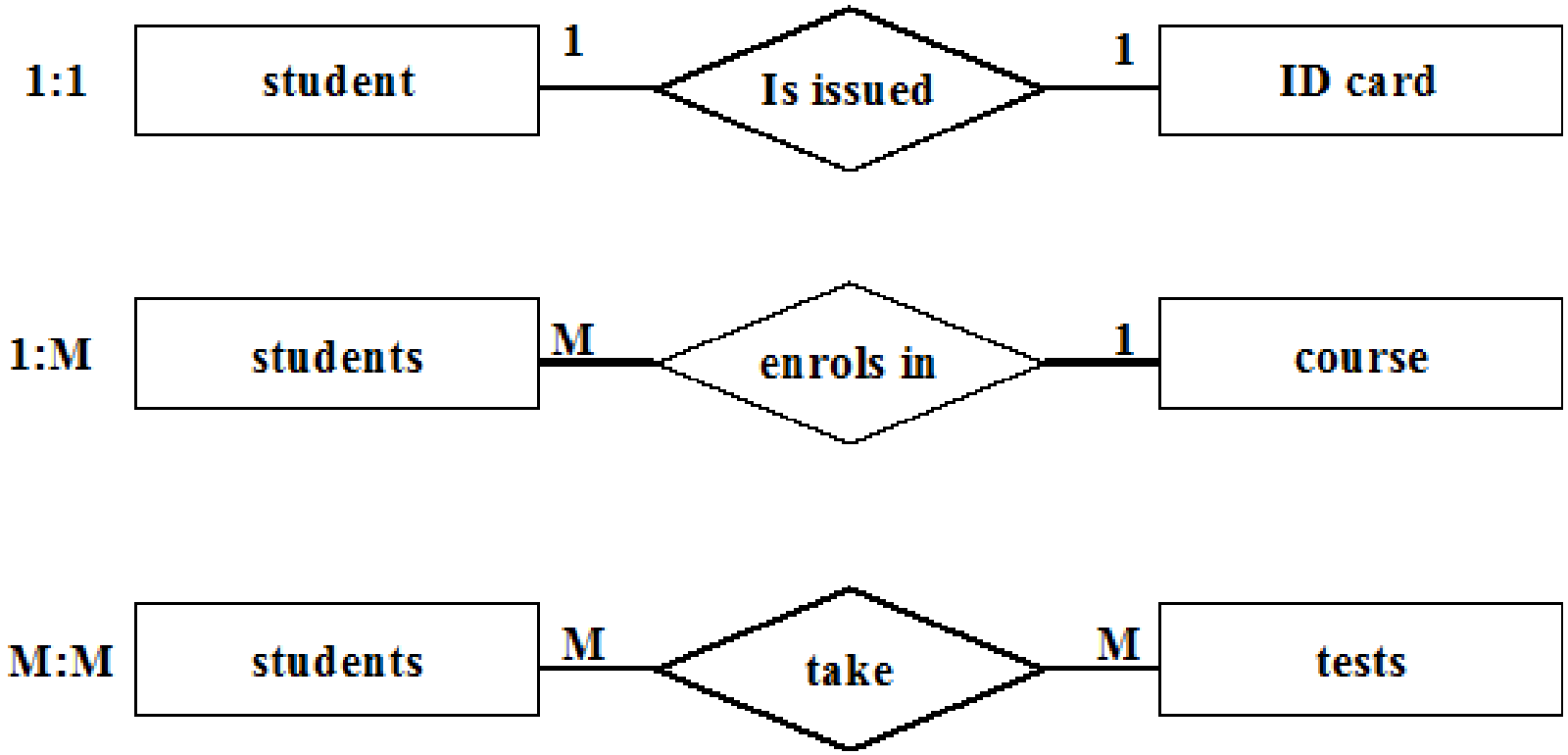
- Many-to-One



- Many-to-Many



Mapping Cardinalities



Relationship Type

- A relationship type between two entity types defines the set of all association between these entity types. A relationship relates two or more distinct entities with a specific meaning.
- or
- A relationship type R among n entity types E_1,$E_{2,}$,……., E_n defines a set of associations.

Relationship set

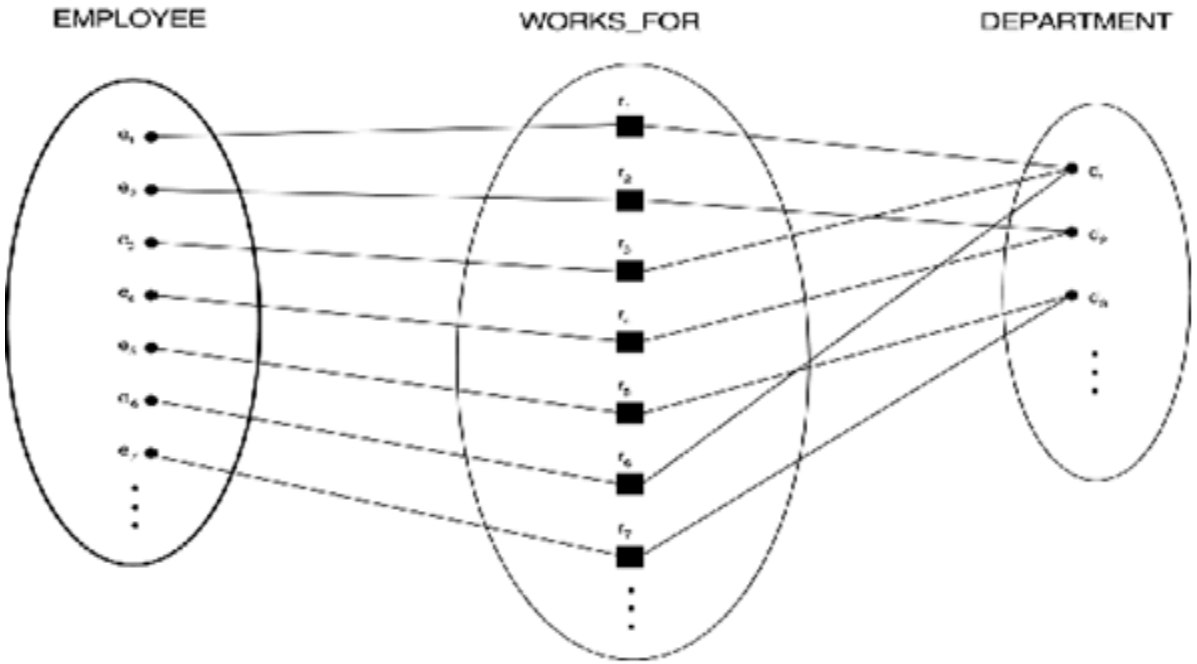
- An instance of a relationship set is a set of relationship.

Relationship instance

- Each instance of the relationship between members of these entity types is called a relationship instance.
- $\mathbf{r_i = (e_1,...., e_n)}$

Mathematically, the relationship set R is a set of relationship instance r_i , where each r_i associates n individual entities ($e_{1,\ldots,}$ e_n) and each entity e_i in r_i is a member of entity type E_j ,1<=j<=n.

Where R be the relationship type. E_1,E_2,….E_n be the entity type. e_1 through e_n, are entity set and r_i is the relationship instances.



Degree of a relationship types

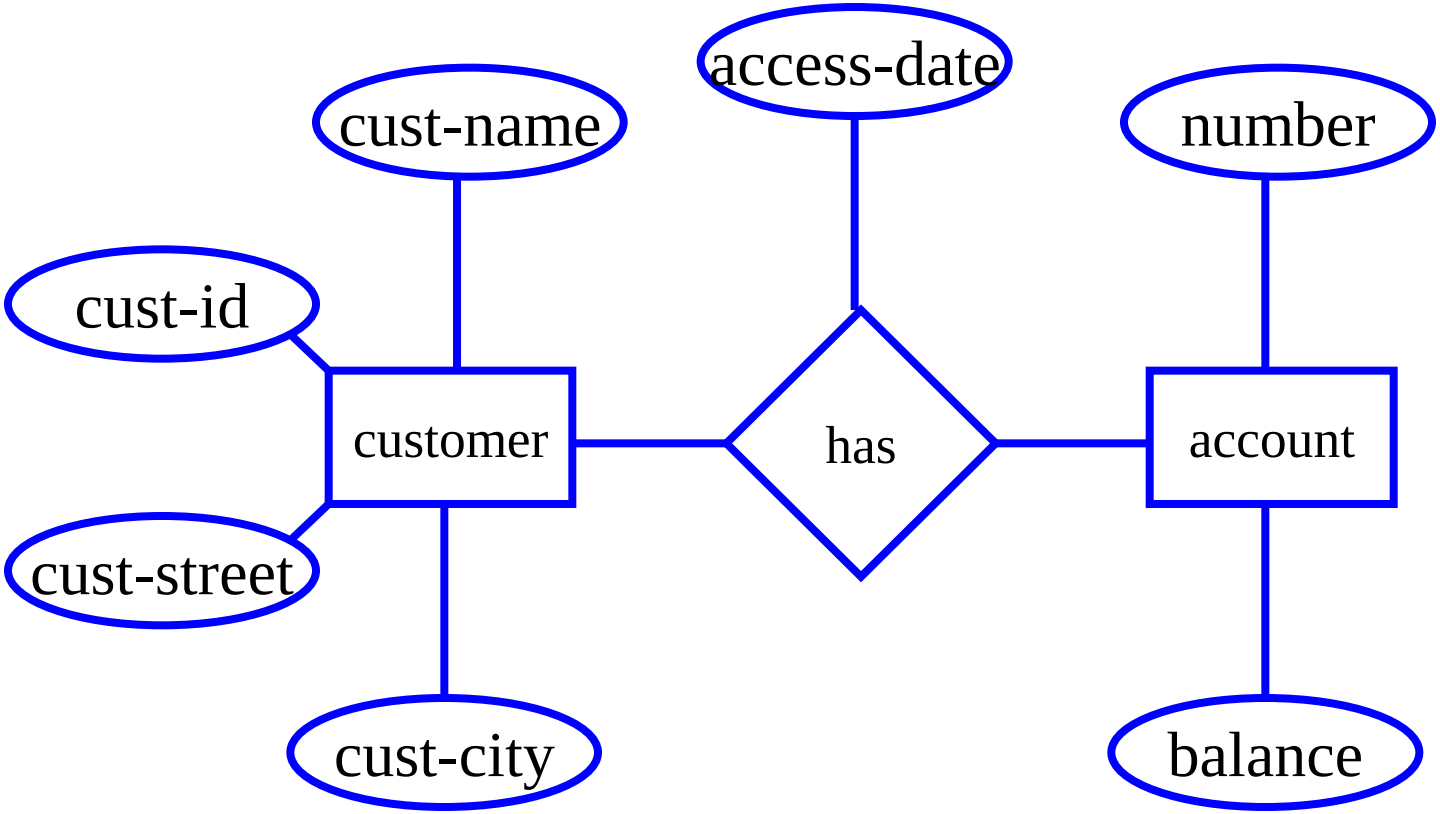
- The ***degree*** of a relationship type number of entity participate in relationship.
  - Binary (connects 2 entity types)
  - Unary/ recursive (connects 1 entity type with itself)
  - n-ary / Complex (connects 3 or more entity types)
    - Ternary (connects 3)

Degree

- Structural constraints - ***cardinality***
  - one to one (1:1)
  - one to many (1:m)
  - many to many (m:n)
- Structural constraints – ***participation***
  - Full(total) / mandatory
  - or partial /optional

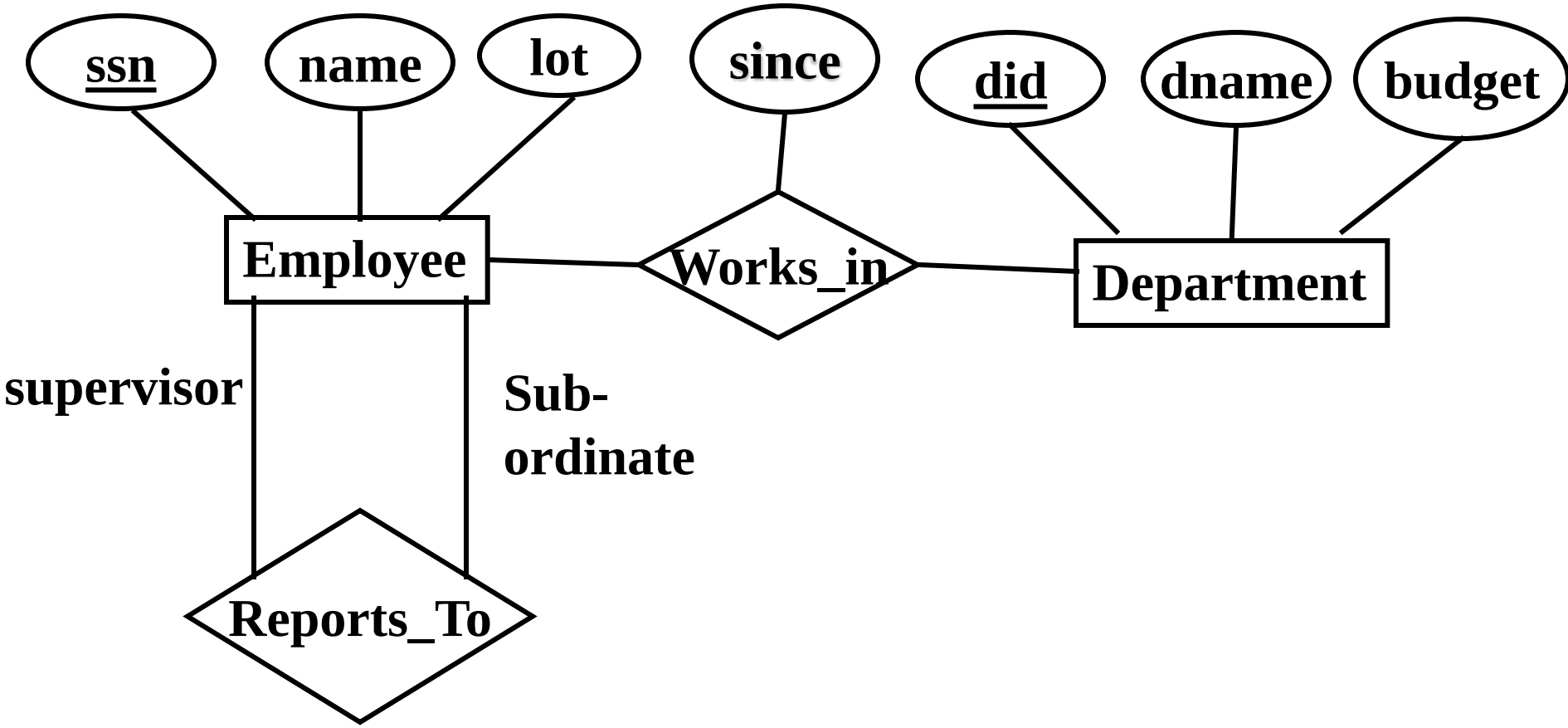
Multiplicity

ER Model: Simple Example

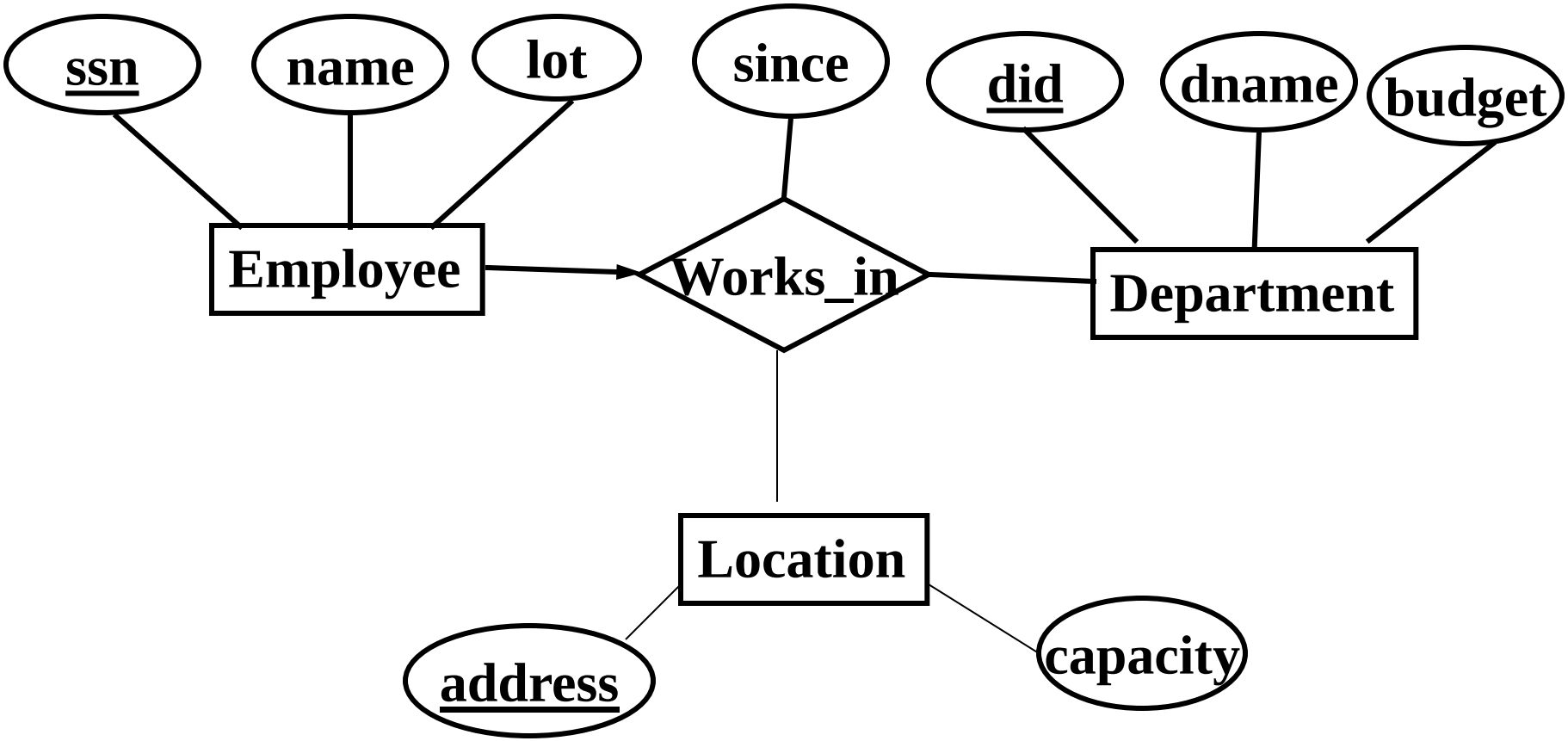


- Rectangles: entity types
- Diamonds: relationship types
- Ovals: attributes

ER Model: Another Example



Ternary Relationships



ER Model (Contd.)

Works_ In

SSN	DID	SINCE
123-22-3666	51	1/1/91
123-22-3666	56	3/3/93
231-31-5368	51	2/2/92

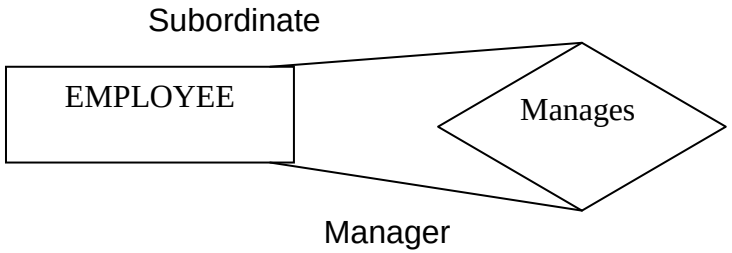
```
CREATE TABLE Works_ In(
ssn CHAR (11),
did INTEGER,
since DATE,
PRIMARY KEY (ssn, did),
FOREIGN KEY (ssn)
REFERENCES Employees,
FOREIGN KEY (did)
REFERENCES Departments)
```

Role names

- Each entity type that participates in a relationship type plays a particular role in the relationship.
- Role names may be added to make the meaning more explicit.

When to use role name?

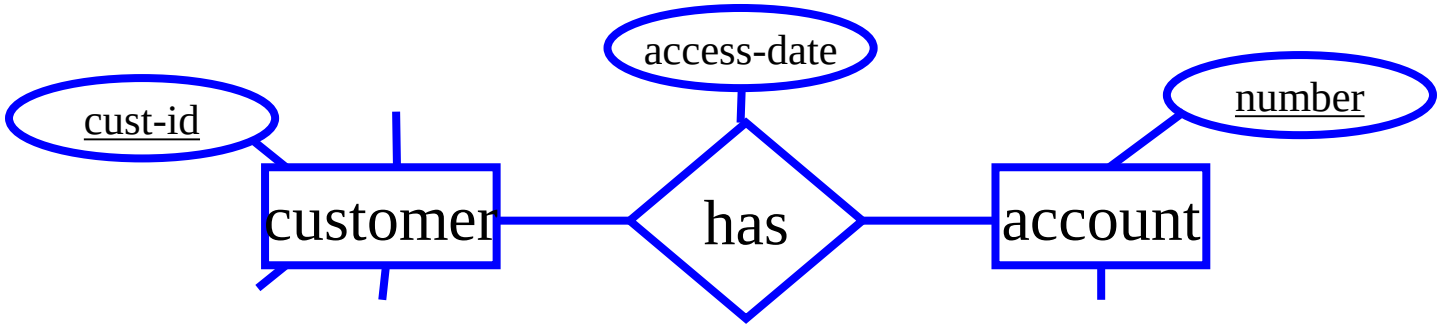
- The participating entity types are distinct, no need to mention role name.
- If the same entity type participates more than once in an r/n type in different roles.
- In such cases the role name becomes essential for distinguishing the meaning of each participation. Such r/n types are called recursive relationship.



Attributes of relationship types

- Relationship types can also have attribute similar to those of entity type.

- What attributes are needed to represent a relationship completely and uniquely ?
  - Union of primary keys of the entities involved, and relationship attributes



  - *{cust-id, access-date, account-number}* describes a relationship completely

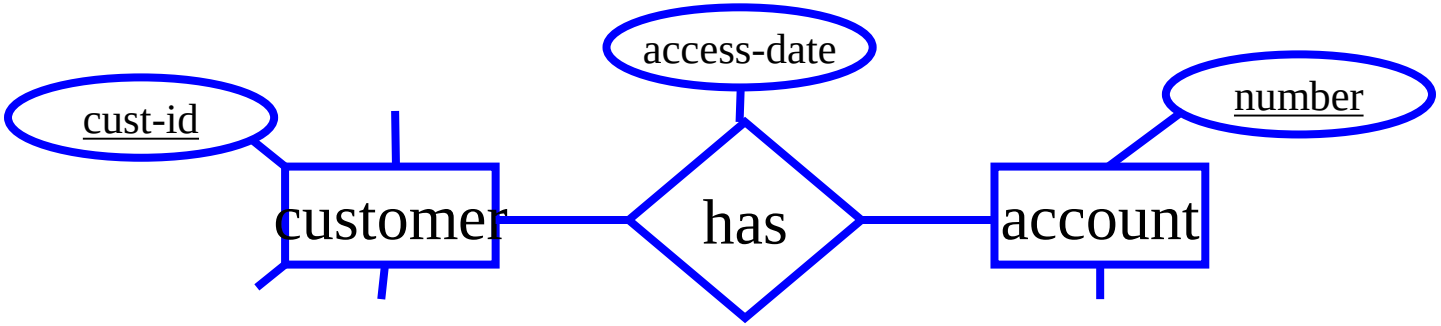
Relationship Set Keys

- ***Note: Placing the attribute in relationship type i.e 1:1, 1:N and M:M . Representing this attribute in one of the entity type.***

- General rule for binary relationships
  - one-to-one: primary key of either entity set
  - one-to-many: primary key of the entity set on the many side
  - many-to-many: union of primary keys of the associate entity sets
- n-ary relationships
  - More complicated rules

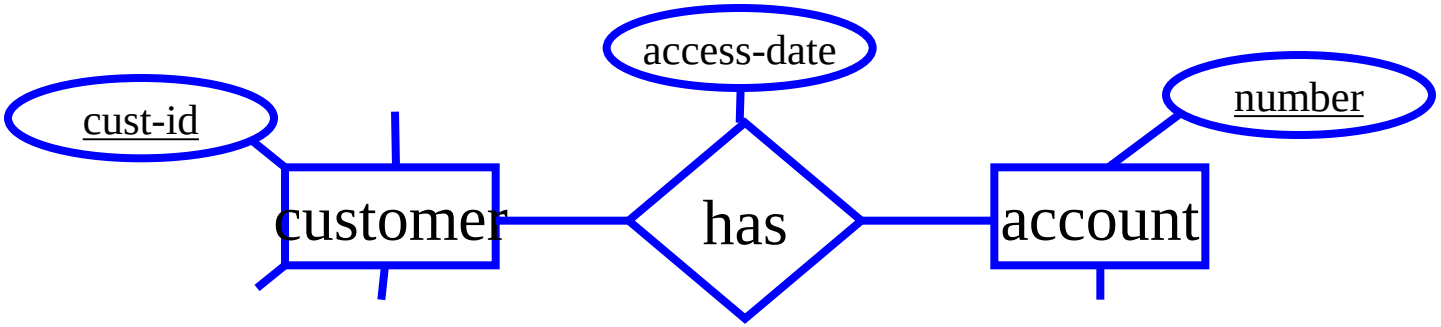
Relationship Set Keys

- Is *{cust-id, access-date, account-number}* a candidate key ?
  - No. Attribute *access-date* can be removed from this set without losing key-ness
  - In fact, union of primary keys of associated entities is always a superkey



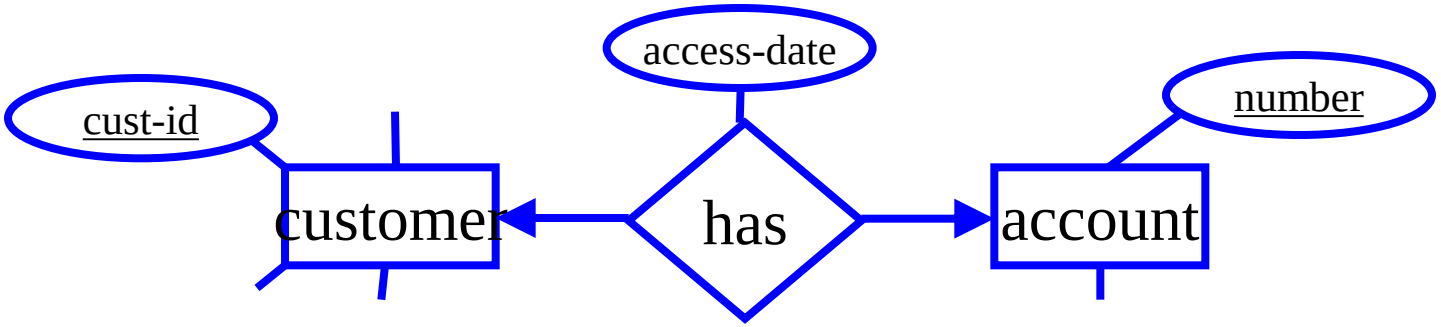
Relationship Set Keys

- Is {cust-id, account-number} a candidate key ?
  - Depends



Relationship Set Keys

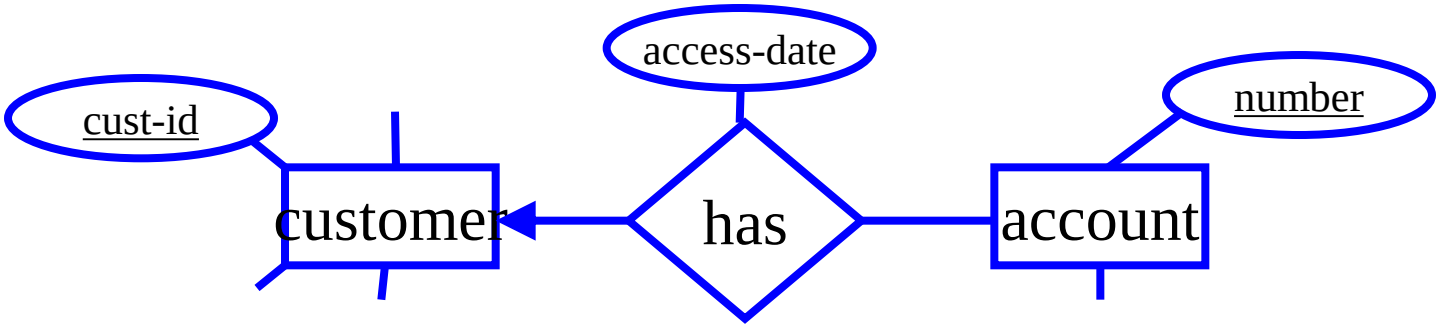
- Is {cust-id, account-number} a candidate key ?
  - Depends



- If one-to-one relationship, either *{cust-id}* or *{account-number}* sufficient
  - Since a given *customer* can only have one *account*, He / She can only participate in one relationship
  - Ditto *account*

Relationship Set Keys

- Is {cust-id, account-number} a candidate key ?
  - Depends



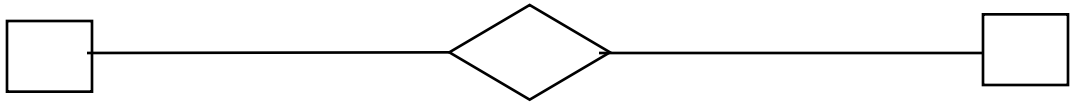
- If one-to-many relationship (as shown), *{account-number}* is a candidate key
  - A given customer can have many accounts, but at most one account holder per account allowed

Constraints on Relationships Types (constraints means restriction or limitation)

- Relationship types have certain constraints that limit the possible combination of entities that may participate in the corresponding relationship set.
- Example: a company has a rule that each employee must work for exactly one department.
- There two main types of relationship constraints
  - Cardinality ratio (Maximum Cardinality constraint)
  - Participation (also called Minimum Cardinality or participation constraint or existence dependency constraints)

Structural Constraints

- Participation
  - Do all entity instances participate in at least one relationship instance?
- Cardinality
  - How many relationship instances can an entity instance participate in?



(min, max) (min, max)

Participation	Cardinality
0 -- Partial	1 -- one
1 -- Total (Mandatory)	M -- more than one

Constraints ratio for binary Relationship Types

- (Also known as ratio constraints)
- Cardinality ratio (Maximum Cardinality) for a binary relationship specifies the maximum number of relationship instances that an entity can participate in.
    - One-to-one (1:1)
    - One-to-many (1:N)
    - Many-to-one (N:1)
    - Many-to-many (M:N)

Participation constraint

- It specifies whether the existence of an entity depends on its being related to another entity the relationship types.
- This constraint specifies the minimum number of relationship instances that each entity can participate in. This type is called **Minimum Cardinality constraint**.

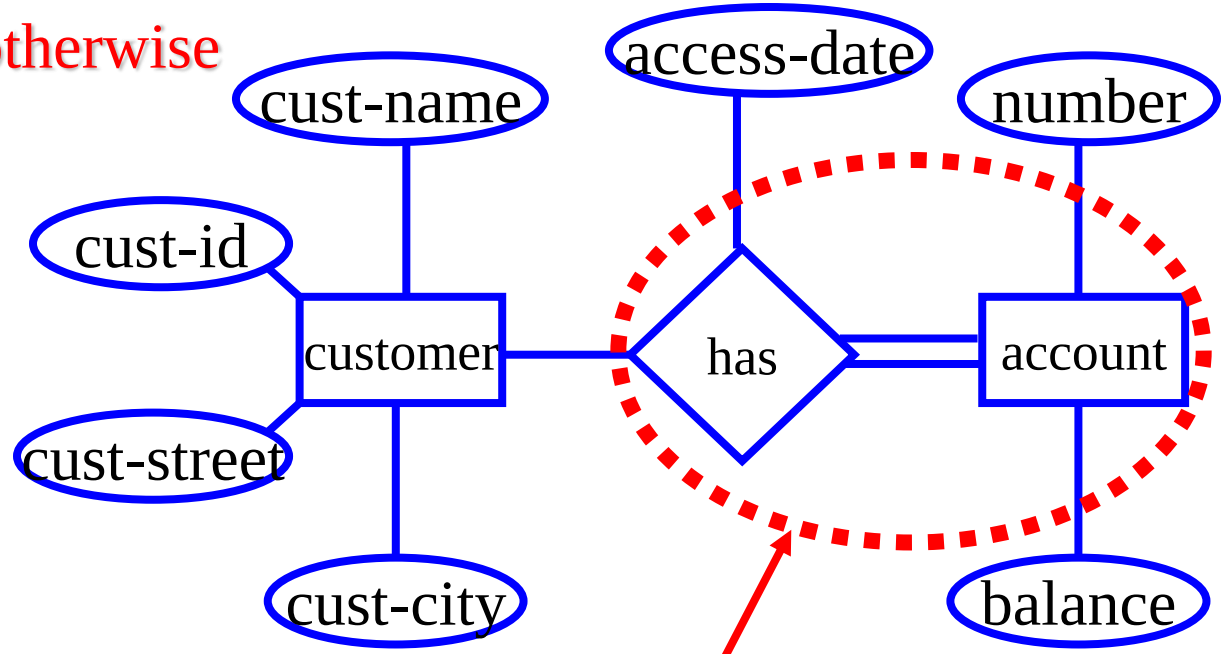
There are two types of Participation constraint

1.Total Participation constraint (or existence dependency):

2.Partial Participation constraint

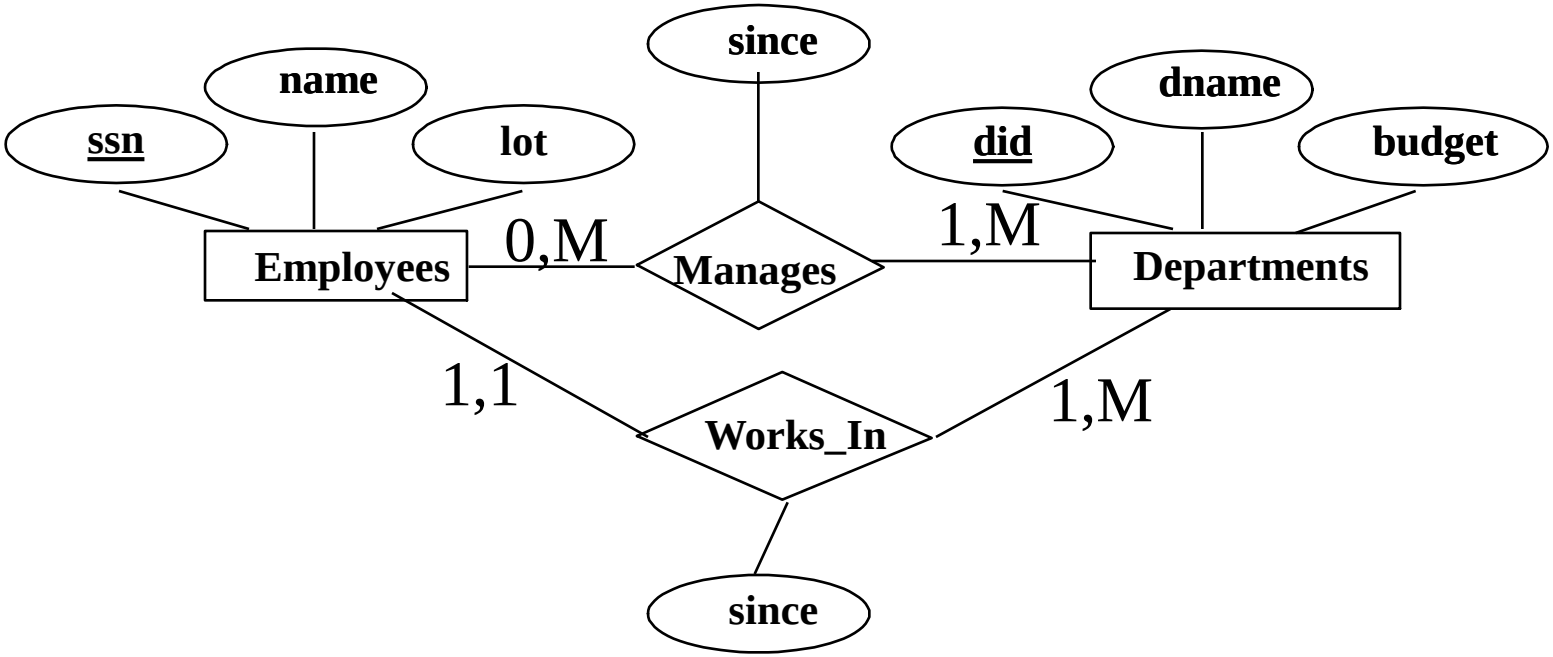
Participation Constraint

- Given an entity set E, and a relationship R it participates in:
  - If every entity in E participates in at least one relationship in R, it is **total participation**
  - partial otherwise



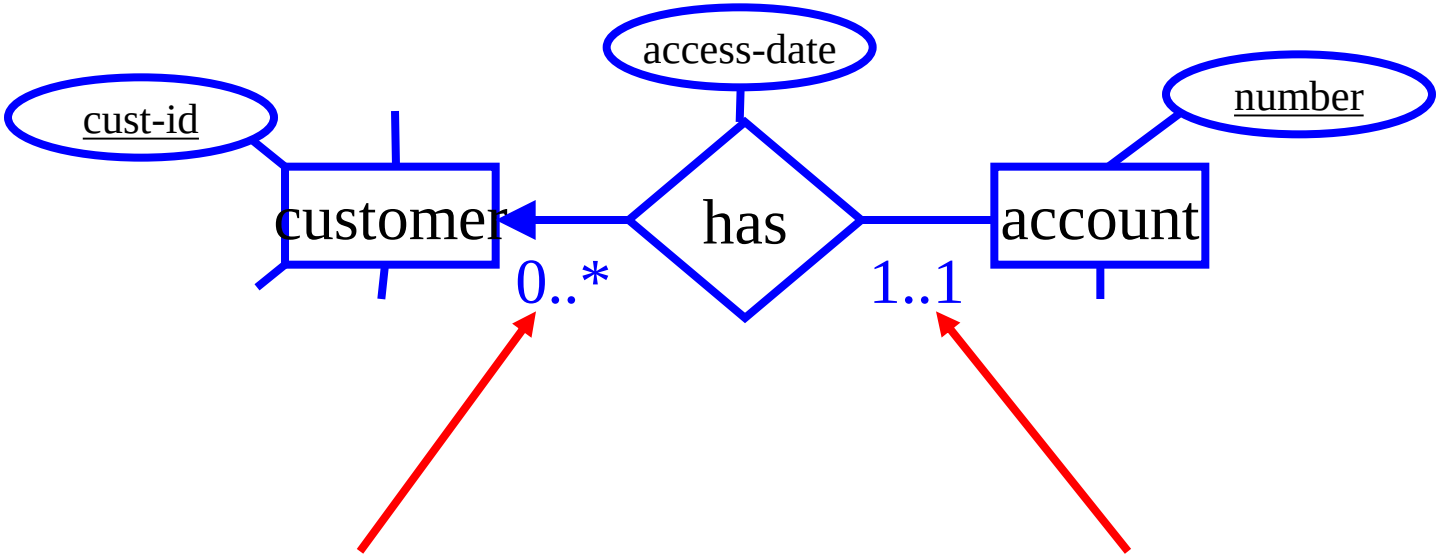
Participation Constraints

- Does every department have a manager?
  - If so, this is a *participation constraint*: the participation of Departments in Manages is said to be *total* (vs. *partial*).
    - Every *did* value in Departments table must appear in a row of the Manages table (with a non-null *ssn* value!)



Cardinality Constraints

How many relationships can an entity participate in ?

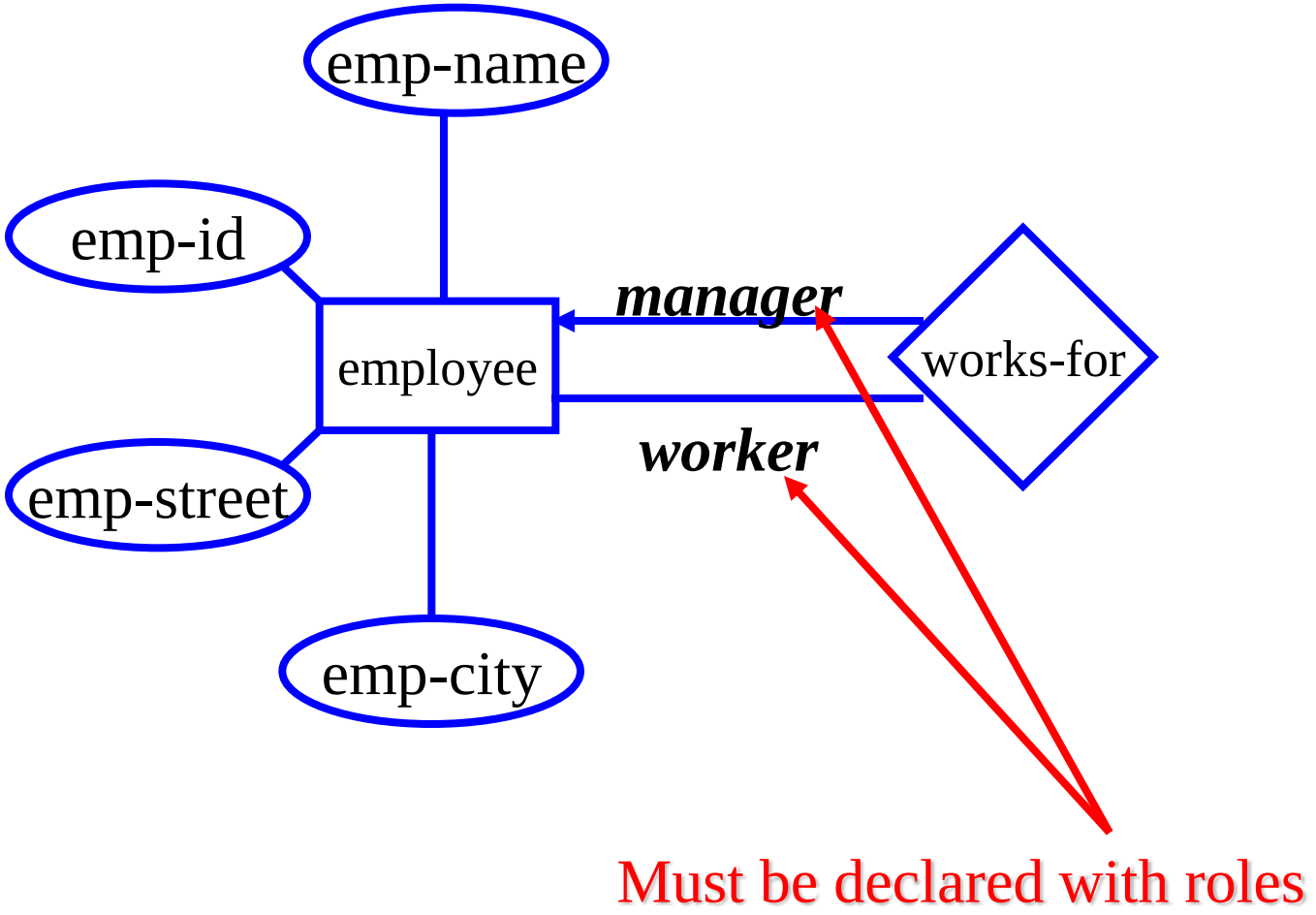


Minimum - 0
Maximum – no limit

Minimum - 1
Maximum - 1

Recursive Relationships

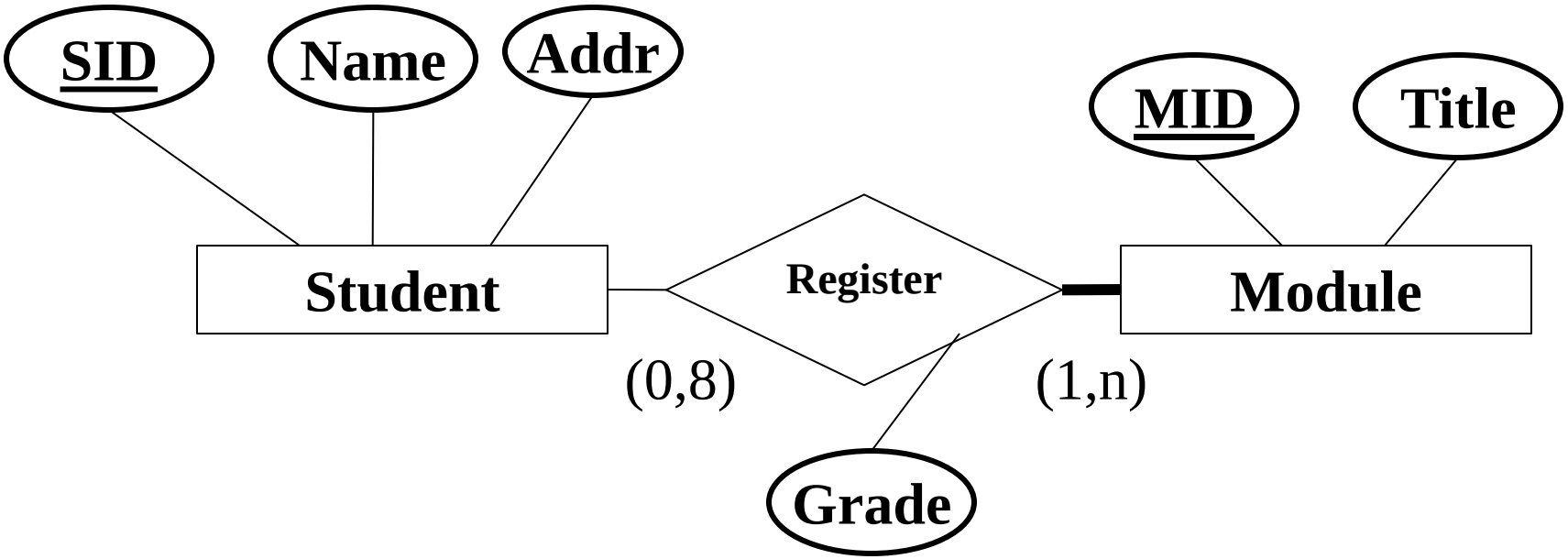
- Sometimes a relationship associates an entity set to itself



Over to You now!

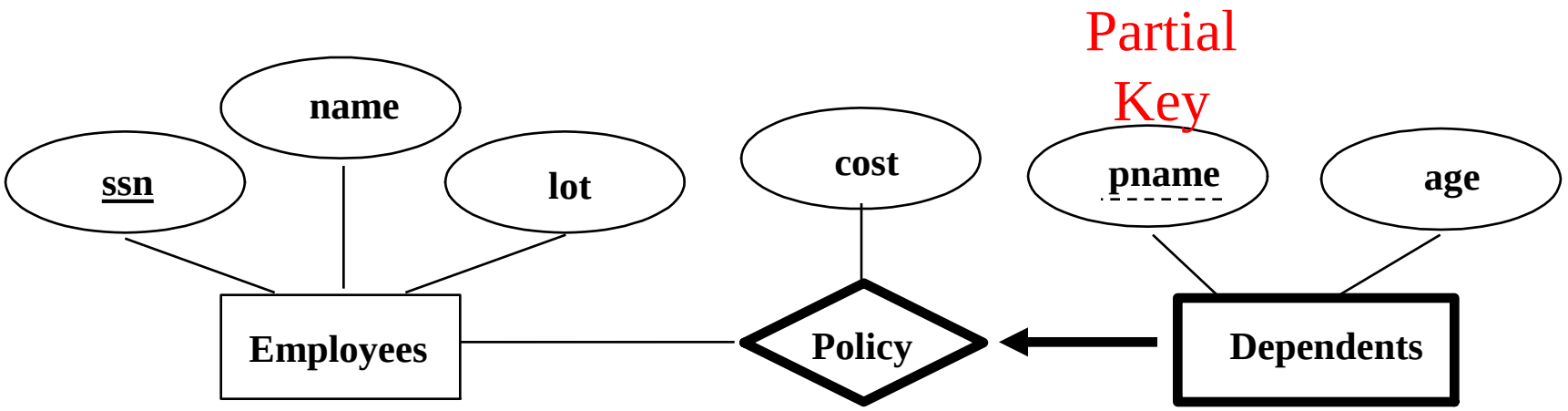
- See if you can draw an E-R diagram for this scenario – you are already familiar with this!
  - "A student registers for up to 8 modules and each module has many students on it. Record the student ID, their full name and address and also each module ID and title. We also want to hold the grade attained by each student for each module"
    - Remember to show in your model:
      - All primary keys,
      - Entities
      - Relationship
      - Attributes

Solution !!!!!



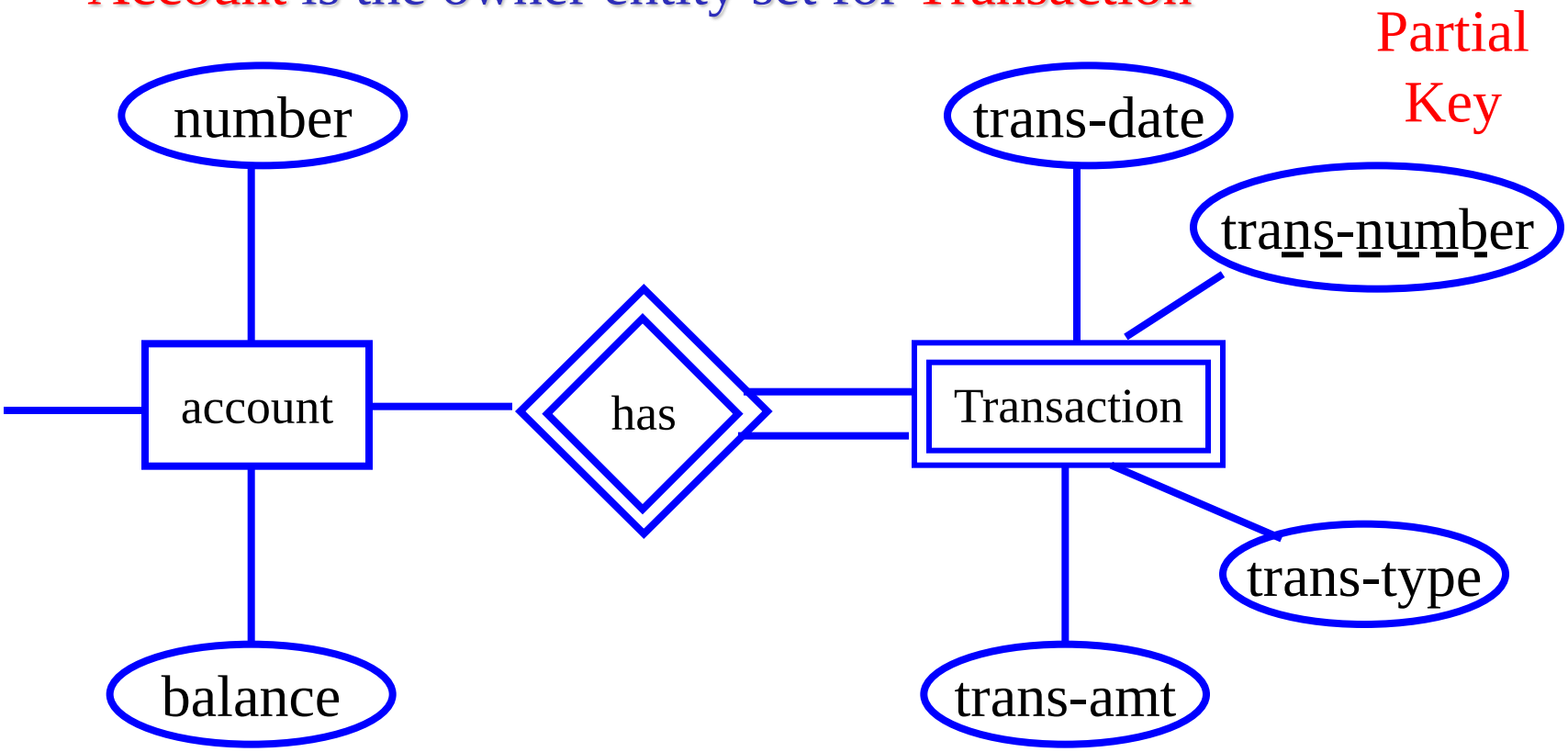
Weak Entities

- A *weak entity* can be identified uniquely only by considering the primary key of another (*owner*) entity.
  - Owner entity set and weak entity set must participate in a one-to-many relationship set (one owner, many weak entities).
  - Weak entity set must have total participation in this *identifying* relationship set.

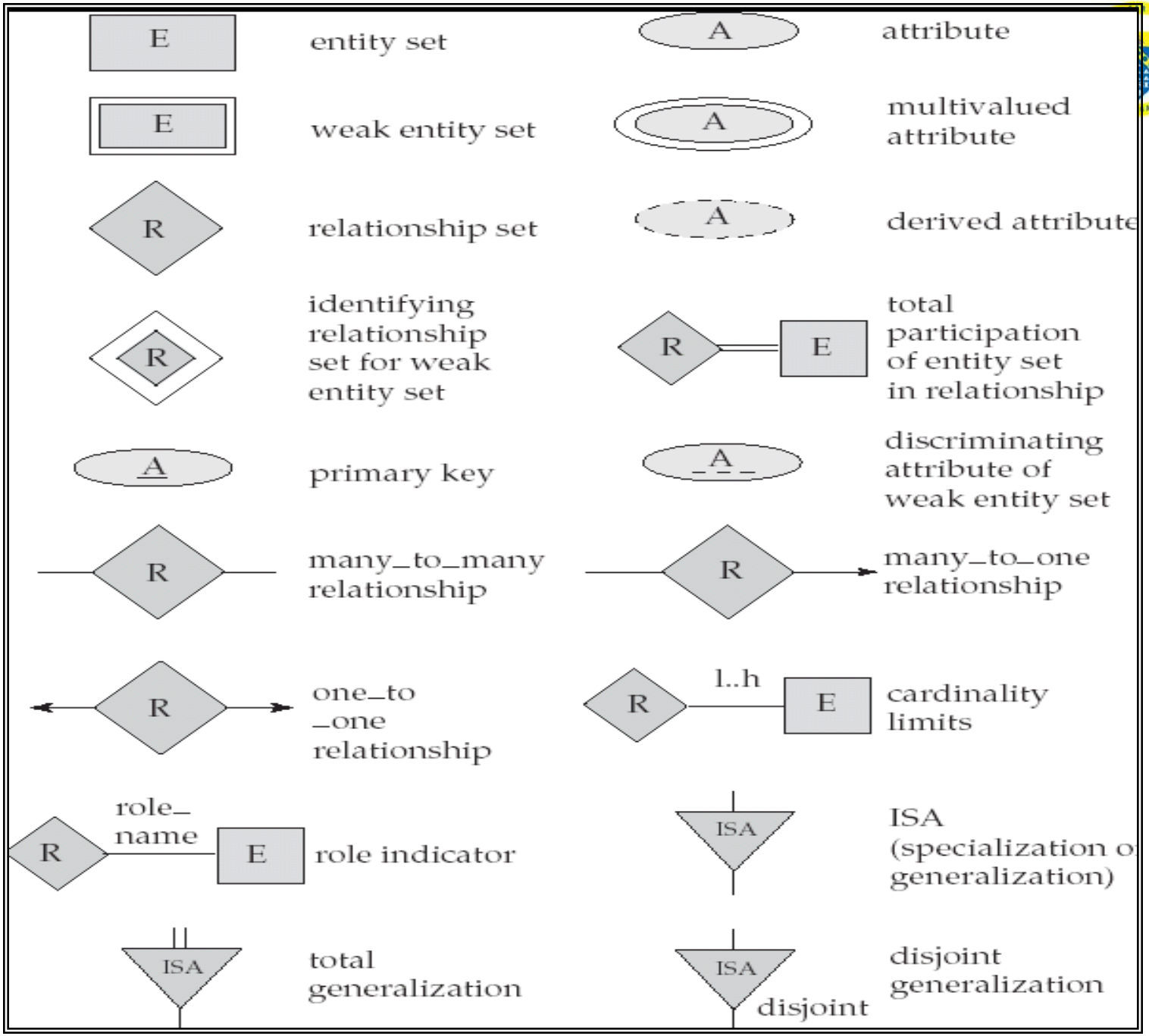


Weak Entity Sets

- A weak entity set must be associated with an identifying or owner entity set
- Account is the owner entity set for Transaction



Summary of Notations in ER Model



ER Model – Design Issues

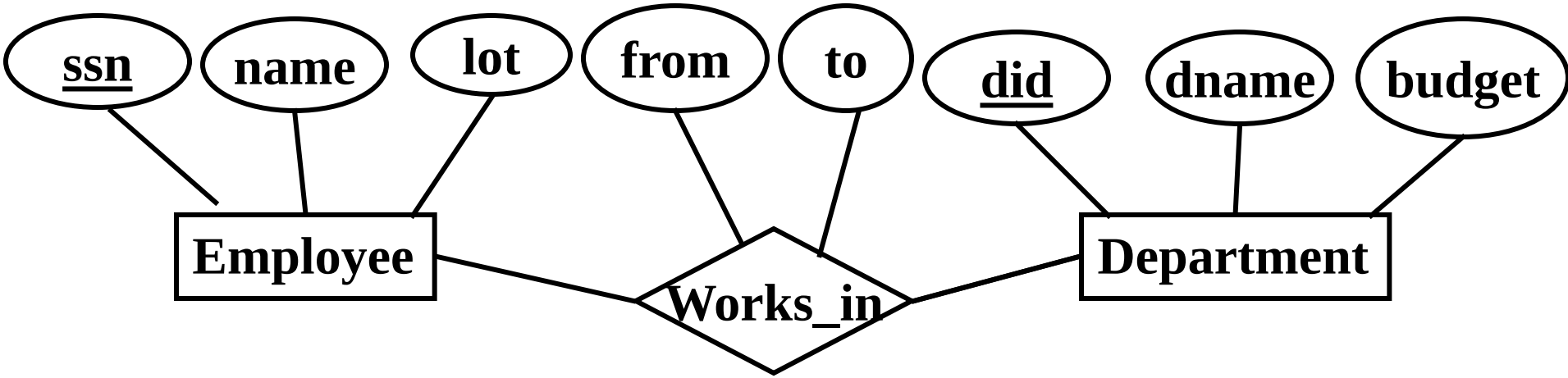
- Design choices:
  - Should a concept be modeled as an entity or an attribute?
  - Should a concept be modeled as an entity or a relationship?
  - Identifying relationships: Binary or ternary? Aggregation?
- Constraints in the ER Model:
  - A lot of data semantics can (and should) be captured.
  - But some constraints cannot be captured in ER diagrams.

Entity vs. Attribute

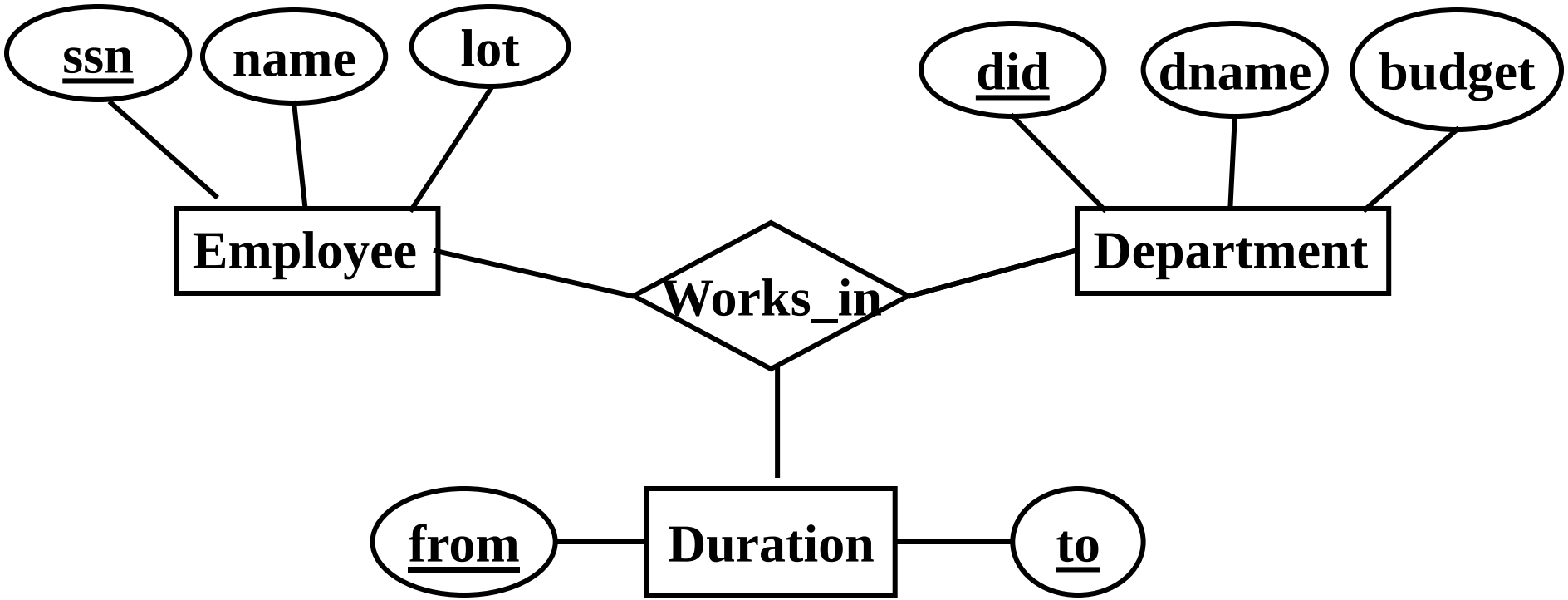
- Should *address* be an attribute of Employees or an entity (connected to Employees by a relationship)?
- Depends upon the use we want to make of address information, and the semantics of the data:
  - If we have several addresses per employee, *address* must be an entity (since attributes cannot be set-valued).
  - If the structure (city, street, etc.) is important, e.g., we want to retrieve employees in a given city, *address* must be modeled as an entity (since attribute values are atomic).

Entity vs. Attribute

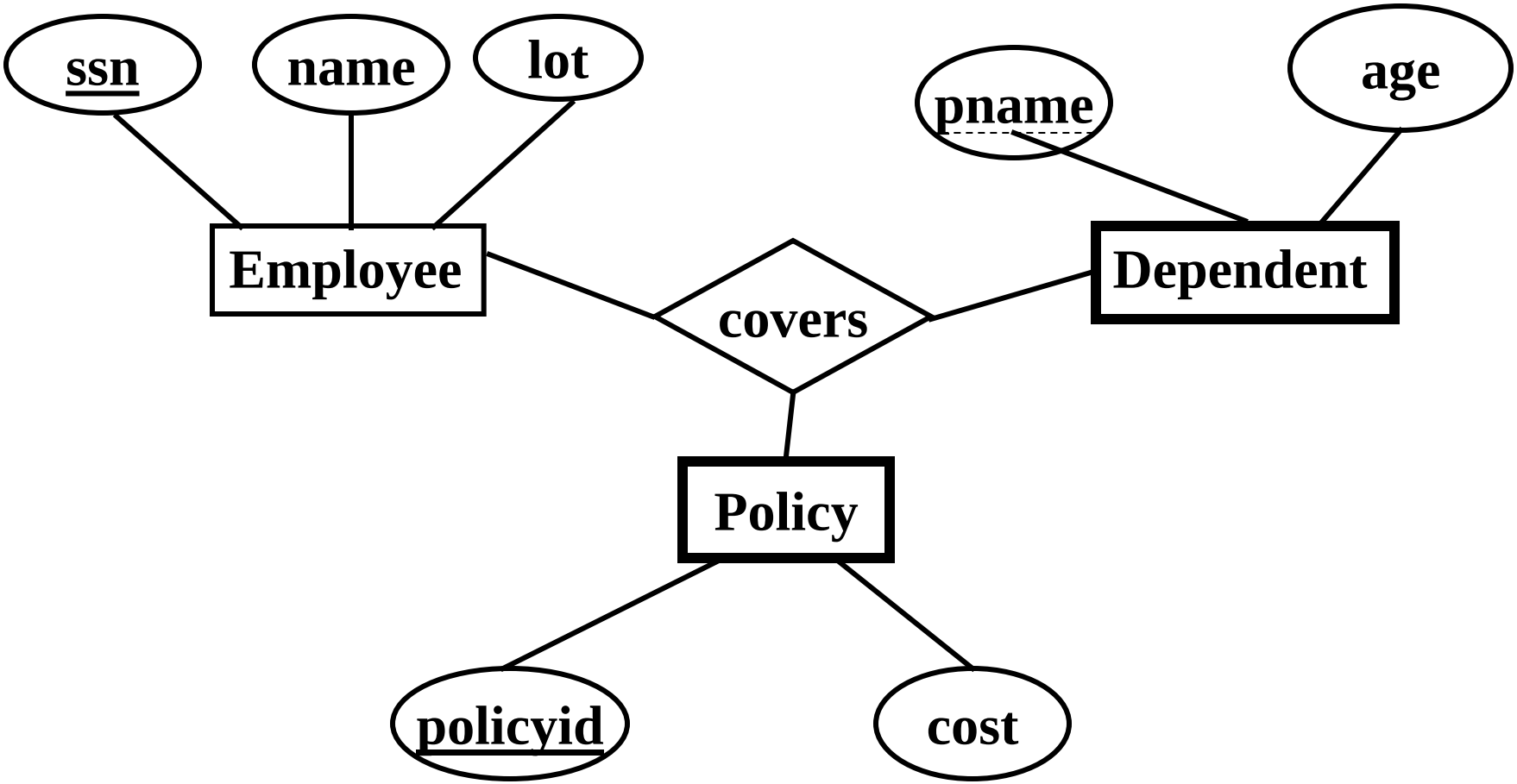
Works_In does not allow an employee to work in a department for two or more periods (why?)



Entity vs. Attribute (Contd.)

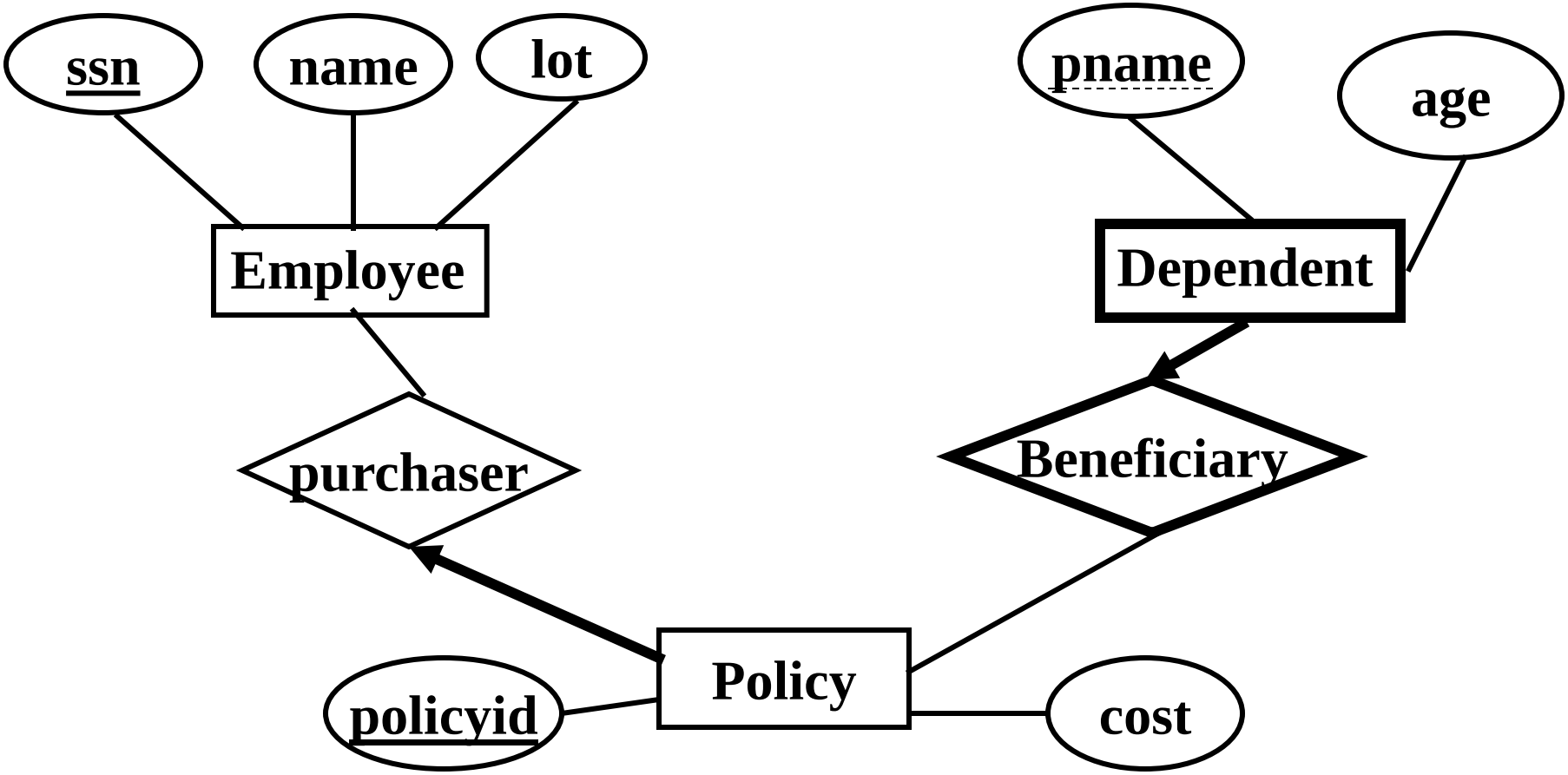


Binary vs. Ternary Relationships



Binary vs. Ternary Relationships

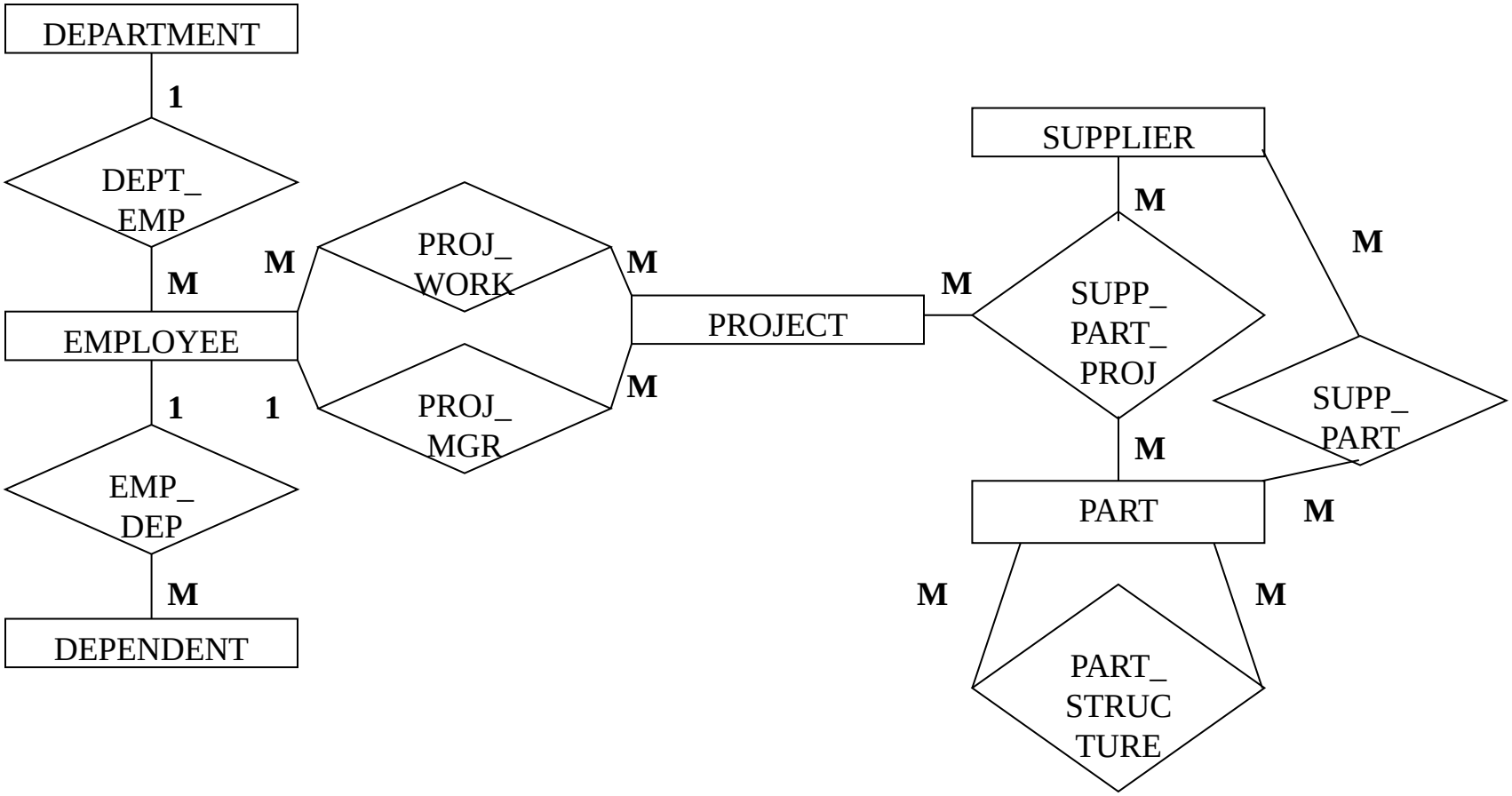
Better Design



Constraints Beyond the ER Model

- Some constraints cannot be captured in ER diagrams:
  - Functional dependencies
  - Inclusion dependencies
  - General constraints

E-R Diagram : Example



Algorithm for ER-to-Relational mapping

Step 1: Mapping of Regular Entity Types

Step 2: Mapping of Weak Entity Types

Step 3: Mapping of Binary 1:1 Relationship Types

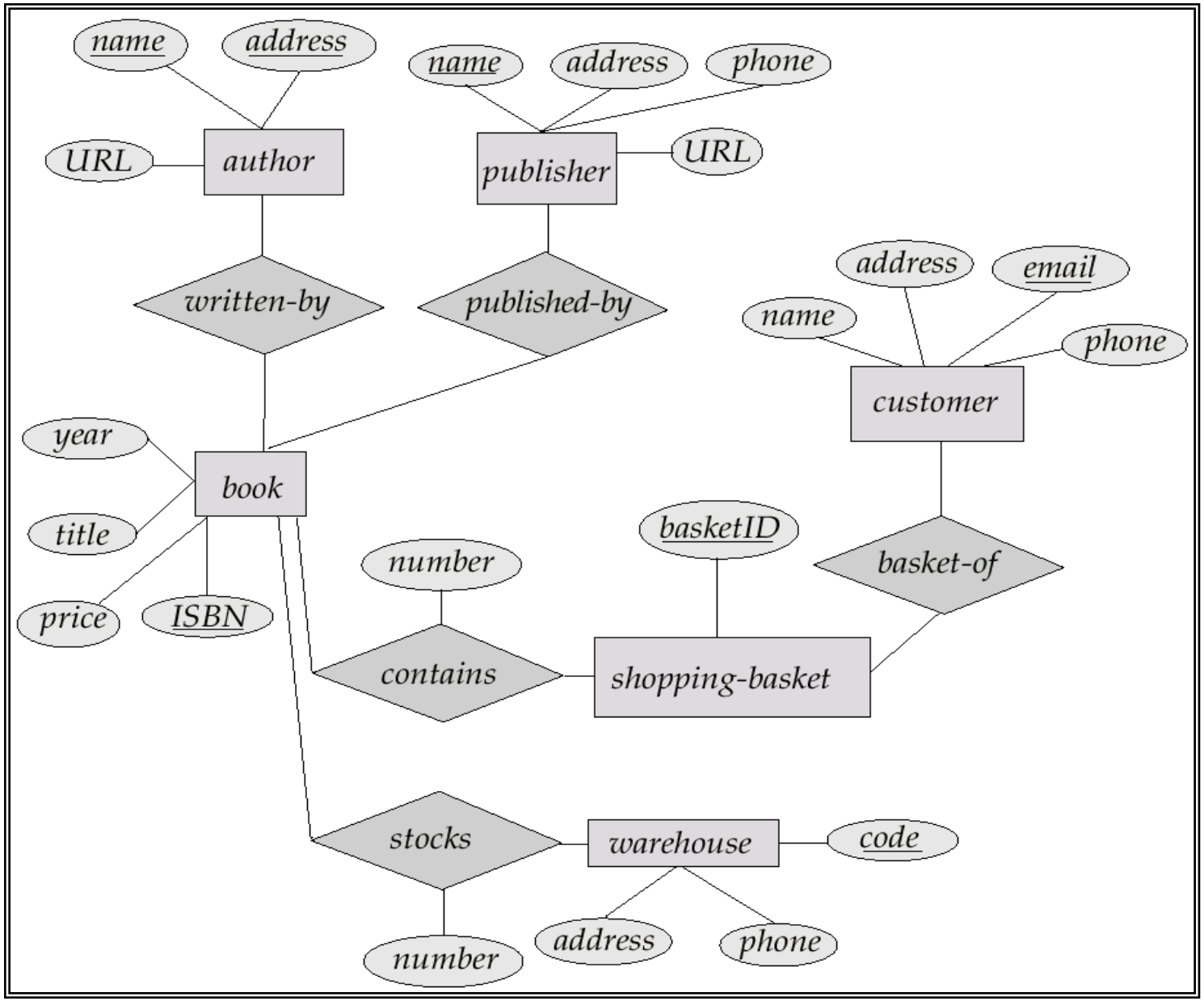
Step 4: Mapping of Binary 1:*N* Relationship Types

Step 5: Mapping of Binary *M*:*N* Relationship Types

Step 6: Mapping of Multivalued/Composite Attributes

Step 7: Mapping of *N*-ary Relationship Types

ER Diagram – Internet Book Shop



Exercise – ER Model….

- An Example Database Application called COMPANY which serves to illustrate the ER Model concepts and their schema design.

The following are collection from the Client.

Analysis…

- Company :
  Organized into Departments, Each Department has a name, no and manager who manages the department. The Company keeps track of the date that employee managing the department. A Department may have a Several locations.

Analysis…

- Department :
  A Department controls a number of Projects each of which has a unique name , no and a single Location.
- Employee:

  Name, Age, Gender, BirthDate, SSN, Address, Salary. An Employee is assigned to one department, may work on several projects which are not controlled by the department. Track of the number of hours per week is also controlled.

- Keep track of the dependents of each employee for insurance policies: We keep each dependent first name, gender, Date of birth and relationship to the employee.

Now to our Company…

DEPARTMENT

(Name, Number, { Locations }, Manager, Start Date)

PROJECT

(Name, Number, Location, Controlling Department)

EMPLOYEE

(Name (Fname, Lname), SSN, Gender, Address, Salary
Birthdate, Department, Supervisor, (Workson (Project , Hrs))

DEPENDENT

(Employee, Name, Gender, Birthdate, Relationship)

Example …Relationships

- Manage:
  - Department and Employee
  - Partial Participation
  - Relation Attribute: StartDate.
- Works For:
  - Department and Employee
  - Total Participation

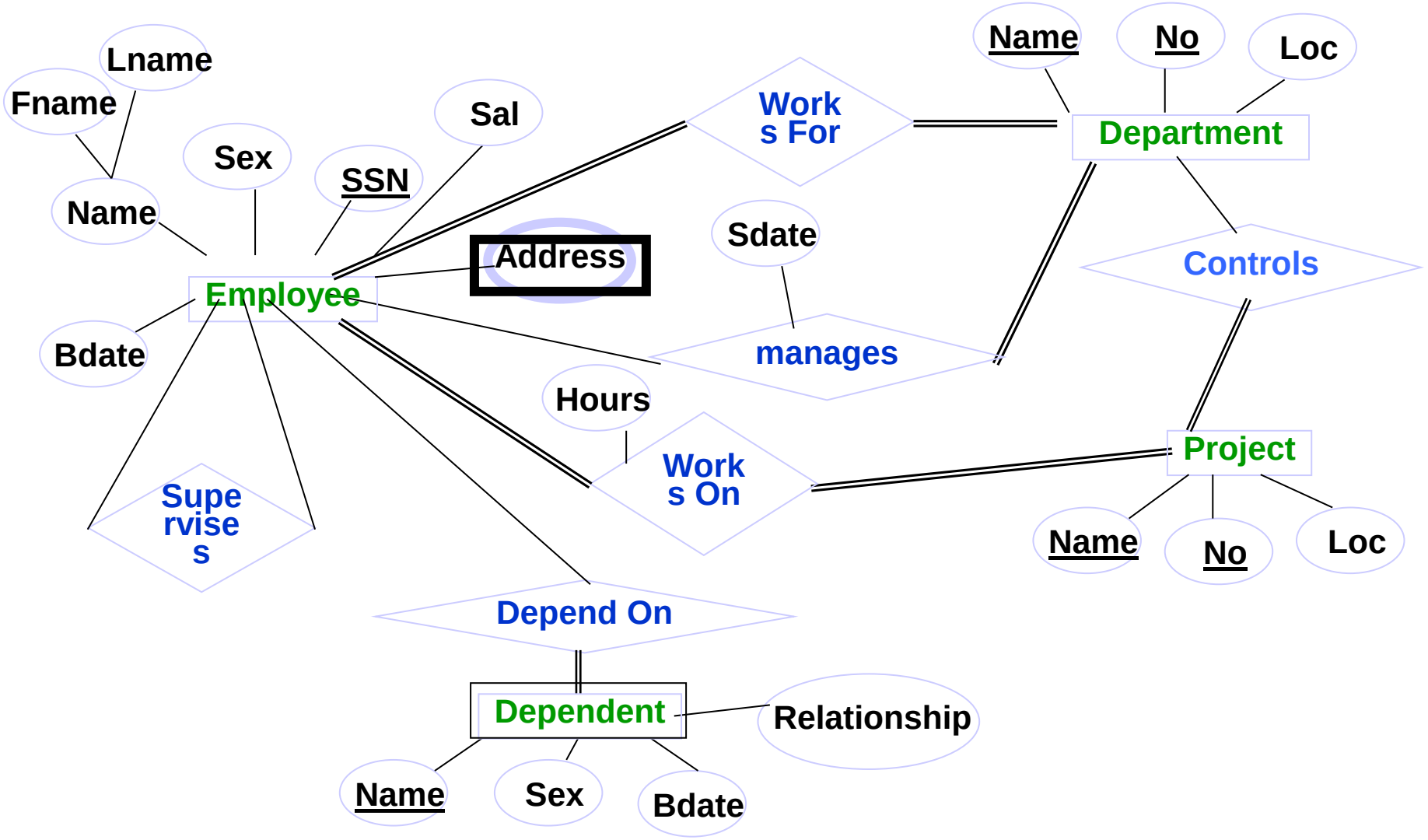
Example…

- Control :
  - Department , Project
  - Partial Participation from Department
  - Total Participation from Project
- Supervisor :
  - Employee, Employee
  - Partial and Recursive

Example …

- Works – On :
  - Project , Employee
  - Total Participation
  - Hours Worked is a RKA.
- Dependants of:
  - Employee, Dependant
  - Dependant is a Weaker entity
  - Dependant is Total , Employee is Partial.

One Possible mapping of the Problem Statement



Company Schema with Structural Constraints

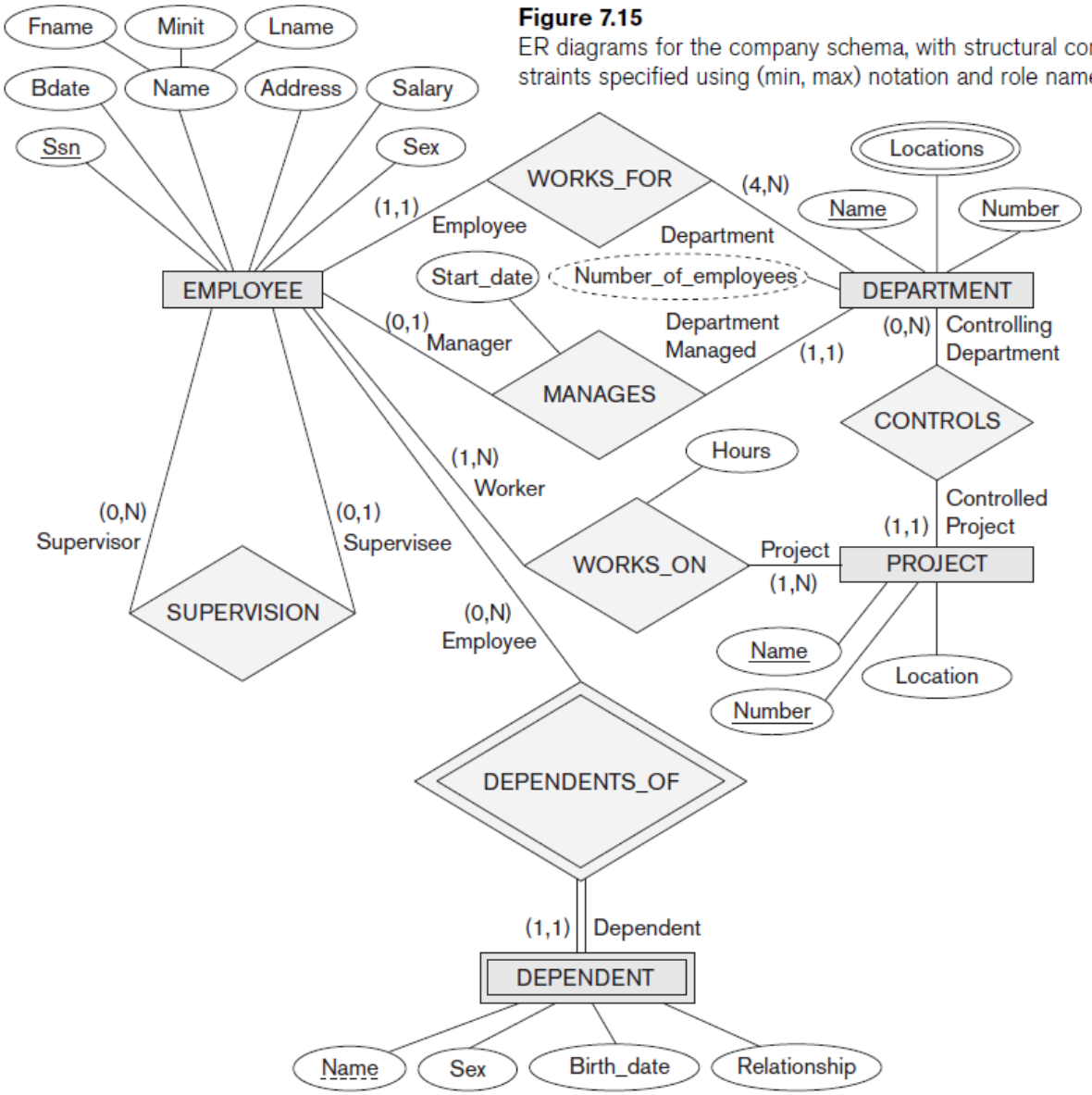


Figure 7.15
ER diagrams for the company schema, with structural constraints specified using (min, max) notation and role names.