



Conceptual Modeling of Data

Prof. S. Mehrotra

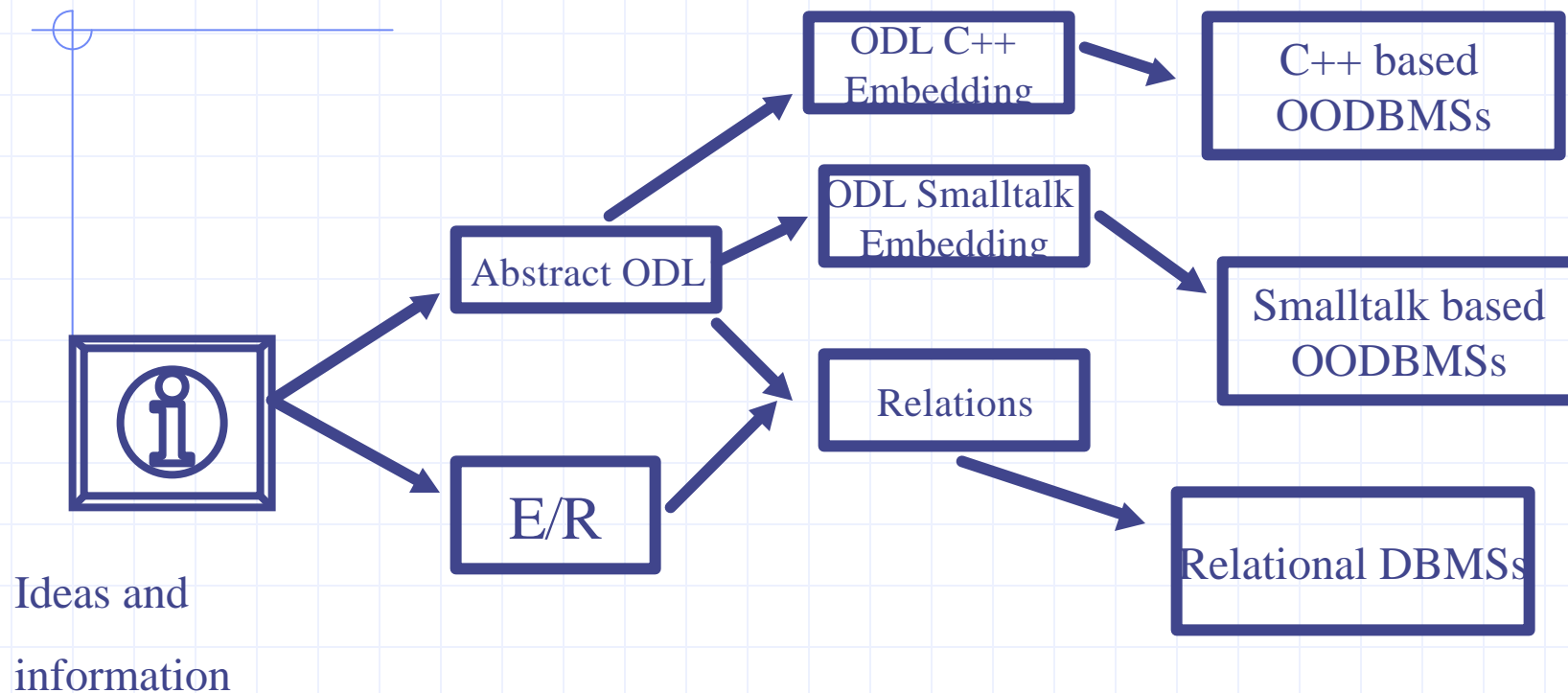
Information and Computer Science
Department

University of California at Irvine

Outline

- ◆ Database design process
- ◆ Entity/Relationship Model
 - Entity sets
 - Relationship sets
 - Constraints on entity sets
 - Constraints on relationship sets
 - Weak entity sets
 - Superclass/subclass relationships
 - Aggregation
- ◆ Good Design Principles
- ◆ Examples

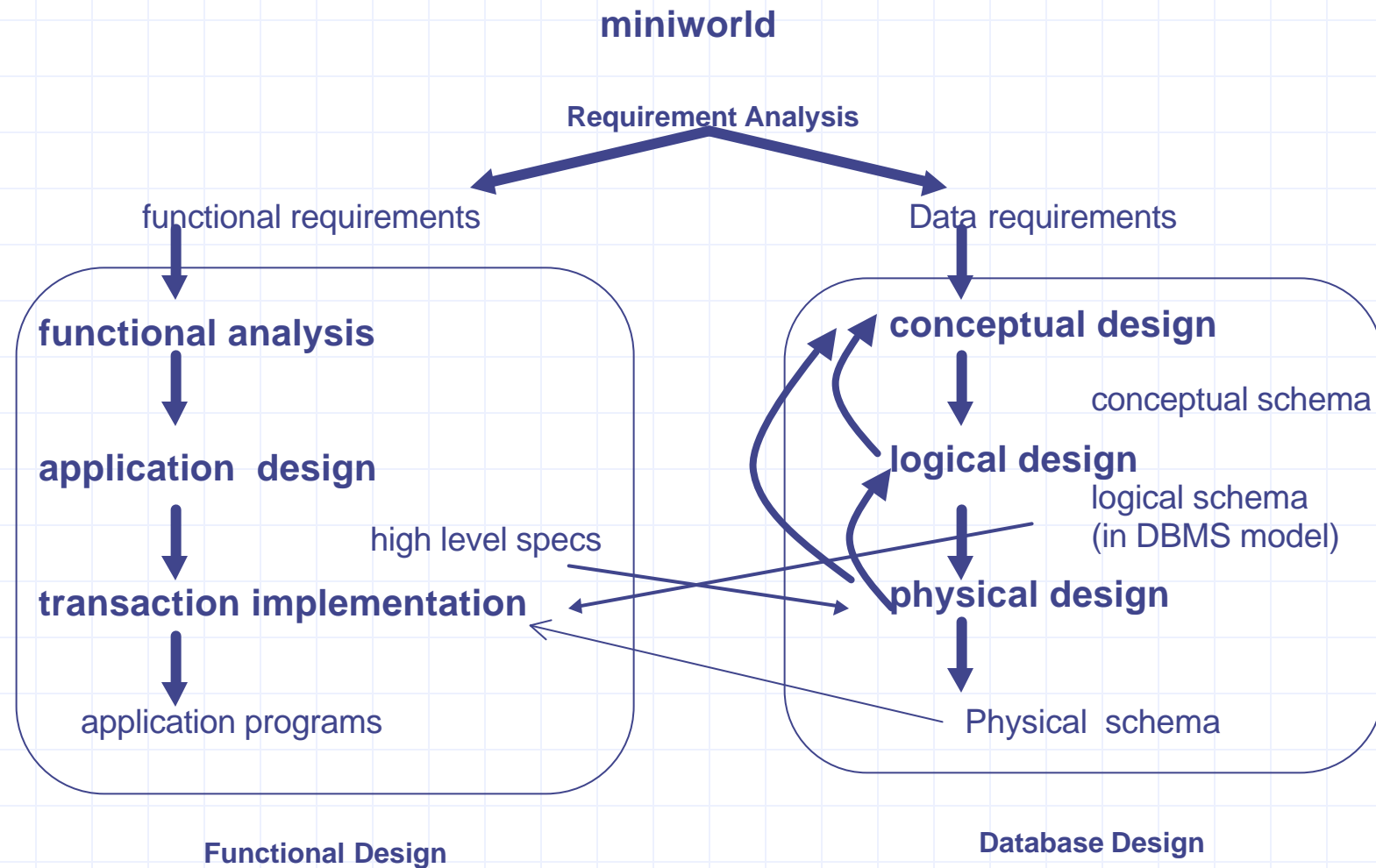
Conceptual Database Design



The design process depends upon the target DBMS

- E/R and ODL are popular models used for conceptual design
- ODL -- Object Definition Language is an emerging standard for OODBMSs

Database Design Process



Database Design Tools

- ◆ Help partially automate the design cycle.
- ◆ Graphical interface to specify conceptual schemas.
- ◆ Partially automated techniques to map to logical (DBMS dependent) model.
- ◆ Features of a good design tool:
 - *Iterative*: errors /shortcomings of original design found later can be corrected without full restart.
 - *Interactive*: any design choices made by system during design should be based on interaction with designer.
 - *Feedback*: a designer's change made at logical and/or physical levels should be automatically translated to changes at higher levels.
- ◆ Example Design tools: ERwin by LogicWorks.
- ◆ Database design tools integrated into CASE tools and supported by most modern DBMSs.

Requirements of a Conceptual Data Model

- ◆ *Expressiveness*: should be expressive enough to allow modeling of different types of relationships, objects and constraints of the miniworld.
- ◆ *Simplicity*: non-specialists should be able to understand
- ◆ *Minimality*: few basic powerful concepts that are non-overlapping
- ◆ *Diagrammatic Representation*: to ease interpretation
- ◆ *Formality*: There should be no ambiguity in the specification

Overview of Entity/Relationship (E/R) Model

- ◆ Entities
- ◆ Relationships
- ◆ Roles of entities in a relationship
- ◆ Constraints on entities:
 - domain constraints
 - key constraints
- ◆ Constraints on relationships
 - Cardinality Constraints (mapping constraints in SKS)
 - Participation Constraints (existence dependencies in SKS)
- ◆ Weak Entity Sets
- ◆ Multiway relationships
- ◆ Subclass/superclass Relationships
- ◆ Aggregation

Entities and Entity Sets

◆ Entities

- nouns, 'things' in the world.
- E.g., students, courses, employees, departments, flights, patients, ...

◆ Attributes

- properties of entities.
- E.g., course name, deptname, departure time, age, room#, ...

◆ Entity set -- a set of entities that have the same attributes.

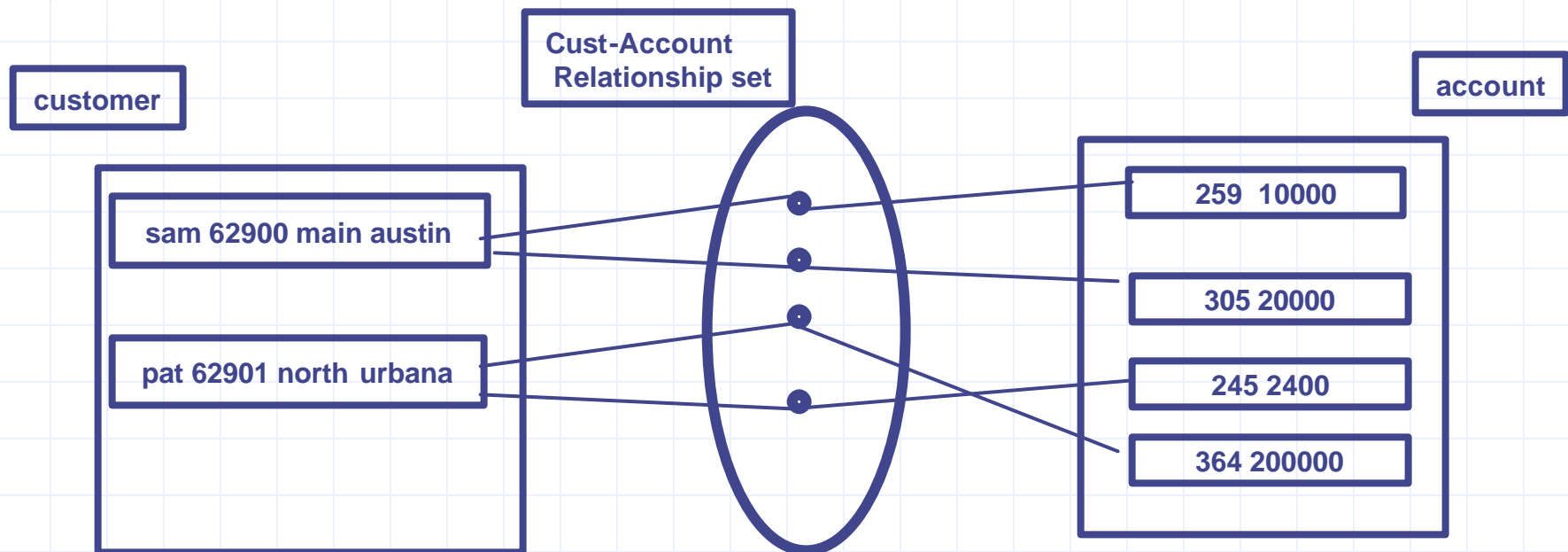
- In OO terminology, an entity set is similar to a class, and an entity similar to an instance

Attributes

- ◆ *single-valued vrs multi-valued:*
 - color of car could be multi-valued
 - salary of employee is single-valued
- ◆ *atomic vrs composite:*
 - age of a person is atomic
 - address of a person could be composite
- ◆ *stored vrs derived:*
 - derived attributes are those that can be derived from other attributes or entities, e.g., age can be derived from date of birth.
 - All other attributes are stored attributes

Relationships

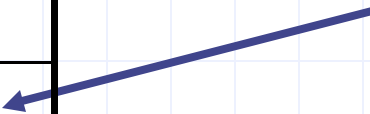
- ◆ *Relationship*:
 - association between multiple entities
- ◆ *Relationship Set*:
 - set of relationships over the same entity sets
- ◆ Binary, Ternary, 4-nary, ... relationship sets



Visualizing ER Relationships as a Table

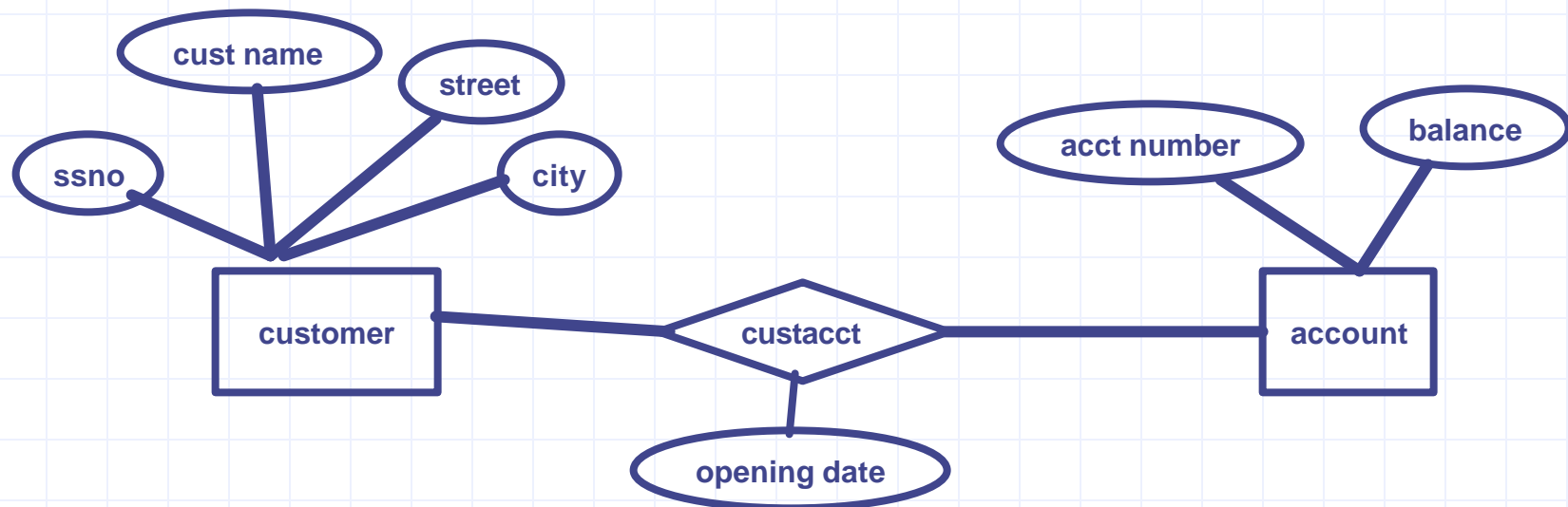
Customer	Account
John	1001
Megan	1001
Megan	2001

**Row in the table
represents the pair of
entities participating
in the relationship**



Relationship Set Corresponding to the Relationship Cust-Account

ER Diagram -- graphical representation of ER schema



- Entity set -- rectangles; attributes -- ellipses; dashed ellipse -- derived attribute; double ellipse -- multivalued attribute; relationship set -- diamonds; lines connect the respective relationship set with entity sets;
- Relationship sets may have 1 or many attributes associated with them -- known as relationship attributes.

Roles in a Relationship

- ◆ The function that an entity plays in a relationship is called its role
- ◆ Roles are normally not explicitly specified unless the meaning of the relationship needs clarification
- ◆ Roles needed when entity set is related to itself via a relationship.



Constraints on Entity Sets

◆ *Key Constraint:*

- With each entity set a notion of a *key* can be associated.
- A key is a set of attributes that uniquely identify an entity in entity set.
- Examples:
 - ◆ designer may specify that {ssno} is a key for a entity set customer entity with attributes {ssno, accountno, balance, name, address}
 - ◆ designer may specify that {accountno} is also a key , that is, no joint accounts are permitted.
- Denoted in ER diagram by underlining the attributes that form a key
- multiple keys may exist in which case one chosen as *primary key* and underlined. Other keys called secondary keys either not indicated or listed in a side comment attached to the diagram.

Constraints on Entity Sets (cont.)

- ◆ *Domain constraint:*
 - with each simple attribute a domain is associated. The value of the attribute for each entity is constrained to be in the domain.

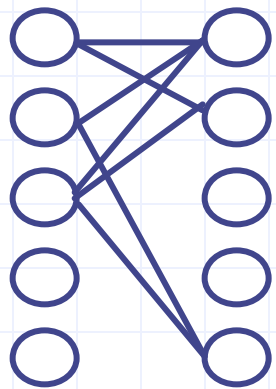
Cardinality Constraints on Relationship Sets

- ◆ Consider binary relationship set R between entity sets A and B
- ◆ *One to one*: an entity in A is associated with at most one entity in B , and an entity in B is associated with at most one entity in A .
 - an *employee* has only one *spouse* in a *married-to* relationship.
- ◆ *Many to One*: An entity in A is associated with many entity in B , an entity in B is associated with at most one entity in A .
 - an *employee* works in a single *department* but a *department* consists of many *employees*.

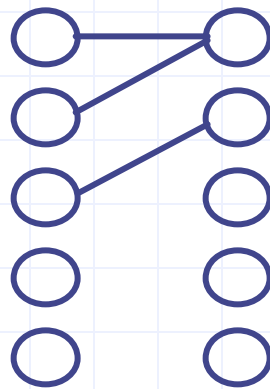
Cardinality Constraints on Relationship Sets (cont.)

- ◆ *Many to Many*: An entity in A is associated with many entities in B, and an entity in B is associated with many entities in A.
 - A *customer* may have many *bank accounts*. *Accounts* may be joint between multiple *customers*.

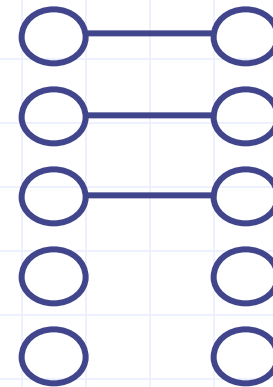
Multiplicity of Relationships



Many-to-many



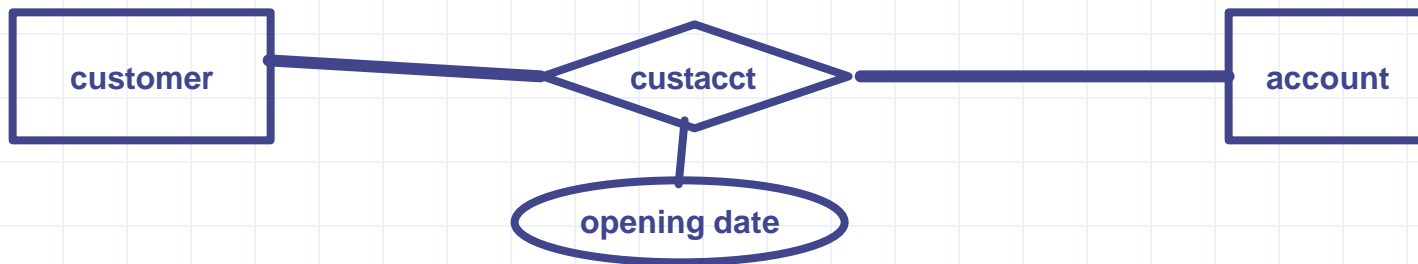
Many-to-one



One-to-one

multiplicity of relationship in ER diagram represented by an arrow pointing to “one”

Many to Many Relationship



Customer	Account	Start Date
John	1001	Jan 20 th 1999
Megan	1001	March 16 th 1999
Megan	2001	Feb 18 th 1994

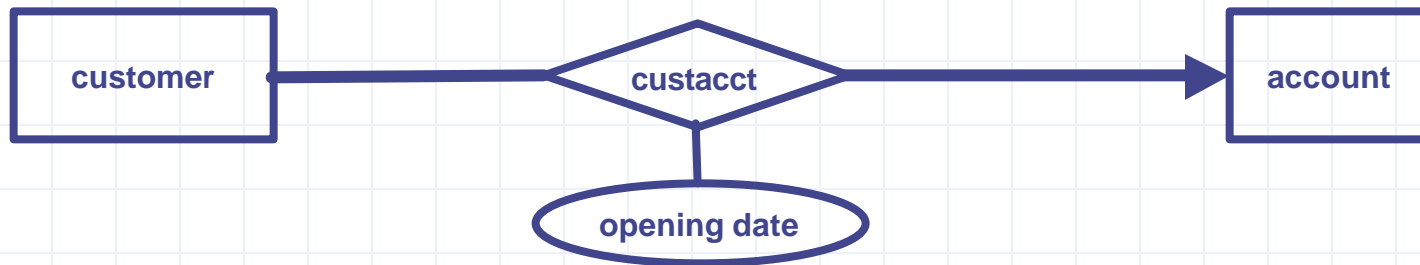
legal

Customer	Account	Start Date
John	1001	Jan 20 th 1999
Megan	1001	March 16 th 1999

legal

- ◆ Multiple customers can share an account
- ◆ Many accounts may have one owner

Many to One Relationship



Customer	Account	Start Date
John	1001	Jan 20 th 1999
Megan	1001	March 16 th 1999
Megan	2001	Feb 18 th 1994

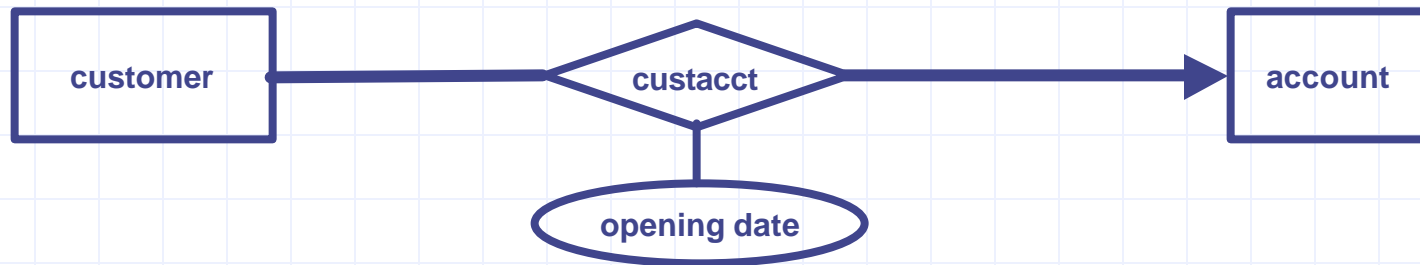
Illegal

Customer	Account	Start Date
John	1001	Jan 20 th 1999
Megan	1001	March 16 th 1999

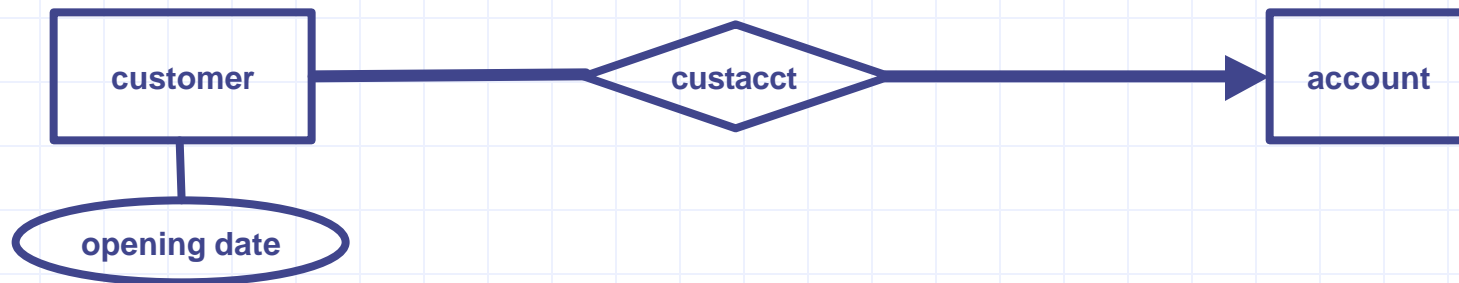
legal

- ◆ Multiple customers can share an account but one customer can have only one account.

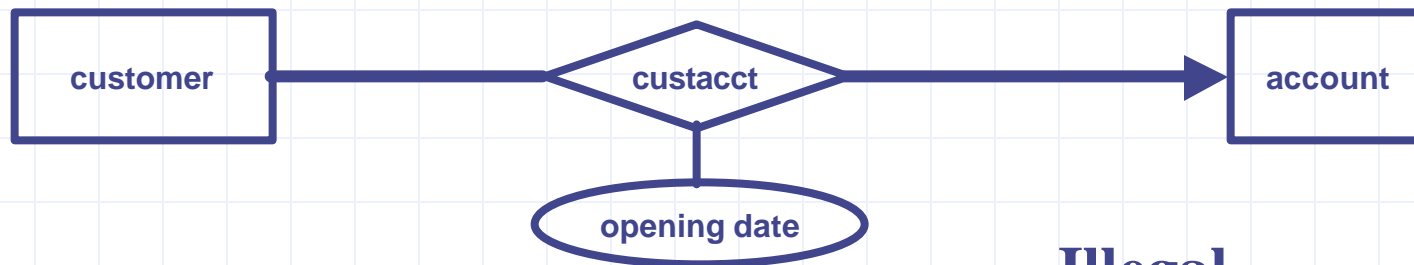
Relationship Attribute in a Many to One Relationship



- ◆ In a Many-One relationship, relationship attributes can be repositioned to the entity set on the many side.



One to One Relationship



- ◆ 1 customer can have 1 account.
- ◆ One account can be owned by 1 customer
- ◆ relationship attributes can be shifted to either of the entity sets

Illegal

Customer	Account	Start Date
John	1001	Jan 20 th 1999
Megan	1001	March 16 th 1999

Illegal

Customer	Account	Start Date
Megan	1001	March 16 th 1999
John	2001	Feb 18 th 1994

Legal

Customer	Account	Start Date
Megan	1001	March 16 th 1999
Megan	2001	Feb 18 th 1994

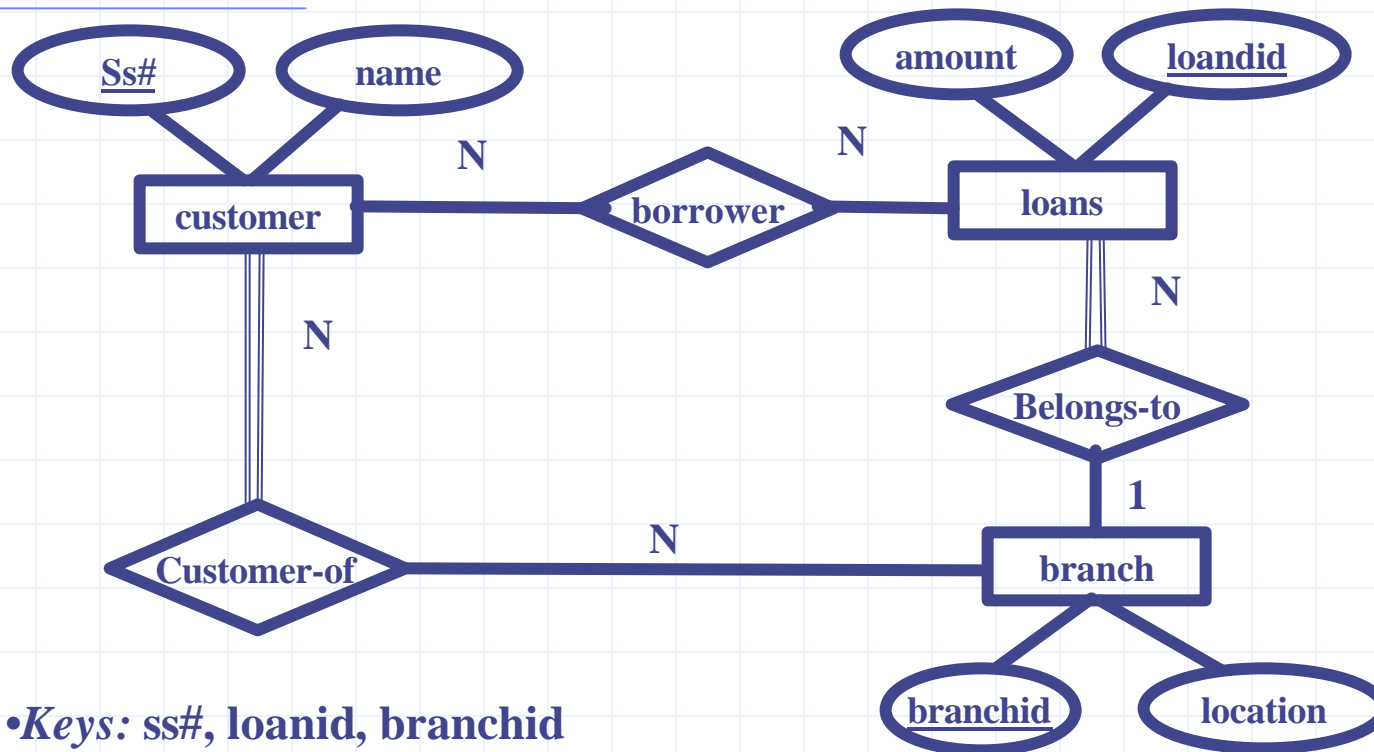
Participation Constraints

- ◆ Participation of an entity set A in the relationship set $R1$ can be *total*
- ◆ Each entity in entity set A is constrained to be related to other entities via relationship $R1$.
- ◆ *Examples*
 - participation of entity set *employee* in the relationship *belongs-to* with the entity set *department* may be total.
 - Each *employee* must *belong to* at least one *department*.

Participation Constraints

- ◆ *total participation* is also called *existential dependency*
- ◆ If an entity does not have a *total participation* in a relationship, it is said to have a *partial participation*
- ◆ In ER diagram, total participation represented using a double line between the relationship and entity set that totally participates in the relationship

Example

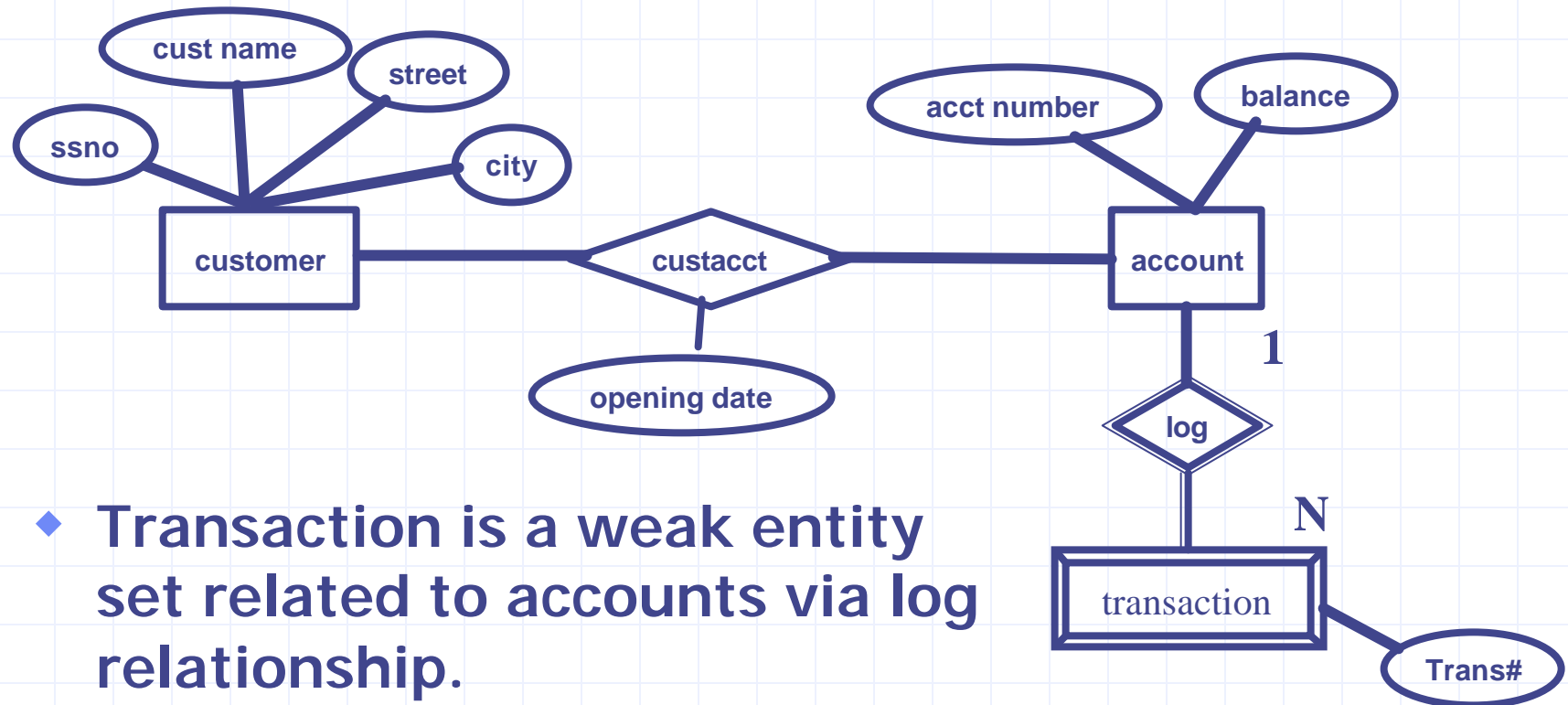


- **Keys:** *ss#*, *loanid*, *branchid*
- **Cardinality constraint:** each loan belongs to a single branch
- **Participation constraints:**
- **Each customer must be a customer of at least one branch**
- **Each loan must belong to some branch**

Weak Entity Sets

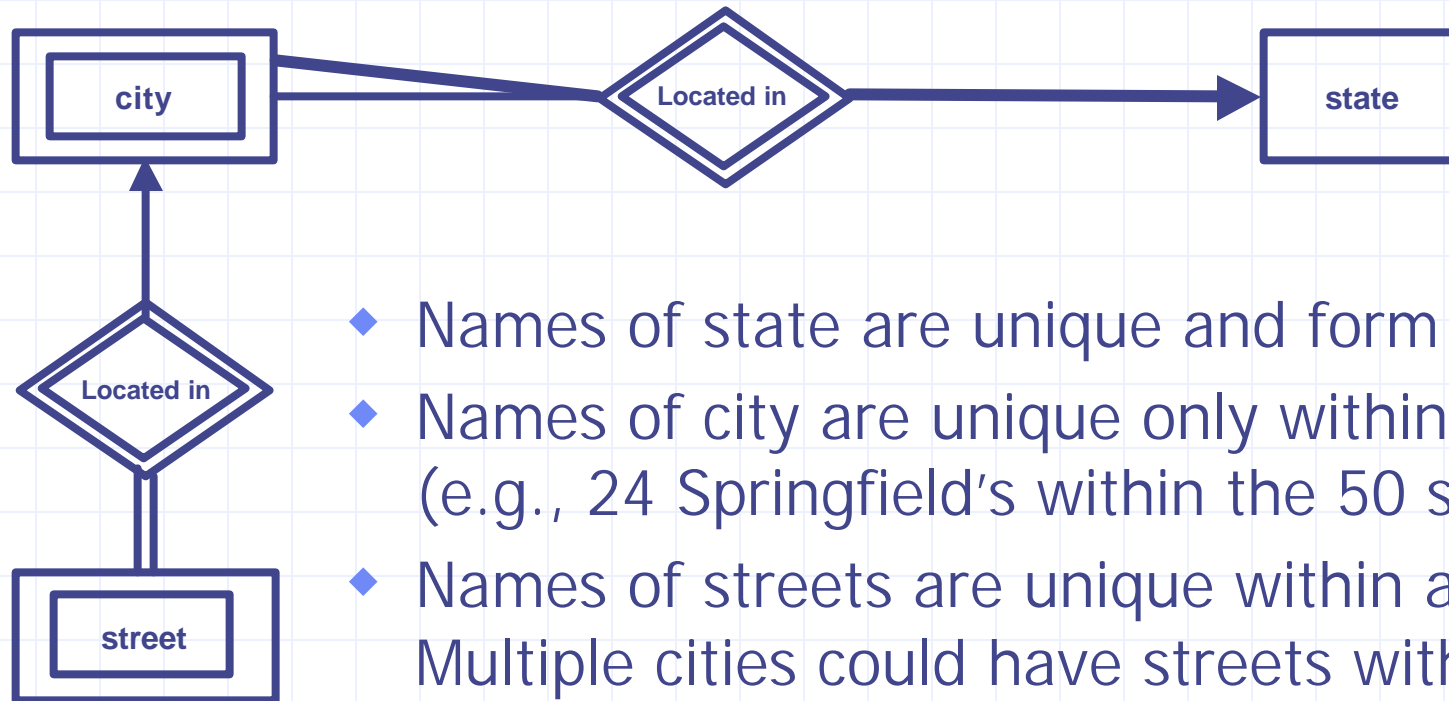
- ◆ Entity sets that do not have sufficient attributes to form a key are called *weak entity sets*.
- ◆ A weak entity set *existentially depend* upon (one or more) strong entity sets via a one-to-many relationship from whom they derive their key
- ◆ A weak entity set may have a *discriminator* (or a *partial key*) that distinguish between weak entities related to the same strong entity
- ◆ key of weak entity set = Key of owner entity set(s) + discriminator

Weak Entity Sets (cont.)



- ◆ Transaction is a weak entity set related to accounts via log relationship.
- ◆ Trans# distinguish different transactions on same account

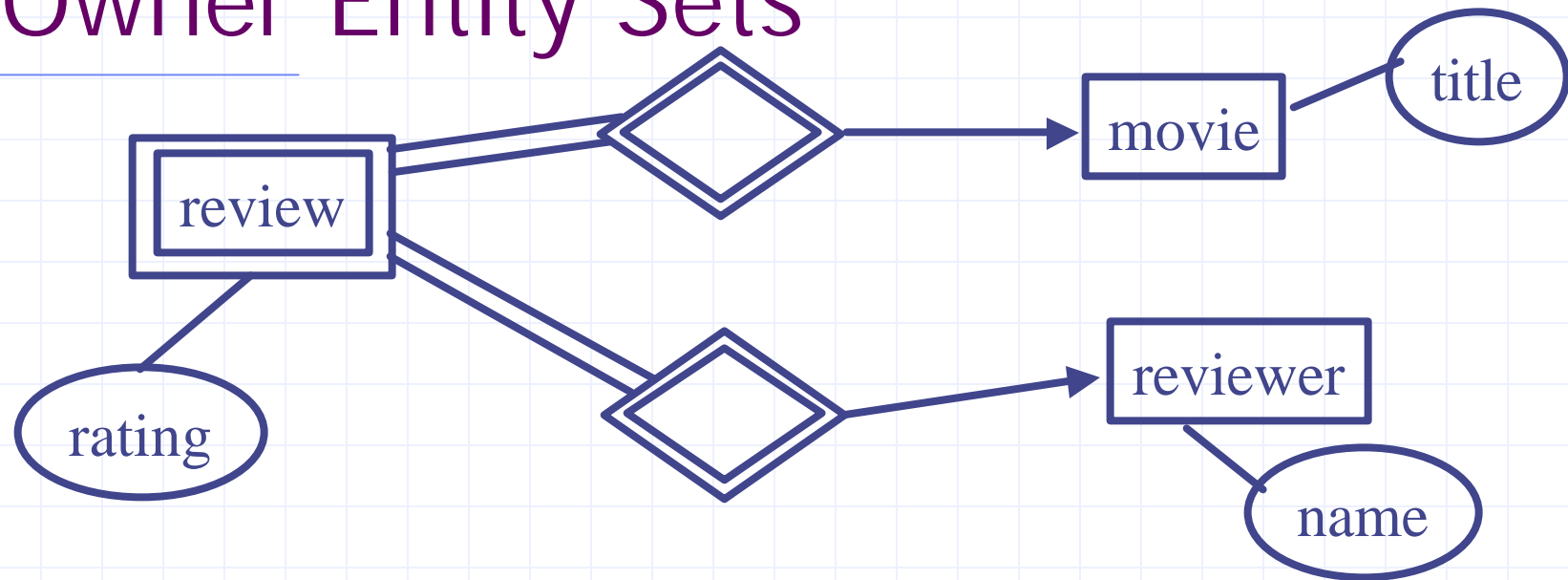
A Chain of Weak Entity Sets



- ◆ Names of state are unique and form the key.
- ◆ Names of city are unique only within a state (e.g., 24 Springfield's within the 50 states).
- ◆ Names of streets are unique within a city. Multiple cities could have streets with the same name.

Example illustrating that a weak entity set might itself participate as owner in an identifying relationship with another weak entity set.

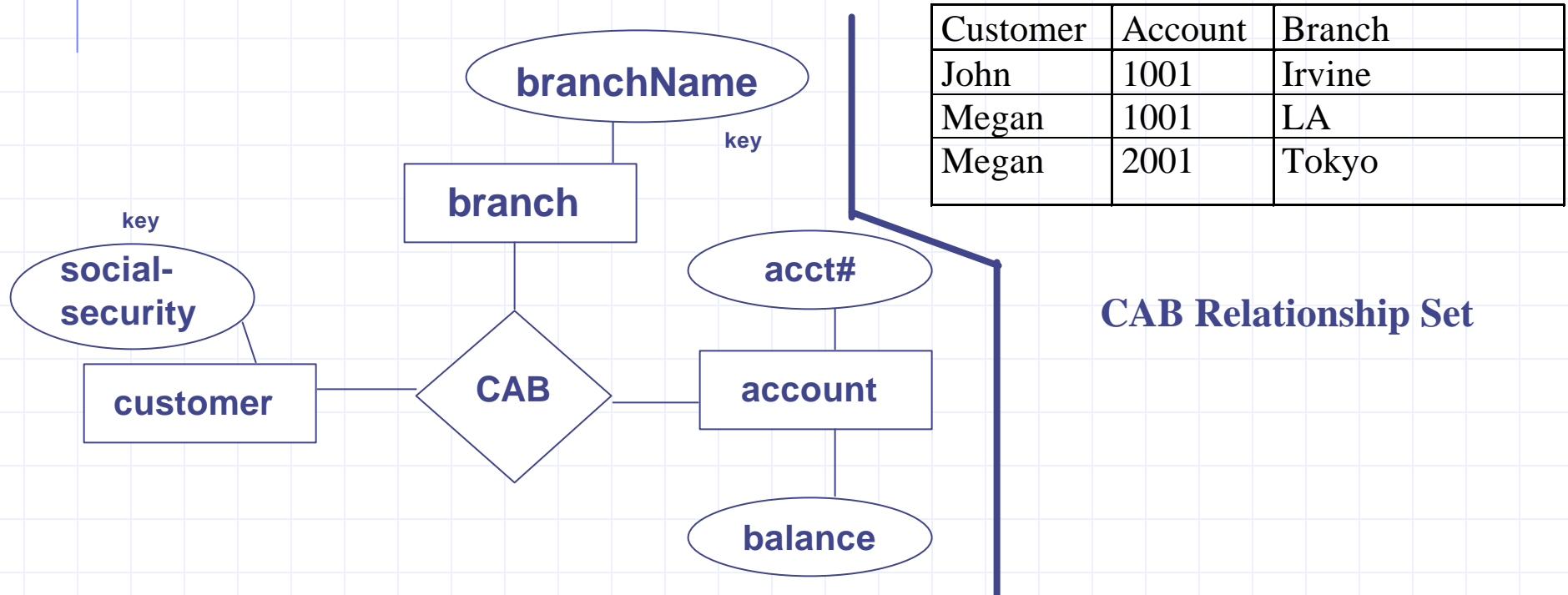
A Weak Entity Set with Multiple Owner Entity Sets



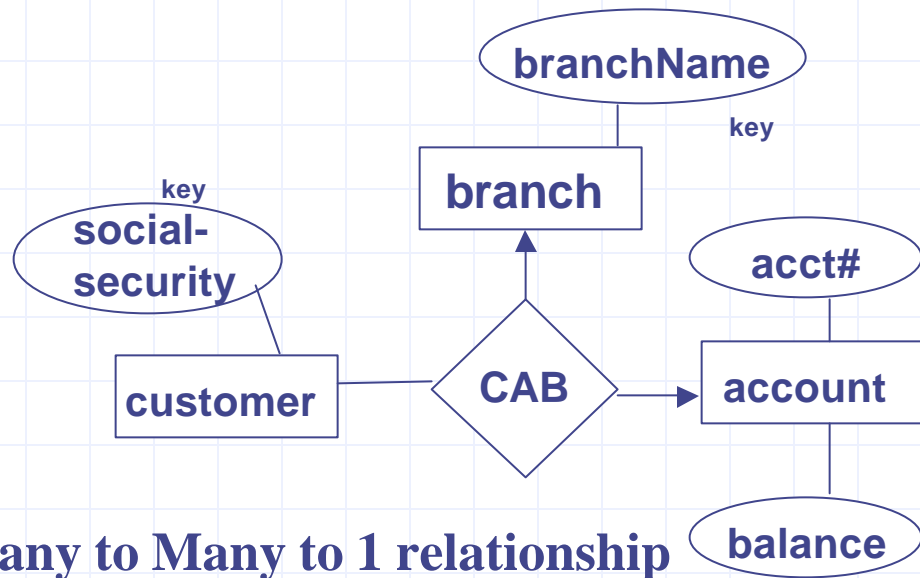
- ◆ Reviewers review movie and assign a rating -- thumb up/thumbs down.
- ◆ Review is a weak entity set whose owner sets correspond to both the movie and the reviewer entity sets.
- ◆ Key for the review entity set = key of movie + key of reviewer

Multiway Relationships

- ◆ Usually binary relationships (connecting two E.S.) suffice.
- ◆ However, there are some cases where three or more E.S. must be connected by one relationship.
- ◆ Similar to binary relationship, cardinality and participation constraints defined over multiway relationships



Cardinality Constraint over Multiway Relationships



Many to Many to 1 relationship

Customer	Account	branch
John	1001	Irvine
Megan	1001	Dallas
Megan	1002	Tokyo
Megan	1001	Tokyo

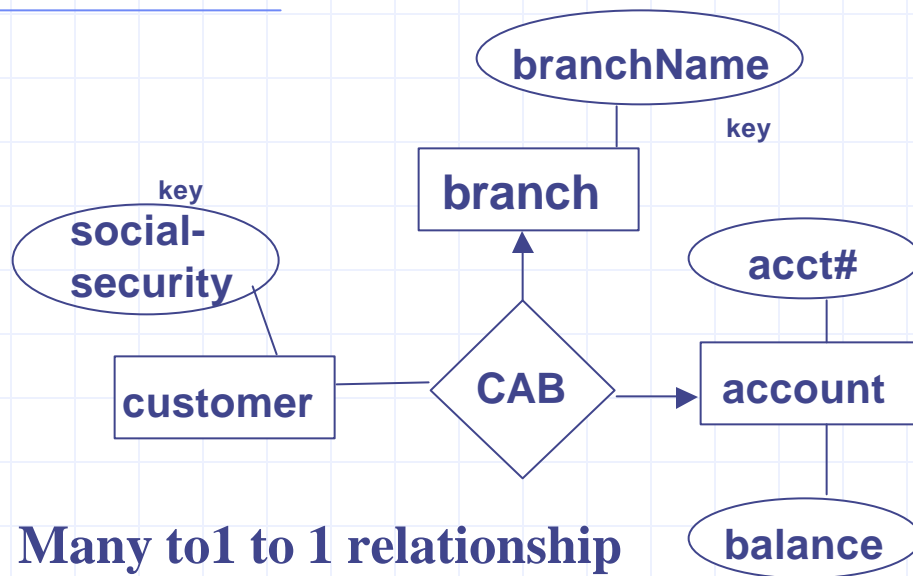
Illegal: Megan has account 1001 at 2 branches

- ◆ Interpretation:
 - Each pair of customer and account determine the branch (that is, have a single branch related to them).

Customer	Account	branch
John	1001	Irvine
Megan	1001	Dallas
Megan	1002	Tokyo
Megan	1003	Tokyo

Legal

Cardinality Constraint over Multiway Relationships



Customer	Account	branch
John	1001	Irvine
Megan	1001	Dallas
Megan	1002	Tokyo
Megan	1003	Tokyo

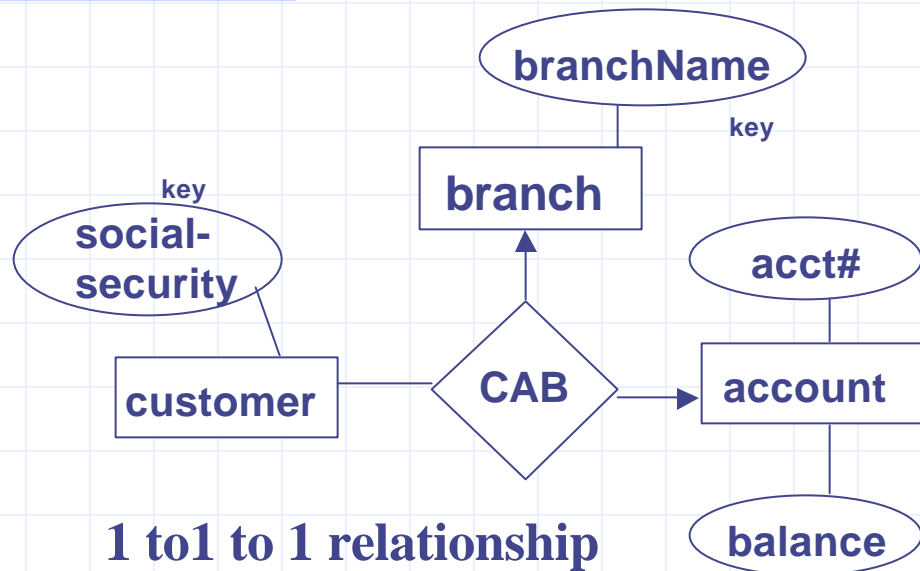
Illegal: Megan has 2 accounts in Tokyo Branch

- ◆ Interpretation:
 - Each (customer, branch) related to a single account
 - Each (customer, account) pair related to a single branch

Customer	Account	branch
John	1001	Irvine
Megan	1001	Dallas
Megan	1002	Tokyo
John	1002	Tokyo

Legal

Cardinality Constraint over Multiway Relationships



Customer	Account	branch
John	1001	Irvine
Megan	1001	Dallas
Megan	1002	Tokyo
John	1002	Tokyo

Illegal: Both John and Megan have account 1002 in Tokyo Branch

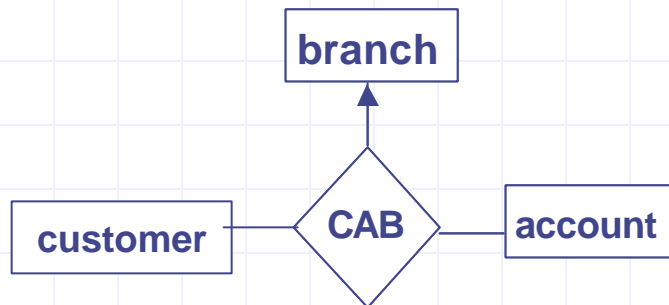
◆ Interpretation:

- Each (customer, branch) related to a single account
- Each (customer, account) pair related to a single branch
- Each (branch, account) pair can have single customer

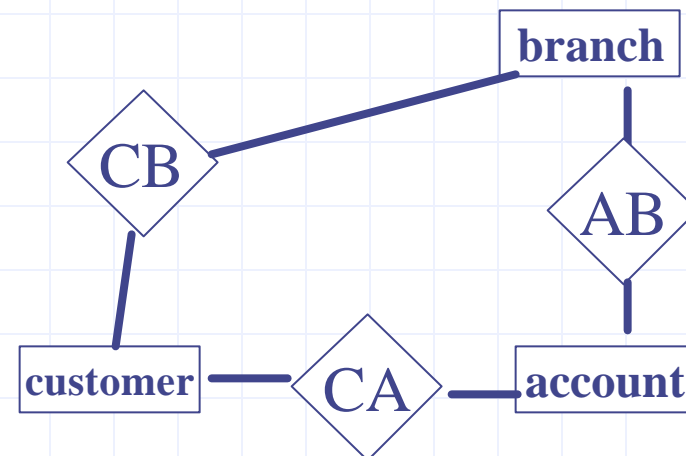
Customer	Account	branch
John	1001	Irvine
Megan	1001	Dallas
Megan	1002	Tokyo

Legal

Representing Ternary Relationship Using Binary Relationships



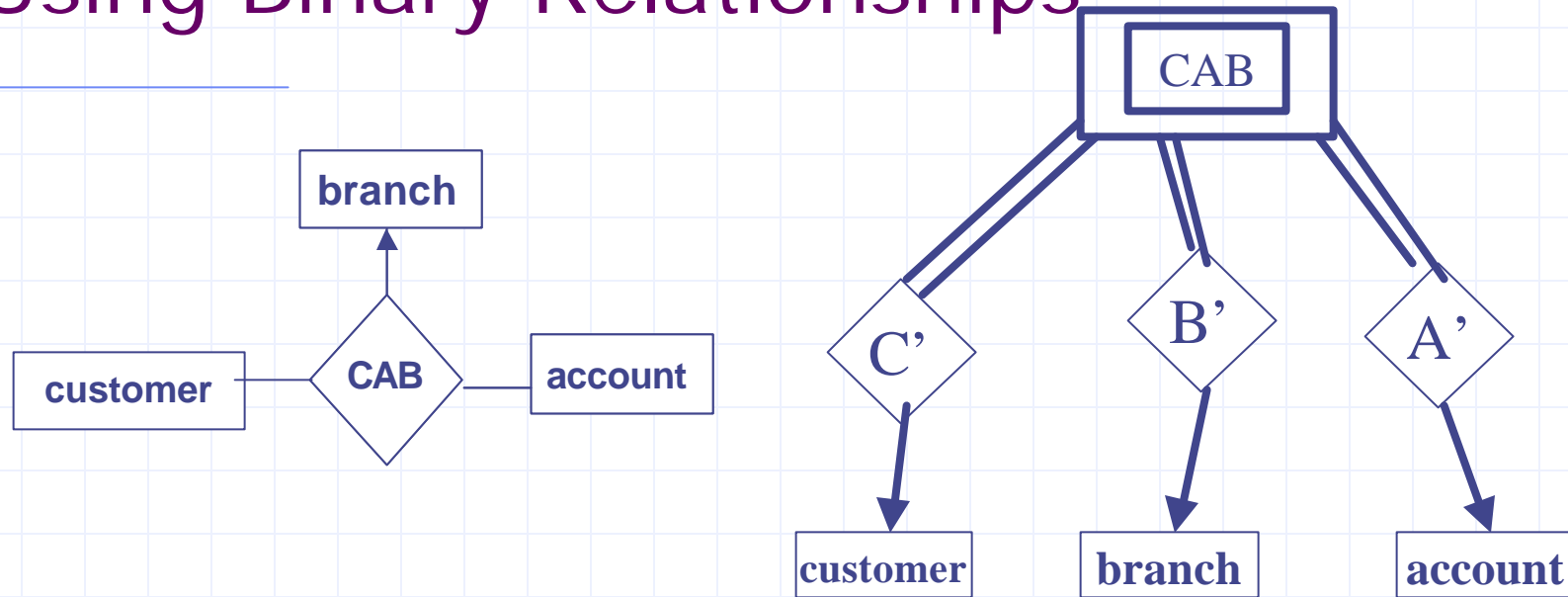
Customer	Account	branch
C2	A1	B1
C1	A2	B1
C1	A1	B2



The above CAB relationship Set cannot be represented using the Schema consisting of binary relationships shown above!!

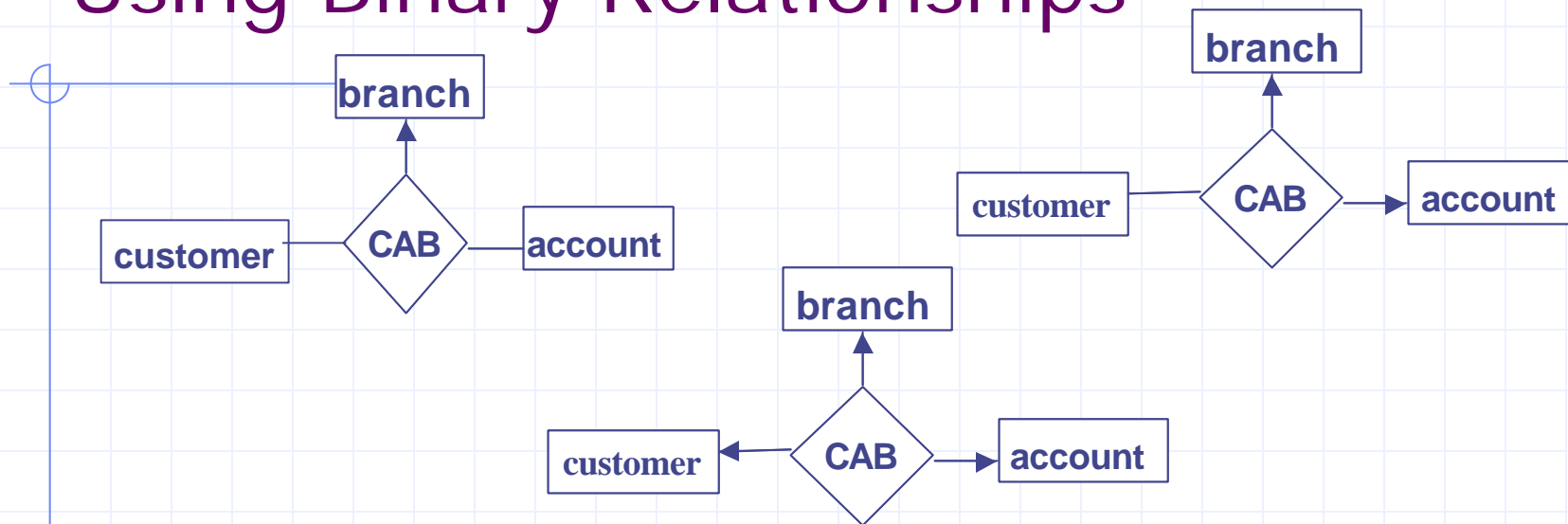
- ◆ Hence, above Schema using binary relationships does not correctly capture the information represented by the ternary relationship.

Representing Ternary Relationship Using Binary Relationships



- ◆ The CAB relationship is represented as a weak entity set that depends upon the customer, branch and account entity sets.
- ◆ This schema using binary relationship fully captures the ternary relationship.

Representing Ternary Relationship Using Binary Relationships



- ◆ Previous mapping technique works for many-many-many relationship.
- ◆ How to convert the many-many-1, many-1-1, 1-1-1 ternary relationships into binary relationships?
- ◆ In general, it is always possible to convert any ternary (or multiway relationship) into a collection of binary relationships without losing information!!
- ◆ However, the conversions can be quite complex and resulting schema unnatural

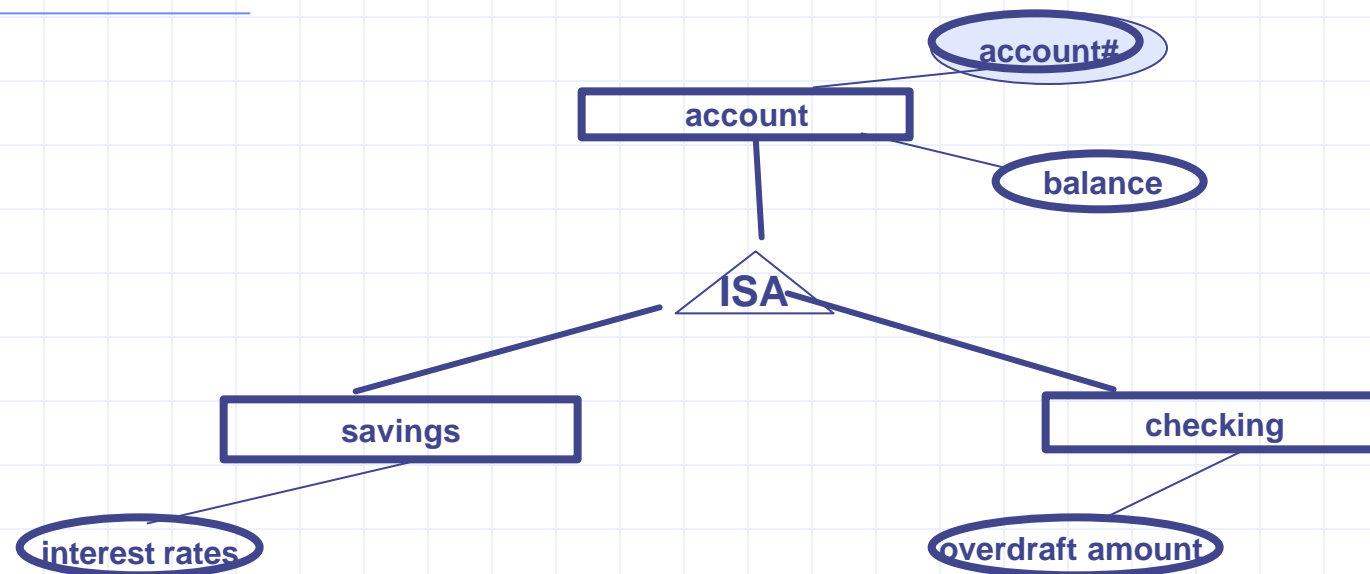
Limitations of the Basic ER Model Studied So Far

- ◆ Lots of times an entity set has members that have special properties not associated with all the members of the entity set.
- ◆ E.g., the set of checking accounts and savings accounts are a subset of the set of accounts. Checking has a *overdraft amount*, and savings has a *interest-rate*.

Limitations of the Basic ER Model Studied So Far

- ◆ How to represent this in the ER model:
 - associate an attribute -- account-type with the accounts entity set
 - Problems:
 - ◆ different attributes may be associated with the account depending on its type
 - checking: overdraft amount
 - savings: interest rate
 - ◆ depending upon its type, savings and checking accounts may participate in different relationships.
 - Another approach:
 - ◆ entity sets: checking, savings, and accounts.
 - ◆ relationships: 1-1 between checking and accounts, and 1-1 between savings and accounts
 - Problems:
 - ◆ Not intuitive: checking and savings are represented as entities different from accounts, even though they are accounts
 - ◆ Redundancy of information: info about accounts represented both in checking / savings as well as account entity set
 - ◆ Potential Errors: Same account could be erroneously associated with both checking as well as savings.

Subclass/Supersclass Relationships



- ◆ savings and checking are subclasses of the account entity set
- ◆ account is a superclass of savings and checking entity sets
- ◆ An entity in a subclass has to belong to superclass as well -- that is, every savings account is also an account. Similarly every checking account is also an account
- ◆ Attribute Inheritance: subclasses inherit all the attributes of the superclass. Similarly, subclasses inherit all relationships in which the superclass participates

Reason why Superclass/Subclass relationships arise in ER Schemas

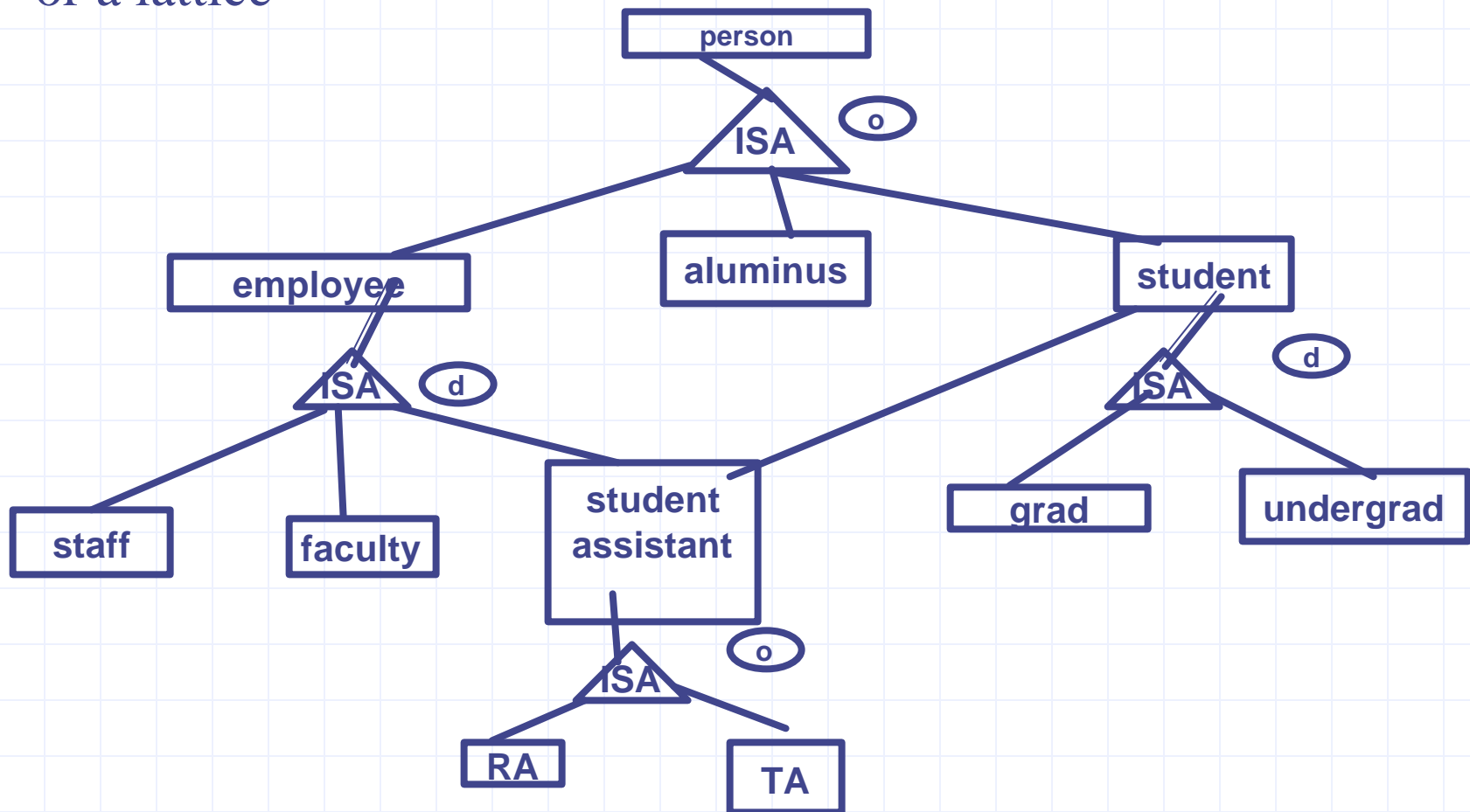
- ◆ Superclass and Subclass relationships arise during schema design due to the process of *specialization* and *generalization*
- ◆ *Specialization*: process of classifying a class of objects into more specialized subclasses
 - E.g., during design, we begin with an employee entity set. We then specialize the employee set into different types of employees.
- ◆ *Generalization*: Reverse of specialization -- it is a process of synthesis of two or more (lower level) entity sets to produce a higher-level entity set.
 - E.g., during design, we have identified a car, a sports utility vehicle, and a truck. We generalize these classes to create an automobile entity set.

Types of Class/Subclass Relationships

- ◆ Disjoint vrs Overlapping:
 - if the subclasses of the entity set do not overlap then it is disjoint (denoted by a 'd' next to ISA triangle).
 - Else, overlapping (denoted by a 'o' next to ISA triangle)
- ◆ Total vrs Partial:
 - If an entity in a superclass belongs to atleast one of the subclasses, then total. (denoted by a double line from superclass to ISA triangle)
 - Else, partial
- ◆ Key of entity set corresponding to the subclass is the same as the key for the superclass.

Superclass/Subclass Lattice

Class/Subclass relationships might form a hierarchy (tree) or a lattice

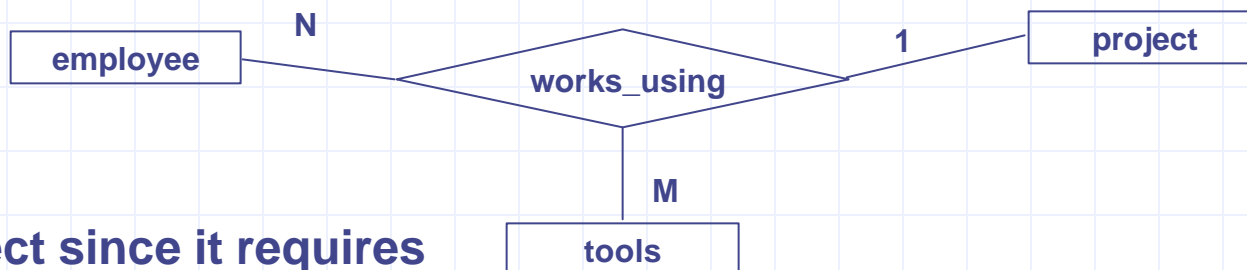


Multiple Inheritance

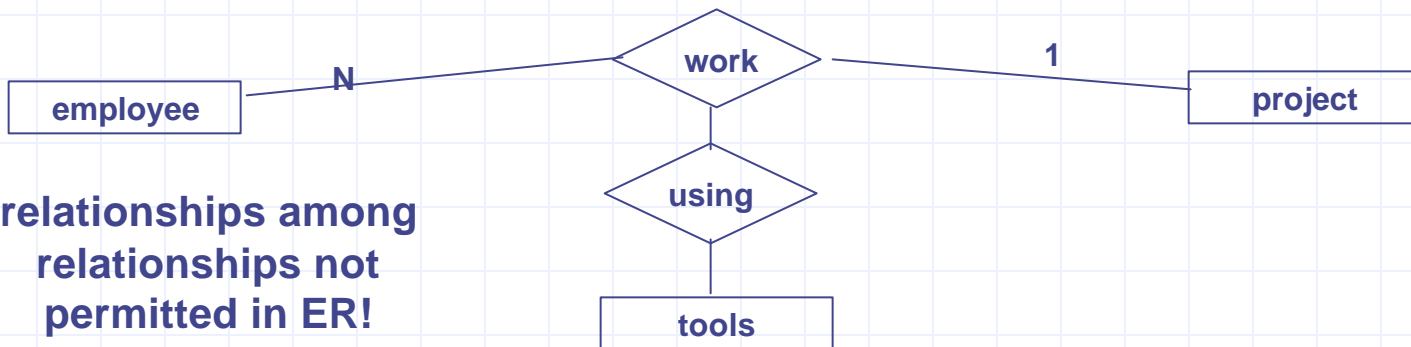
- ◆ In a class/subclass relationship, the subclass inherits all its attributes from the superclass.
- ◆ If a subclass has 2 or more superclasses, then subclass inherits from all the superclasses (multiple inheritance)
- ◆ How should conflicts be resolved?
- ◆ Example:
 - Employee Entity Set: with an attribute country denoting the country of citizenship
 - Asians Entity Set: with an attribute country denoting the country from which a particular person originated.
 - Asian_Employee Entity set is a subclass of both Employee and Asians. However, what does country attribute of the Asian_Employee correspond to.
- ◆ ER model mute on multiple inheritance

Limitations of ER Model

We wish to represent that an employee works on a specific project possibly using multiple tools

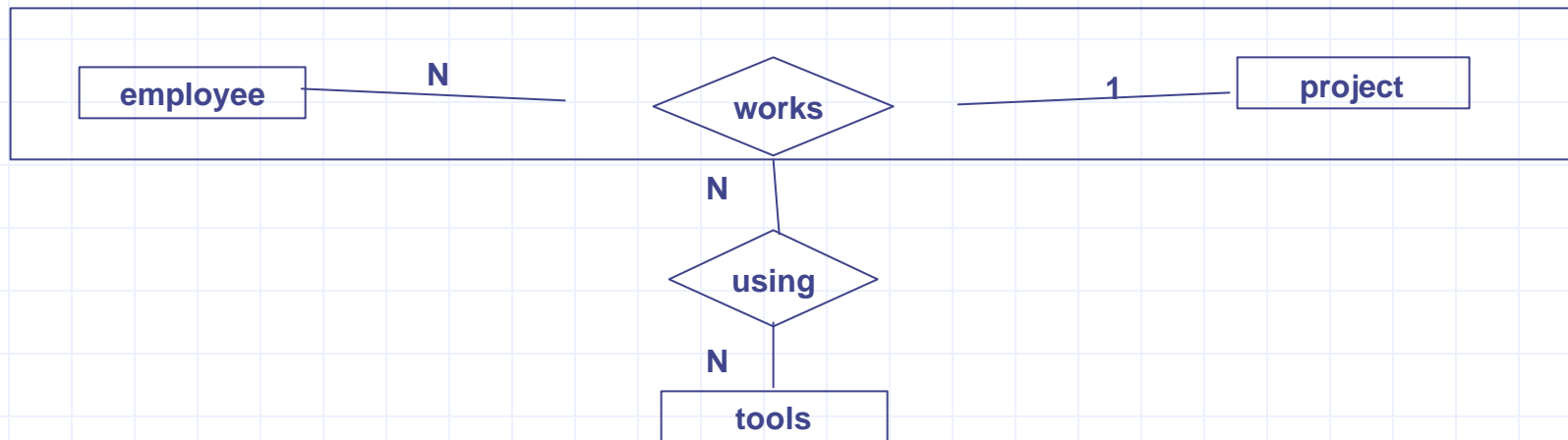


incorrect since it requires each project to use tools



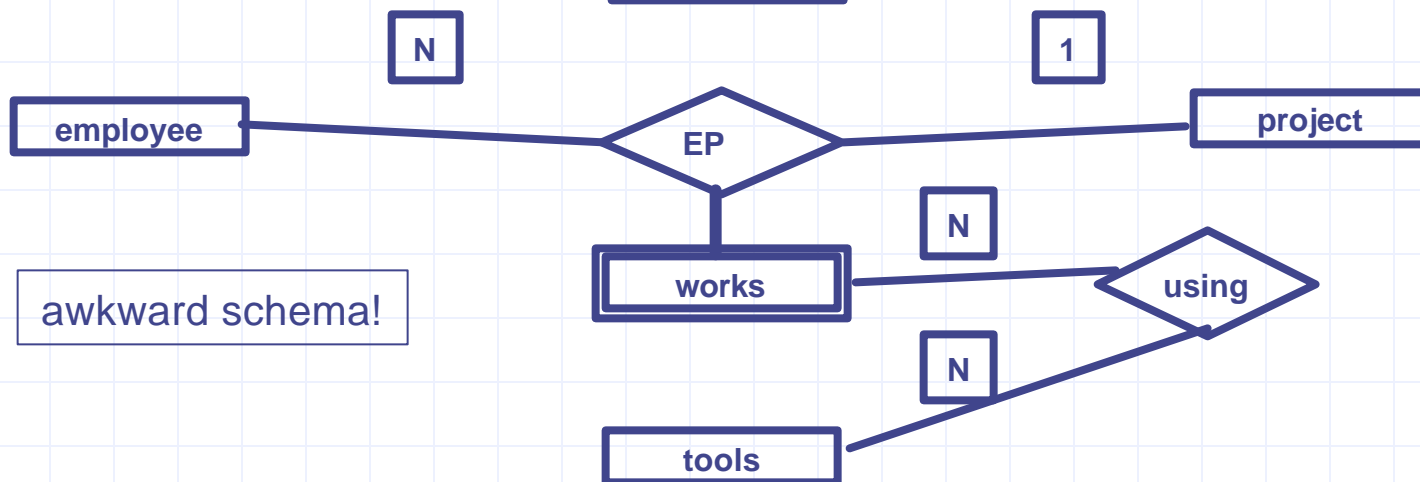
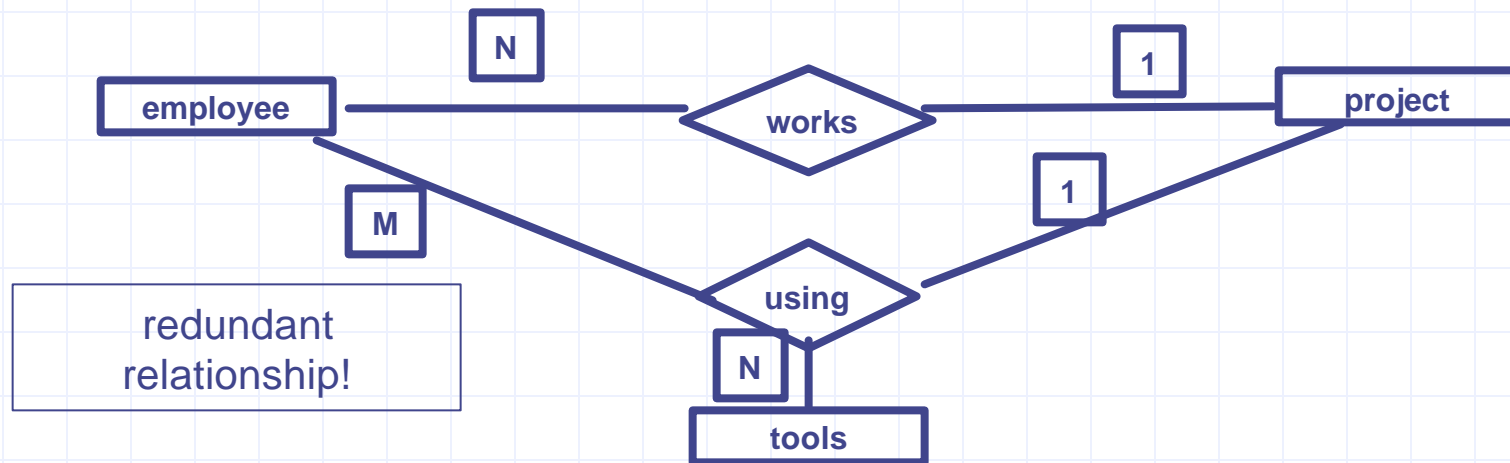
relationships among relationships not permitted in ER!

Aggregation



- ◆ Treat the relationship set work and the entity sets employee and projects as a higher level entity set-- an *aggregate entity set*
- ◆ Permit relationships between aggregate entity sets and other entity sets

Representation without Aggregation in ER Model



Review of ER Model

◆ Basic Model:

- Entities : strong, weak
- Attributes associated with entity sets and relationships
- Relationships: binary, ternary, ...
- Role of entity sets in a relationship
- Constraints on entity set: domain constraints, key constraint
- Constraint on relationships: cardinality -- 1-1, 1-N, M-N, participation (also called existential) --total vrs partial

◆ Extended Model:

- Notion of superclass and subclass
- Superclass/subclass relationships: disjoint vrs overlapping, total vrs partial
- Notion of aggregation

E/R Design Cycle

- ◆ Good design important since schemas do not change often
- ◆ The first version is almost always wrong.

Typical Schema Design Cycle

1: Requirement Analysis: Learn about the application.

- what problem does the application solve, what questions does the application ask about the data, what data does the application need to answer these questions.

2: Design a trial schema

- **top-down strategy:** define high level concepts and then use successive refinements
- **bottom-up strategy:** start with schema containing basic abstractions and then combine or add to them

3: Evaluate schema for quality and completeness.

- **consider the future: how is the application likely to change? Account for change**

4: Iterate until satisfied

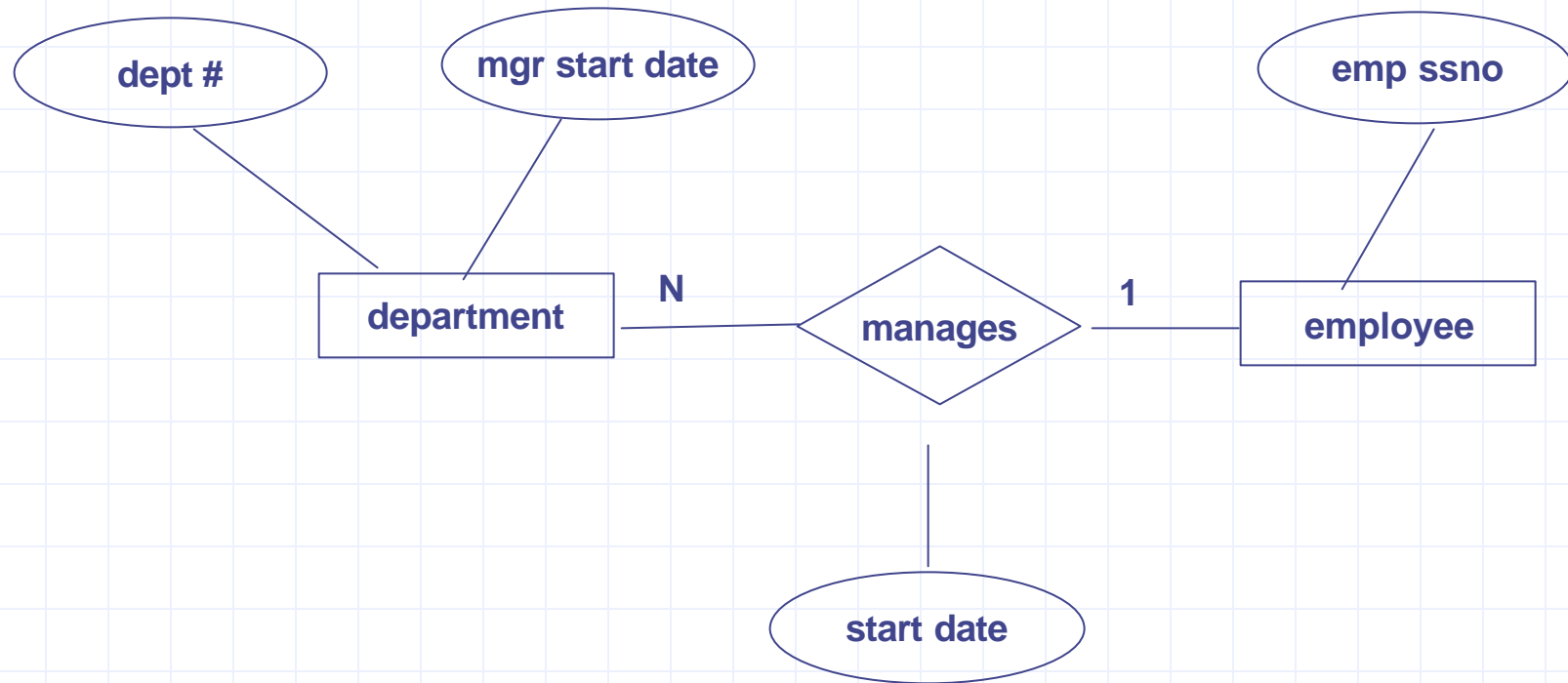
Schema Design Issues

- ◆ **Observation:** there may be multiple ER schemas describing the same target database or miniworld.
- ◆ Decisions that need to be made:
 - whether to use an attribute or entity set to represent an object
 - whether to model a concept as a relationship or an entity set
 - whether to use ternary relationship or a set of binary ones
 - whether to use a strong entity set or a weak entity set
 - whether using generalization/specializations is appropriate
 - whether using aggregates is appropriate
- ◆ Unfortunately, there are no straightforward answers to these questions
- ◆ No two design teams will come up with the same design.
- ◆ However, there are some simple design principles that should be followed during ER design.

E/R Design Principles

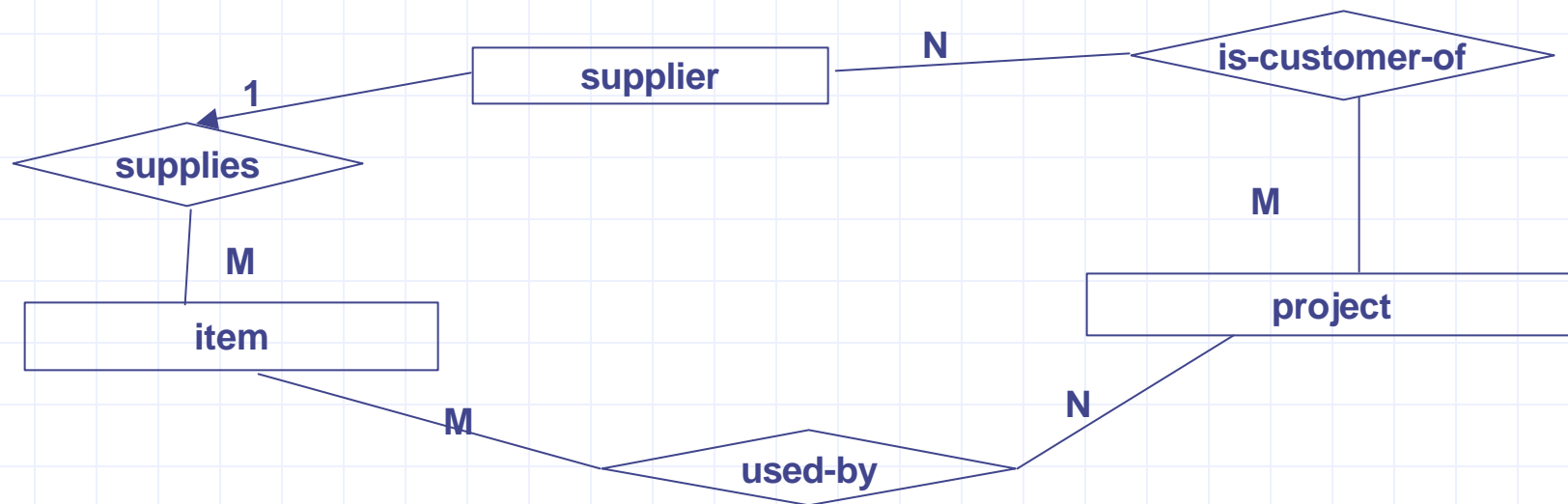
- ◆ Schemas should not change often. So store frequently changing information as instances.
 - currently each project consists of 10 members. Since later projects may have more or less employees, do not hard code the 10 employees as 10 attributes of the project entity
- ◆ Schemas should prevent representing the same facts multiple times (avoid redundancy).
 - An attribute/relationship is redundant if deleting it does not result in a loss of any information
 - redundancy may cause:
 - ◆ wastage of space in storing data
 - ◆ application programming to be more difficult -- applications need to update all instances of a fact else risk inconsistency of database
- ◆ Consistent and clear naming policy for attributes, entities, and relationships

Redundant Attributes



Managers start date stored twice -- redundancy.

Redundant Relationship



- The fact that a project is-customer-of a supplier can be derived from the relationships between supplier and item and between item and project.
- That is, a project is-customer-of a supplier if there is a item that the supplier supplies which is used by the project.
- Redundancy analysis can be tricky -- if supplies is a N:N relationship, then schema does not contain redundancy.

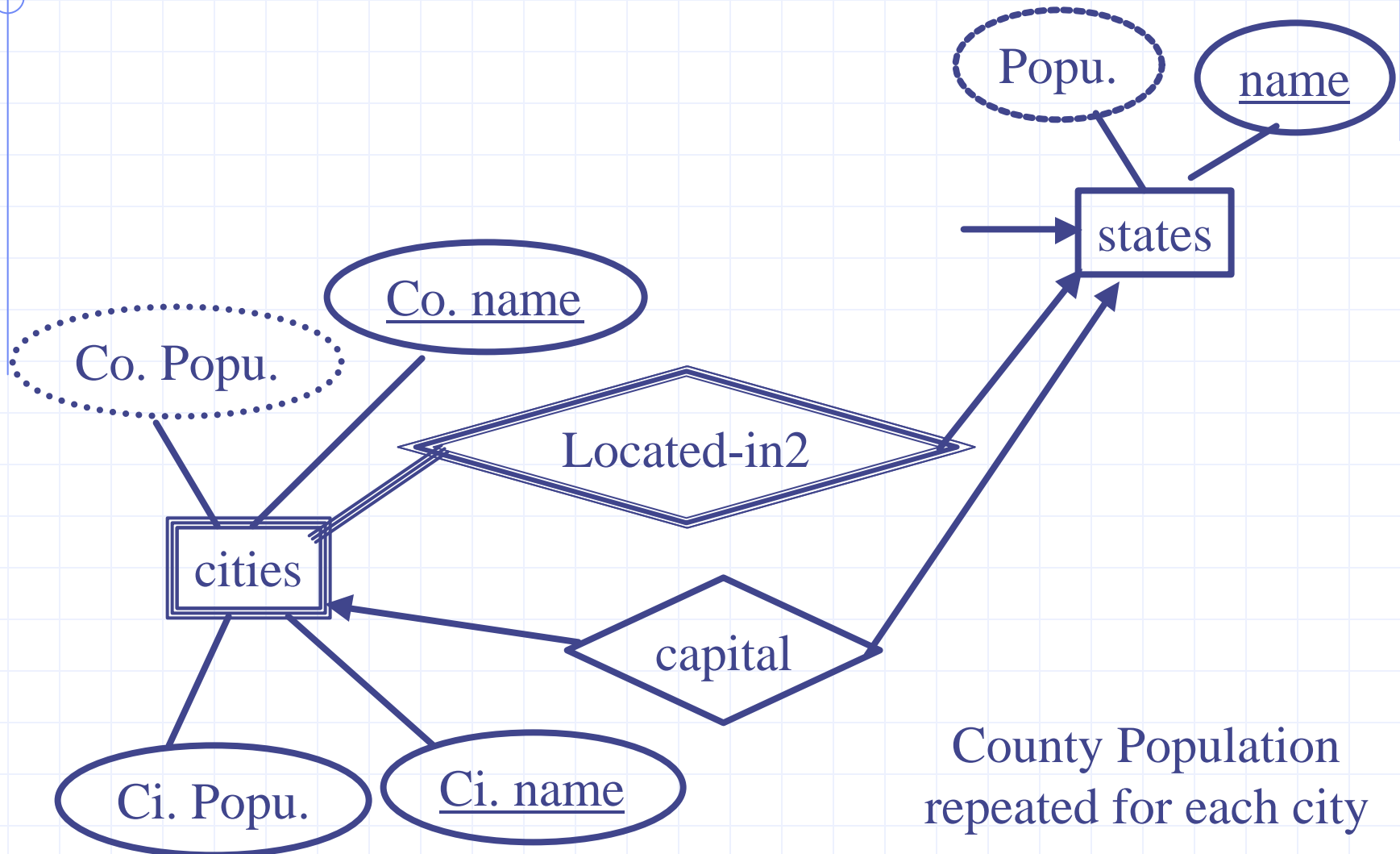
A Design Problem

- ◆ We wish to design a database representing cities, counties, and states in the US.
- ◆ For states, we wish to record the name, population, and state capital (which is a city).
- ◆ For counties, we wish to record the name, the population, and the state in which it is located.
- ◆ For cities, we wish to record the name, the population, the state in which it is located and the county in which it is located.

Uniqueness assumptions:

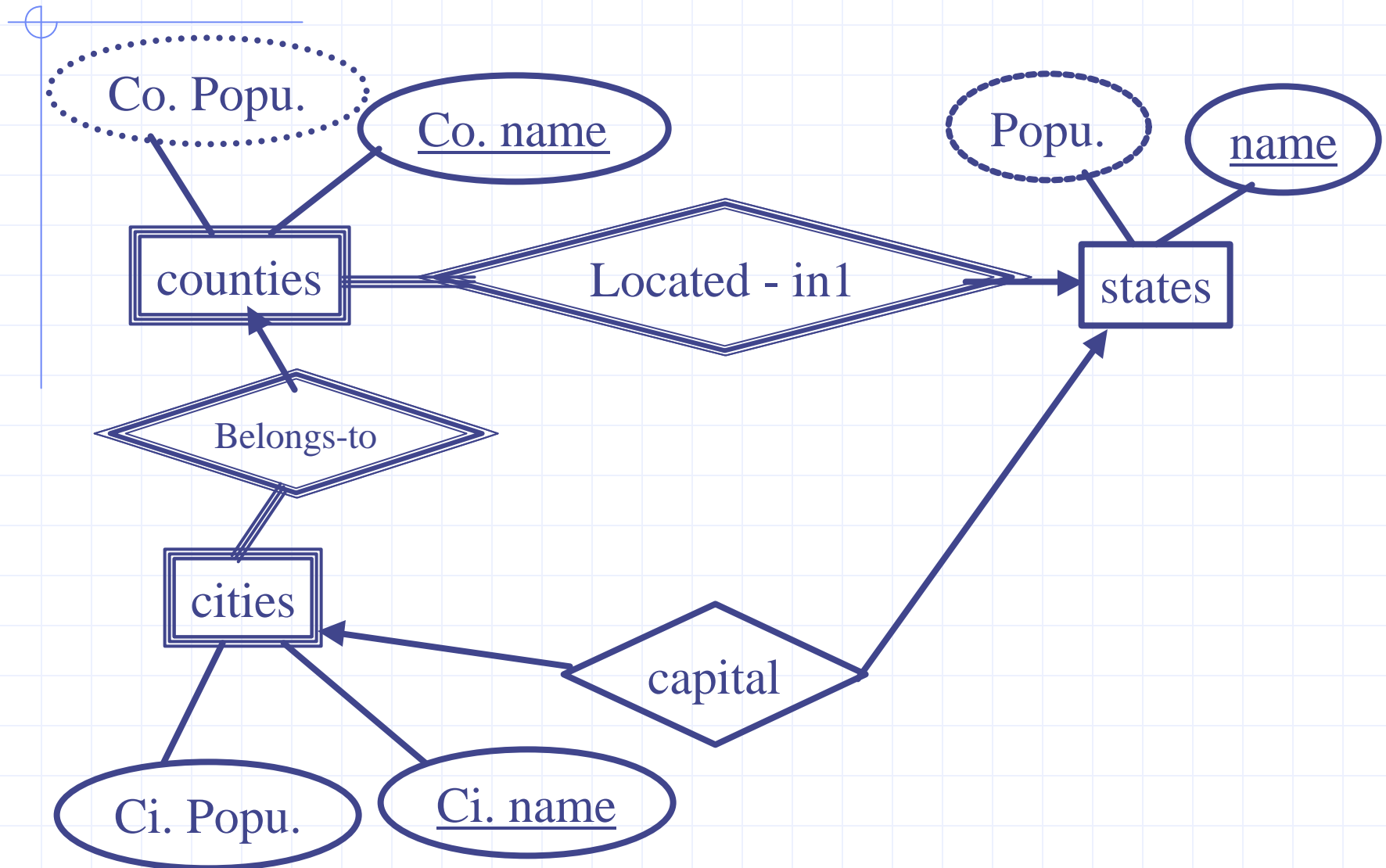
- ◆ Names of states are unique.
- ◆ Names of counties are only unique within a state (e.g., 26 states have Washington Counties).
- ◆ Cities are likewise unique only within a state (e.g., there are 24 Springfields among the 50 states).
- ◆ Some counties and cities have the same name, even within a state (example: San Francisco).
- ◆ All cities are located within a single county.

Design 1: Bad design



County Population repeated for each city

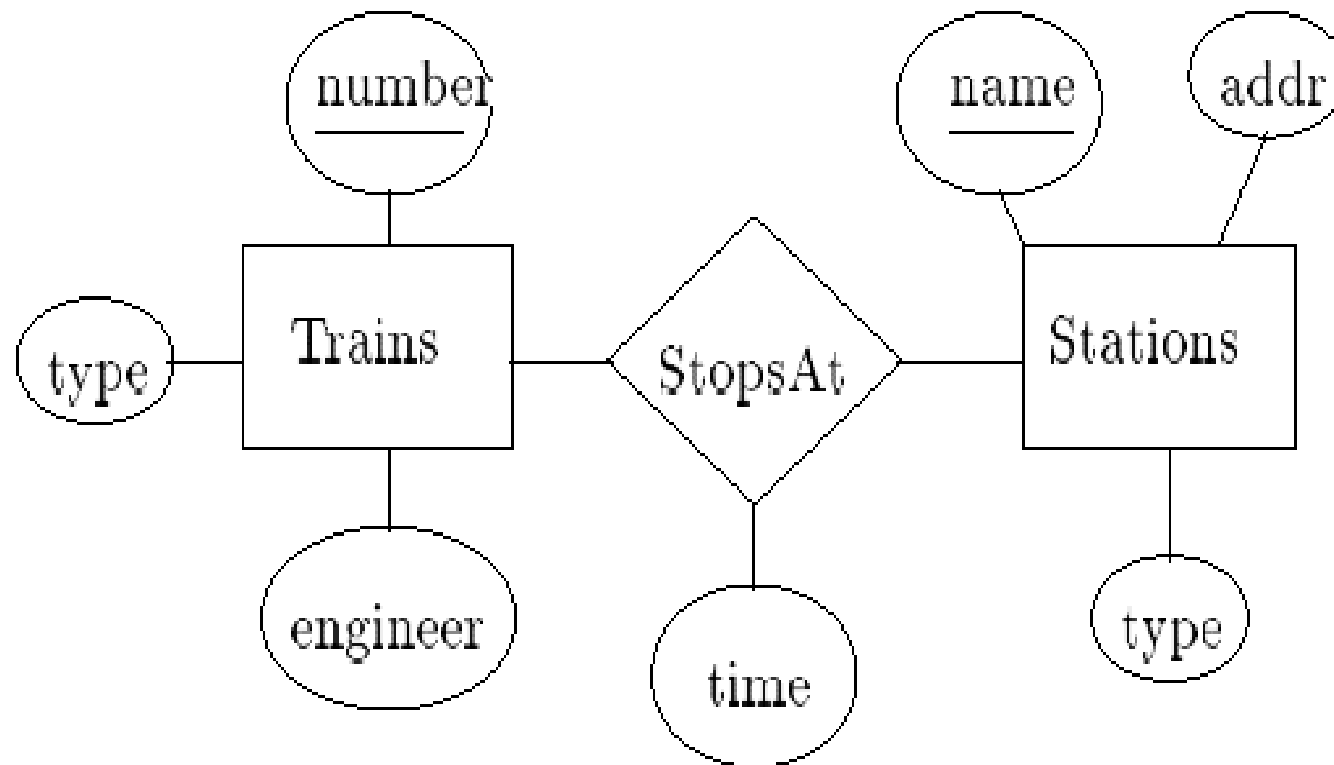
Design 2 -- good design



Another Design Problem

- ◆ We wish to design a database consistent with the following facts.
- ◆ Trains are either local trains or express trains, but never both.
- ◆ A train has a unique number and an engineer.
- ◆ Stations are either express stops or local stops, but never both.
- ◆ A station has a name (assumed unique) and an address.
- ◆ All local trains stop at all stations.
- ◆ Express trains stop only at express stations.
- ◆ For each train and each station the train stops at, there is a time.

Design 1: Bad design



Does not capture the constraints that express trains only stop only at express stations and local trains stop at all local stations

Design 2: Better Design

