# Advanced Database Design

IT 4153 Advanced Database

J.G. Zheng
Spring 2012

SOUTHERN POLYTECHNIC STATE UNIVERSITY

## Overview

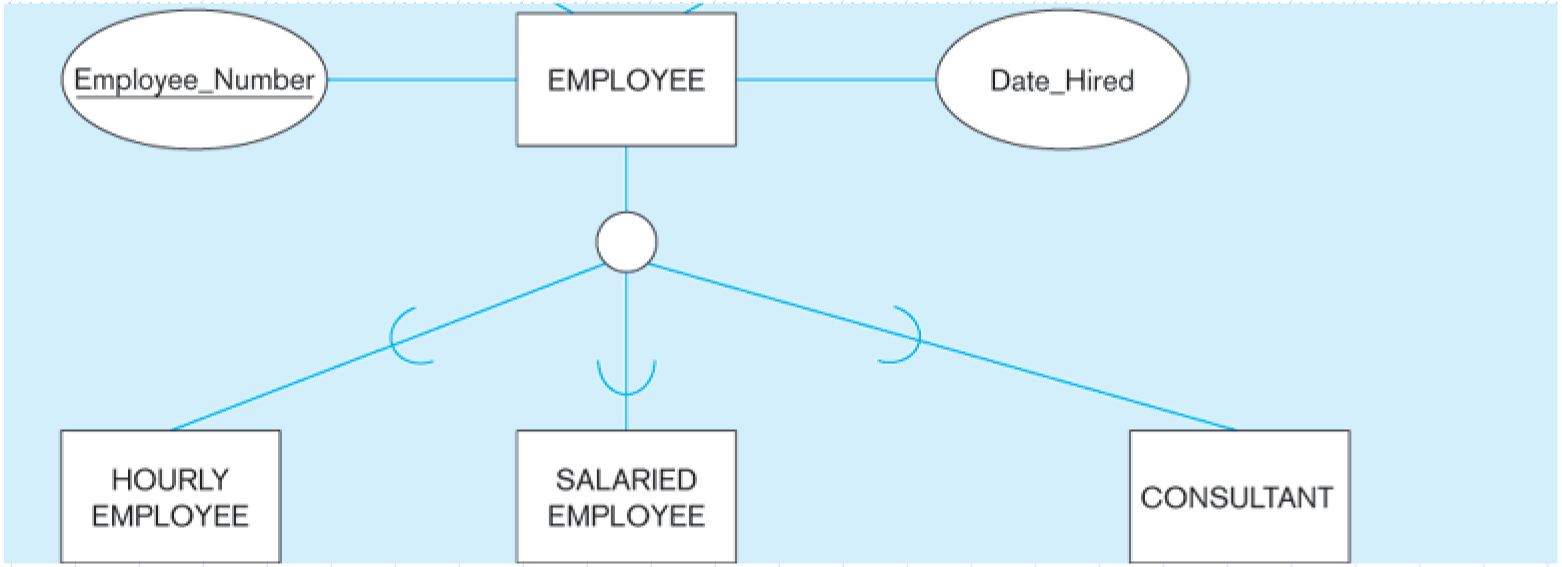
- EER
- Database modeling and design issues at the conceptual and logical level

## EER

- Extended (or enhanced) entity relationship (EER) model
  - A conceptual data model incorporating extensions to the original ER model, used in the design of databases.
  - It was developed to add more semantic constructs found in relationships.
  - Includes all of the concepts introduced by the ER model. Additionally it includes the concepts of subtypes and supertypes, along with the concepts of specialization and generalization.

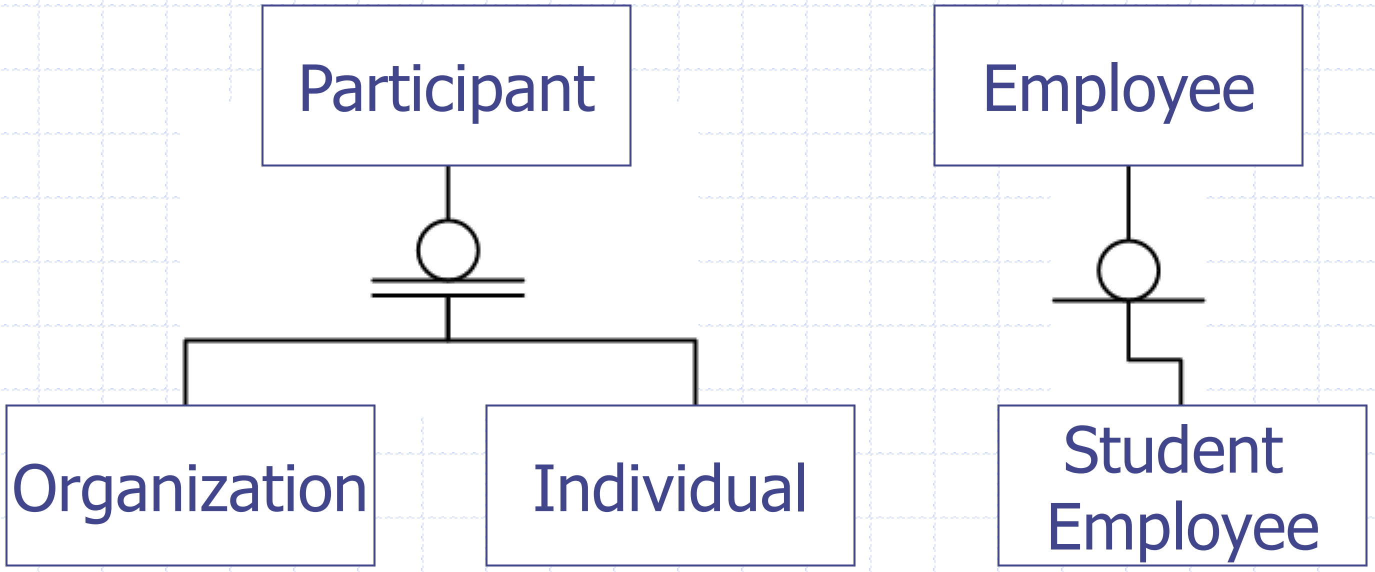
# Super and Sub Types

- A Sub-type is a special case, or a category, of a Super-type
  - Student → Graduates, Undergraduates
  - Employee → full-time, part-time, contractor



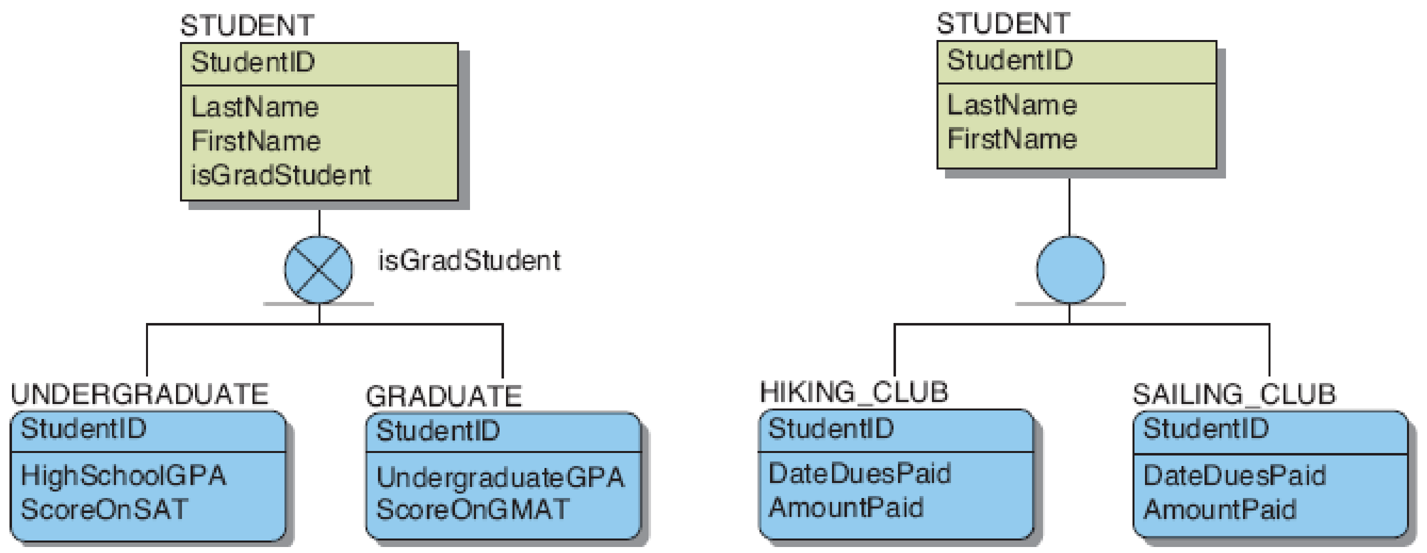
## Sub-Type Completeness

- Completeness
  - Is every super type instance in at least one sub-type entity?



# Sub-Type Disjointness

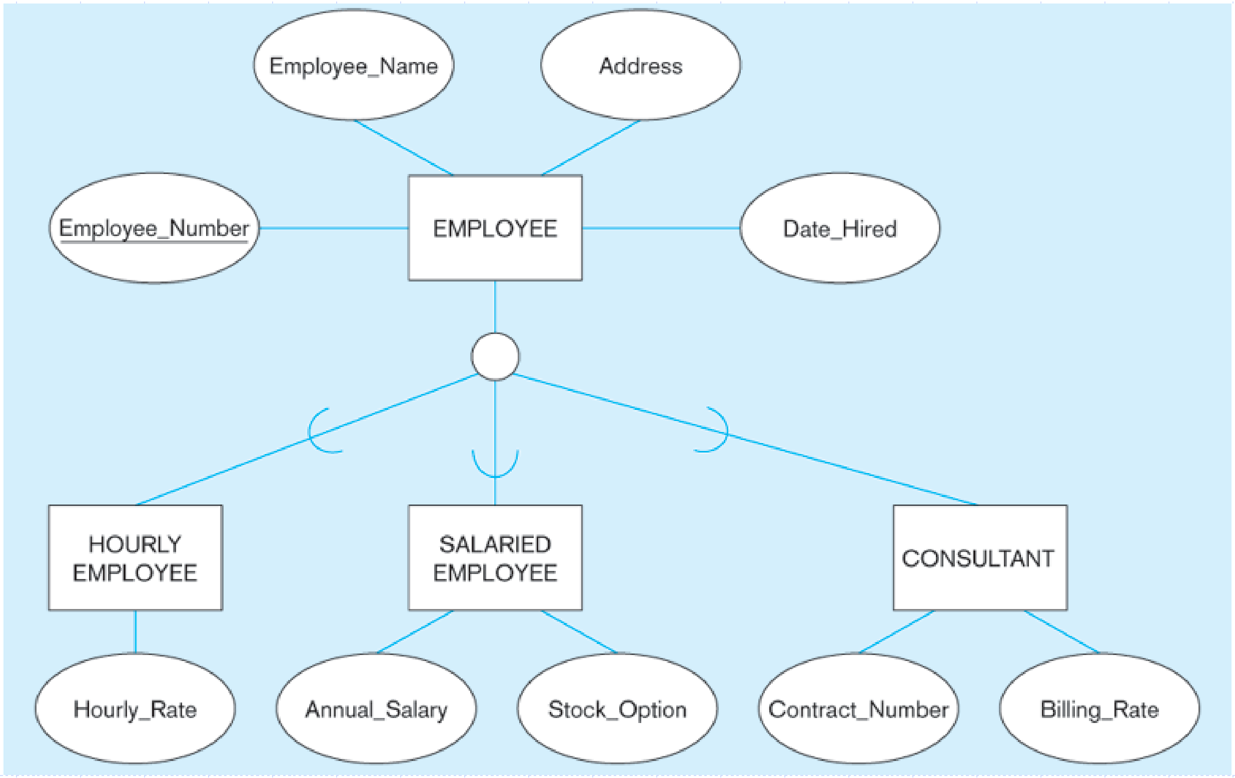
- Disjointness: does any instance appear in multiple subtype entities?
  - Yes: Overlap (Inclusive)
  - No: Disjoint (Exclusive)



(a) Exclusive Subtypes with Discriminator (b) Inclusive Subtypes

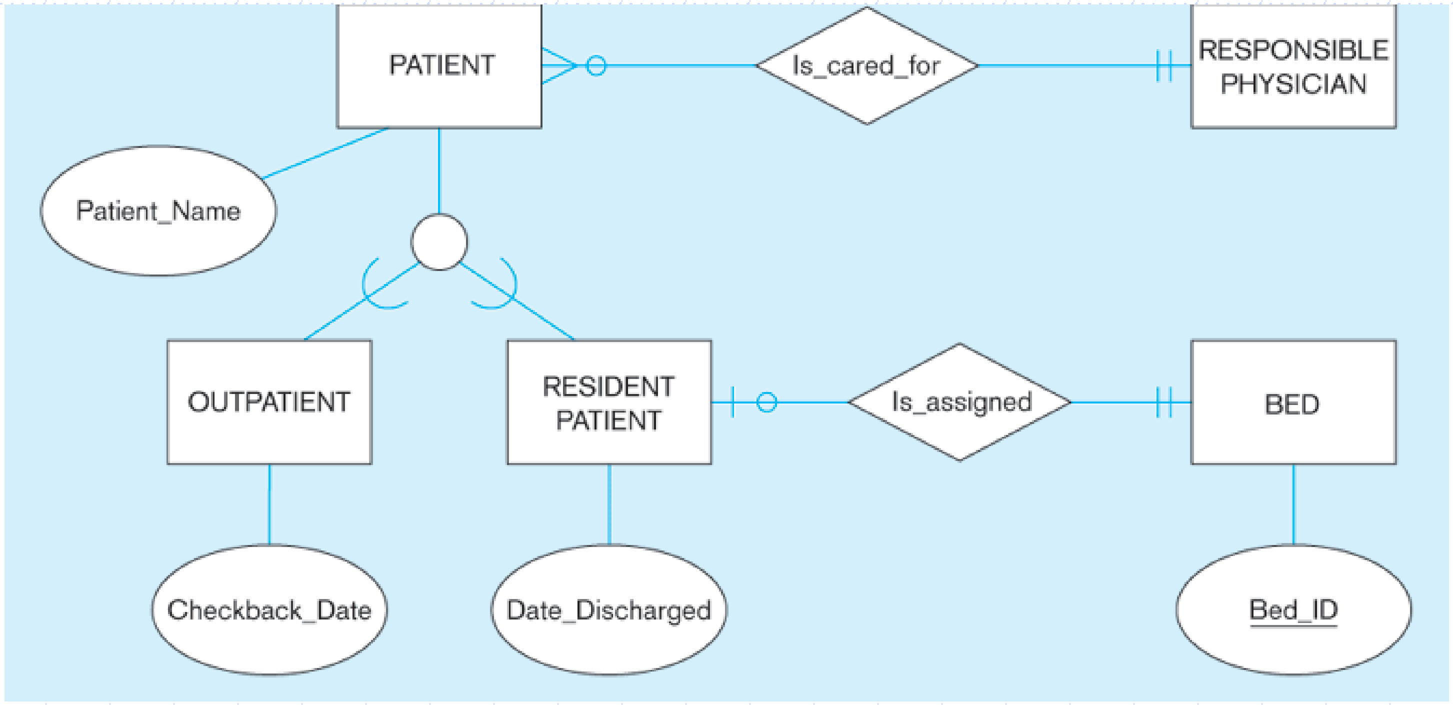
## When to use sub-type (specification)?

- When there are attributes that apply to some (but not all) of the instances of an entity type



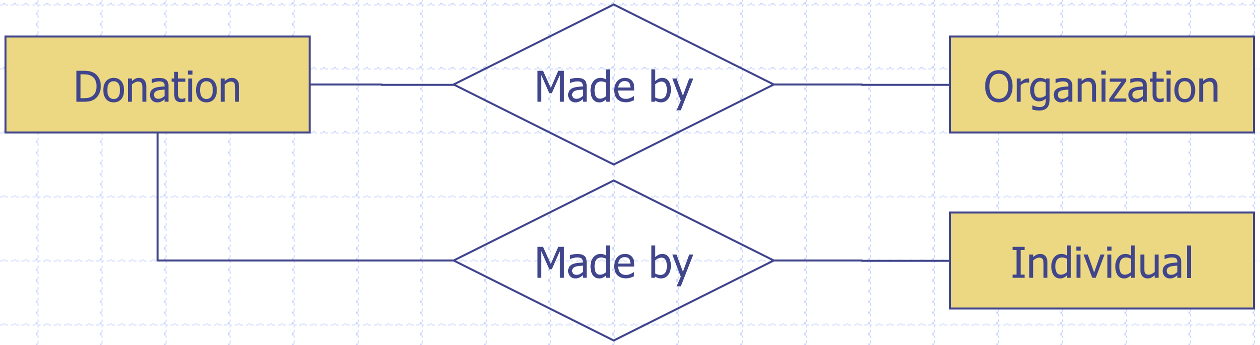
# When to use sub-type (specification)?

- When instances of a subtype participate in a relationship unique to that subtype



## When to use super-type (generalization)?

- When several entities have same major attributes, see if they are constantly treated together.
  - Faculty, staff, student assistant
  - Employee, customer, business partner (or, supplier)
- Multiple entities participate in a common relationship with the same entity (also see modeling issue #8)



## Relational Model for EER

- Generally each supertype or subtype is transformed into a table, which shares the same primary key

- A discriminator is added to the super type table.
  - Disjoint: one column
  - Overlap: multiple columns

# Practical Design Issues

- Table and column
    1. Columns or tables
    2. Fields or values (columns or rows)
    3. Primary key selection issues
- Relationship and constraints
    4. Minimum cardinality design
    5. Fan trap
    6. Redundant relationship
    7. Ternary relationships
    8. Identical relationships
- Specific data structure and requirements
    9. Time variant data
    10. Hierarchies

# Issue #1: Column(s) or Table

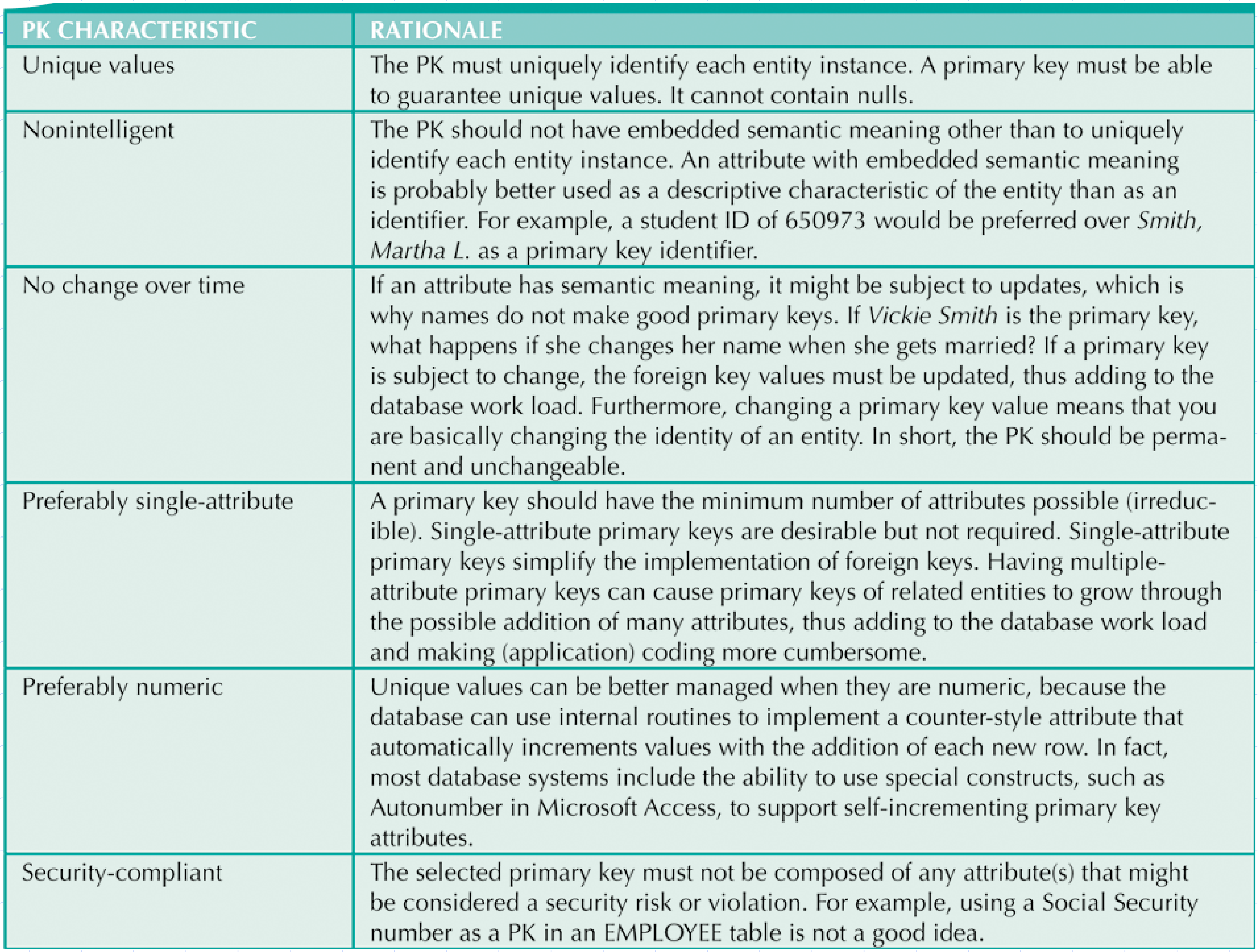
- What data structure level to use for the following data?
  - "Company", "Department", "ZIP", etc.
  - Composite attribute: an attribute that can be further divided into more attributes
    - Example: Name, Address, Date, etc.
  - Multi-Value Attribute: an attribute that allow multiple values
    - Example: skills, phone numbers, etc.

- Depending on business requirements and contexts

## Issue #2: Fields or Values?

- How to model the following scenario?
  - Contacts
    - A sales person can be contacted by “Fax Number”, “Cell Phone Number”, “Home Phone Number”, “Work Phone Number”, “Work Email”, etc.
  - Advising Hours
    - Faculty members have specific advising hours on 5 week days: “Monday”, “Tuesday”, etc.
- Entity-attribute-value model
  - http://en.wikipedia.org/wiki/Entity-attribute-value_model

# Issue #3: What's a Good PK?



| PK CHARACTERISTIC | RATIONALE |
|---|---|
| Unique values | The PK must uniquely identify each entity instance. A primary key must be able to guarantee unique values. It cannot contain nulls. |
| Nonintelligent | The PK should not have embedded semantic meaning other than to uniquely identify each entity instance. An attribute with embedded semantic meaning is probably better used as a descriptive characteristic of the entity than as an identifier. For example, a student ID of 650973 would be preferred over *Smith, Martha L.* as a primary key identifier. |
| No change over time | If an attribute has semantic meaning, it might be subject to updates, which is why names do not make good primary keys. If *Vickie Smith* is the primary key, what happens if she changes her name when she gets married? If a primary key is subject to change, the foreign key values must be updated, thus adding to the database work load. Furthermore, changing a primary key value means that you are basically changing the identity of an entity. In short, the PK should be permanent and unchangeable. |
| Preferably single-attribute | A primary key should have the minimum number of attributes possible (irreducible). Single-attribute primary keys are desirable but not required. Single-attribute primary keys simplify the implementation of foreign keys. Having multiple-attribute primary keys can cause primary keys of related entities to grow through the possible addition of many attributes, thus adding to the database work load and making (application) coding more cumbersome. |
| Preferably numeric | Unique values can be better managed when they are numeric, because the database can use internal routines to implement a counter-style attribute that automatically increments values with the addition of each new row. In fact, most database systems include the ability to use special constructs, such as Autonumber in Microsoft Access, to support self-incrementing primary key attributes. |
| Security-compliant | The selected primary key must not be composed of any attribute(s) that might be considered a security risk or violation. For example, using a Social Security number as a PK in an EMPLOYEE table is not a good idea. |

## Surrogate Key – When (Not)?

- It’s common to add a surrogate key to replace the composite key in intersection tables
- Also use surrogate key when a composite primary key will be used as a foreign key
  - This is much simpler
- When not to use?
  - If there is a perfect natural key already
  - If the composite key has critical information.

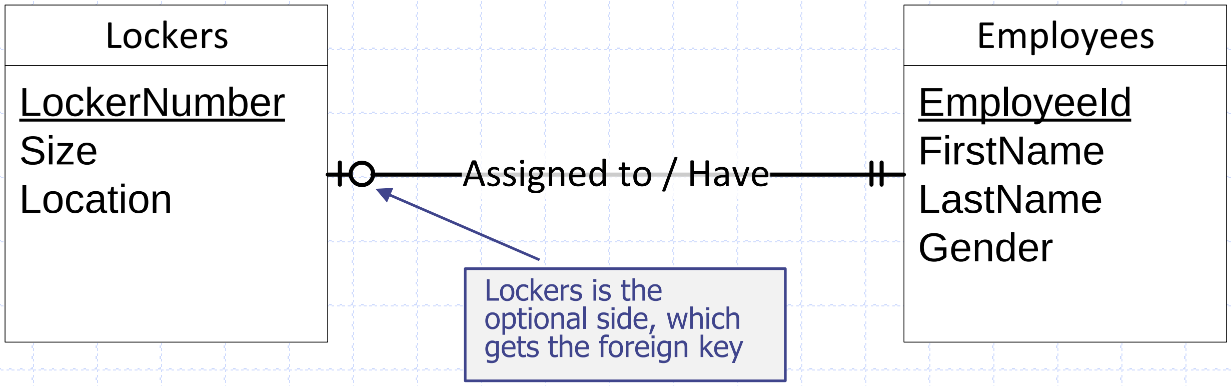
## Issue #4: Minimum Cardinality

- Minimum cardinality describes the minimum number of instances that must participate in a relationship for any one instance
  - 0 (optional): participation in the relationship by the entity is optional.
  - 1 (mandatory): participation in the relationship by the entity is mandatory.
- How minimum cardinality affects modeling and design in 1:1 relationship
- How is minimum cardinality enforced?

# Minimum Cardinality 1:0

- When only one side is optional, foreign key is placed on the optional side, to avoid null values → kind of like normalization.

- Example:
  - A locker is optional for an employee (an employee may not get one)
  - An employee is mandatory for a locker (a locker has to be assigned to someone)

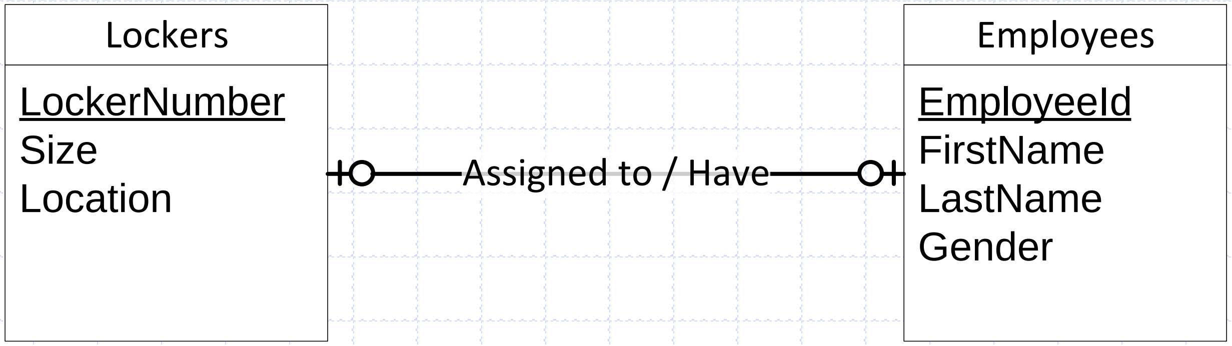


Employees (<u>EmployeeId</u>, FirstName, LastName, Gender)
Lockers (<u>LockerNumber</u>, Size, Location, AssginedTo)
    FK: AssignedTo → Employees.EmployeeId

# Minimum Cardinality 0:0

- When both sides are optional, foreign key is placed in the table which causes minimum null values
- Or, create a new intersection table → kind of like normalization.

- Example:
  - A locker is optional for an employee (an employee may not get one)
  - An employee is optional for a locker (a locker may not be assigned to someone)



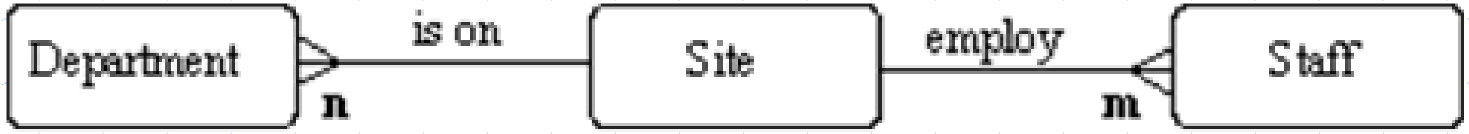
Employees (<u>EmployeeId</u>, FirstName, LastName, Gender)
Lockers (<u>LockerNumber</u>, Size, Location)
LockerAssignment (*<u>LockerNumber</u>*, *EmployeeId*)
FK: LockerNumber → Lockers.LockerNumber
FK: EmployeeId → Employees.EmployeeId

The intersection table with two attributes: one is primary key and the other is unique.

Two foreign keys

## Issue #5: Fan Trap

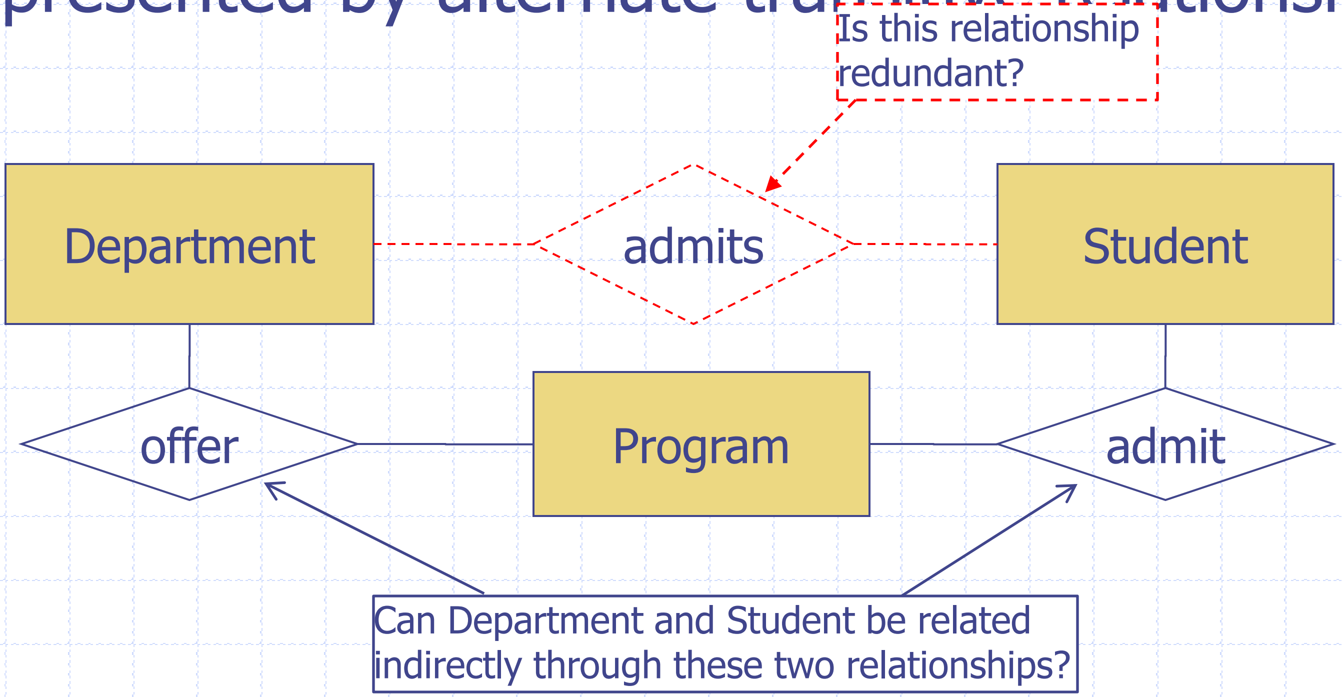
- Avoid the Fan Trap



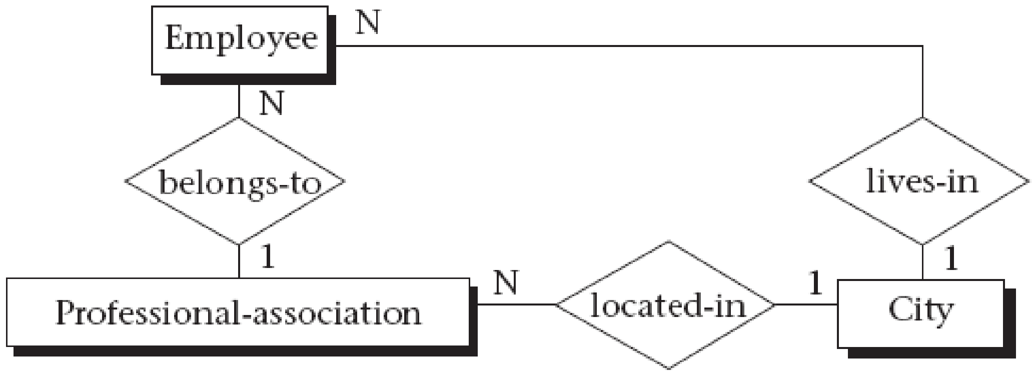
Ambiguous (broken) relationship between Department and Staff

## Issue #6: Redundant Relationship

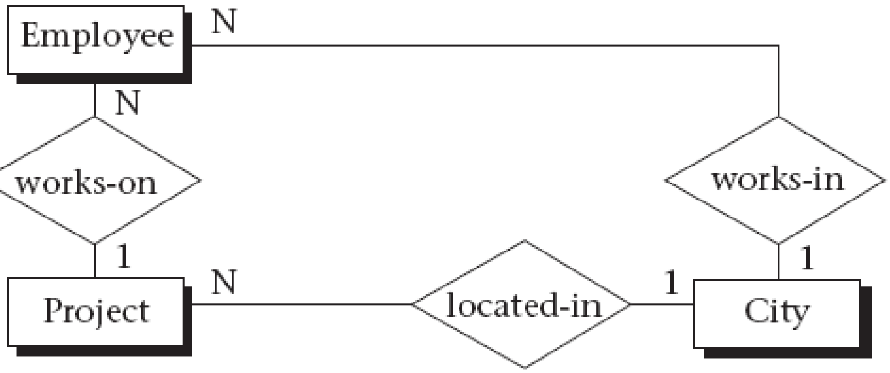
- Entities can be related indirectly by two relationships.
- A relationship is redundant if it can be completely represented by alternate transitive relationships


Is this relationship redundant?
Department
admits
Student
offer
Program
admit
Can Department and Student be related indirectly through these two relationships?

# Redundant Relationship?



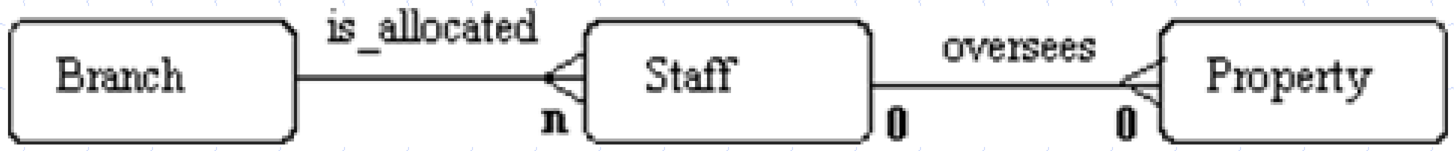
(a) Nonredundant relationships



(b) Redundant relationships using transitivity

## Chasm Trap

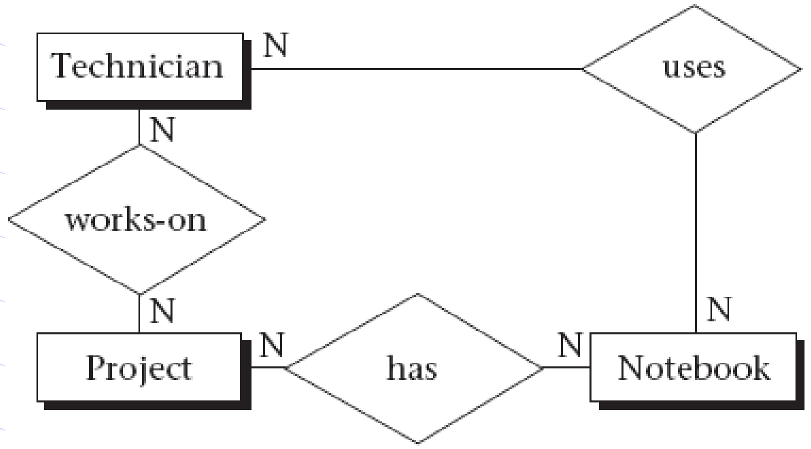
- Avoid the Chasm Trap



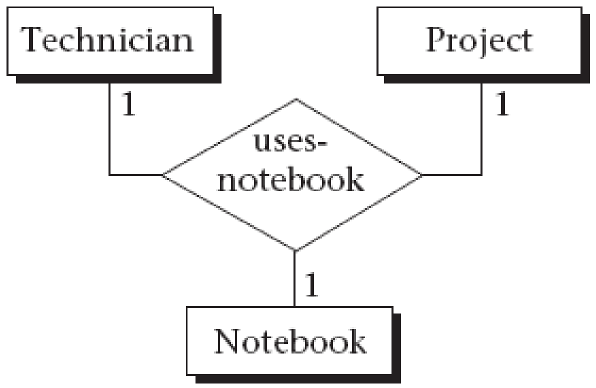
Ambiguous (broken) relationship between Branch and Property

# Issue #7: Ternary Relationship

- 3 binary relationships or a ternary one?



(a) Binary relationships

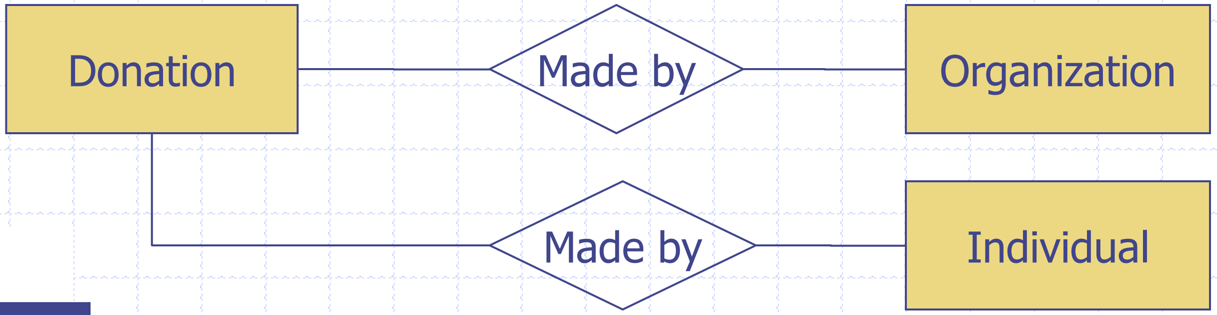


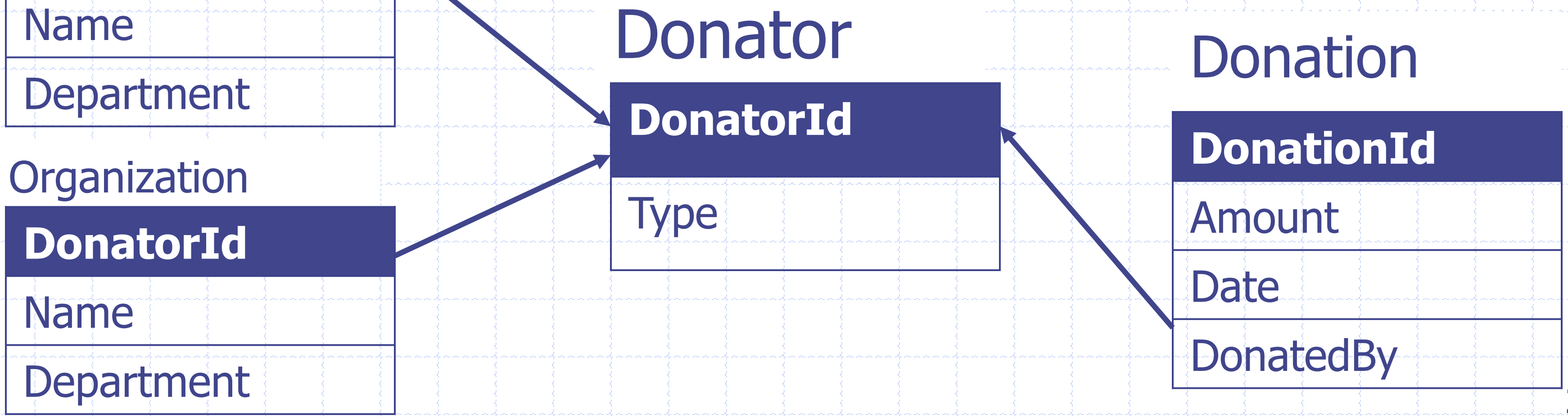
(b) Different meaning using a ternary relationship

Database Modeling and Design: Logical Design, 4th Edition by Toby J. Teorey, Sam S. Lightstone, and Tom Nadeau, 2005

## Issue #8: Common relationship trap

- Avoid the same (identical) relationship with multiple entities





## Issue #9: Time Variant Data

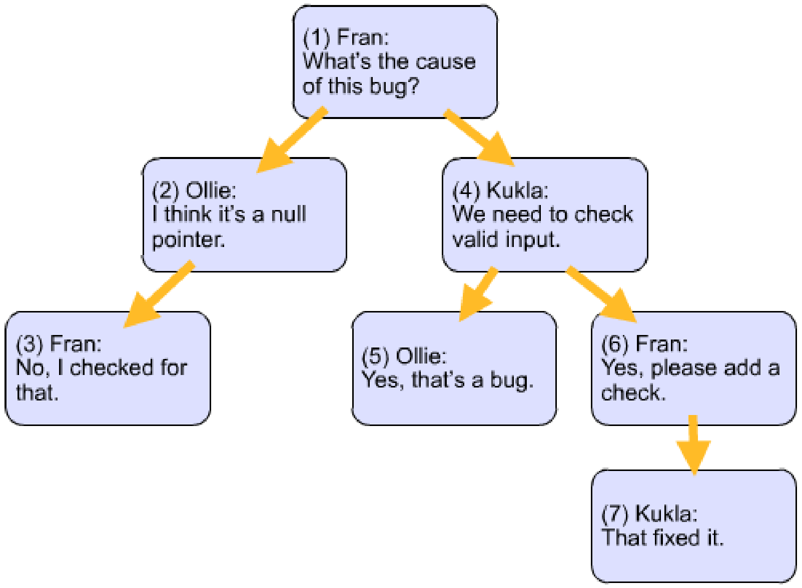
- Normally, existing attribute values are replaced with new value without regard to previous value
- Time-variant data:
  - Values change over time
  - Must keep a history of data changes
- Keeping history of time-variant data equivalent to having a multi-value attribute

## Issue #10: Hierarchical Data (1)

- If it involves multiple different entities
  - Use a chain of one-to-many relationship

# Issue #10: Hierarchical Data (2)

- If within the same entity


(1) Fran: What's the cause of this bug?
(2) Ollie: I think it's a null pointer.
(4) Kukla: We need to check valid input.
(3) Fran: No, I checked for that.
(5) Ollie: Yes, that's a bug.
(6) Fran: Yes, please add a check.
(7) Kukla: That fixed it.

## Adjacency list

| comment_id | parent_id | author | comment |
|---|---|---|---|
| 1 | NULL | Fran | What's the cause of this bug? |
| 2 | 1 | Ollie | I think it's a null pointer. |
| 3 | 2 | Fran | No, I checked for that. |
| 4 | 1 | Kukla | We need to check valid input. |
| 5 | 4 | Ollie | Yes, that's a bug. |
| 6 | 4 | Fran | Yes, please add a check |
| 7 | 6 | Kukla | That fixed it. |

## Materialized path

| comment_id | path | author | comment |
|---|---|---|---|
| 1 | 1/ | Fran | What's the cause of this bug? |
| 2 | 1/2/ | Ollie | I think it's a null pointer. |
| 3 | 1/2/3/ | Fran | No, I checked for that. |
| 4 | 1/4/ | Kukla | We need to check valid input. |
| 5 | 1/4/5/ | Ollie | Yes, that's a bug. |
| 6 | 1/4/6/ | Fran | Yes, please add a check |
| 7 | 1/4/6/7/ | Kukla | That fixed it. |

# Analysis and Modeling Tips

- Modeling is an iterative refinement process; let entities, attributes, and relationships eventually emerge and refined
  - Start with entities and attributes first: look for basic and obvious facts or concepts; these are the preliminary entities and attributes.
  - Determine how entities are related
    - Quickly identify binary relationships and maximum cardinality first.
    - Identify minimum cardinality
  - Refine relationships and check for common modeling traps.
    - Check for redundant relationships and missing relationships.
    - Consider n'ery relationships if necessary
  - Add, combine, or split entities and attributes as needed. Check relationships and constraints after changes.
  - Repeat some steps and refine the model until satisfactory.
- View integration
  - Start with specific function areas (user views) and integrate them later
- Ensure the consistency between requirements, ERD and the data dictionary