浙江大学
ZHEJIANG UNIVERSITY

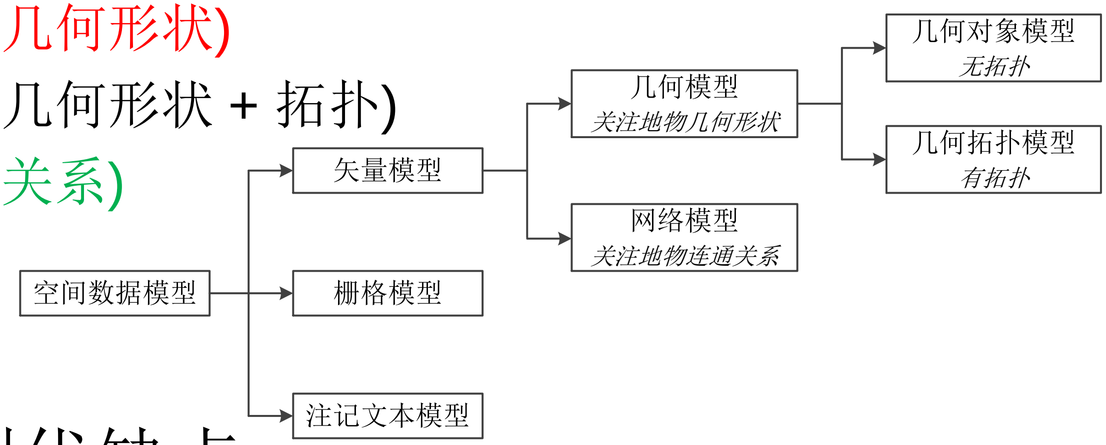
第五章 空间扩展E/R图

陶煜波

计算机科学与技术学院

几何对象模型与查询回顾

- 空间数据模型分类
  - 矢量模型
    - 几何对象模型 (地物几何形状)
    - 几何拓扑模型 (地物几何形状 + 拓扑)
    - 网络模型 (地物连通关系)
  - 栅格模型
  - 注记文本模型
- 矢量模型和栅格模型优缺点
- 概念模型 → 逻辑模型 → 物理模型



空间数据模型

矢量模型

栅格模型

注记文本模型

几何模型

关注地物几何形状

网络模型

关注地物连通关系

几何对象模型

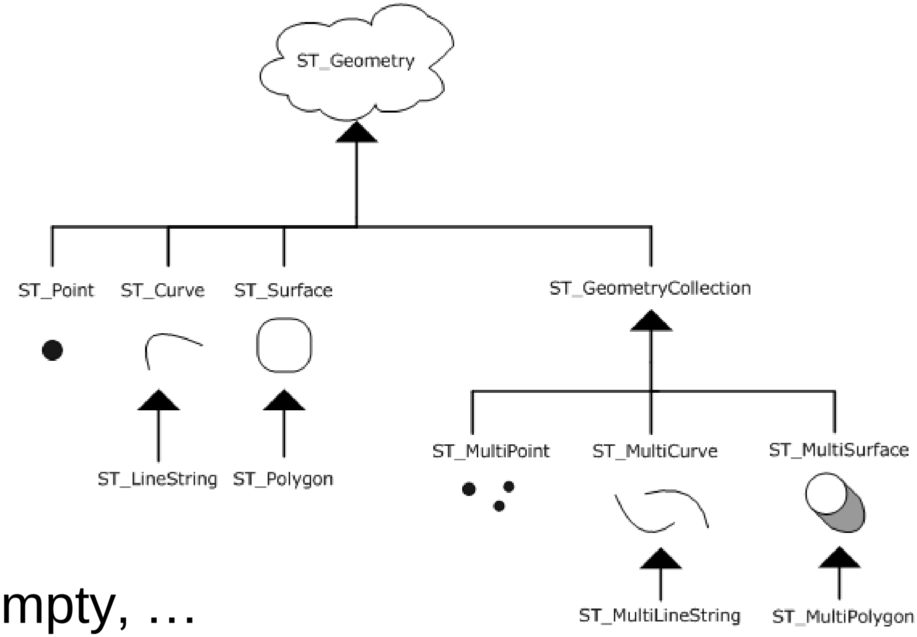
无拓扑

几何拓扑模型

有拓扑

几何对象模型

- 对象关系数据库：数据+方法 (C++中的类)
- 几何对象层次关系
  - 坐标维数和几何维数
  - 边界、内部、外部
  - 九交矩阵



- 几何对象方法
  - 常规方法 (12种)
    - Dimension, Boundary, IsEmpty, …
  - 常规GIS分析方法 (7种)
    - Distance, Buffer, ConvexHull, Intersection, Union, Difference, SymDifference
  - 空间查询方法 (11种)
    - 包含于(within): 若$a \cap b = a$，且$\mathrm{I}(a) \cap E(b) = \emptyset$，则a包含于b内

空间拓扑关系

T = True
F = False
* = True / False, i.e., Don’t care

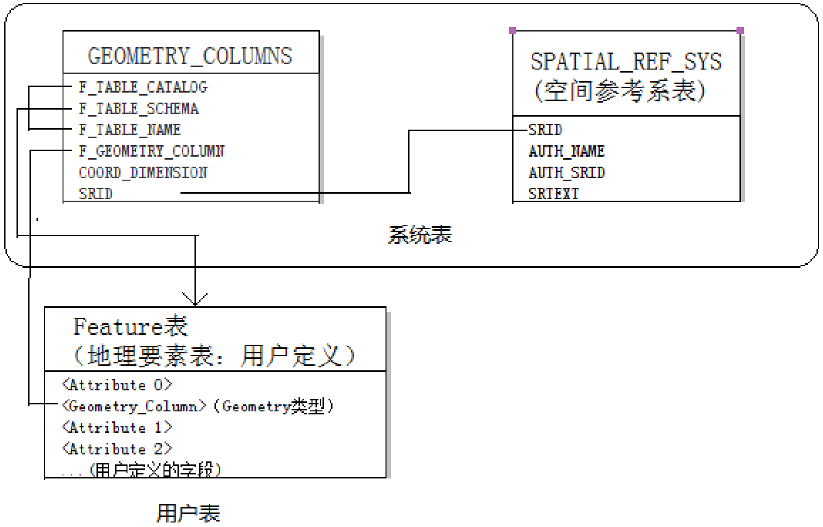
空间关系	定义	九交矩阵
Equals	$a \subseteq b$，$a \supseteq b$	TFFFTFFFT (同类：点/点, 线/线, 面/面)
Overlaps	Dim(I(a)) = Dim(I(b)) = Dim(I(a) ∩ $I(b)$)，$a \cap b \neq a$，$a \cap b \neq b$	T*T***T** (点/点, 面/面) 1*T***T** (线/线) (同类)
Disjoint	$a \cap b = \emptyset$	FF*FF**** (点线面所有组合)
Intersects	a.Intersects(b) ←→ !b.Disjoint(b)	? (点线面所有组合)
Within	$a \cap b = a$，I(a) ∩ $E(b) = \emptyset$	T*F**F*** (除点/点，点线面所有组合)
Contains	a.Contains(b) ←→ b.Within(a)	? (除点/点，点线面所有组合)
Touches	I(a) ∩ $I(b) = \emptyset$，$a \cap b \neq \emptyset$	FT*******, F**T*****, F***T**** (除点/点，点线面所有组合)
Crosses	I(a) ∩ $I(b) \neq \emptyset$ ，$a \cap b \neq a$，$a \cap b \neq b$	T*T****** (点/线, 点/面, 线/面) 0******** (线/线) (线/面，面/面, 多点/线(面))

几何对象模型

- 空间查询方法 (11种)
  - Equals, Disjoint, Intersects, Touches, Crosses, Within, Contains, Overlaps, Relates
  - LocateAlong, LocateBetween
- 空间关系 (8种)
  - 相离(disjoint), 相交(intersects), 相等(equals)
    - a.Intersects(b) ↔ !a.Disjoint(b)
  - 交叠(overlaps), 包含于(within), 包含(contains)
    - a.Contains(b) ↔ b.Within(a)
    - a.Equals(b) → a.Overlaps(b) ?
  - 相接(touches), 穿越(crosses)
    - a.Touches(b) (或a.Crosses(b))→ a.Intersects(b) ?
    - a.Touches(b) → a.Crosses(b) ?

几何对象模型

- 逻辑模型
  - 基于预定义数据类型的实现
    - numeric和BLOB
  - 基于扩展几何类型的实现
    - Geometry类
- 表模式
  - 系统表
    - GEOMETRY_COLUMNS和SPATIAL_REF_SYS
  - 用户表
    - Feature和Geometry
- 物理模型
  - WKB和WKT



基于扩展Geometry类型的要素表模式

空间数据库

- 空间数据库 = 对象关系/关系数据库 + 空间扩展
  - Oracle + Oracle Spatial
  - SQL Server + SQL Server Spatial
  - PostgreSQL + PostGIS
  - MySQL + MySQL Spatial
  - SQLite + SQLite Spatialite
- PostGIS
  - 提供了空间数据类型、空间函数和空间索引
  - geom geometry (Point/LineString/Polygon/Multixxx, 空间参考系)
  - ST_GeomFromText (‘WKT表示’, 空间参考系)
  - ST_XXX: ST_Distance, ST_Intersects, ST_DWithin, …
  - GiST

应用举例：打车软件

- 需要哪些空间数据？
  - 出租车空间位置
    - Taxi(ID, driverID, ……, status, pos(Point, 4326))
  - 乘客空间位置
    - User(ID, name, ……, pos(Point, 4326))
  - 道路
    - Road(ID, name, ……, line(LineString, 4326))
- 乘客(ID = A)的附近1公里内的空车？(注意距离单位)

  -
    ```
    Select T.ID, T.position
    From Taxi T, User U
    Where U.ID = A and ST_Distance(T.pos, U.pos) < 1000
     and T.status = 0;
    ```
- 出租车(ID = B)附近1公里内的乘客叫车？

应用举例：打车软件

- 需要哪些空间数据？
  - Taxi(ID, driverID, ……, status, pos(Point, 4326))
  - User(ID, name, ……, pos(Point, 4326))
  - Road(ID, name, ……, line(LineString, 4326))
- 利用出租车数量评估道路拥堵情况，出租车在道路100米内认为在该道路上

  -
    ```
    Select R.ID, count(*)
    From Taxi T, Road R
    Where ST_Distance(T.pos, T.line, false) < 100
    Group by R.ID;
    ```
  - 是否输出当前没有出租车所在的道路？
  - 这样的评价方式合理吗？

应用举例：打车软件

- 通常，数据库保留所有时间上的信息，即
  - Taxi(ID, driverID, ……, status, pos(Point, 4326), time)
  - User(ID, name, ……, pos(Point, 4326), time)
  - Road(ID, name, ……, line(LineString, 4326))
- 应如何修改上述SQL语句？
- 出租车(ID = B)附近1公里内的乘客叫车？
  - Select T.ID, T.position

    From Taxi T, User U

    Where T.ID = B and ST_Distance(T.pos, U.pos) < 1000

    and T.time ……

    and U.time ……
  - 这样的实现方式合理吗？

应用举例：公共自行车站点

- 道路关系Road(rid, name, centerline)和公共自行车站点Station(sid, name, pos)，查询没有自行车站点的道路？
- A. Select rid, name From Road, Station

  Where not ST_Intersects(pos, centerline)

- B. Select rid, name From Road Where not exists (select * from station where ST_Intersects(pos, centerline));

- C. Select rid, name From Road Where not exists (select * from station where ST_Within(pos, centerline));

- D. Select rid, name From Road, Station

  Where ST_Intersects(pos, centerline)

  Group by rid, name

  Having count(*) = 0

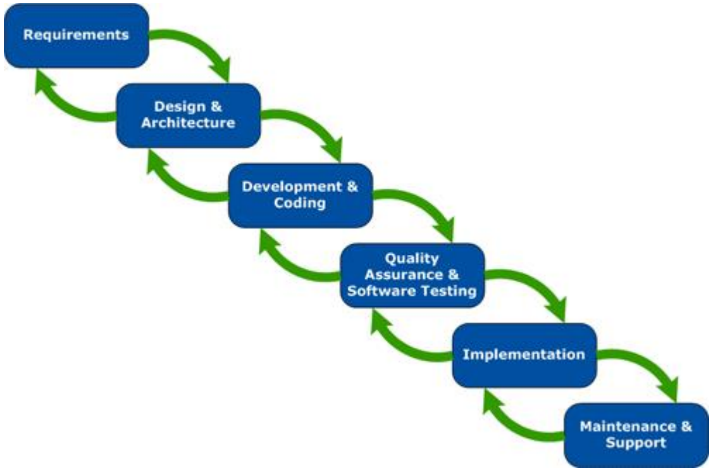
第五章 空间扩展E/R图

- 5.1 Database Design
- 5.2 E/R Basic: Entities & Relations
- 5.3 E/R Design Considerations
  - 5.3.1 Relations: multiplicity, multi-way
  - 5.3.2 Design considerations
  - 5.3.3 Conversion to SQL
- 5.4 Advanced E/R Concepts
  - 5.4.1 Subclasses & connection to OO
  - 5.4.2 Constraints
  - 5.4.3 Weak entity sets
- 5.5 Spatial Database Design Example

参考教材：
数据库系统概念 7.1-7.10
空间数据库管理系统概论 5.1-5.4

5.1 Database Design

- Database design: Why do we need it?
  - Agree on structure of the database before deciding on a particular implementation



Software Engineering

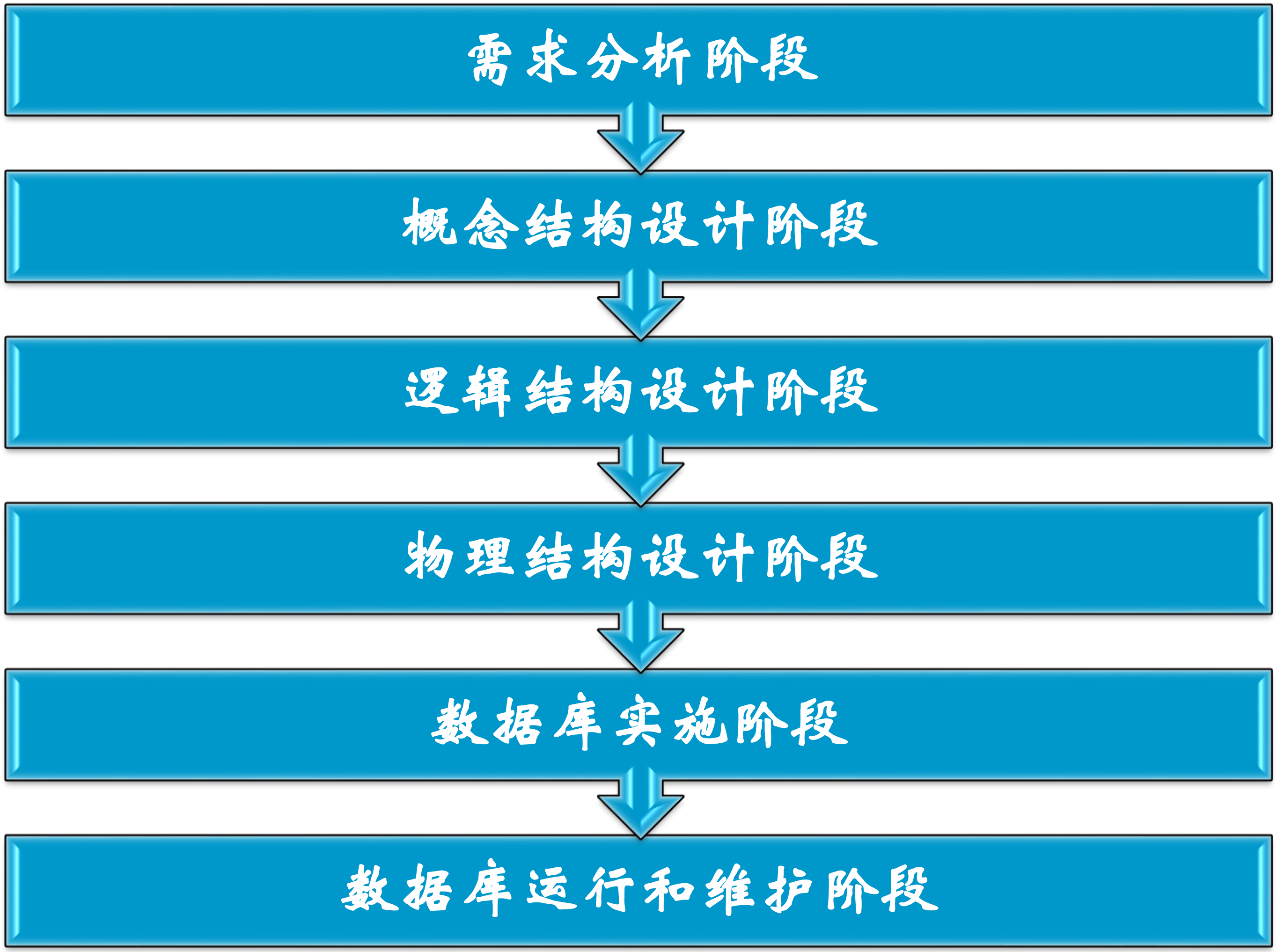
- Consider issues, such as
  - What entities to model
  - How entities are related
  - What constraints exist in the domain
  - How to achieve good designs
- Several formalisms exist
  - We discuss one flavor of E/R diagrams

Database Design Process

- 数据库的设计任务
  - 对某个给定的应用领域，为某一个部门或组织设计出某种数据库管理系统所支持的一个结构合理、使用方便、效率较高的数据库及其应用系统
- 数据库设计应该与应用系统设计相结合
  - 结构(数据)设计：设计数据库框架或数据库结构
  - 行为(处理)设计：设计应用程序、事务处理等
- 典型方法
  - 新奥尔良(New Orleans)方法, S.B.Yao方法, I.R.Palmer方法
- 计算机辅助设计
  - ORACLE Designer, SYBASE PowerDesigner, Erwin, Visio, BDB, Rational Rose

Database Design Process

- 基本步骤



需求分析阶段

概念结构设计阶段

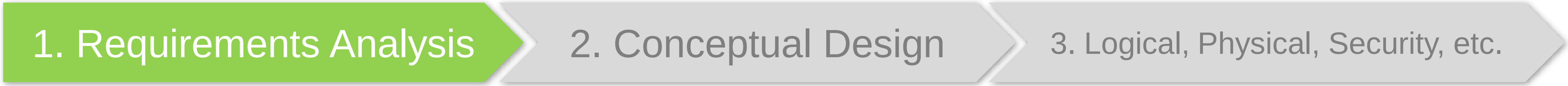
逻辑结构设计阶段

物理结构设计阶段

数据库实施阶段

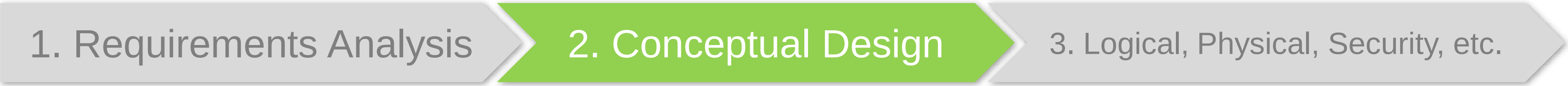
数据库运行和维护阶段

Database Design Process



- 需求分析阶段：全面了解与分析用户需求，需求分析做的是否充分与准确，决定了构建数据库系统的速度与质量
- Requirements analysis (技术人员和非技术人员)
  - 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?

Database Design Process

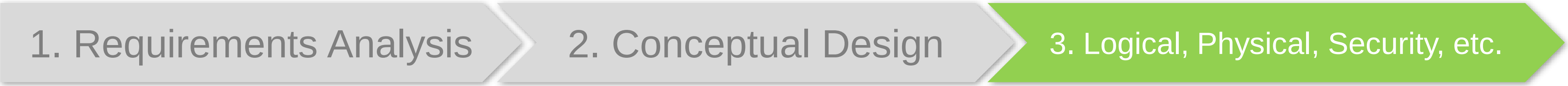

1. Requirements Analysis

2. Conceptual Design

3. Logical, Physical, Security, etc.

- 概念结构设计阶段：是整个数据库设计的关键，通过对用户需求进行综合、归纳与抽象，形成一个独立于具体DBMS的概念模型
- Conceptual Design (E/R图)
  - A high-level description of the database
  - Sufficiently precise that technical people can understand it
  - But, not so precise that non-technical people can't participate

Database Design Process



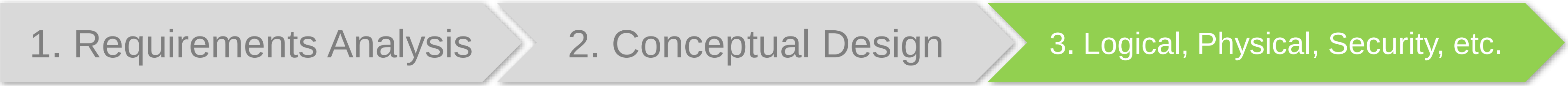
1. Requirements Analysis

2. Conceptual Design

3. Logical, Physical, Security, etc.

- 逻辑结构设计阶段：将概念结构转换为某个DBMS所支持的数据模型，对其进行优化
- 物理结构设计阶段：为逻辑数据模型选取一个最适合应用环境的物理结构，使数据库的运行达到某种性能要求，如响应时间、处理频率、存储空间、维护代价等，保证数据库的安全性，如用户权限等

Database Design Process



1. Requirements Analysis

2. Conceptual Design

3. Logical, Physical, Security, etc.

- 数据库实施阶段：运用DBMS提供的数据语言、工具及宿主语言，根据逻辑设计和物理设计的结果建立数据库、编制与调试应用程序、组织数据入库、并进行试运行
- 数据库运行和维护阶段：数据库应用系统经过试运行后即可投入正式运行。在数据库系统运行过程中必须不断地对其进行评价、调整与修改

- 数据库设计过程是迭代设计过程，不断重复修正

Database Design Process

数据库设计一般要经过以下几个步骤：

- 需求分析阶段 → 数据流图和数据字典 (软工)
- 概念结构设计阶段 → E/R图或UML图
- 逻辑结构设计阶段 → 关系数据库模式 (规范化)
- 数据库物理设计阶段 → 存储方式、索引和用户权限
- 数据库实施阶段
- 数据库运行和维护阶段

5.1 数据库设计

Conceptual Model:

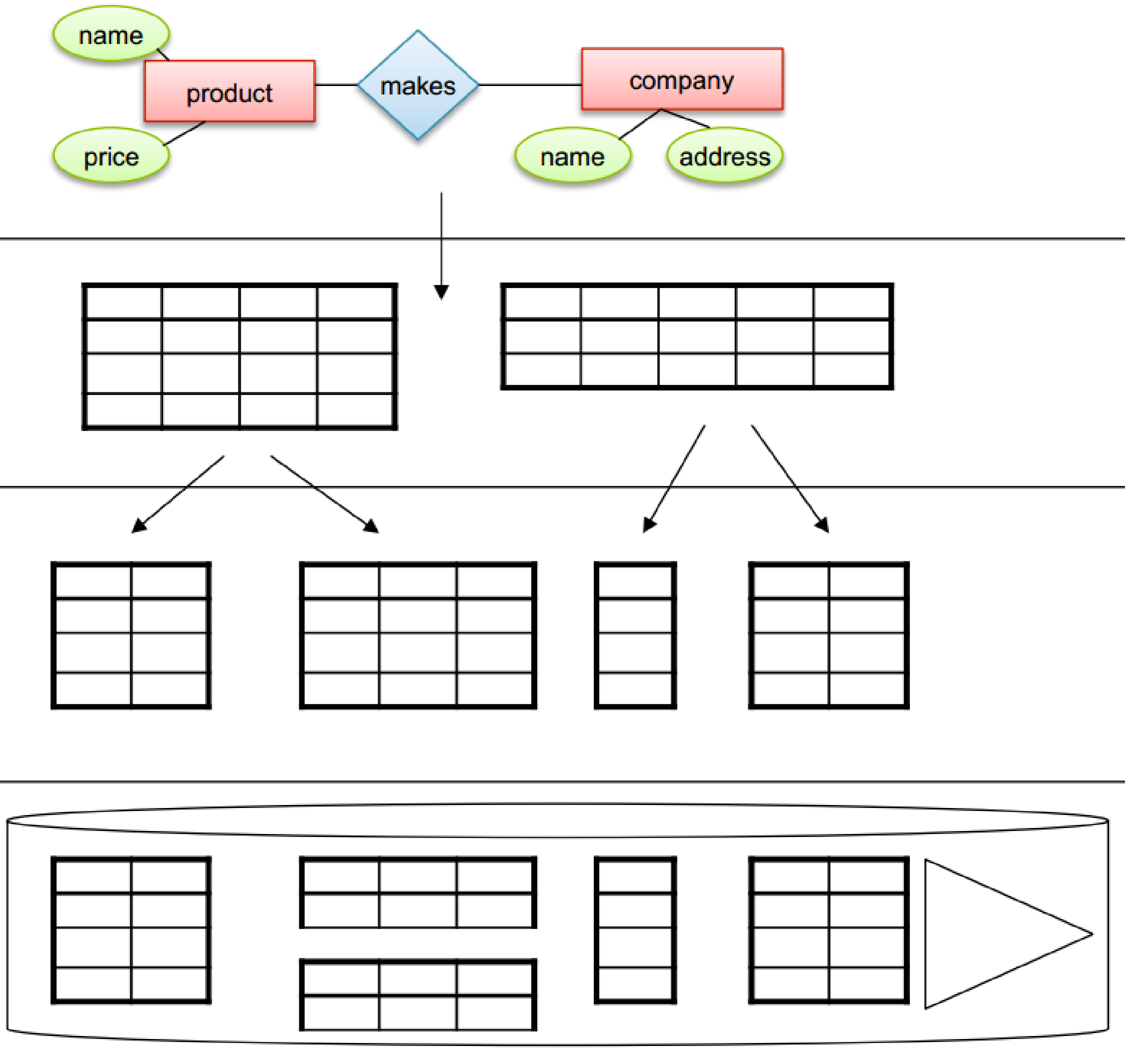
Relational Model:
Tables + constraints
And also functional dep.

Normalization:
Eliminates anomalies

Conceptual Schema

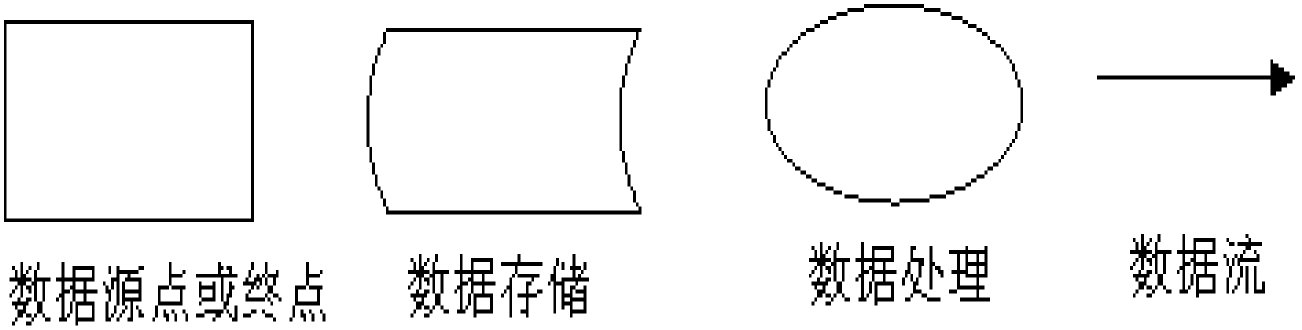
Physical storage details

Physical Schema



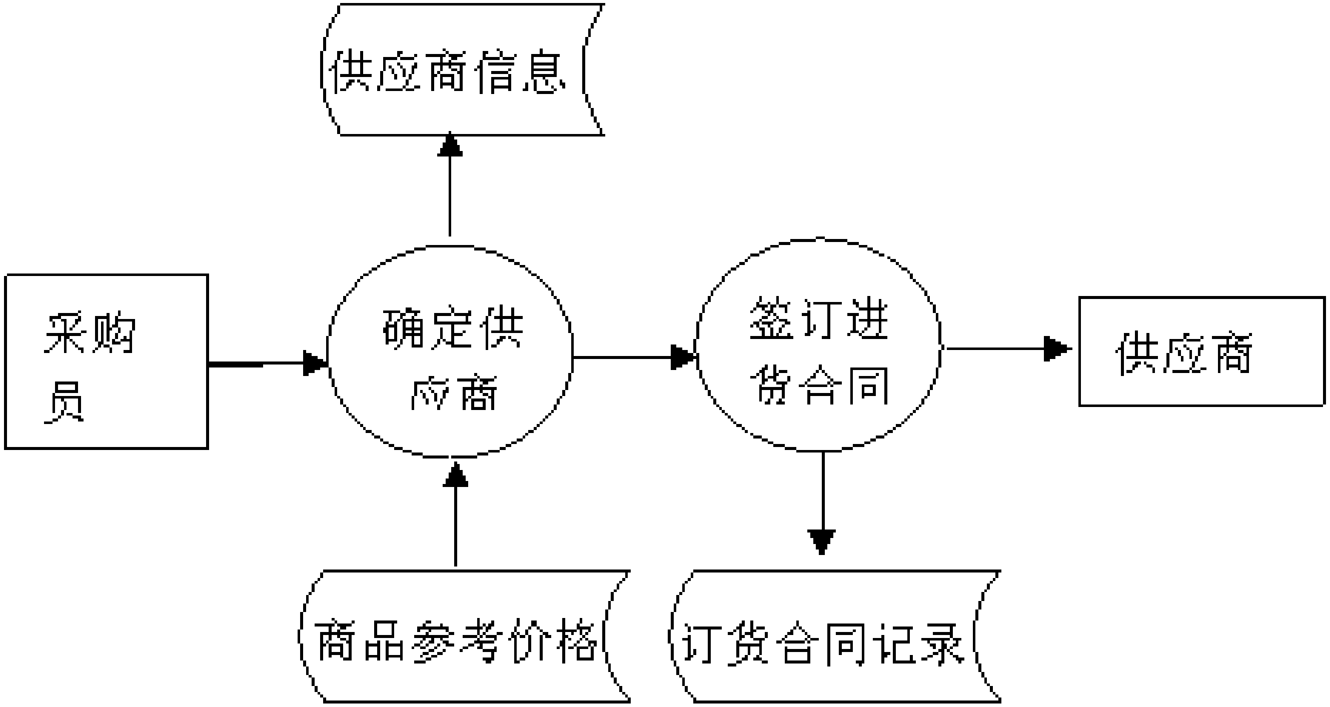
数据流图

- 数据流图
  - 用于表达和描述系统的数据流向和对数据的处理功能
  - 是现行系统的一种逻辑抽象，独立于系统的实现
- 数据流图使用的符号如下



数据流图

- 数据流图示例



数据字典

- 数据字典
  - 数据字典是各类数据描述的集合
  - 数据字典是进行详细的数据收集和数据分析所获得的主要结果
- 数据字典的内容
  - 数据项 (数据的最小组成单位)
  - 数据结构 (若干数据项组成一个数据结构)
  - 数据流
  - 数据存储
  - 处理过程
- 数据字典通过对数据项和数据结构的定义来描述数据流、数据存储的逻辑内容

从数据流图/数据字典到E/R图

- E/R is a visual syntax for DB design which is precise enough for technical points, but abstracted enough for non-technical people
  - *“The Entity-Relationship model – toward a unified view of data”*, Peter Chen, 1976
- 数据抽象对需求分析阶段收集到的数据进行分类、组织(聚集)，形成
  - 实体
  - 实体的属性，标识实体的码
  - 确定实体之间的联系类型(1:1，1:n，m:n)
- 设计分E/R图的步骤
  - 选择局部应用
  - 逐一设计分E/R图

E/R图设计

- 设计分E/R图首先需要根据系统的具体情况，在多层的数据流图中选择一个适当层次的数据流图，让这组图中每一部分对应一个局部应用，然后以这一层次的数据流图为出发点，设计分E/R图
- 通常以中层数据流图作为设计分E/R图的依据，原因
  - 高层数据流图只能反映系统的概貌
  - 中层数据流图能较好地反映系统中各局部应用的子系统组成
  - 低层数据流图过细

E/R图设计

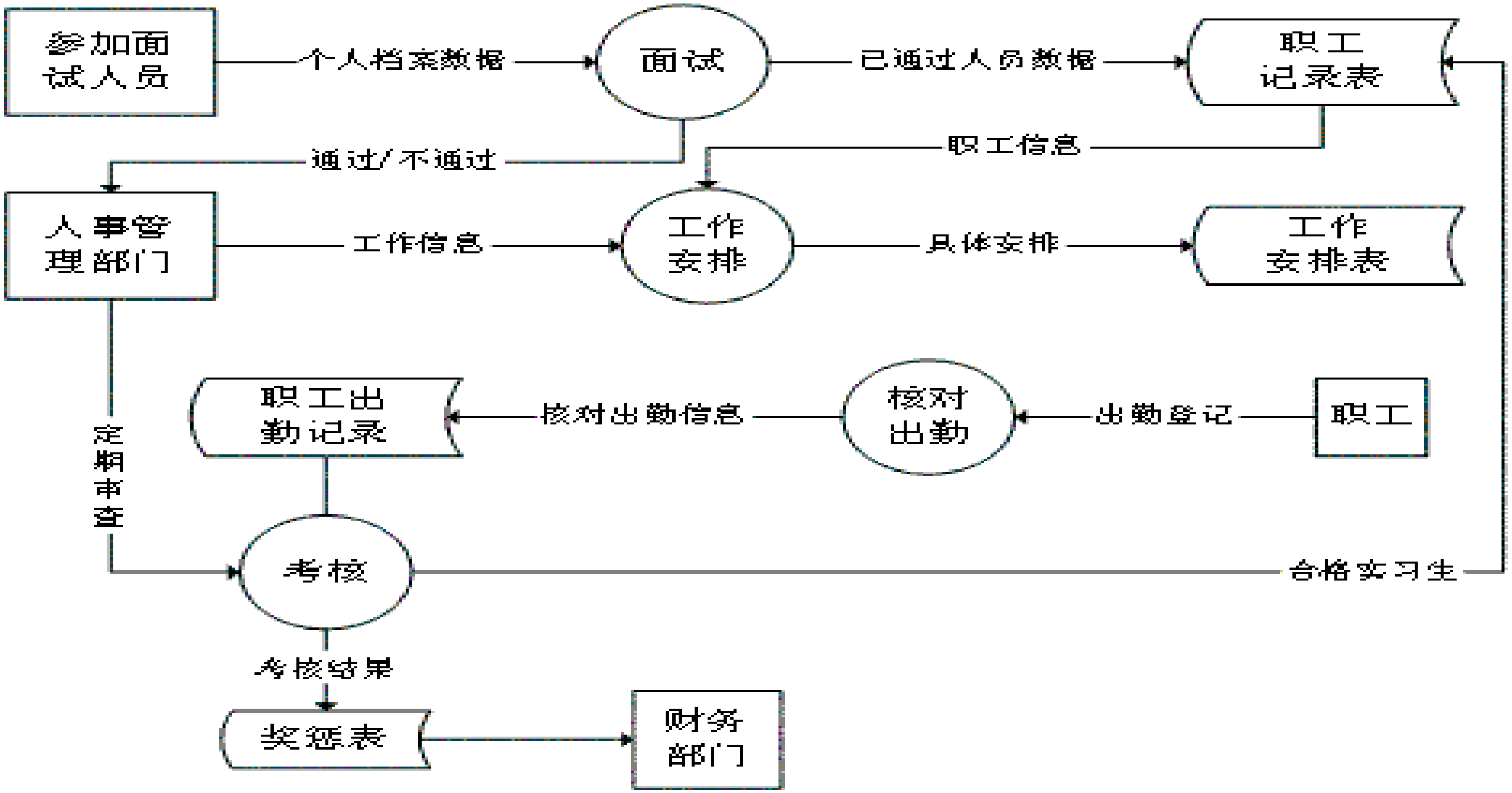
- 将各局部应用涉及的数据分别从数据字典中抽取出来，参照数据流图，标定各局部应用中的实体、实体的属性、标识实体的码，确定实体之间的联系及其类型(1:1, 1:n, m:n)

E/R图设计举例

- 例：某单位人事管理子系统分E/R图设计
- 需求调查、信息流程分析和数据收集，明确子系统的功能
  - 统计各部门人员需求，生成招聘需求信息
  - 查询面试者信息，对初选合格者进行面试
  - 根据面试记录，对面试合格者建立职工记录
  - 建立工作安排表，查询职工排班情况
  - 建立职工出勤记录，生成职工出勤统计表，作为发放奖金的依据

E/R图设计举例

- 该子系统的数据流图



参加面试人员

个人档案数据

面试

已通过人员数据

职工记录表

通过/不通过

职工信息

人事管理部门

工作信息

工作安排

具体安排

工作安排表

职工出勤记录

核对出勤信息

核对出勤

出勤登记

职工

定期审查

考核

合格实习生

考核结果

奖惩表

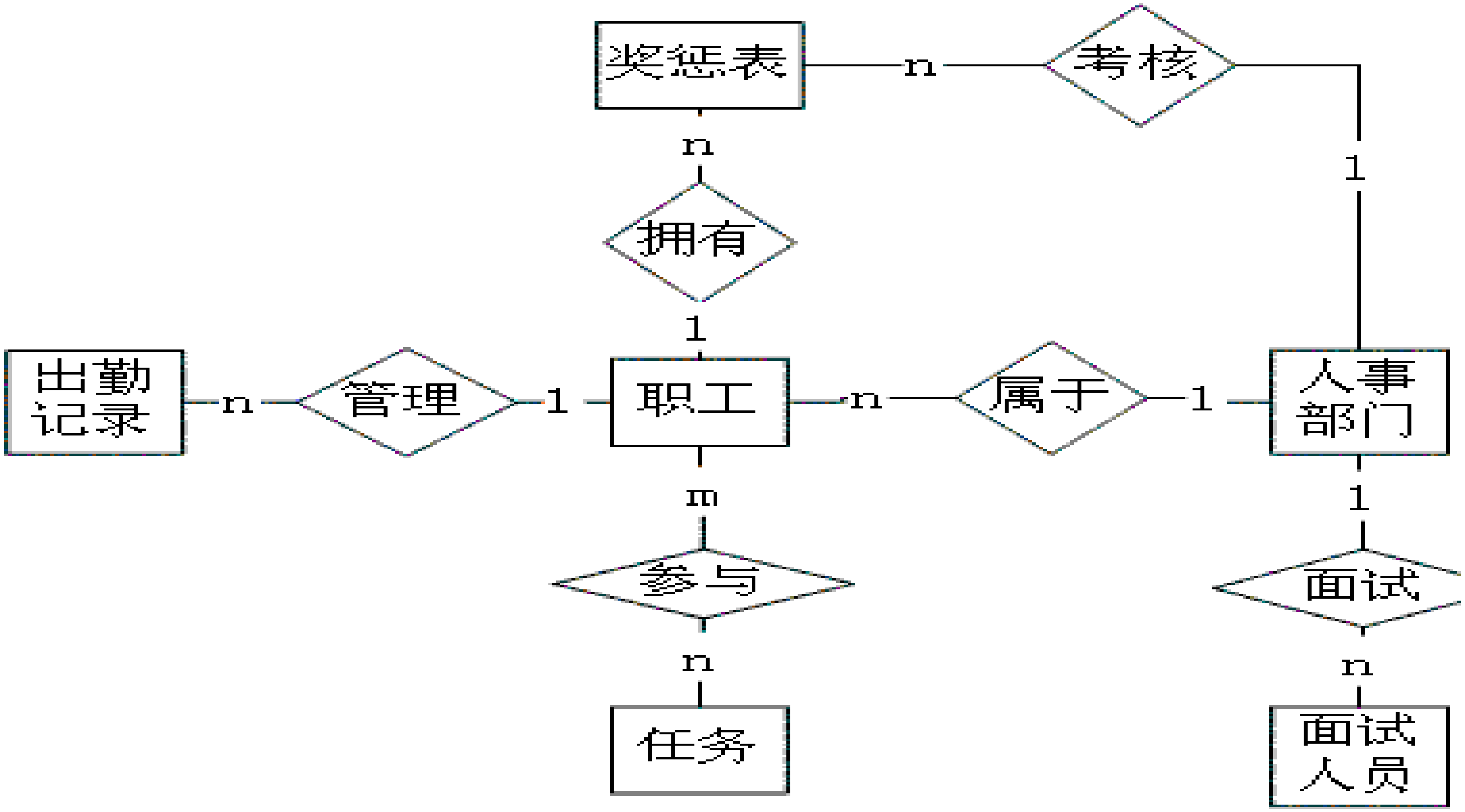
财务部门

E/R图设计举例

- 利用前面介绍的抽象机制，将数据源“参加面试人员”和“人事管理部门”分别抽象为实体“面试人员” 和“人事部门”
- 将数据源“职工”和数据存储“职工记录表”合并在一起，抽象为实体“职工”
- 将数据存储“工作安排表”、“员工出勤记录”、“奖惩表”分别抽象为实体“任务”、“出勤记录” “奖惩表”

E/R图设计举例

- 最后得到的分E/R图



局部E/R图到全局E/R图

- 集成局部视图
  - 一次集成
    - 一次集成多个分E/R图
    - 通常用于局部视图比较简单时
  - 逐步累积式
    - 首先集成两个局部视图 (通常是比较关键的两个局部视图)
    - 以后每次将一个新的局部视图集成进来
  - 平衡集成式
    - 首先将局部视图进行两两合并，然后再将合并后的视图继续进行合并
- 集成中需要解决的问题
  - 解决冲突
  - 修改与重构

局部E/R图到全局E/R图

- 解决冲突
  - 两类属性冲突
    - 属性域冲突：属性的类型、取值范围或取值集合不同
    - 属性单位冲突
  - 两类命名冲突
    - 同名异义
    - 异名同义 (一义多名)
  - 三类结构冲突
    - 同一对象在不同应用中具有不同的抽象
    - 同一实体在不同局部视图中所包含的属性不完全相同，或者属性的排列次序不完全相同
    - 实体之间的联系在不同局部视图中呈现不同的类型

局部E/R图到全局E/R图

- 修改与重构
  - 冗余的数据是指可由基本数据导出的数据，冗余的联系是指可由其他联系导出的联系
  - 冗余数据和冗余联系容易破坏数据库的完整性，给数据库维护增加困难
  - 消除不必要的冗余后的初步E/R图称为基本E/R图
  - 分析方法
    - 以数据字典和数据流图为依据，根据数据字典中关于数据项之间逻辑关系的说明来消除冗余

第五章 空间扩展E/R图

- 5.1 Database Design
- 5.2 E/R Basic: Entities & Relations
- 5.3 E/R Design Considerations
  - 5.3.1 Relations: multiplicity, multi-way
  - 5.3.2 Design considerations
  - 5.3.3 Conversion to SQL
- 5.4 Advanced E/R Concepts
  - 5.4.1 Subclasses & connection to OO
  - 5.4.2 Constraints
  - 5.4.3 Weak entity sets
- 5.5 Spatial Database Design Example

参考教材：
数据库系统概念 7.1-7.10
空间数据库管理系统概论 5.1-5.4

5.2.1 Entities and Entity Sets

- Entities & entity sets are the primitive unit of the E/R model
  - Entities are the individual objects, which are members of entity sets
    - Ex: A specific person or product

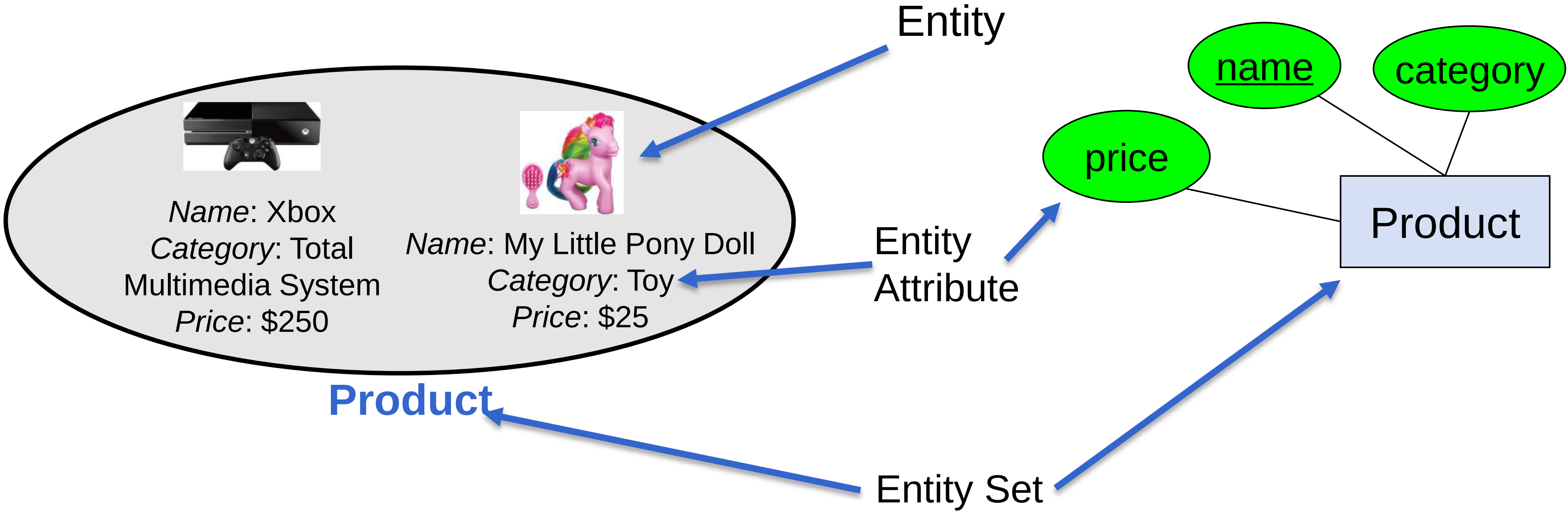
  - Entity sets are the *classes* or *types* of objects in our model
    - Ex: Person, Product
    - These are what is shown in E/R diagrams - as rectangles
    - Entity sets represent the sets of all possible entities

Product

Person

Entities and Entity Sets

- An entity set has attributes
  - Represented by ovals attached to an entity set
  - Shapes are important, colors are not
- Entities are not explicitly represented in E/R diagrams



Entity

Name: Xbox
Category: Total Multimedia System
Price: $250

Name: My Little Pony Doll
Category: Toy
Price: $25

Product

Entity Attribute

Entity Set

name

category

price

Product

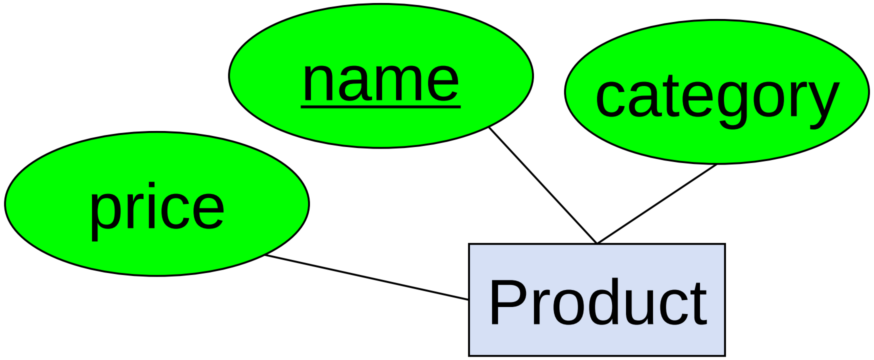
Keys

- A key is a minimal set of attributes that uniquely identifies an entity

Denote elements of the primary key by underlining

Here, {name, category} is **not** a key (it is not *minimal*).

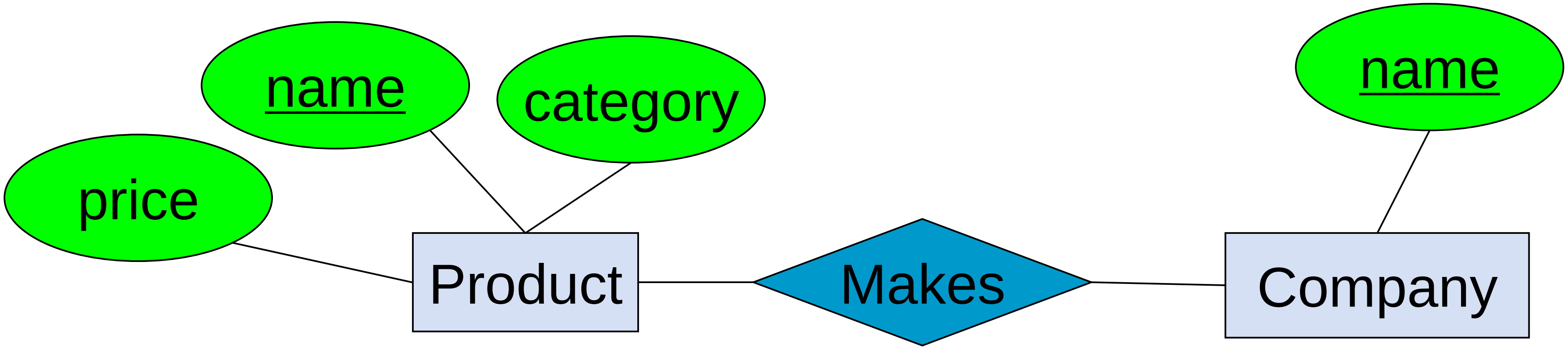
If it were, what would it mean?

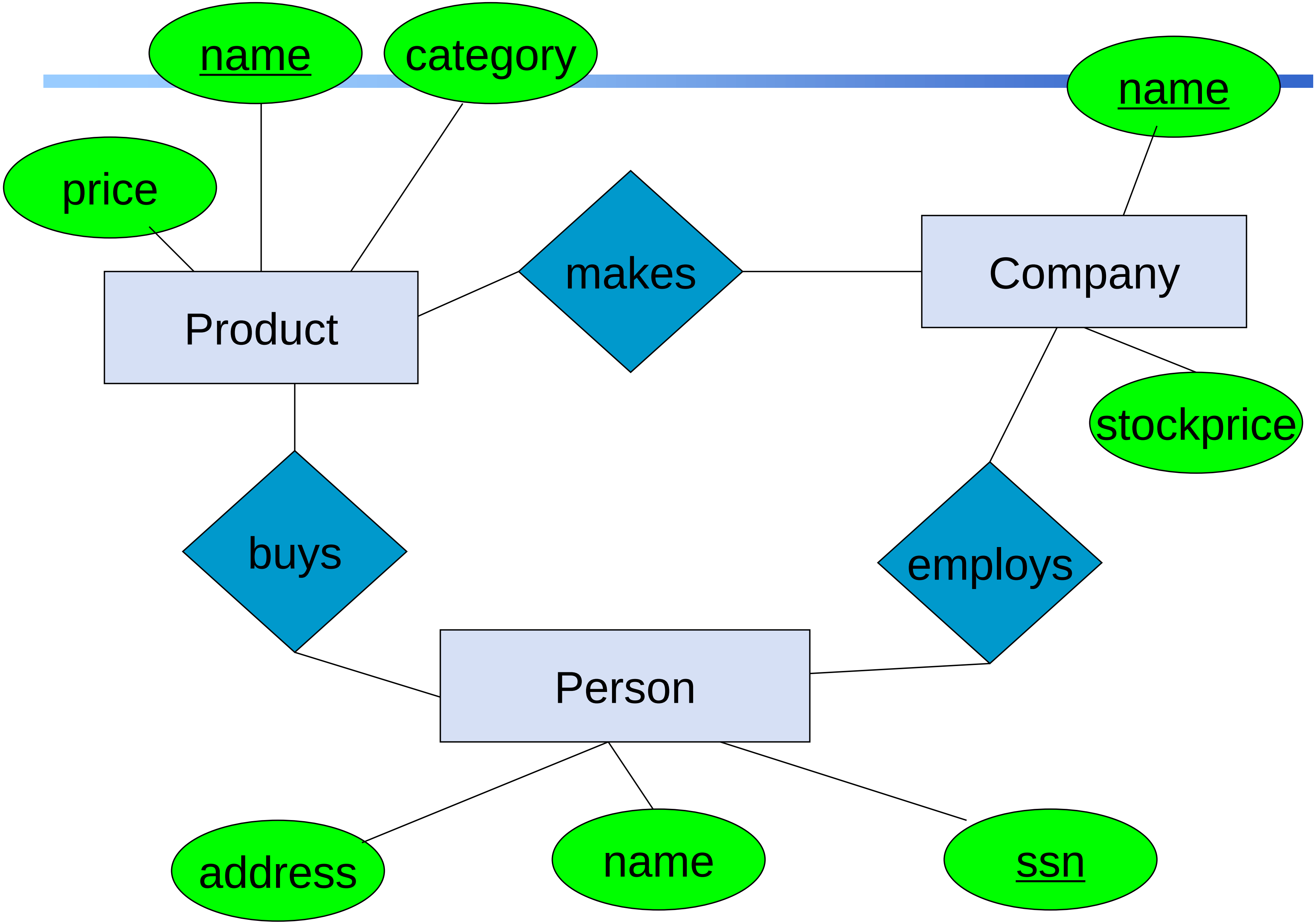


The E/R model forces us to designate a single primary key, though there may be multiple candidate keys

5.2.2 The R in E/R: Relationships

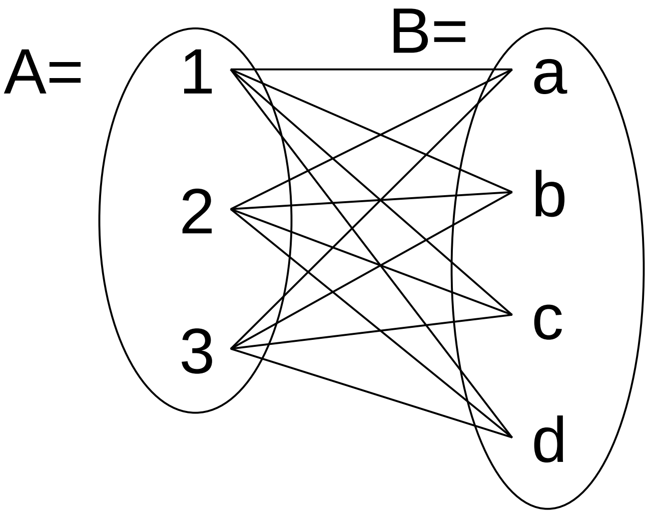
- A relationship is between two entities

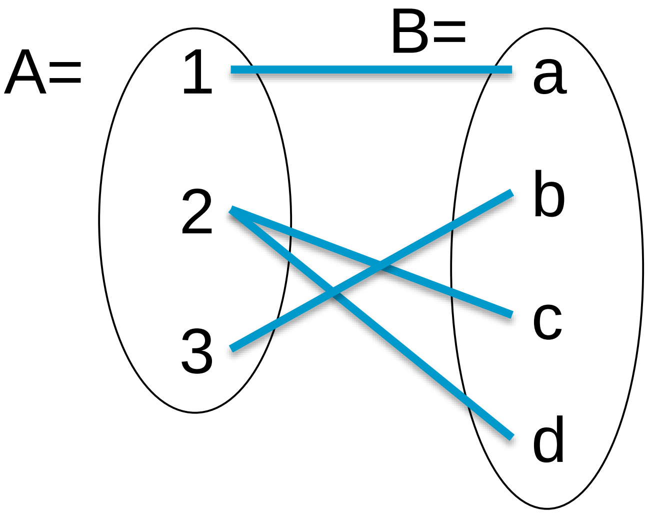



name
category
name
price
makes
Company
Product
stockprice
buys
employs
Person
address
name
ssn

What is a Relationship?

- A mathematical definition
  - Let A, B be sets
    - $A=\{1,2,3\}, \quad B=\{a,b,c,d\}$
  - A x B (the cross-product) is the set of all pairs (a,b)
    - $A \times B = \{(1,a), (1,b), (1,c), (1,d), (2,a), (2,b), (2,c), (2,d), (3,a), (3,b), (3,c), (3,d)\}$
  - We define a relationship to be a subset of A x B
    - $R = \{(1,a), (2,c), (2,d), (3,b)\}$

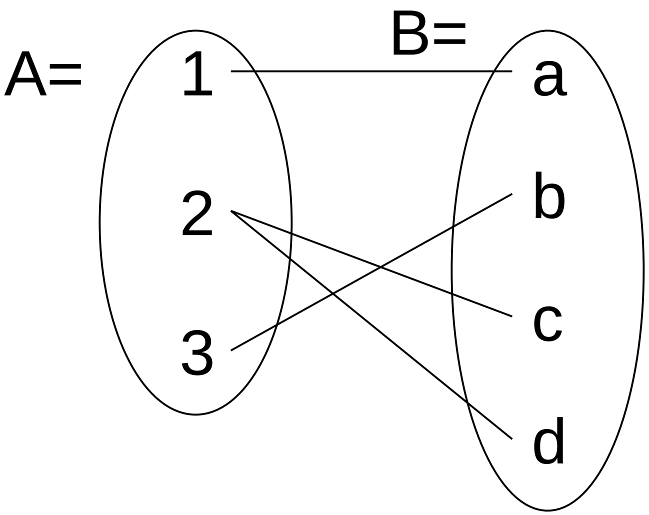


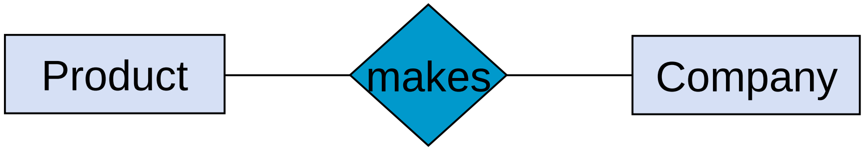


What is a Relationship?

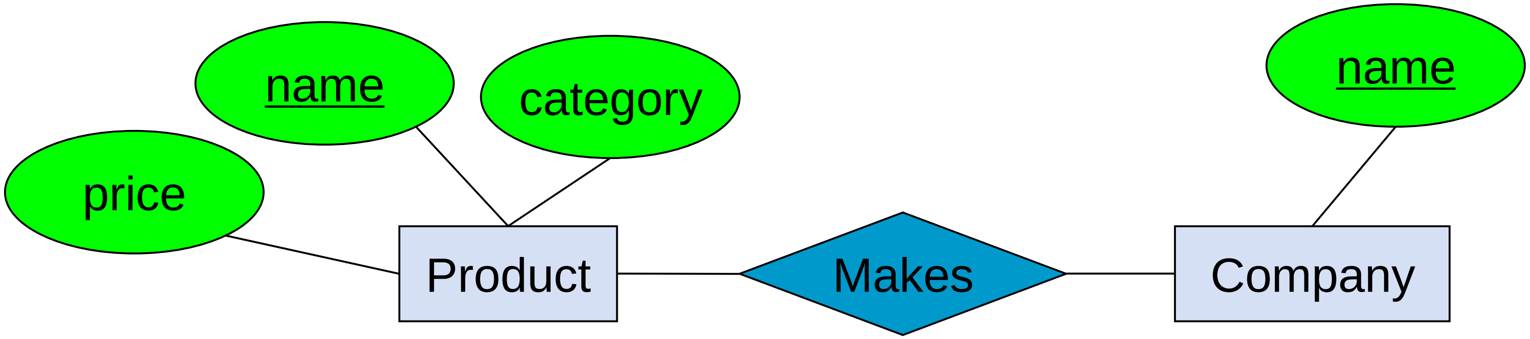
- A mathematical definition
  - Let A, B be sets
  - A x B (the cross-product) is the set of all pairs
  - A relationship is a subset of A x B

- Makes is relationship - it is a subset of Product × Company





What is a Relationship?



A relationship between entity sets P and C is a subset of all possible pairs of entities in P and C, with tuples uniquely identified by P and C’s keys

What is a Relationship?

Company

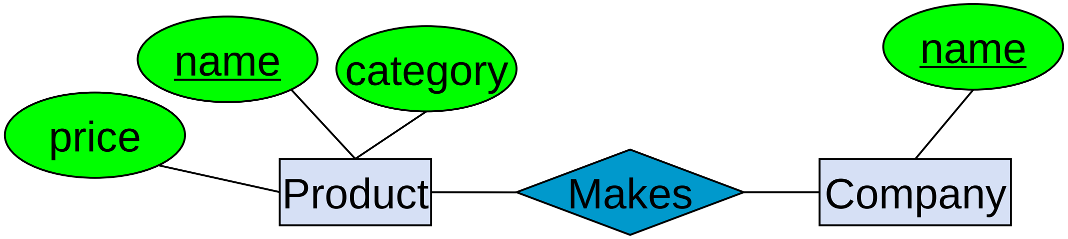
name
GizmoWorks
GadgetCorp

Product

name	category	price
Gizmo	Electronics	$9.99
GizmoLite	Electronics	$7.50
Gadget	Toys	$5.50

Company C × Product P

C.name	P.name	P.category	P.price
GizmoWorks	Gizmo	Electronics	$9.99
GizmoWorks	GizmoLite	Electronics	$7.50
GizmoWorks	Gadget	Toys	$5.50
GadgetCorp	Gizmo	Electronics	$9.99
GadgetCorp	GizmoLite	Electronics	$7.50
GadgetCorp	Gadget	Toys	$5.50



Makes

C.name	P.name
GizmoWorks	Gizmo
GizmoWorks	GizmoLite
GadgetCorp	Gadget

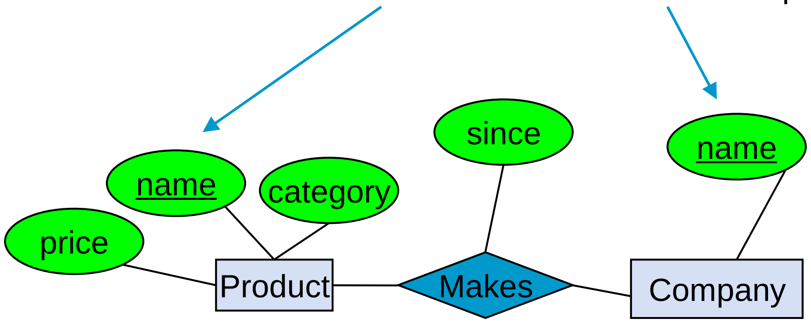
A relationship between entity sets P and C is a subset of all possible pairs of entities in P and C, with tuples uniquely identified by P and C’s keys

What is a Relationship?

- There can only be one relationship for every unique combination of entities
  - This follows from our mathematical definition of a relationship - it's a SET!
- This also means that the relationship is uniquely determined by the keys of its entities
  - *Example: the "key" for Makes is*

{Product.name, Company.name}

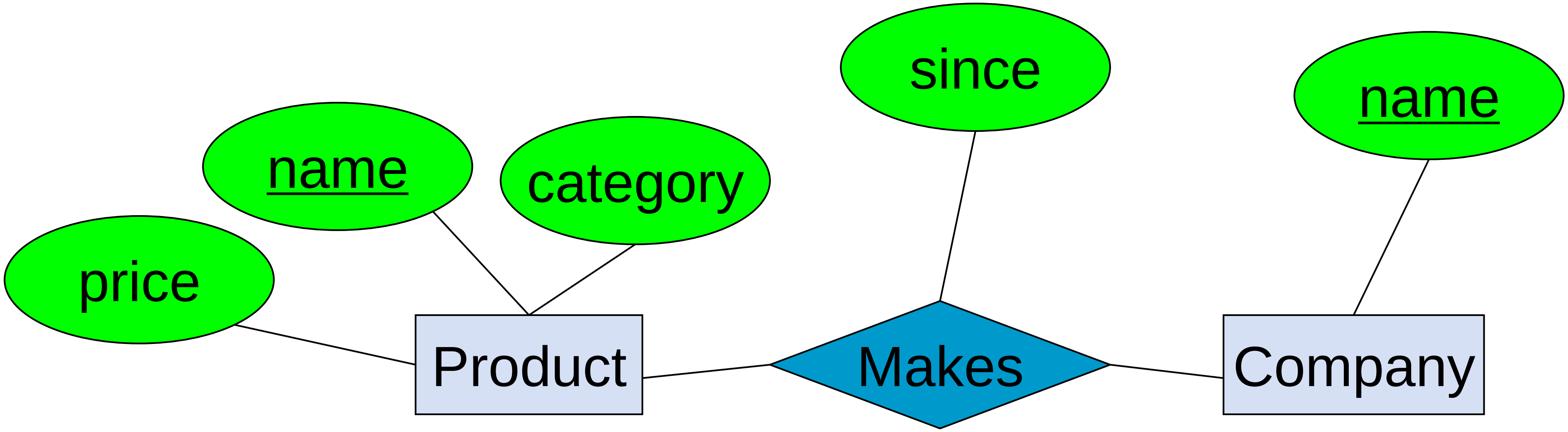
$$Key_{Makes} = Key_{Product} \cup Key_{Company}$$



思考：Why does this make sense?

Relationships and Attributes

- Relationships may have attributes as well



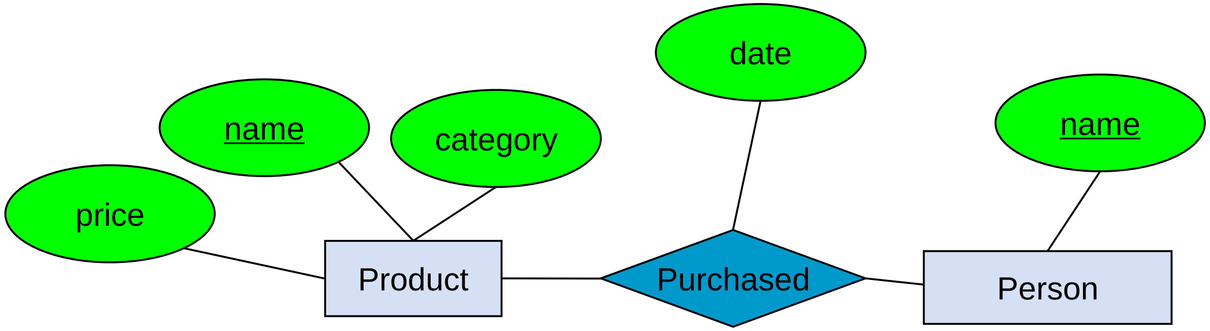
For example: "since" records when company started making a product

Note: "*since*" is implicitly unique per pair here! Why?

Note #2: Why not "how long"?

Decision: Relationship vs. Entity?

- **Q:** What does this say?

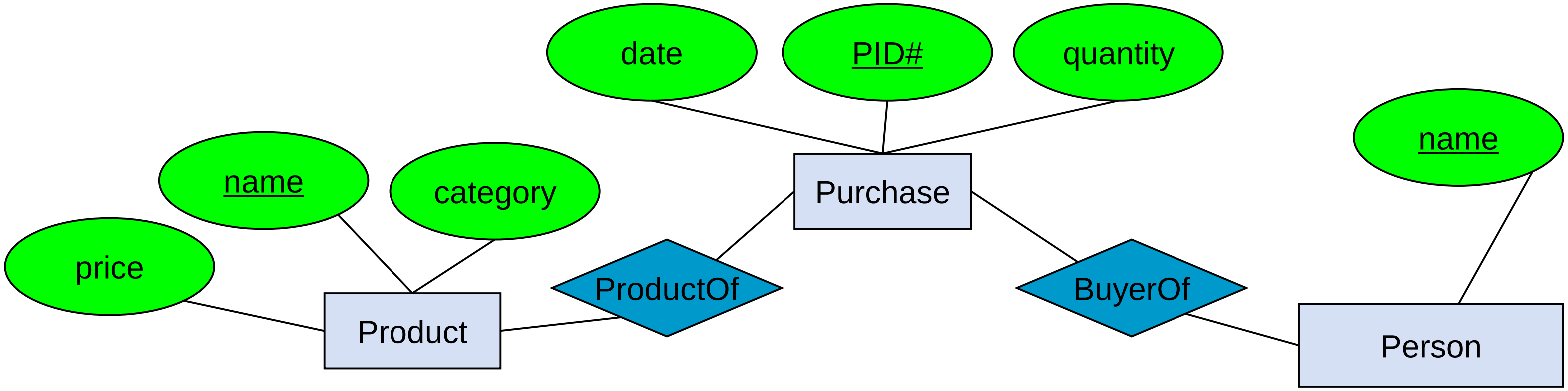


- **A:** A person can only buy a specific product once (on one date)

Modeling something as a relationship makes it unique; what if not appropriate?

Decision: Relationship vs. Entity?

- What about this way?

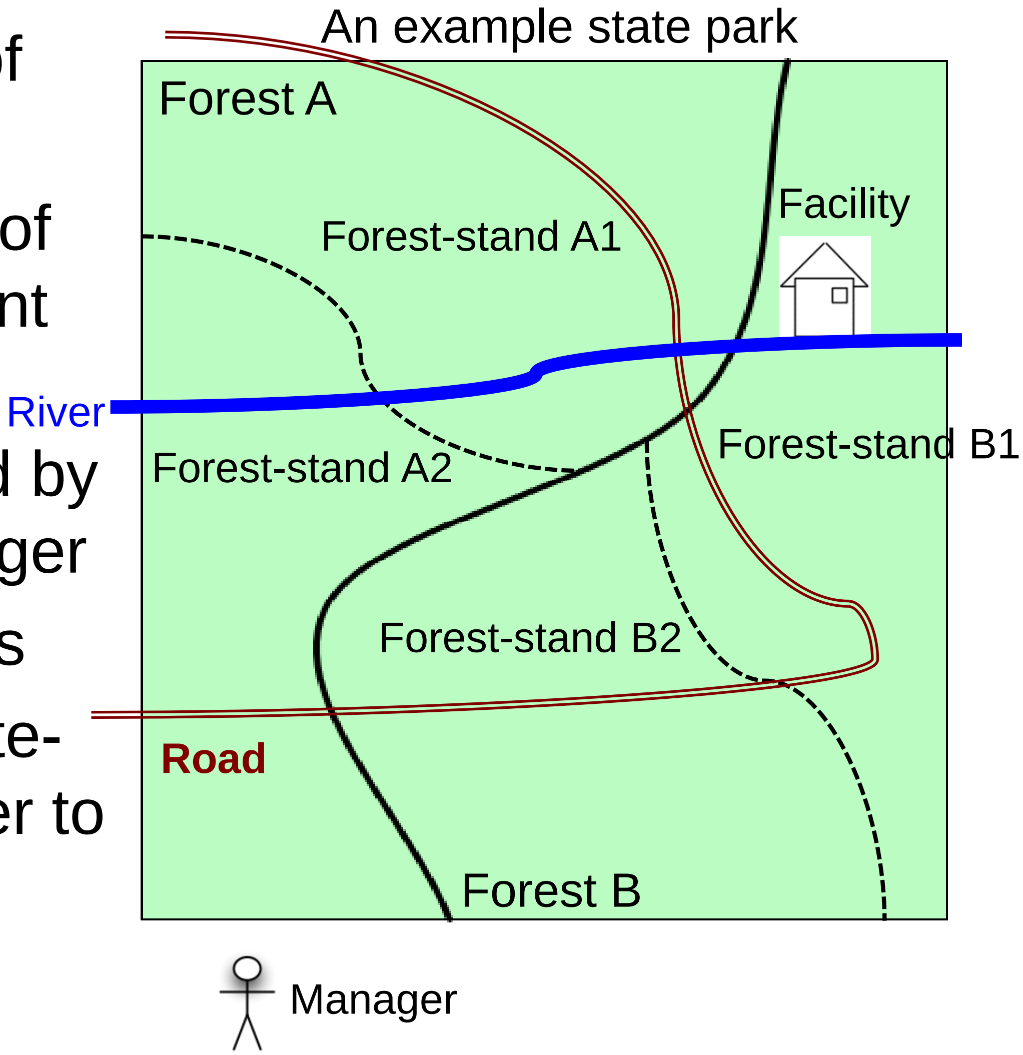


- *Now we can have multiple purchases per product, person pair!*

We can always use a new entity instead of a relationship. For example, to permit multiple instances of each entity combination!

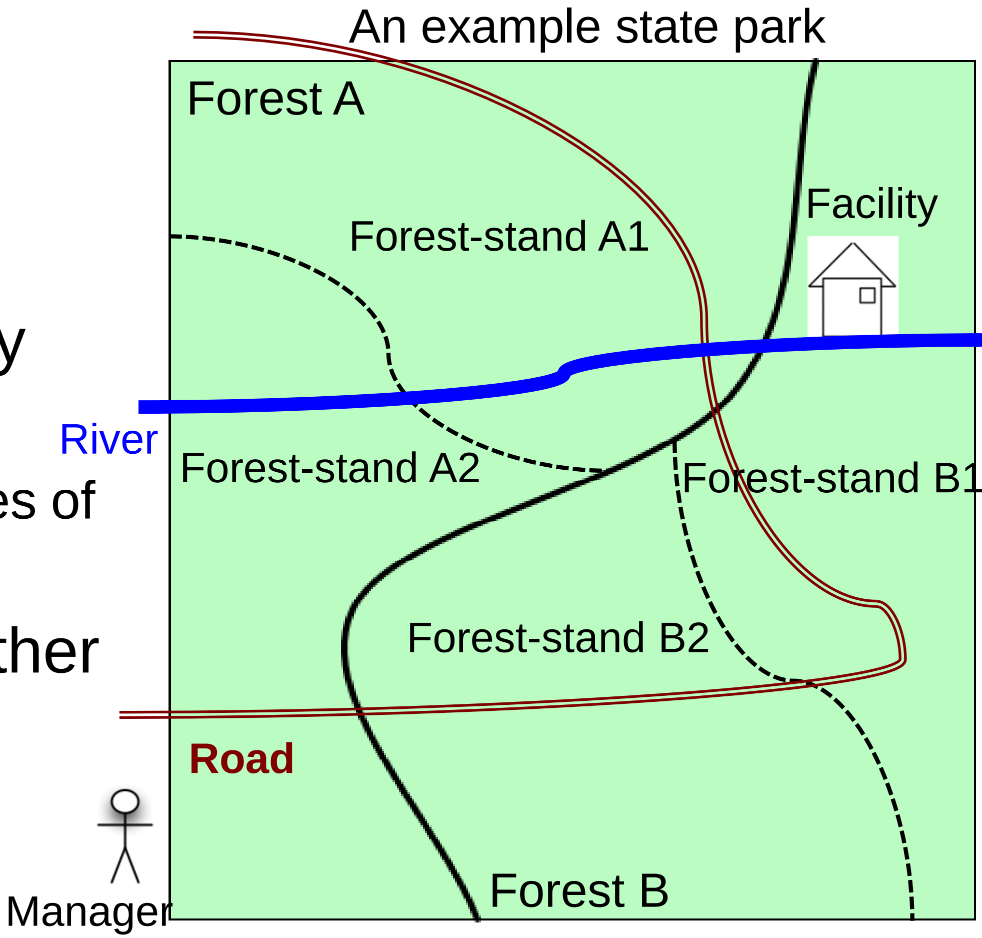
Example Application Domain

- Spatial application domain
  - A state-park consists of forests
  - A forest is a collection of forest-stands of different species
  - State-Park is accessed by roads and has a manager
  - State-Park has facilities
  - River runs through state-park and supplies water to the facilities



Example Application Domain

- Entities have an independent conceptual or physical existence
  - Examples: Forest, Road, Manager, ...
- Entities are characterized by Attributes
  - Example: Forest has attributes of name, elevation, etc.
- An Entity interacts with another Entity through relationships
  - Road allow access to Forest interiors
  - This relationship may be name "Accesses"



第五章 空间扩展E/R图

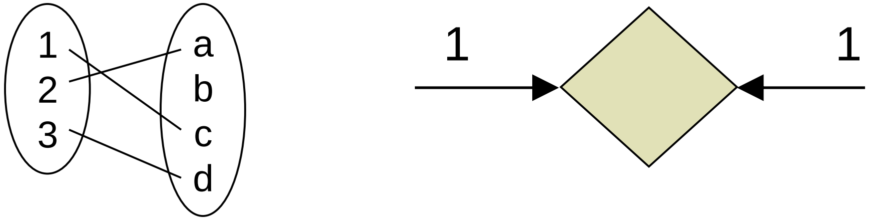
- 5.1 Database Design
- 5.2 E/R Basic: Entities & Relations
- 5.3 E/R Design Considerations
  - 5.3.1 Relations: multiplicity, multi-way
  - 5.3.2 Design considerations
  - 5.3.3 Conversion to SQL
- 5.4 Advanced E/R Concepts
  - 5.4.1 Subclasses & connection to OO
  - 5.4.2 Constraints
  - 5.4.3 Weak entity sets
- 5.5 Spatial Database Design Example

参考教材：
数据库系统概念 7.1-7.10
空间数据库管理系统概论 5.1-5.4

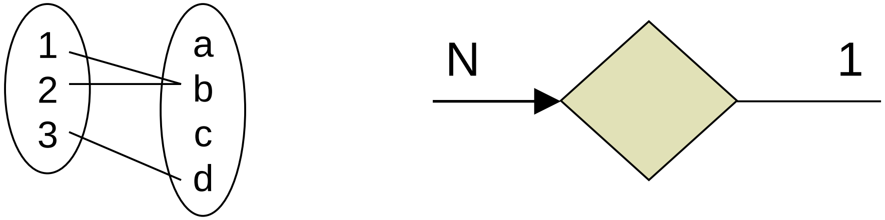
5.3.1 Multiplicity of E/R Relationships

- Indicated using arrows
  - X -> Y means **there exists a function mapping from X to Y** (*recall the definition of a function*)

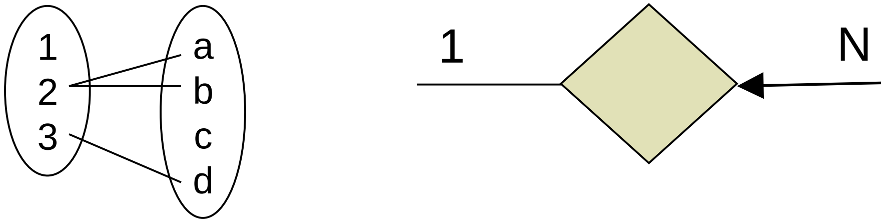
One-to-one:



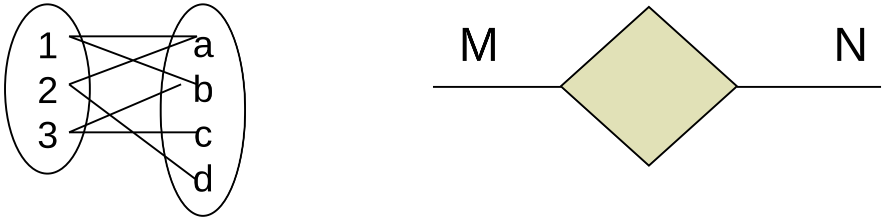
Many-to-one:

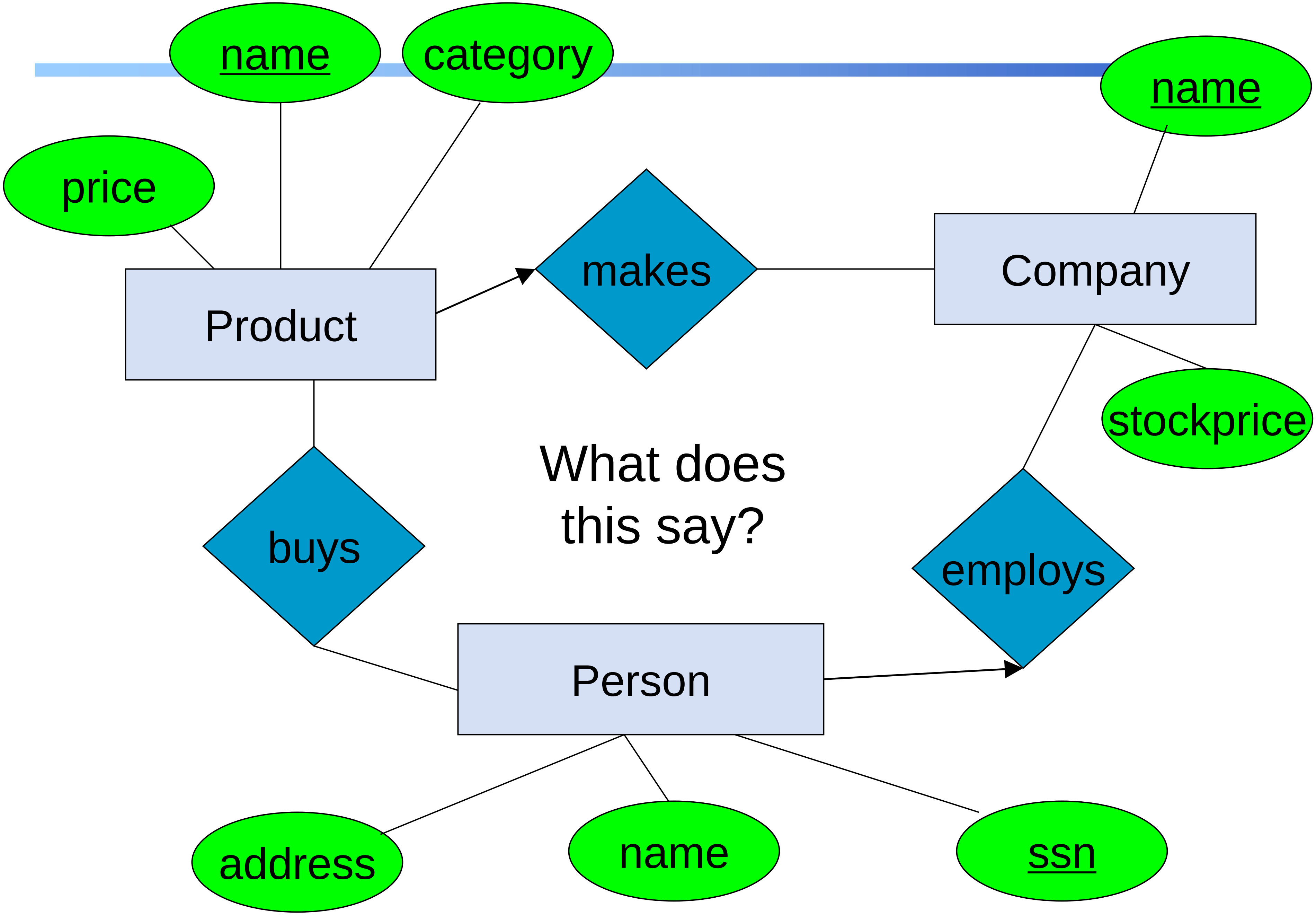


One-to-many:



Many-to-many:

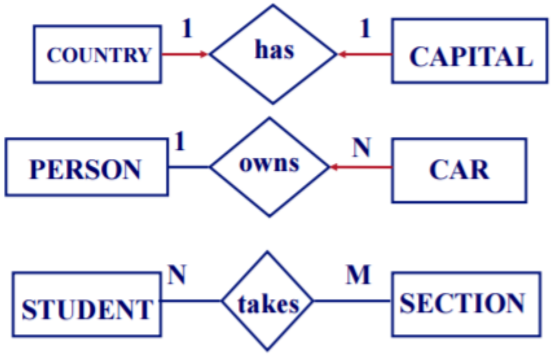




What does this say?

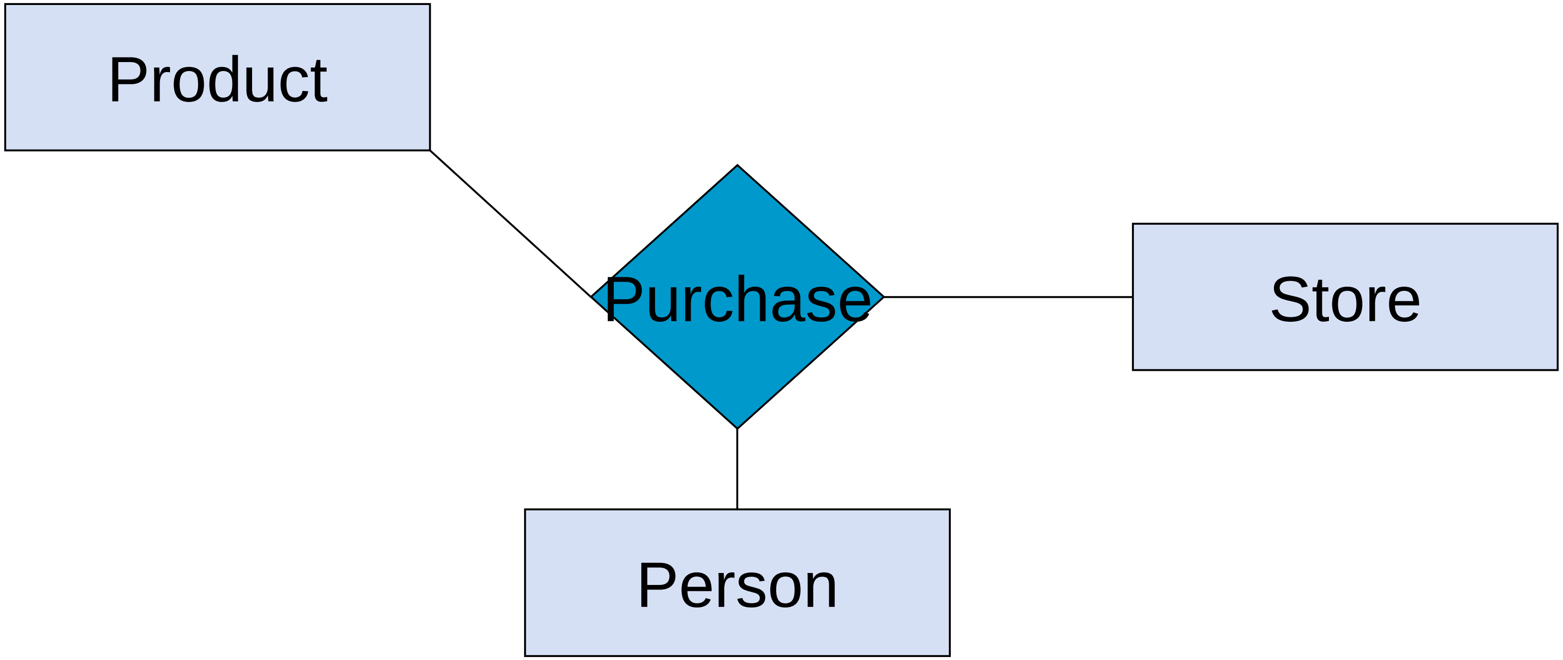
Multiplicity of E/R Relationships

- One-to-One
  - Each state in the United States has one capital, and each capital is in one state
- Many-to-One
  - Each state in the United States has many cities, and each city is in one state
- Many-to-Many
  - Each country may be crossed by multiple rivers, and each river may cross multiple countries



Multi-way Relationships

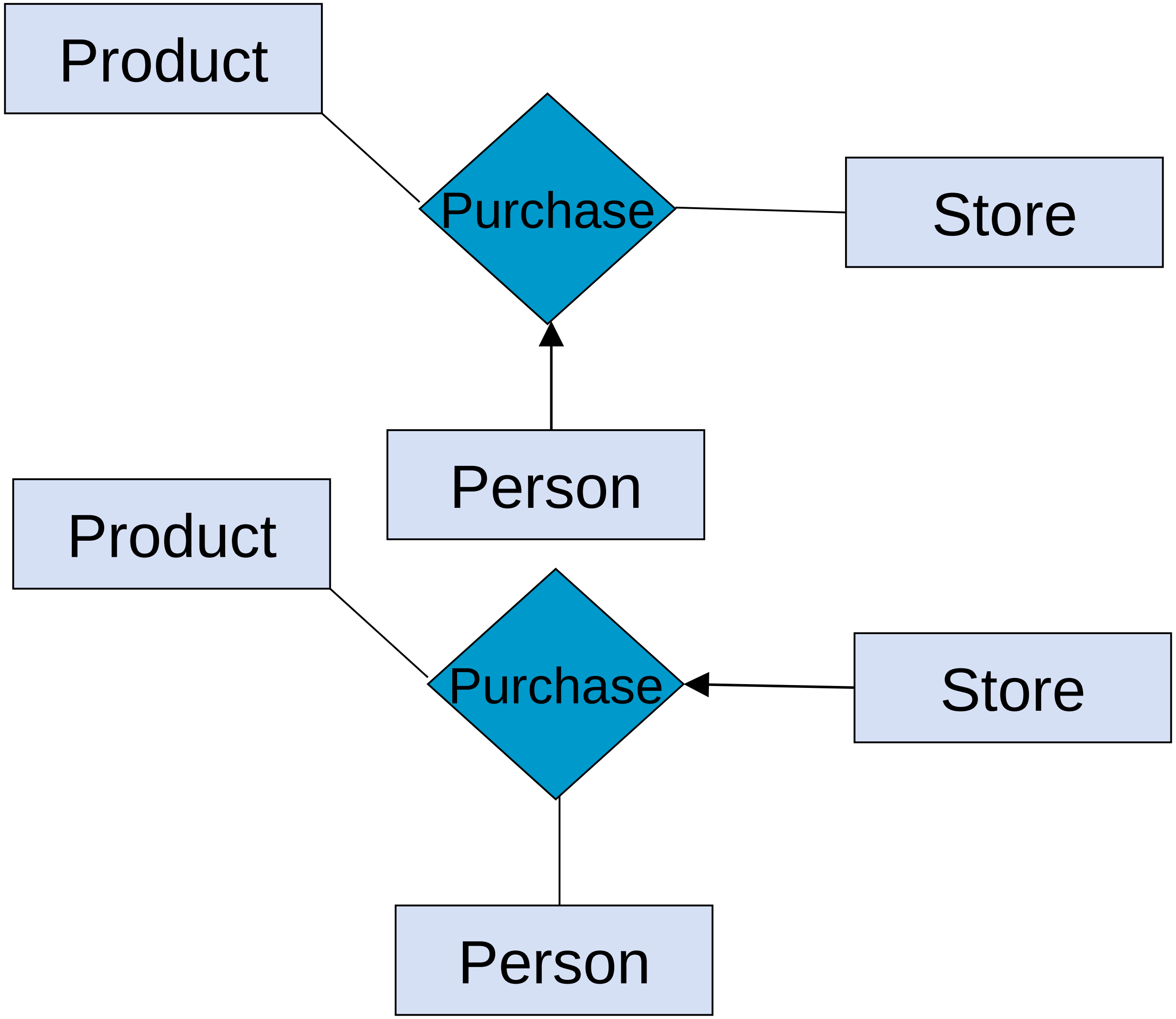
- How do we model a purchase relationship between buyers, products and stores?



NB: Can still model as a mathematical set (how?)

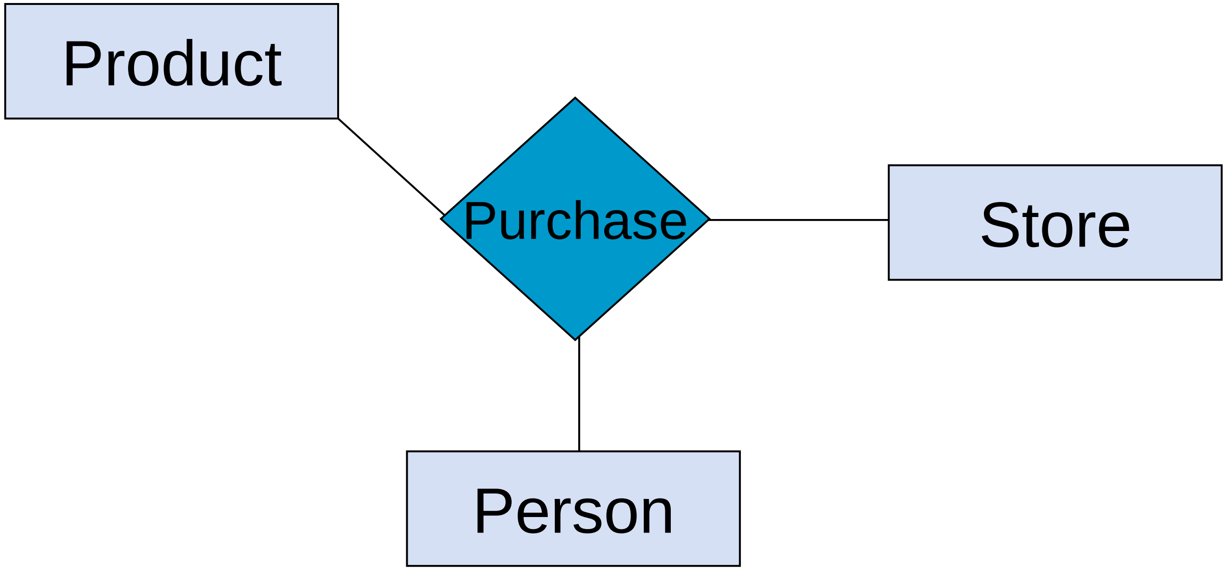
Arrows in Multiway Relationships

- **Q**: What does the arrow mean?



Arrows in Multiway Relationships

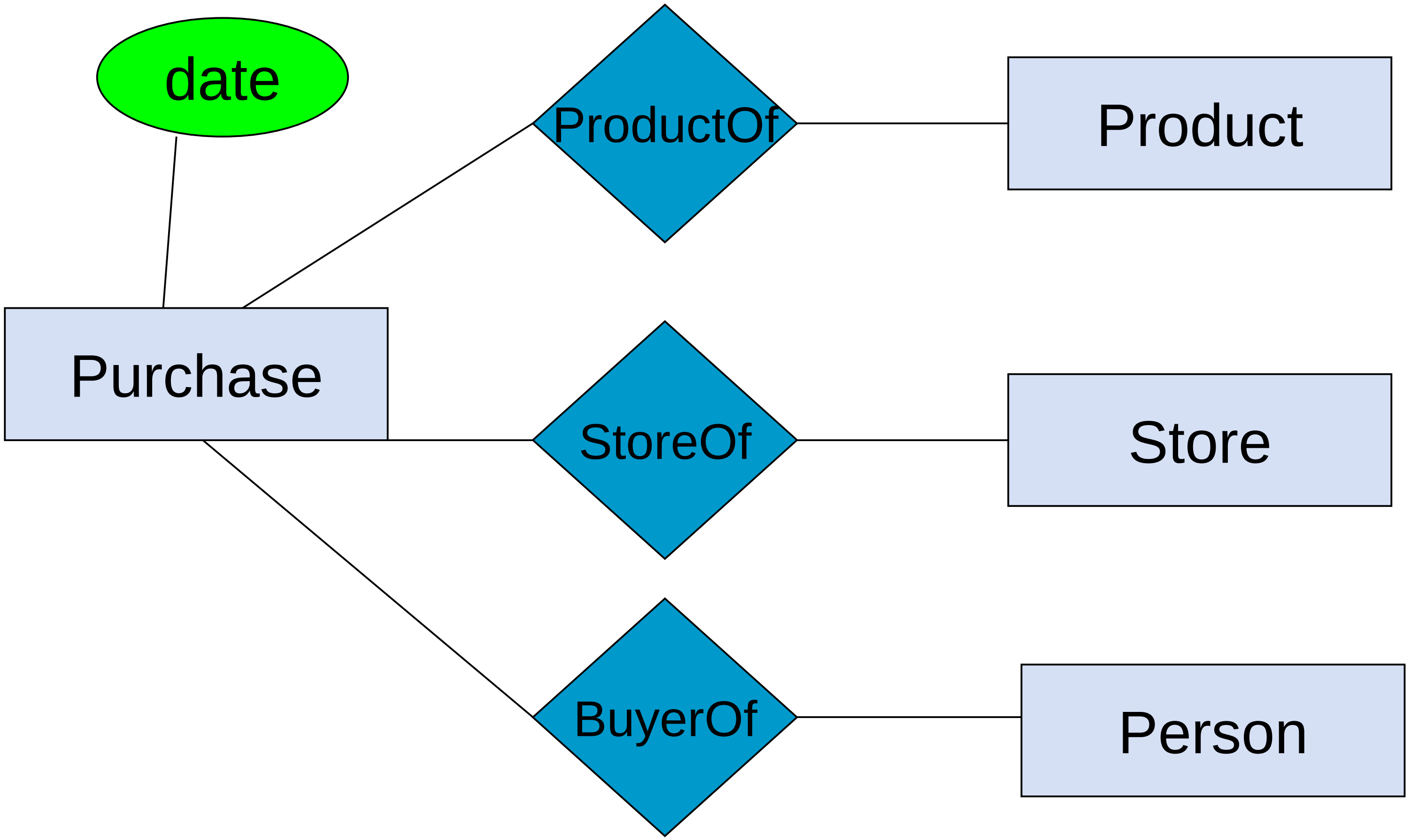
- **Q**: How do we say that every person shops in at most one store?



- **A**: Cannot. This is the best approximation. (Why only approximation?)

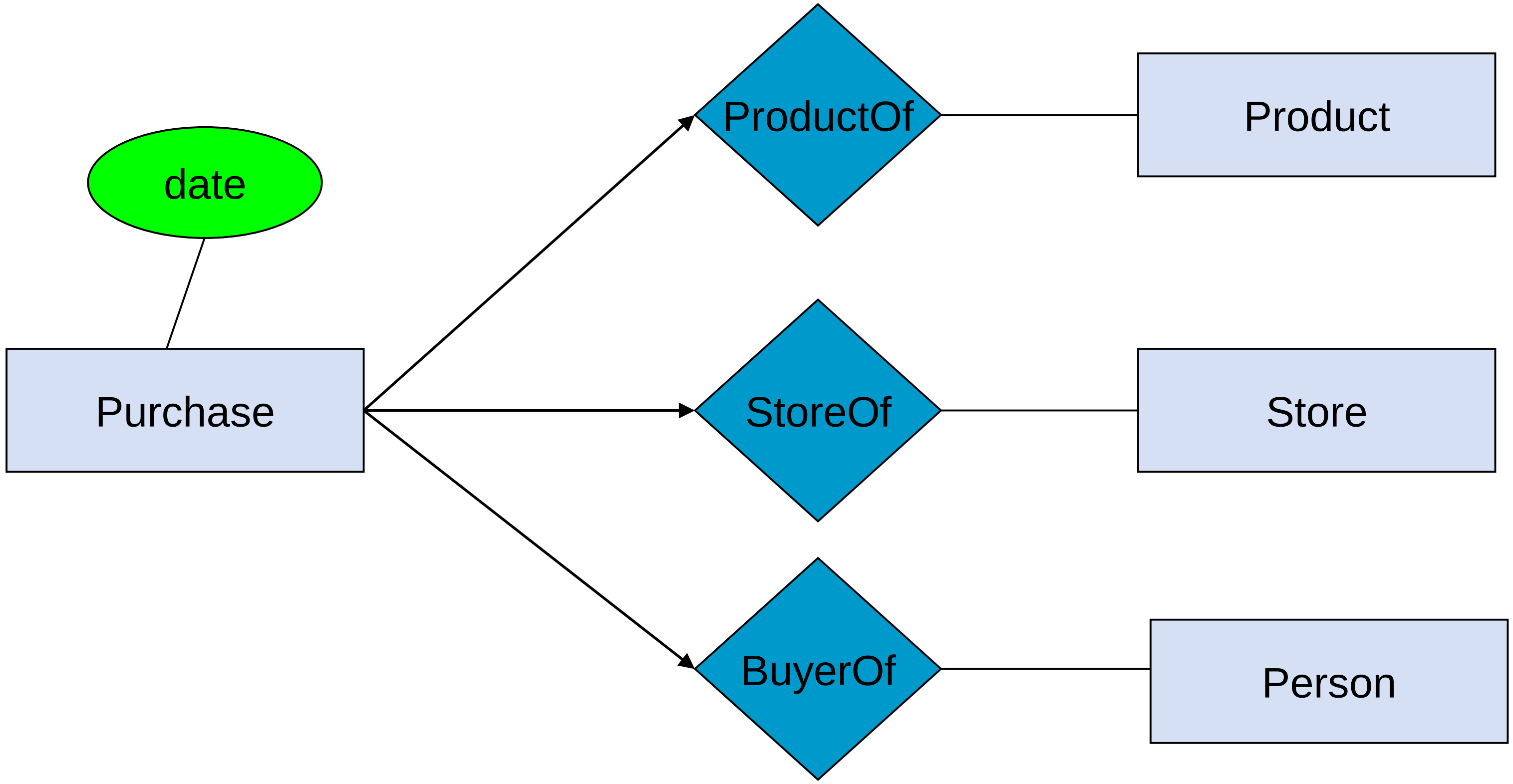
Converting Multi-way Relationships to Entity + Binary

- From what we had on previous slide to this - what did we do?



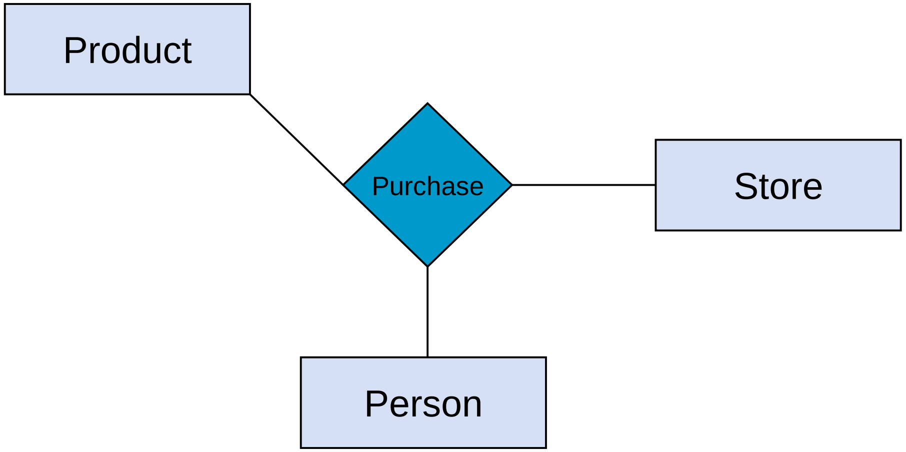
Converting Multi-way Relationships to Entity + Binary

- What arrows should be added here? Are these correct?

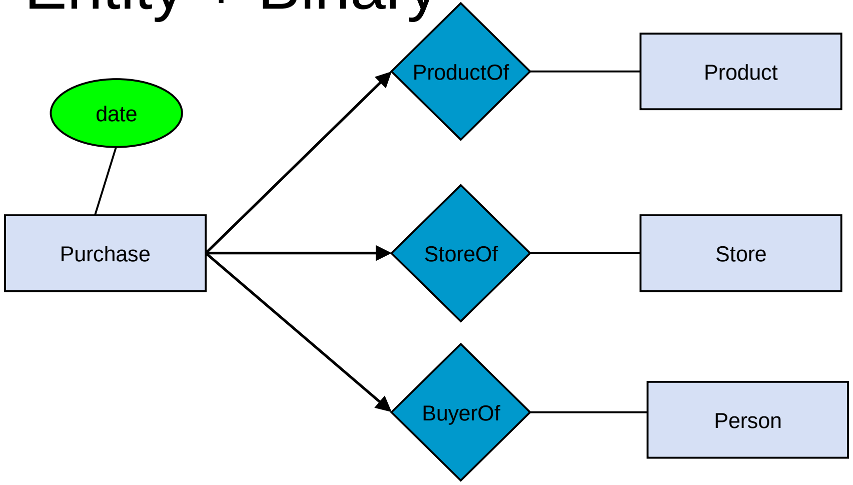


Decision: Multi-way or New Entity + Binary?

Multi-way Relationship



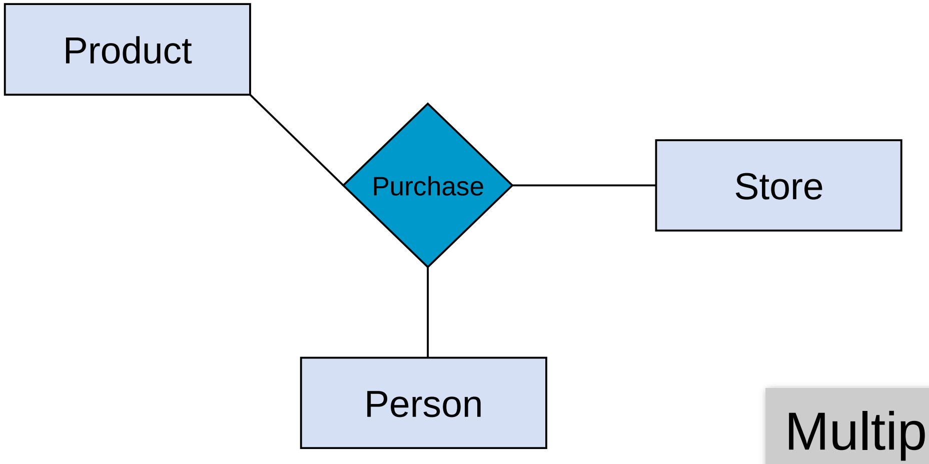
Entity + Binary



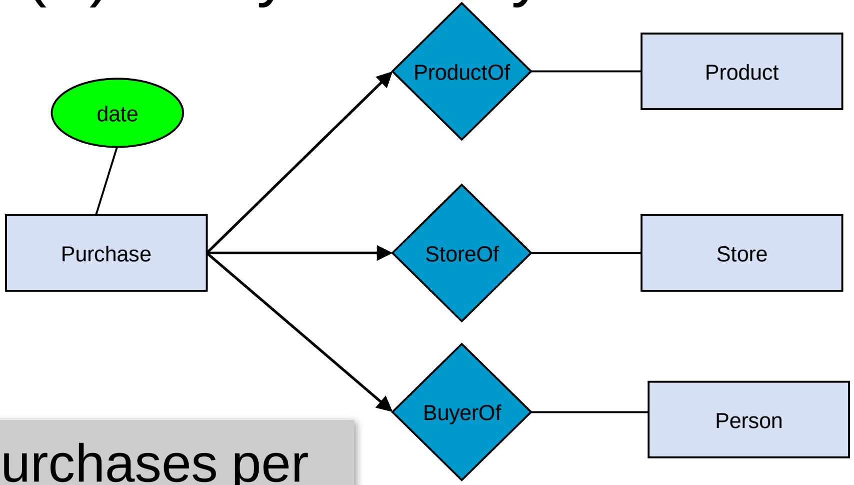
Should we use a single multi-way relationship or a new entity with binary relations**?**

Decision: Multi-way or New Entity + Binary?

(A) Multi-way Relationship



(B) Entity + Binary

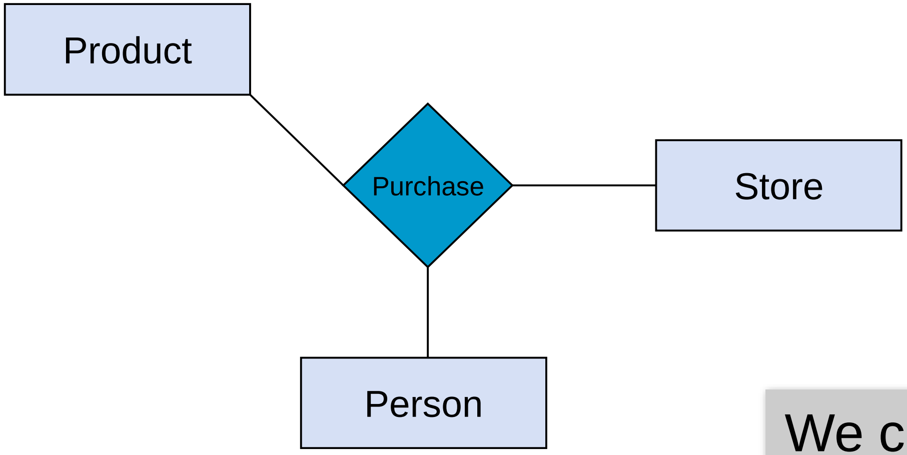


Multiple purchases per (product, store, person) combo possible here!

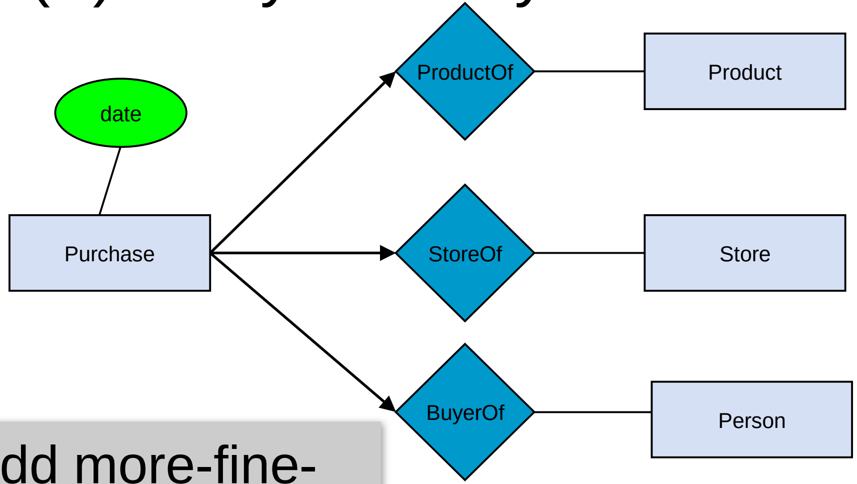
- *Covered earlier:* (B) is useful if we want to have multiple instances of the “relationship” per entity combination

Decision: Multi-way or New Entity + Binary?

(A) Multi-way Relationship



(B) Entity + Binary

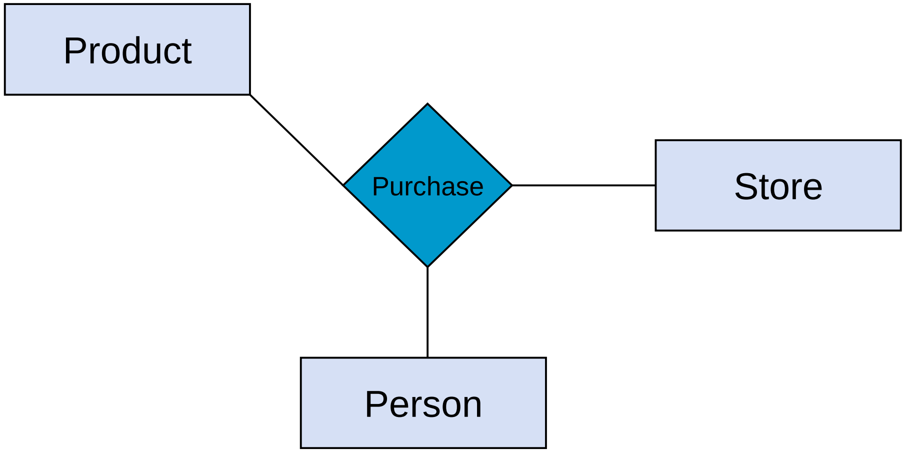


We can add more-fine-grained constraints here!

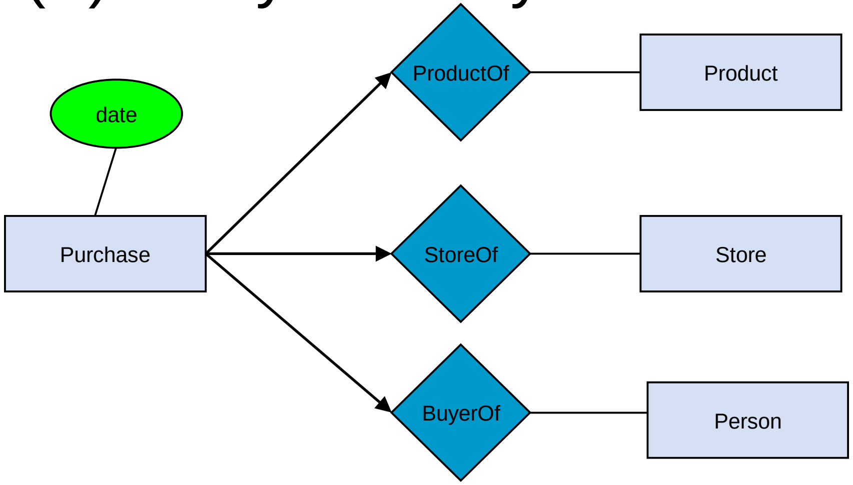
- (B) is also useful when we want to add details (constraints or attributes) to the relationship
  - “A person who shops in only one store”
  - “How long a person has been shopping at a store”

Decision: Multi-way or New Entity + Binary?

(A) Multi-way Relationship



(B) Entity + Binary



- (A) is useful when a relationship really is between multiple entities
  - Ex: A three-party legal contract

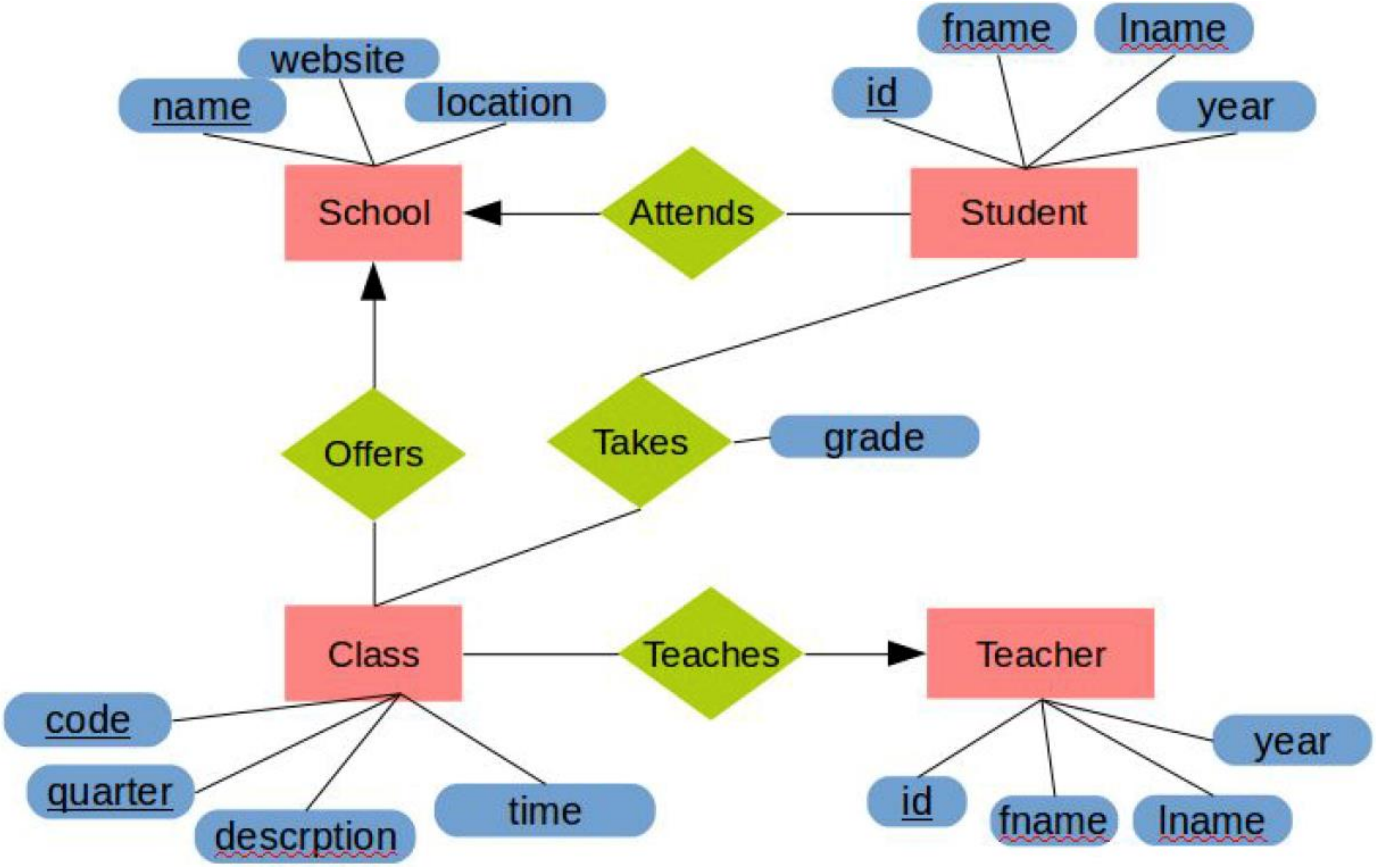
E/R Diagram for School

- Entities
  - A School has a unique name (key), location (address string), and website
  - A Student has attributes id (key), fname , lname , and year
  - A Teacher has a employee id (key), fname , lname , and salary
  - A Class has a class code, quarter, and description, time where a single class may be offered only once each quarter

E/R Diagram for School

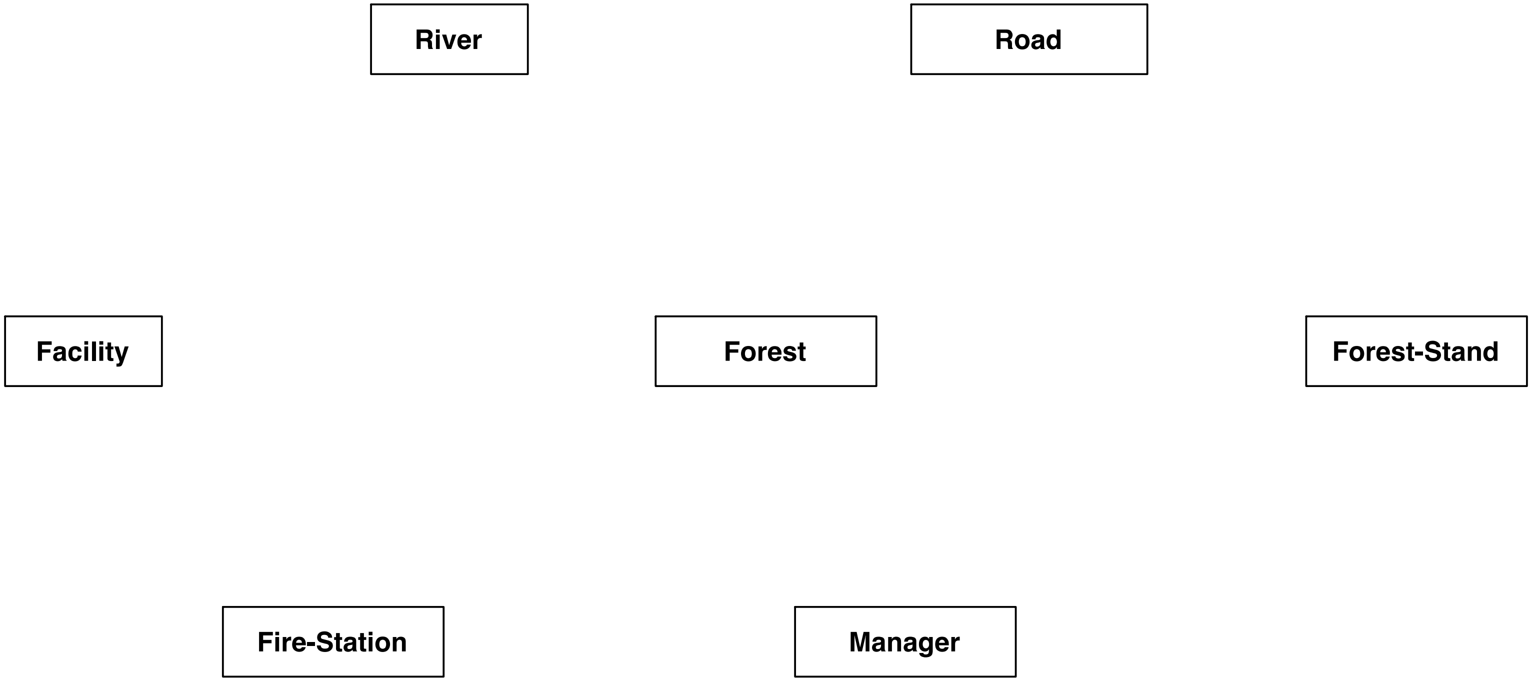
- Entities
  - A School
  - A Student
  - A Teacher
  - A Class
- Relationships
  - A Student Attends only one school but Takes many classes. Each class that a Student takes has a grade associated with it
  - A School offers many Classes but a class can only be offered by one school
  - A Teacher can Teach many classes

E/R Diagram for School



E/R Diagram for “State-Park”

- Entities



River

Road

Facility

Forest

Forest-Stand

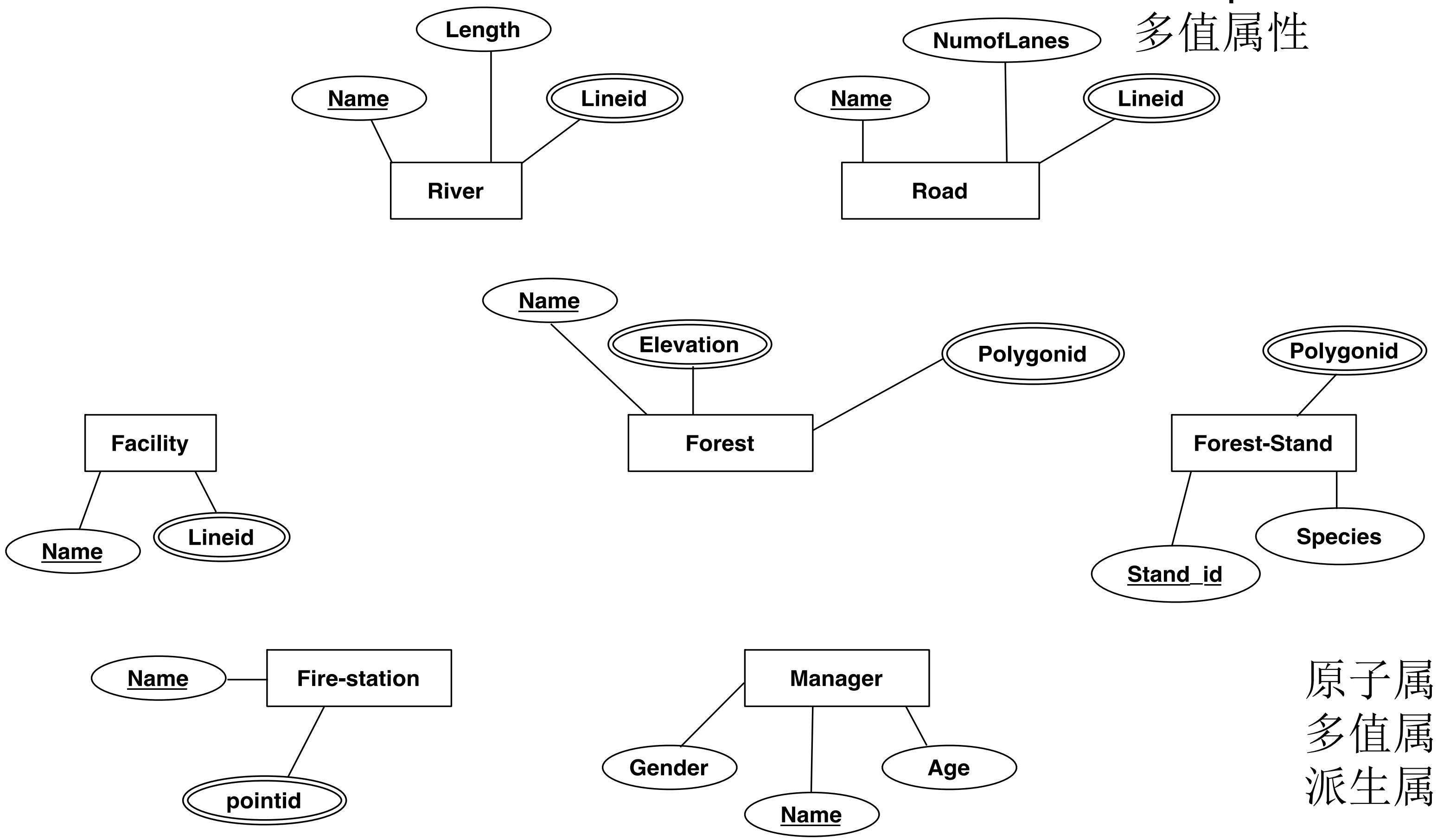
Fire-Station

Manager

E/R Diagram for “State-Park”

- Entities + Attributes

Multiple-value attributes
多值属性

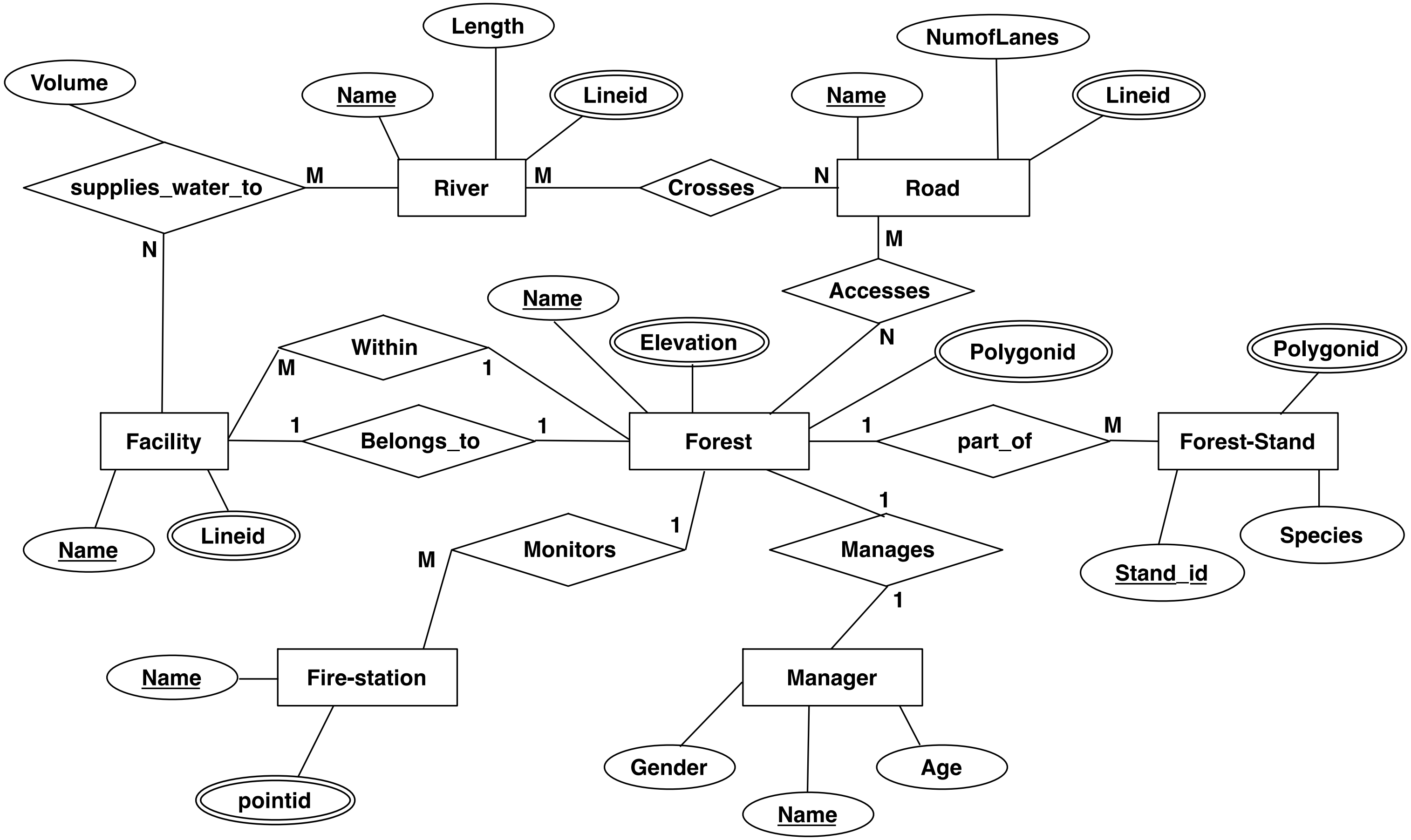


原子属性
多值属性
派生属性

注意几何类型显示编码在属性名中

E/R Diagram for “State-Park”

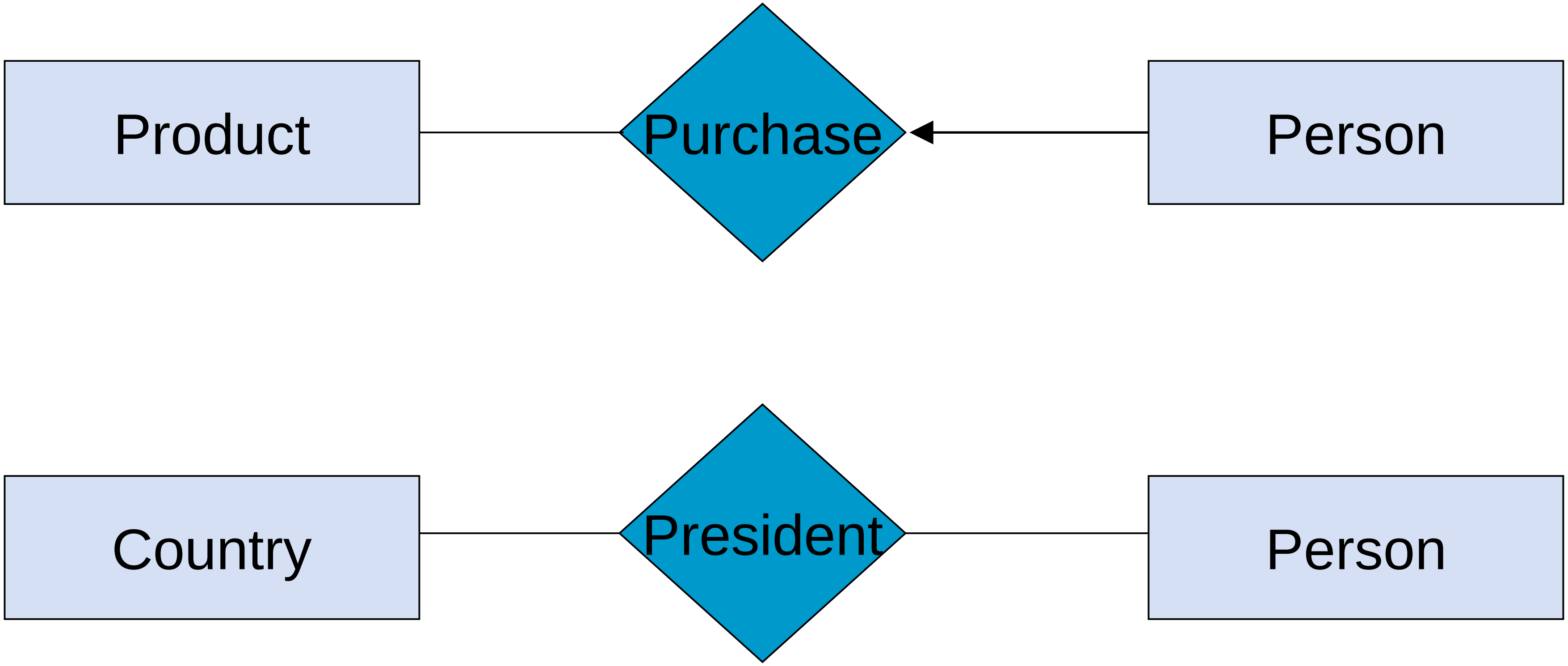
- Entities + Attributes + Relationships



思考：关系是否存在冗余？

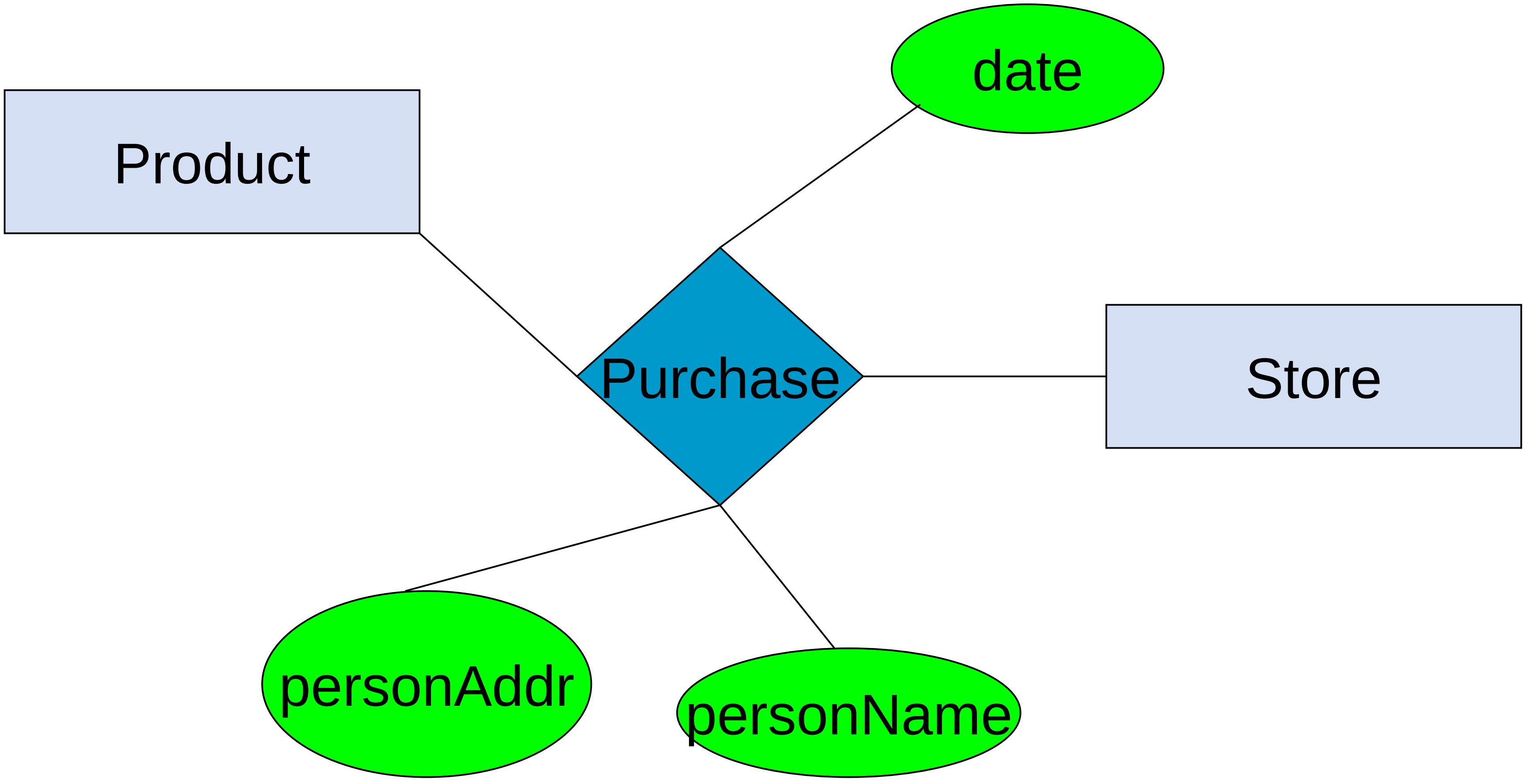
5.3.2 Design Principles

- What's wrong with these examples?



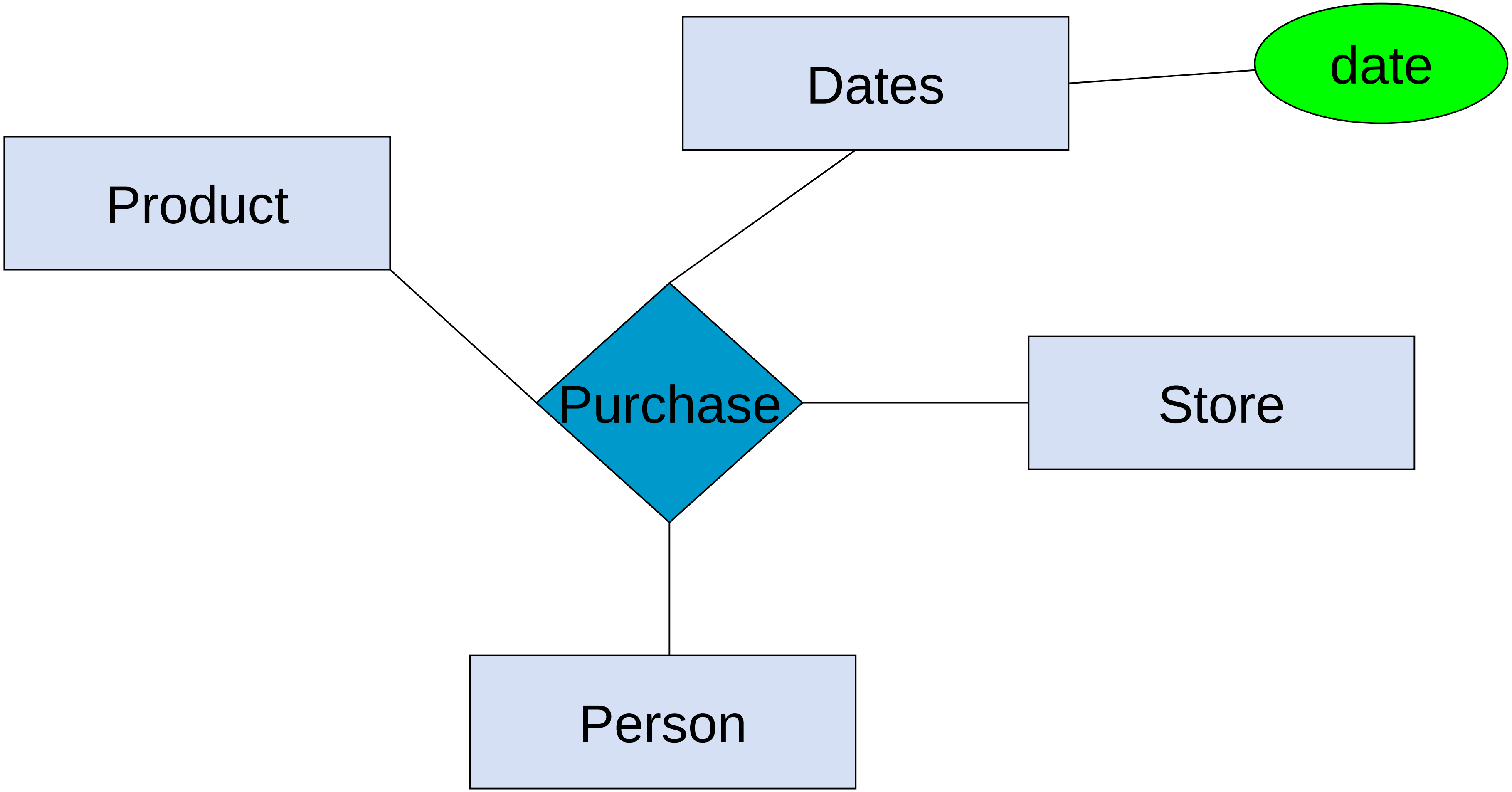
Design Principles

- What’s wrong with these examples?



Design Principles

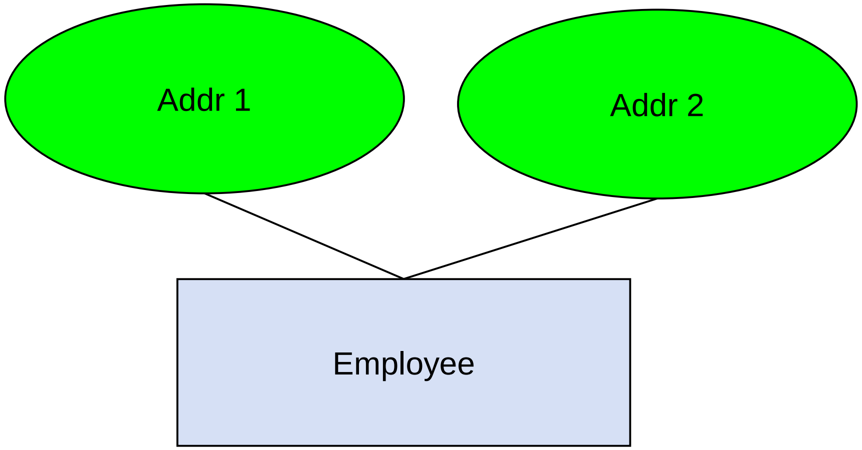
- What’s wrong with these examples?

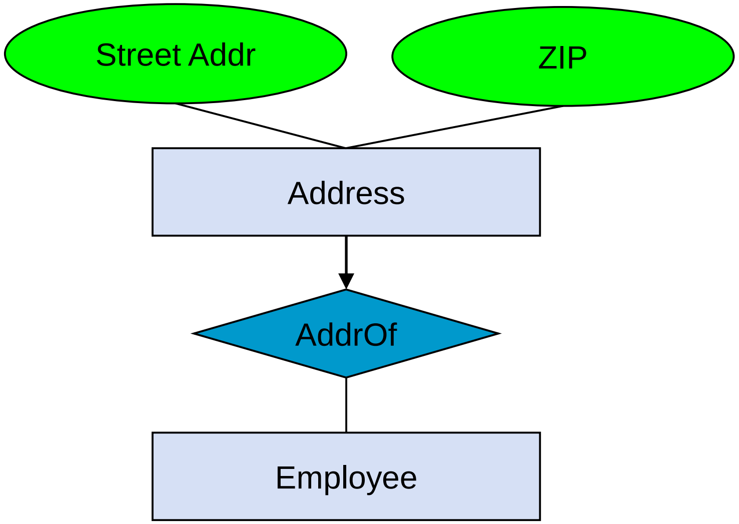


Examples: Entity vs. Attribute

Should address (A) be an attribute?

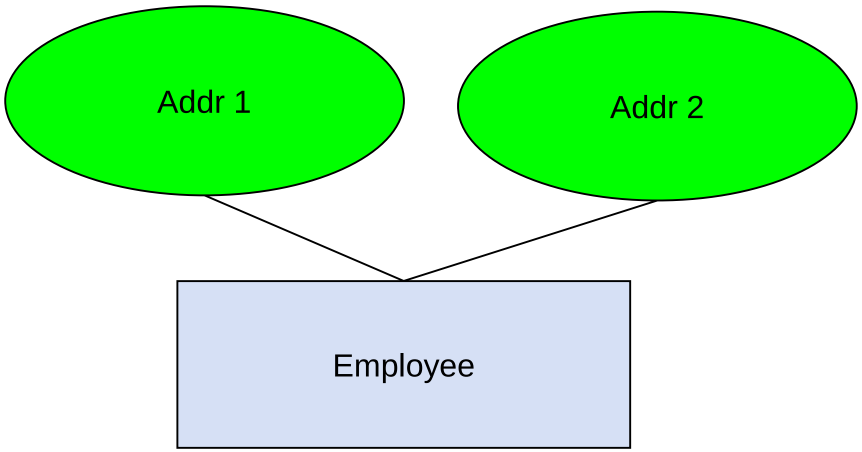
Or (B) be an entity?





Examples: Entity vs. Attribute

Should address (A) be an attribute?

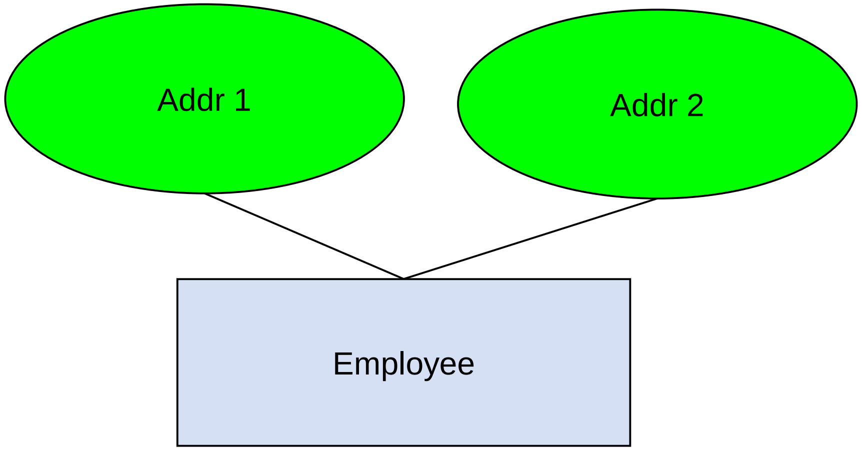


How do we handle employees with multiple addresses here?

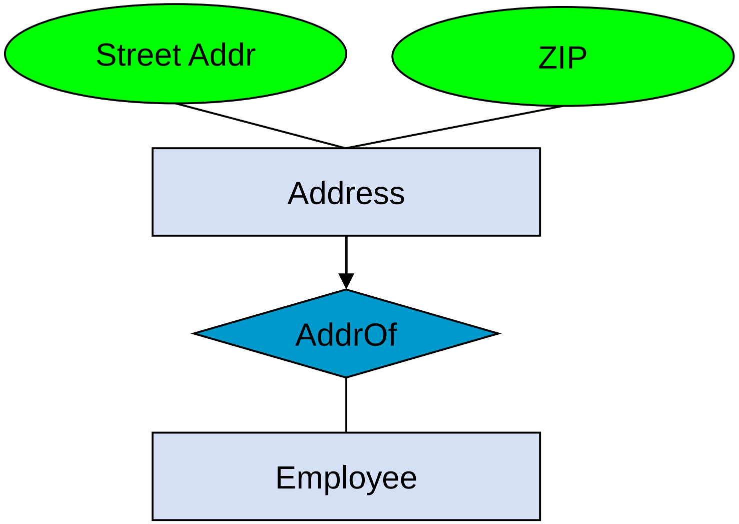
How do we handle addresses where internal structure of the address (e.g. zip code, state) is useful?

Examples: Entity vs. Attribute

Should address (A) be an attribute?



Or (B) be an entity?



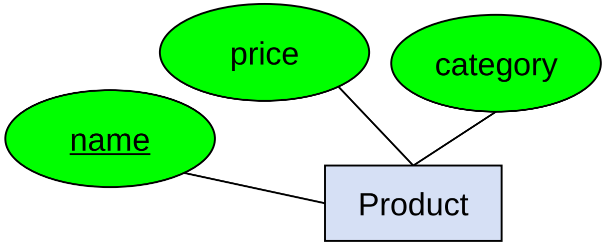
In general, when we want to record several values, we choose new entity

Design Consideration Summary

- Entity vs. Attribute
  - Multi-valued attributes
- Entity vs. Relationship
  - Multiple instances of each entity combination
- Multi-way vs. New Entity + Binary relationships
- Aggreation?

5.3.3 From E/R Diagrams to Relational Schema

- Key concept: Both Entity sets and Relationships become relations (tables in RDBMS)
- An entity set becomes a relation (multiset of tuples / table)
  - Each tuple is one entity
  - Each tuple is composed of the entity's attributes, and has the same primary key



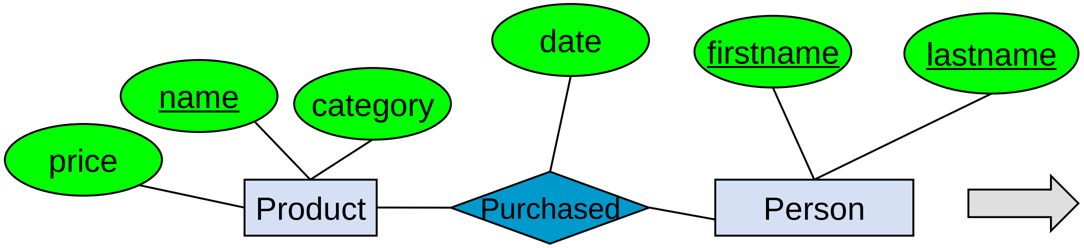
Product

name	**price**	**category**
Gizmo1	99.99	Camera
Gizmo2	19.99	Edible

```
CREATE TABLE Product (
  name     CHAR(50) PRIMARY KEY,
  price     DOUBLE,
  category VARCHAR(30))
```

From E/R Diagrams to Relational Schema

- A relation between entity sets A_1, …, A_N also becomes a multiset of tuples / a table
  - Each row/tuple is one relation, i.e. one unique combination of entities $(a_1,…,a_N)$
  - Each row/tuple
    - is composed of the union of the entity sets’ keys
    - has the entities’ primary keys as foreign keys
    - has the union of the entity sets’ keys as primary key

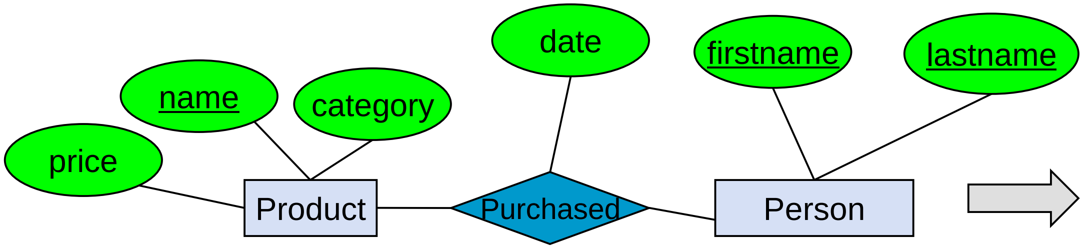


Purchased

name	firstname	lastname	date
Gizmo1	Bob	Joe	01/01/15
Gizmo2	Joe	Bob	01/03/15
Gizmo1	JoeBob	Smith	01/05/15

From E/R Diagrams to Relational Schema

- A relation between entity sets A_1, …, A_N also becomes a multiset of tuples / a table



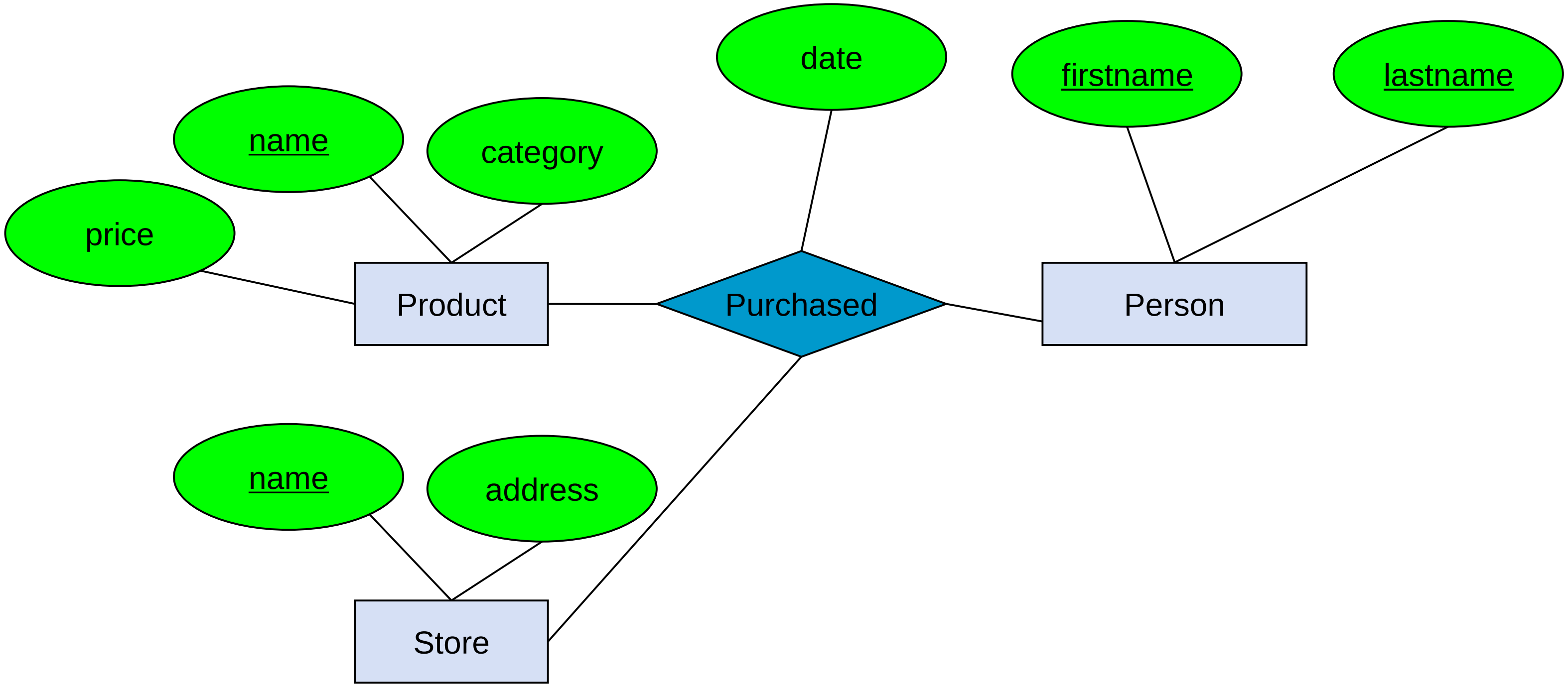
Purchased

name	firstname	lastname	date
Gizmo1	Bob	Joe	01/01/15
Gizmo2	Joe	Bob	01/03/15
Gizmo1	JoeBob	Smith	01/05/15

```
CREATE TABLE Purchased (
  name      CHAR(50),
  firstname CHAR(50),
  lastname  CHAR(50),
  date      DATE,
  PRIMARY KEY (name, firstname, lastname),
  FOREIGN KEY (name) REFERENCES Product,
  FOREIGN KEY (firstname, lastname) REFERENCES Person
)
```

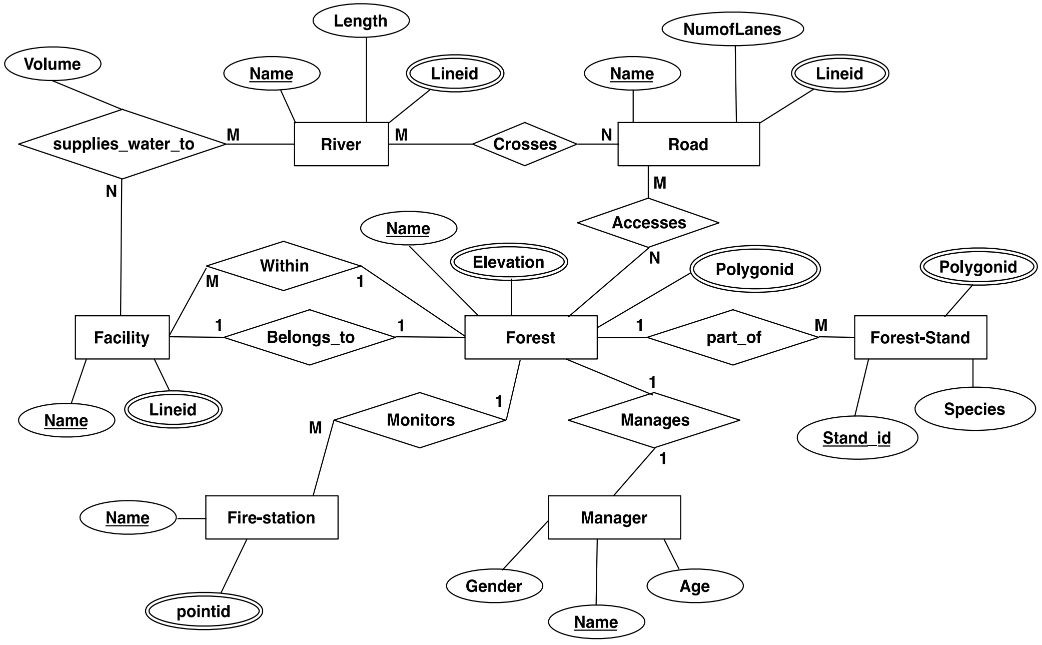
From E/R Diagram to Relational Schema

- How do we represent this as a relational schema?



From E/R Diagrams to Relational Schema

- 非扩展几何类型实现
  - 几何属性实际上作为关系的属性



Forest-Stand

Stand-id	Species	Forest-name
(Integer)	(Varchar)	(Varchar)

River

Name	Length
(Varchar)	(Real)

Road

Name	NumofLanes
(Varchar)	(Integer)

Facility

Name	Forest-name	Forest-name-2
(Varchar)	(Varchar)	(Varchar)

Forest

Name
(Varchar)

Fire-station

Name	ForName
(Varchar)	(varchar)

Supplies_Water_To

FacName	RivName	Volume
(Varchar)	(Varchar)	(Real)

Manager

Name	Age	Gender	ForName
(Varchar)	(Integer)	(Varchar)	(Varchar)

Fstand-Geom

Stand-id	Polygonid
(Integer)	(Integer)

River-Geom

Name	Lineid
(Integer)	(Integer)

Road-Geom

Rname	Lineid
(Varchar)	(Integer)

Facility-Geom

Name	Pointid
(Varchar)	(Integer)

Forest-Geom

Name	Polygonid
(Varchar)	(Integer)

Fstation-Geom

Name	Pointid
(Varchar)	(Integer)

Road-Access-Forest

RoadName	ForName
(Varchar)	(Varchar)

From E/R Diagrams to Relational Schema

- Highlights of translation rules
  - Entity becomes Relation
  - Attributes become columns in the relation
  - Multi-valued attributes become a new relation
    - Includes foreign key to link to relation for the entity
  - Relationships (1:1, 1:N) become foreign keys
    - 注意适用条件
  - M:N Relationships become a relation
    - Containing foreign keys or relations from participating entities

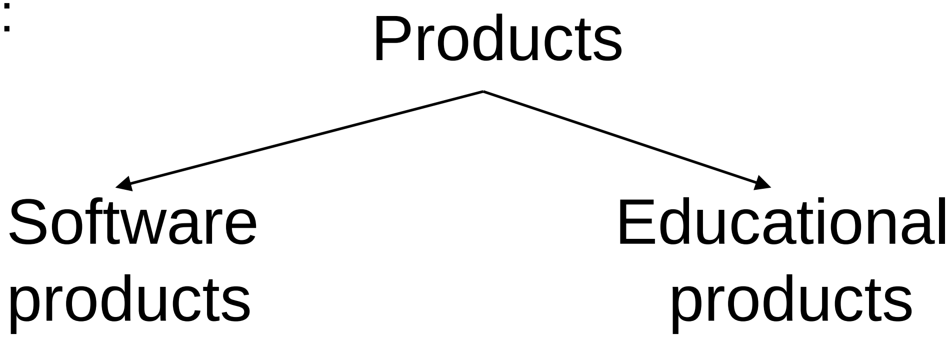
第五章 空间扩展E/R图

- 5.1 Database Design
- 5.2 E/R Basic: Entities & Relations
- 5.3 E/R Design Considerations
  - 5.3.1 Relations: multiplicity, multi-way
  - 5.3.2 Design considerations
  - 5.3.3 Conversion to SQL
- 5.4 Advanced E/R Concepts
  - 5.4.1 Subclasses & connection to OO
  - 5.4.2 Constraints
  - 5.4.3 Weak entity sets
- 5.5 Spatial Database Design Example

参考教材：
数据库系统概念 7.1-7.10
空间数据库管理系统概论 5.1-5.4

5.4.1 Modeling Subclasses

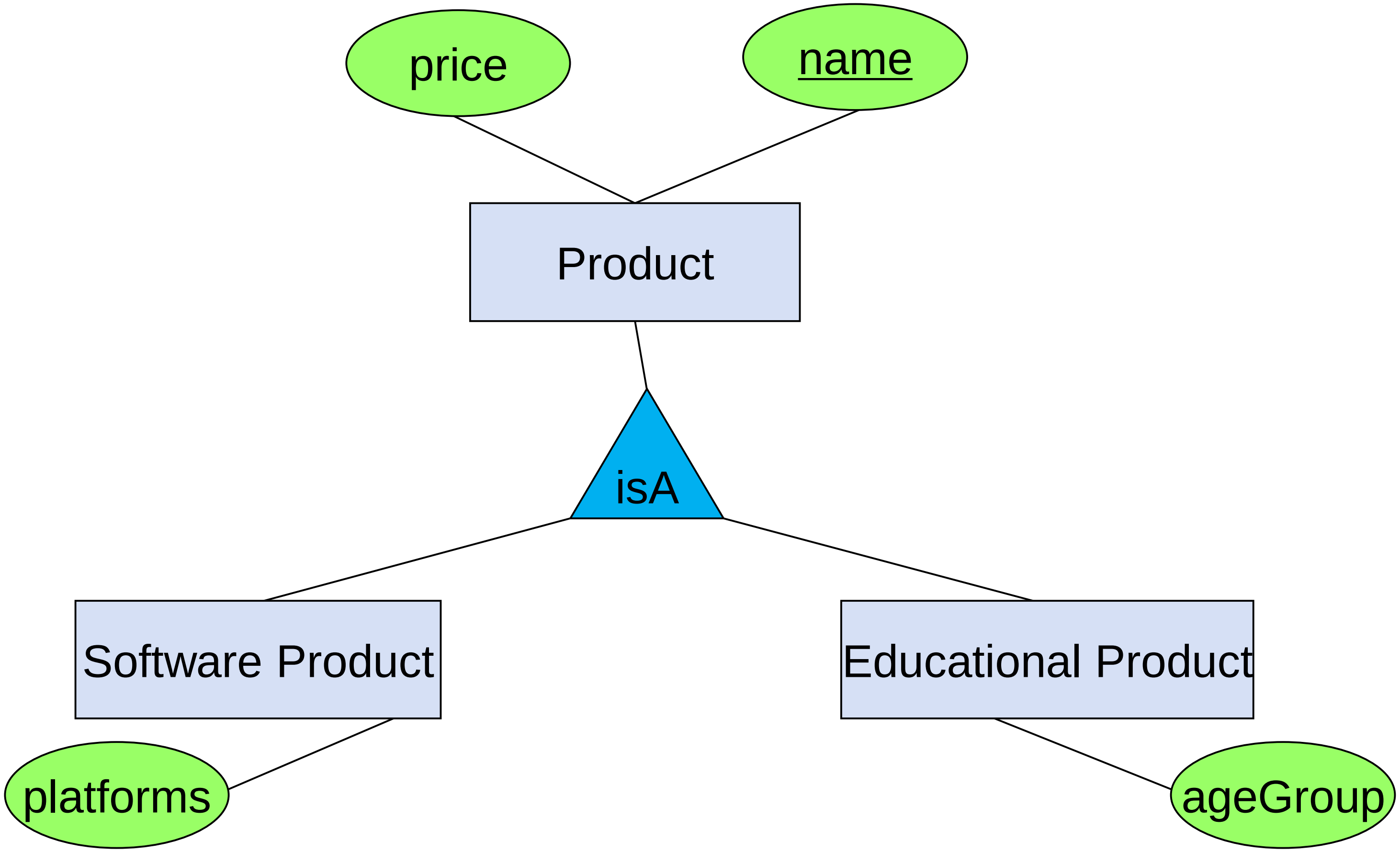
- Some objects in a class may be special, i.e. worthy of their own class
  - Define a new class?
    - But what if we want to maintain connection to current class?
  - Better: define a subclass
    - Ex:



We can define subclasses in E/R!

Modeling Subclasses

- Child subclasses contain all the attributes of all of their parent classes plus the new attributes shown attached to them in the E/R diagram



Understanding Subclasses

- Think in terms of records; ex:
  - Product

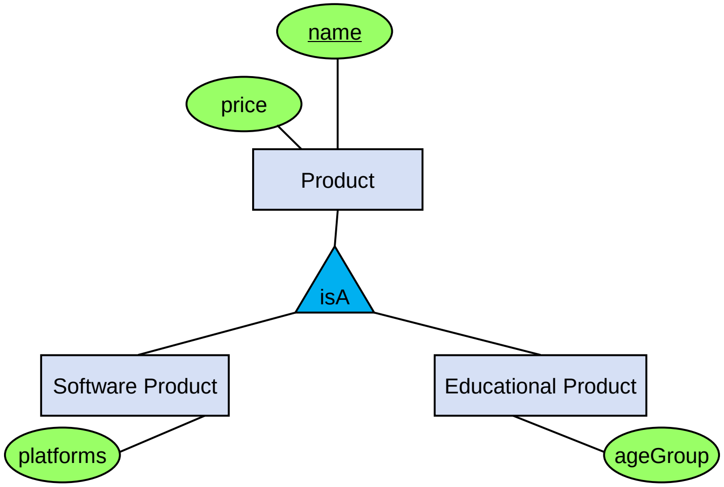
    name
    price

  - SoftwareProduct

    name
    price
    platforms

  - EducationalProduct

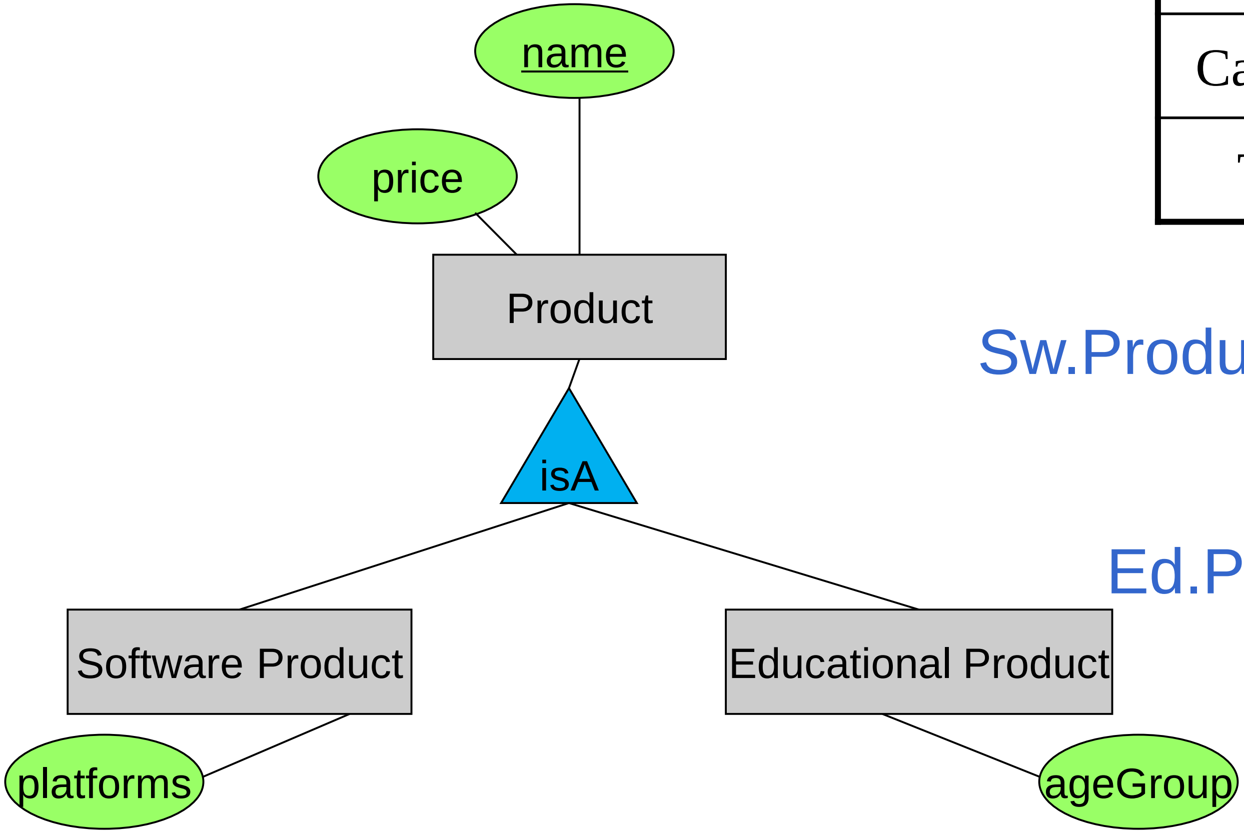
    name
    price
    ageGroup



Child subclasses contain all the attributes of all of their parent classes plus the new attributes shown attached to them in the E/R diagram

Understanding Subclasses

- Think like tables



Product

name	price	category
Gizmo	99	gadget
Camera	49	photo
Toy	39	gadget

Sw.Product

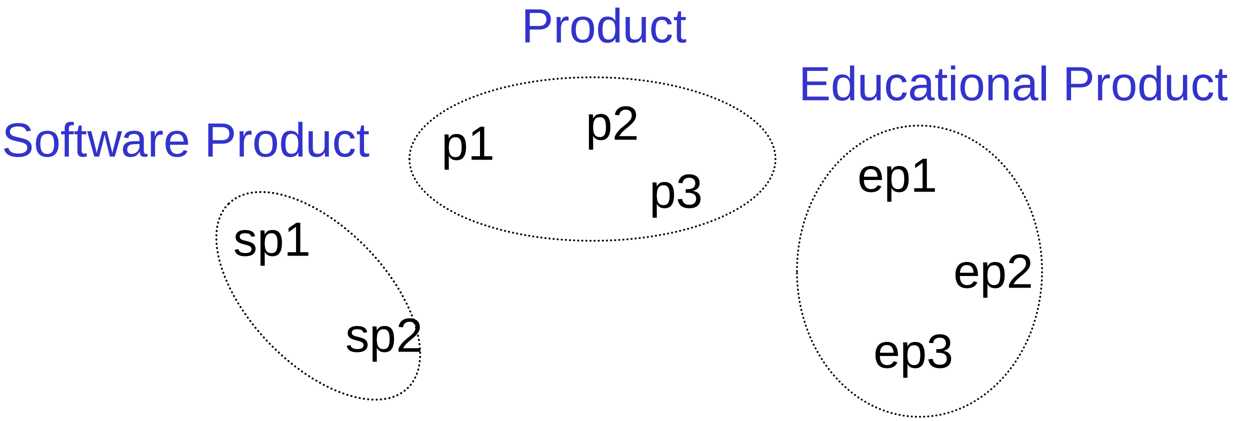
name	platforms
Gizmo	unix

Ed.Product

name	ageGroup
Gizmo	todler
Toy	retired

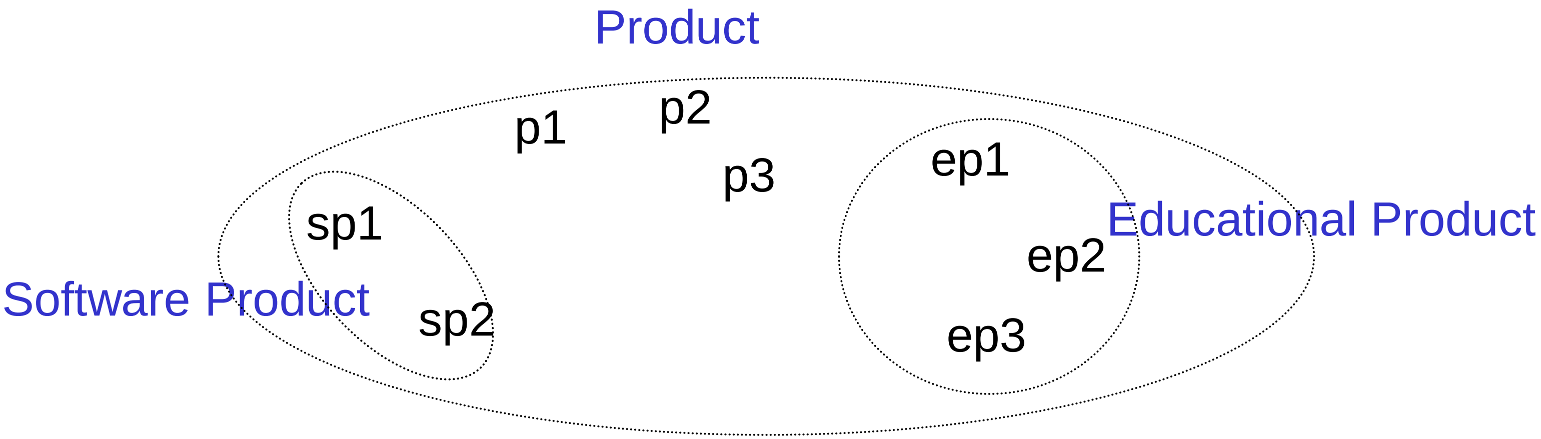
Difference between OO and E/R Inheritance

- OO: Classes are disjoint (same for Java, C++)



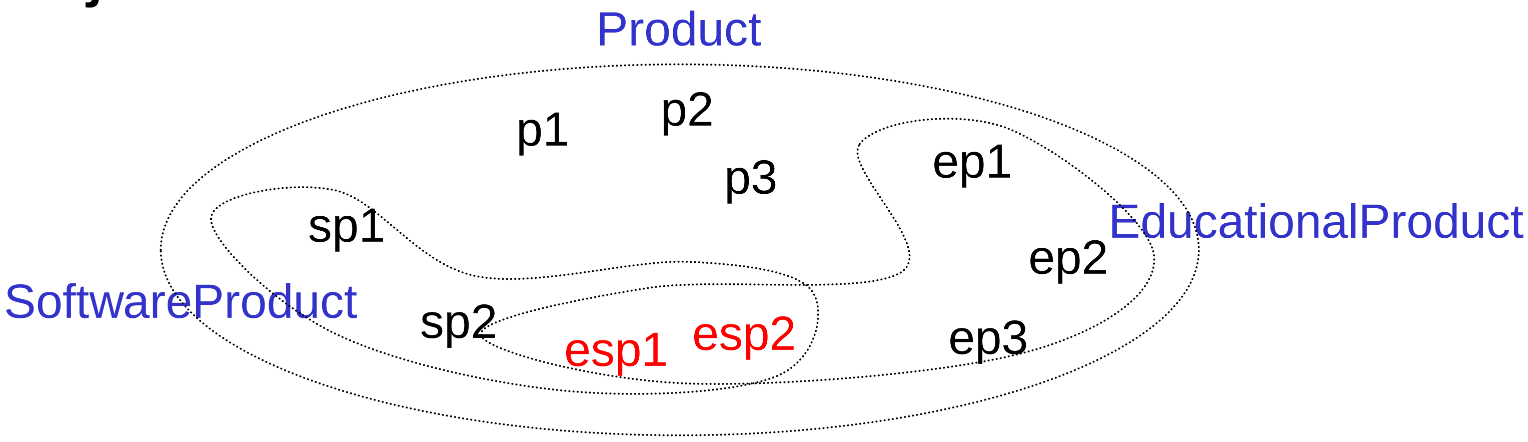
OO = **Object Oriented**. E.g. classes as fundamental building block, etc…

- E/R: Entity sets overlap



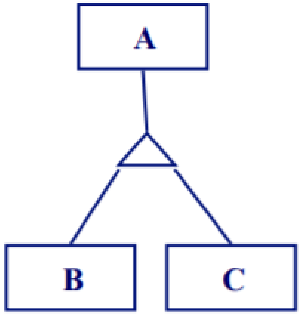
Difference between OO and E/R Inheritance

- We have three entity sets, but four different kinds of objects

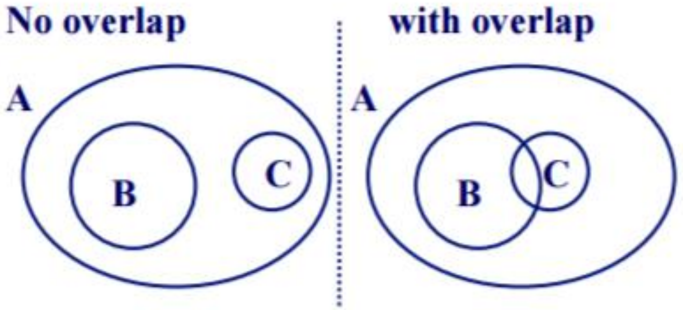


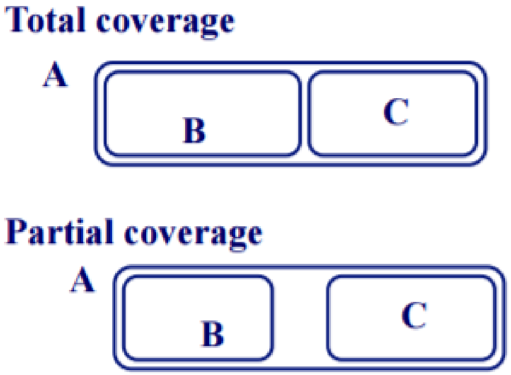
No need for multiple inheritance in E/R

IsA Review



- If we declare A IsA B then every A is a B
  - 分类属性
  - 不相交约束
  - 完备性约束





- We use IsA to
  - Add descriptive attributes to a subclass
  - To identify entities that participate in a relationship

- No need for multiple inheritance

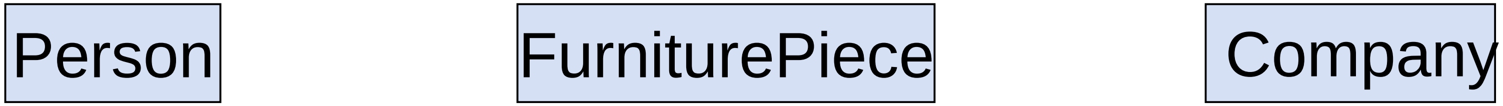
思考：几何对象模型的geometry类设计是否满足不相交约束和完备性约束？

IsA to Relation Schema

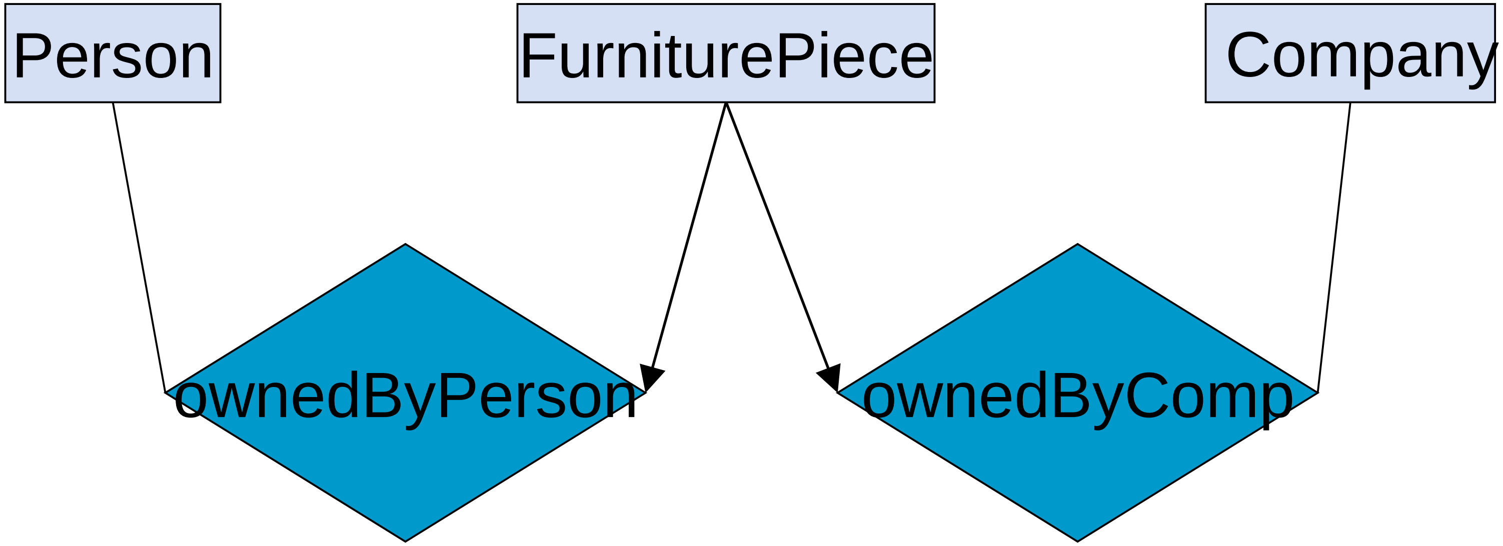
- IsA联系描述了实体型之间的继承关系。在关系模型中仍然使用关系表示IsA联系。一般情况下，父实体和各子实体分别用独自的关系表示，表示父实体的关系属性包括所有父实体的属性，子实体对应的关系除了包含各自的属性外，还必须包含父实体的主键
- 如果IsA联系满足不相交约束，也可以用一个关系表示父实体和所有的子实体
- 如果IsA联系满足完备性约束，也可以去除表示父实体的关系，但是父实体的所有属性在每个子实体的关系中都必须出现

Modeling Union Types with Subclasses

- Suppose each piece of furniture is owned either by a person, or by a company. How do we represent this?

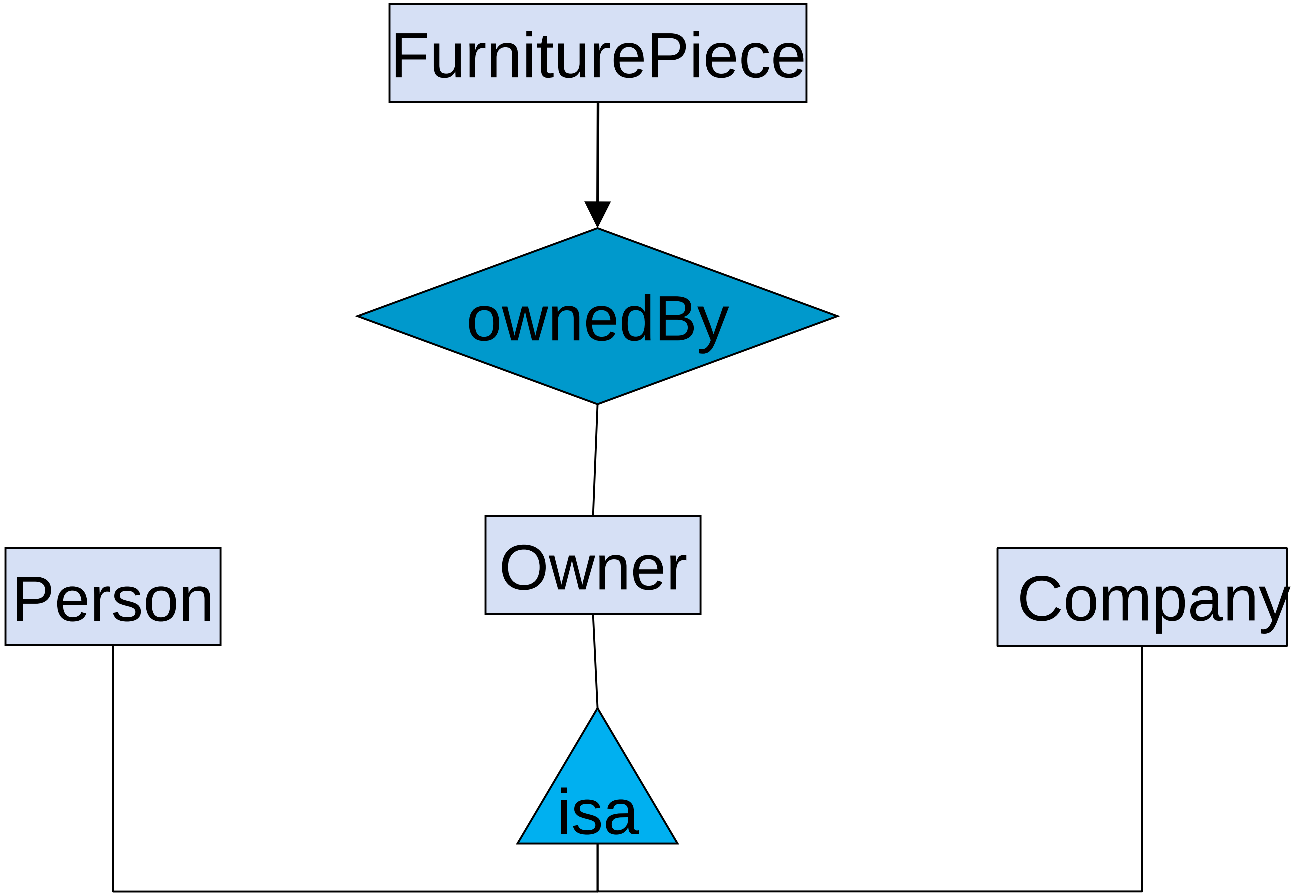


- Solution 1. Acceptable, but imperfect (What’s wrong ?)



Modeling Union Types with Subclasses

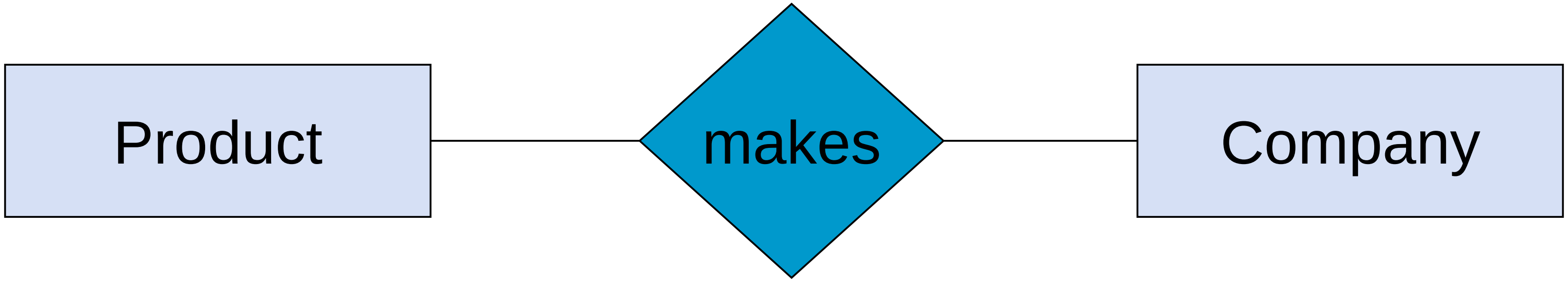
- Suppose each piece of furniture is owned either by a person, or by a company
- Solution 2: better (though more laborious)



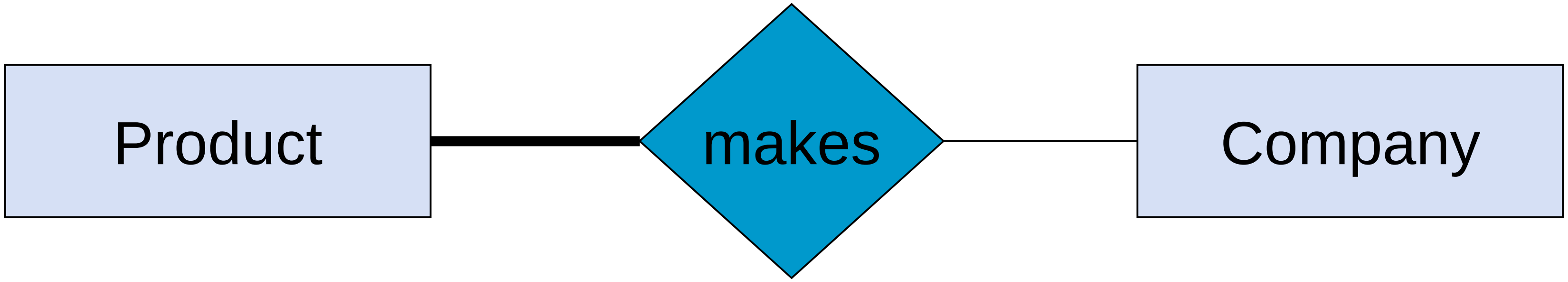
5.4.2 Constraints in E/R Diagrams

- Finding constraints is part of the E/R modeling process. Commonly used constraints are
  - Keys: Implicit constraints on uniqueness of entities
    - Ex: An SSN uniquely identifies a person
  - Single-value constraints
    - Ex: a person can have only one father
  - Referential integrity constraints: Referenced entities must exist
    - Ex: if you work for a company, it must exist in the database
  - Spatial concept constraints - Pictograms
    - Ex: the geometry type of the river is multilinestring
  - Other constraints:
    - Ex: peoples' ages are between 0 and 150

Participation Constraints: Partial v. Total



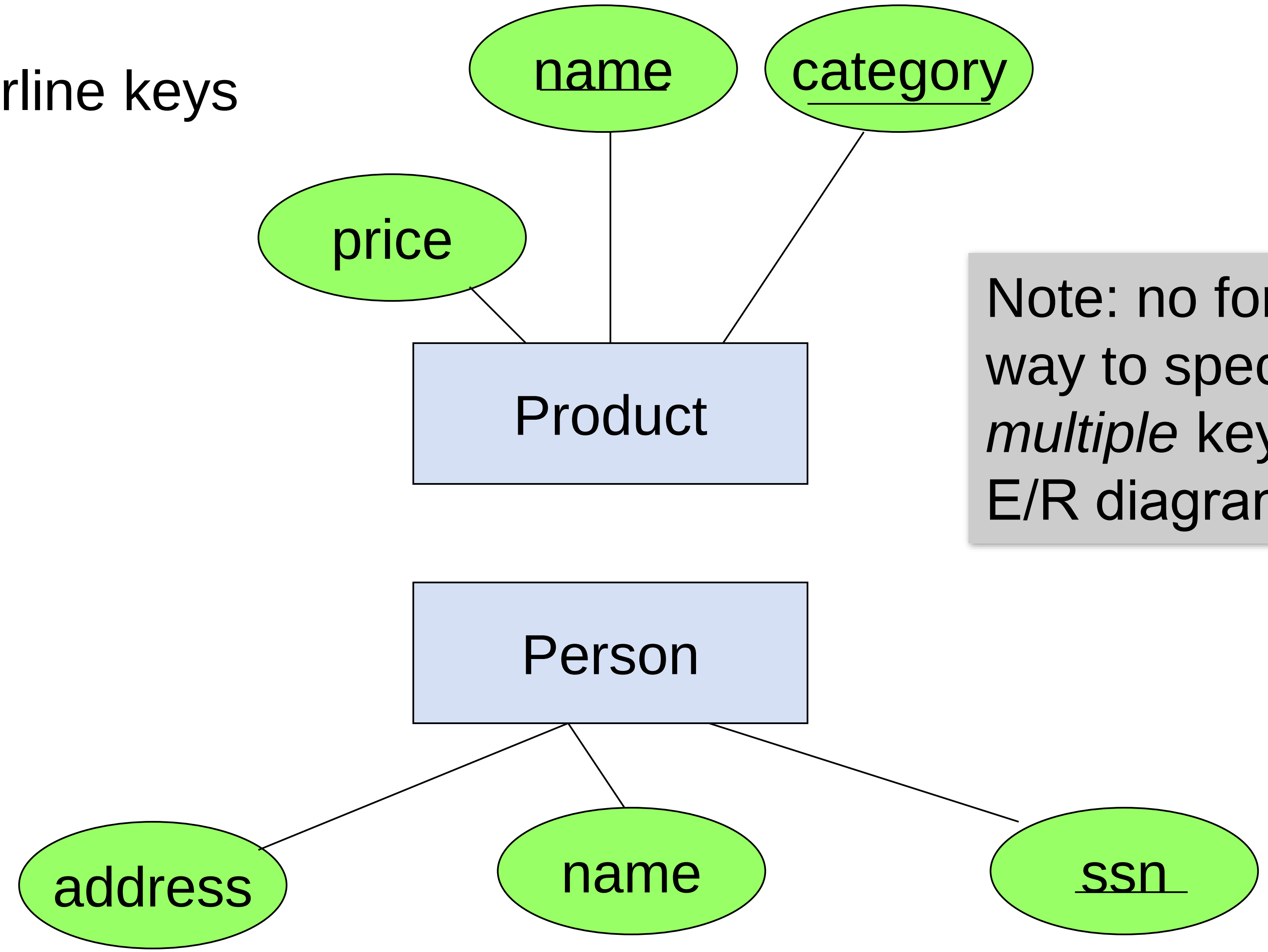
Are there products made by no company?
Companies that don't make a product?



Bold line indicates total participation (i.e. here: all products are made by a company)

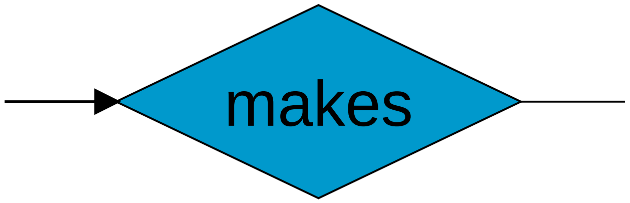
Keys in E/R Diagrams

Underline keys

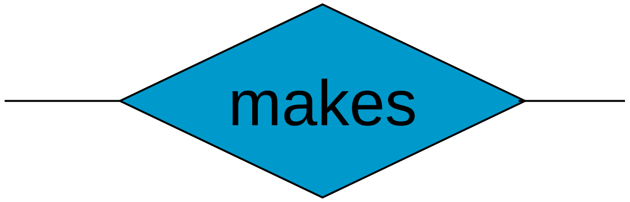


Note: no formal way to specify *multiple* keys in E/R diagrams…

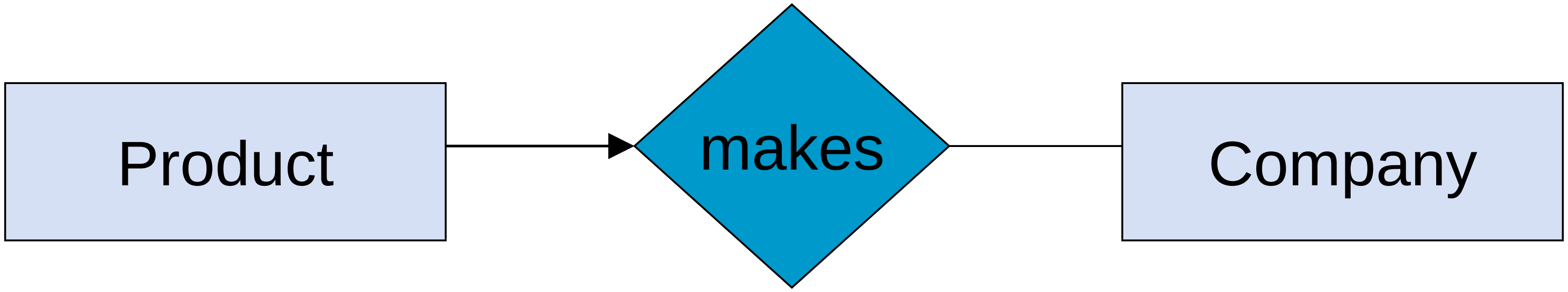
Single Value Constraints



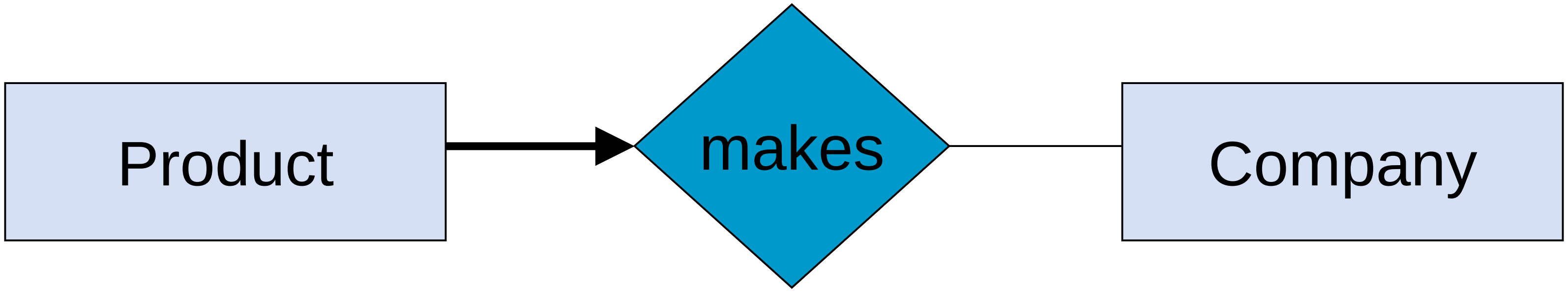
v. s.



Referential Integrity Constraints



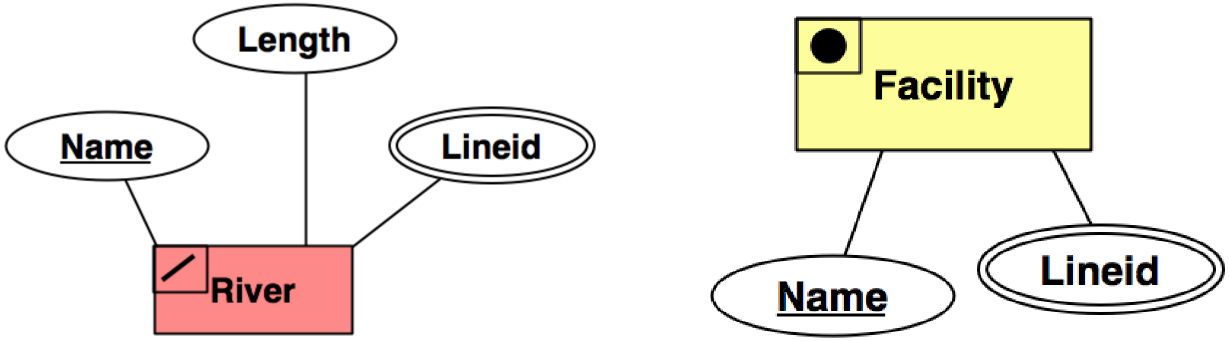
Each product made by at most one company.
Some products made by no company?



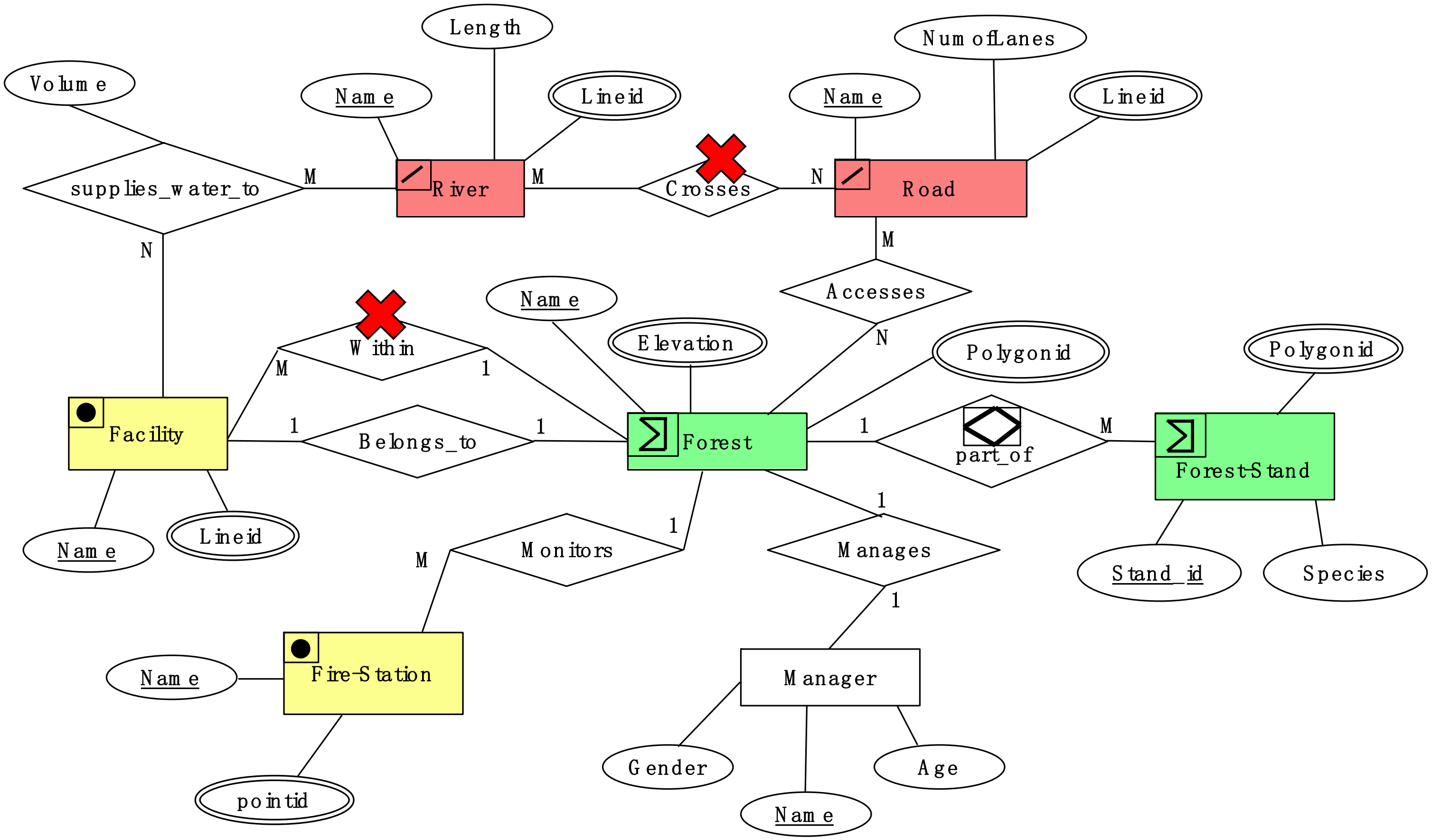
Each product made by exactly one company

Spatial Concept Constraints

- Pictograms
  - Label spatial entities along with their spatial data types
  - Allows inference of spatial relationships and constraints

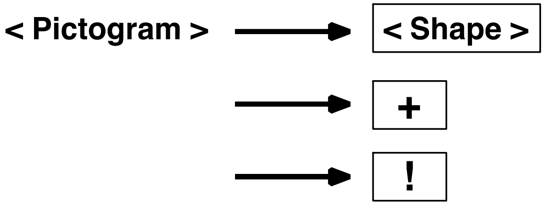


Spatial Concept Constraints

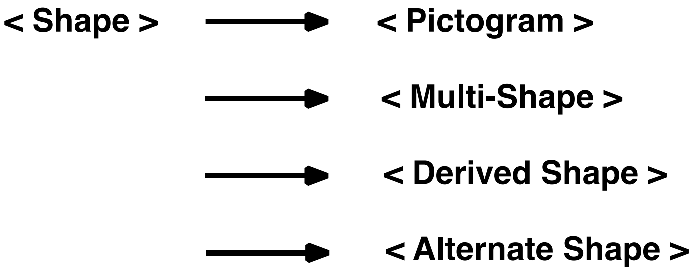


Pictograms

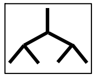
- Grammar based approach
  - Rewrite rule
  - Like English syntax diagrams
- Classes of pictograms
  - Entity pictograms
    - Basic: point, line, polygon
    - Collection of basic
    - ...
  - Relationship pictograms
    - Partition, network



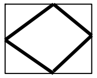
Grammar (for Pictogram)



Grammar (for Shape)



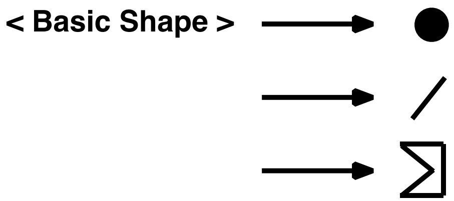
Part_of (Network)



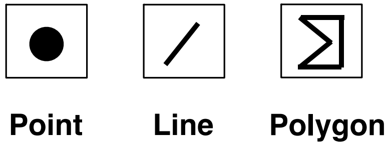
Part_of (Partition)

Pictograms for Relationship

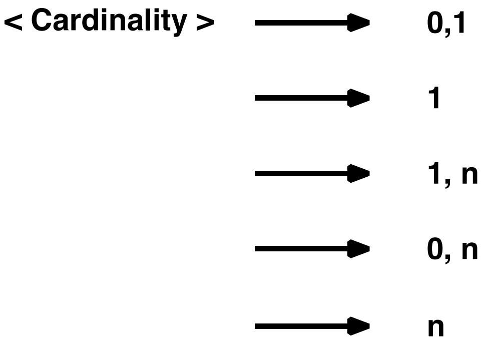
Entity Pictograms: Basic shapes, Collections



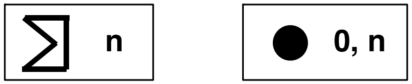
Grammar (for Basic Shape)



Pictograms for Basic Shapes



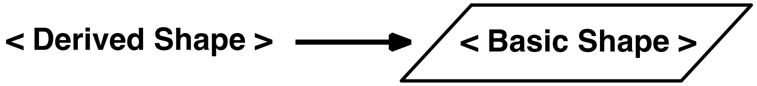
Grammar (for Cardinality)



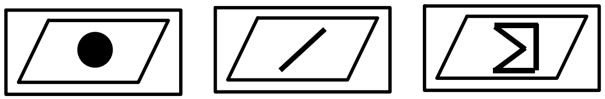
Pictograms Multishapes
(using cardinality)

Entity Pictograms: Derived and Alternate Shapes

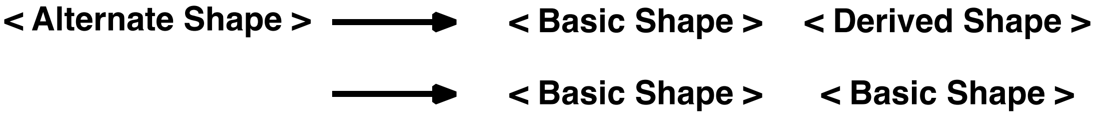
- Derived shape example is city center point from boundary polygon
- Alternate shape example: A road is represented as a polygon for construction
  - or as a line for navigation



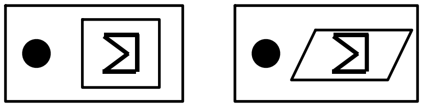
Grammar (for Derived Shape)



Pictograms for Derived Shapes



Grammar (for Alternate Shape)



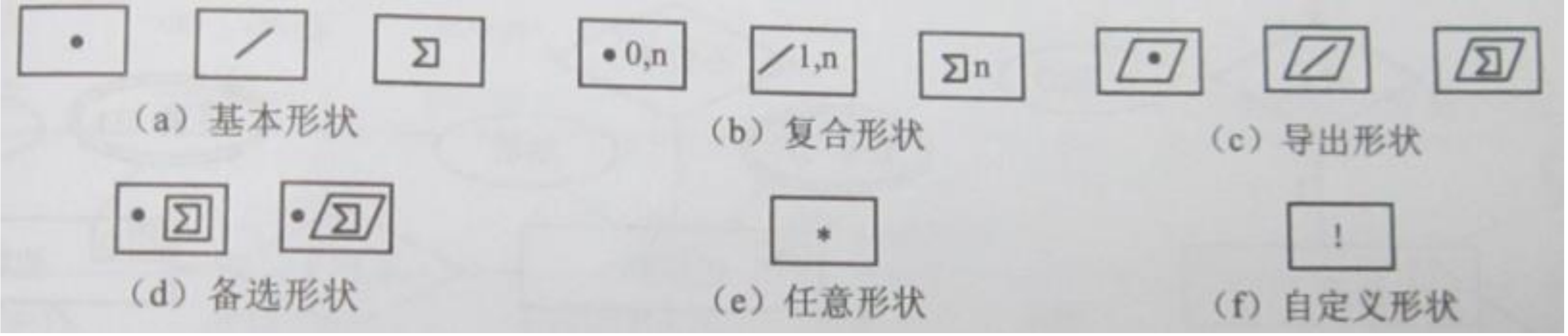
Pictograms for Alternate Shapes

空间扩展

- 具有空间含义的象形图(pictogram)来注释和扩展E/R图
- 实体象形图
  - 实体的方框中插入的一种用于描述空间实体的几何特征的微缩图
- 联系象形图
  - 联系的菱形框中插入的用于描述实体间空间联系特征的微缩图

空间扩展

- 实体象形图
  - 基本形状：包括点、线、面。在一般应用中大多数空间实体可以用这些基本形状表示
  - 复合形状：定义了一组聚合形状，并用基数来量化这些复合形状，用于表示那些不能用某个形状表示的对象。其中复合形状的基数可以有“0,1”、“1”、“1,n”、“0,n”、“n”多种形式
  - 导出形状：若一个形状是由其他形状导出的，那么就用斜体形式来表示。例如可以从美国的州界形状中导出每个的形状

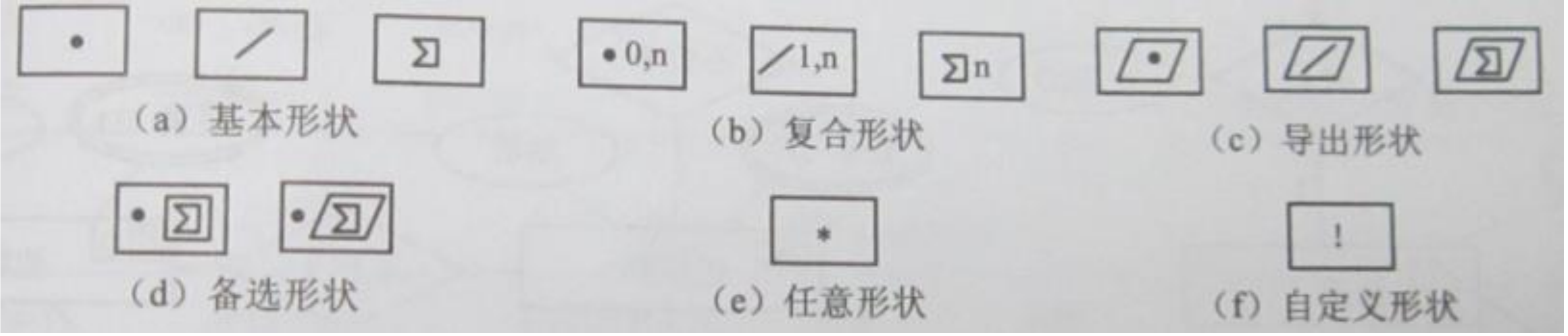


（a）基本形状 （b）复合形状 （c）导出形状

（d）备选形状 （e）任意形状 （f）自定义形状

空间扩展

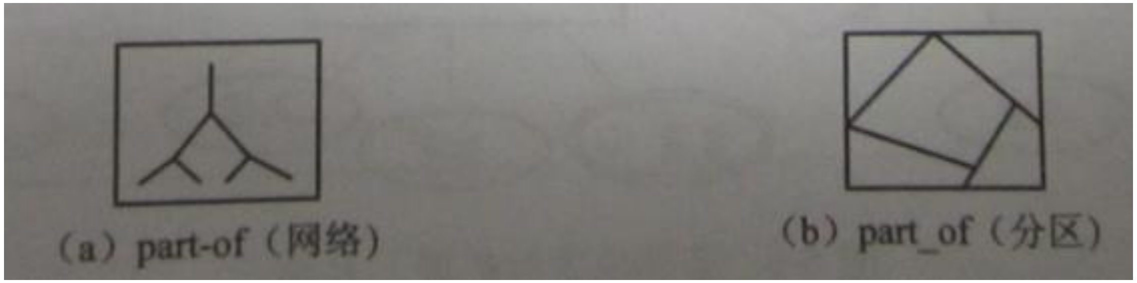
- 实体象形图
  - 备选形状：用于表示某种条件下的同一对象。备选形状的元素可以属于其他的几类形状。例如不同比例尺下，一座房屋可以表示成一个多边形，也可以表示成一个点
  - 任意形状：对于形状的组合，可以用通配符(*)表示，它表示各种形状，例如一个灌溉网是由泵站(点)，水渠(线)以及水库(多边形)所组成
  - 自定义形状：除了点线和多边形这些基本形状外，用户还可以自定义形状。例如用感叹号之类的象形图来表示灌溉图



（a）基本形状
（b）复合形状
（c）导出形状
（d）备选形状
（e）任意形状
（f）自定义形状

空间扩展

- 联系象形图
  - part-of (网络)
  - part_of (分区)
- 例如，道路与道路网之间的联系的part-of微缩图和用于描述把土地划分为地块的part_of微缩图



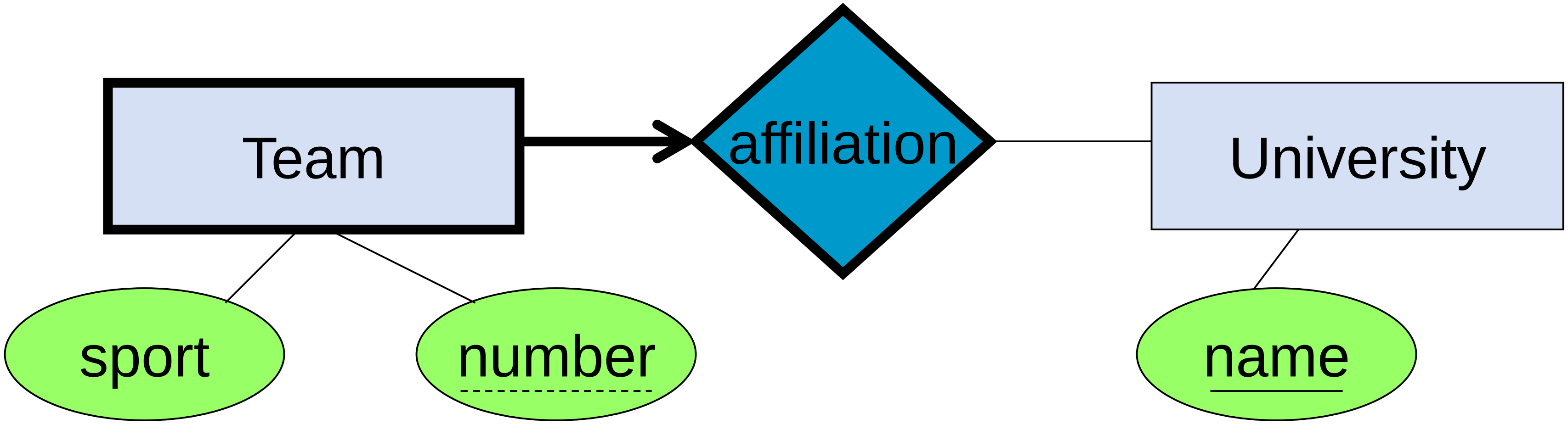
（a）part-of（网络）

（b）part_of（分区）

5.4.3 Weak Entity Sets

- Entity sets are weak when their key comes from other classes to which they are related

“Football team” v. “***The Stanford*** Football team” *(E.g., Berkeley has a football team too, sort of)*



- Number is a partial key. (denote with dashed underline)
- University is called the identifying owner
- Participation in affiliation must be total. Why?

第五章 空间扩展E/R图

- 5.1 Database Design
- 5.2 E/R Basic: Entities & Relations
- 5.3 E/R Design Considerations
  - 5.3.1 Relations: multiplicity, multi-way
  - 5.3.2 Design considerations
  - 5.3.3 Conversion to SQL
- 5.4 Advanced E/R Concepts
  - 5.4.1 Subclasses & connection to OO
  - 5.4.2 Constraints
  - 5.4.3 Weak entity sets
- 5.5 Spatial Database Design Example

参考教材：
数据库系统概念 7.1-7.10
空间数据库管理系统概论 5.1-5.4

E/R Summary

- E/R diagrams are a visual syntax that allows technical and non-technical people to talk
  - For conceptual design
- Basic constructs: entity, relationship, and attributes
- A good design is faithful to the constraints of the application, but not overzealous

从E/R图到关系数据库模式

- 实体 → 关系
  - 主码 → 主码
  - 属性 → 属性
  - 复合属性 → 若干原子属性
  - 多值属性 → 关系 或 转化为联系
  - 实体象形图 → Geometry类型
- 联系 (如何确定主码？)
  - 1:1 → 关系 或 与非强制性的实体合并
  - 1:n → 关系 或 与n端的实体合并
  - m:n → 关系
  - IsA联系 → 三种方法
  - part-of联系 → ???

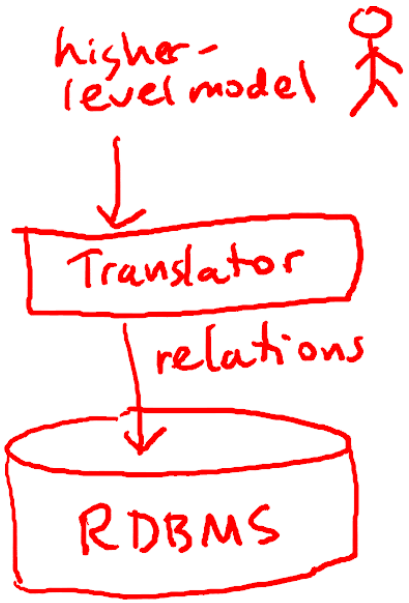
思考：为什么要合并？

数据库设计

- 数据流图→ E/R图/UML图
  - 参照数据流图，标定各局部应用中的实体、实体的属性、标识实体的码，确定实体之间的联系及其类型 (1:1，1:n，m:n)
- E/R图/UML图 → 关系数据库模型
  - E/R图包括实体、实体的属性和实体之间的联系
  - 关系模型的逻辑结构是一组关系模式的集合
  - 什么是设计合理或“好”的关系模型？

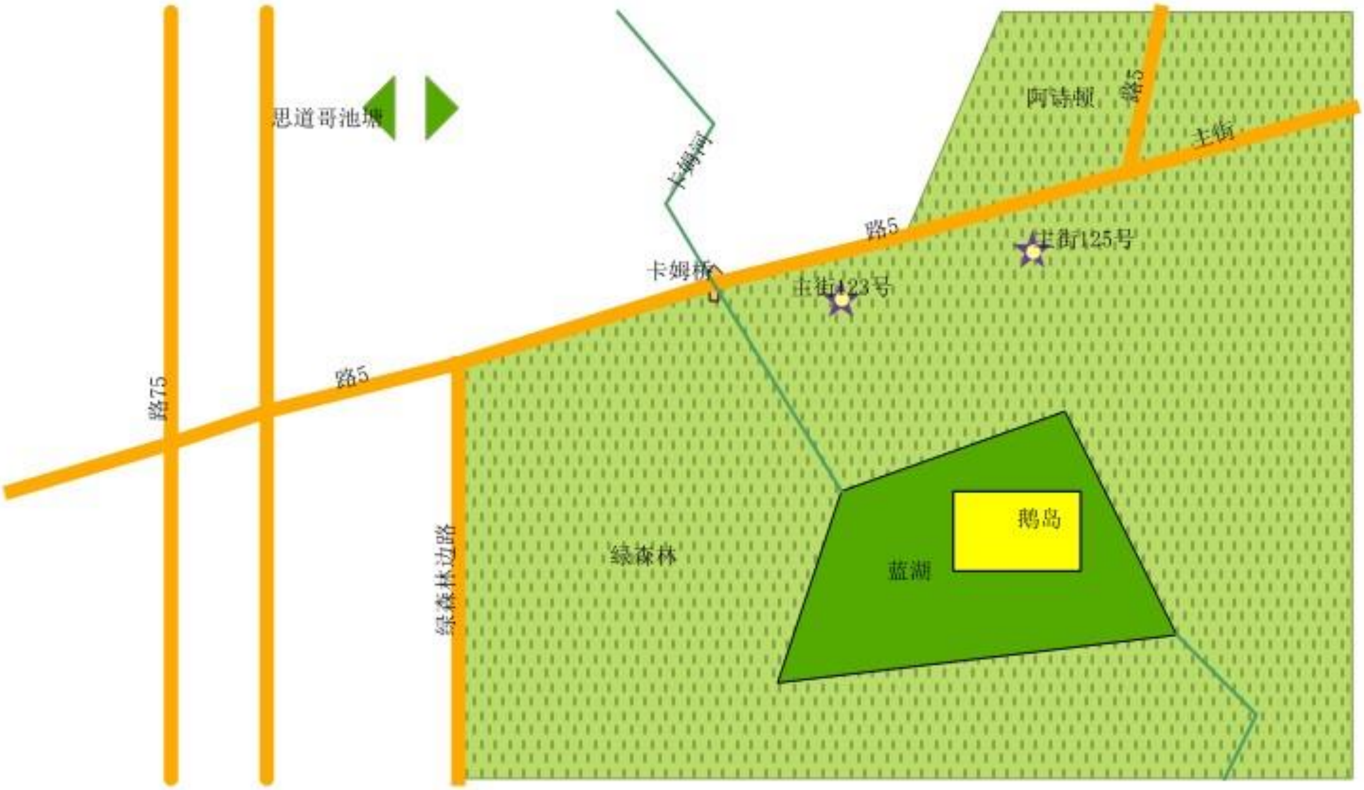
Data Modeling

- How to represent data for applications
  - Relational model – with design principles
  - XML
- Higher-level database design models
  - Entity-Relationship Model (E/R)
  - Unified Modeling Language (UML)
    - Data modeling subset
  - Both are graphical
  - Both can be translated into relations automatically or semi-automatically



5.5 空间数据库设计实例

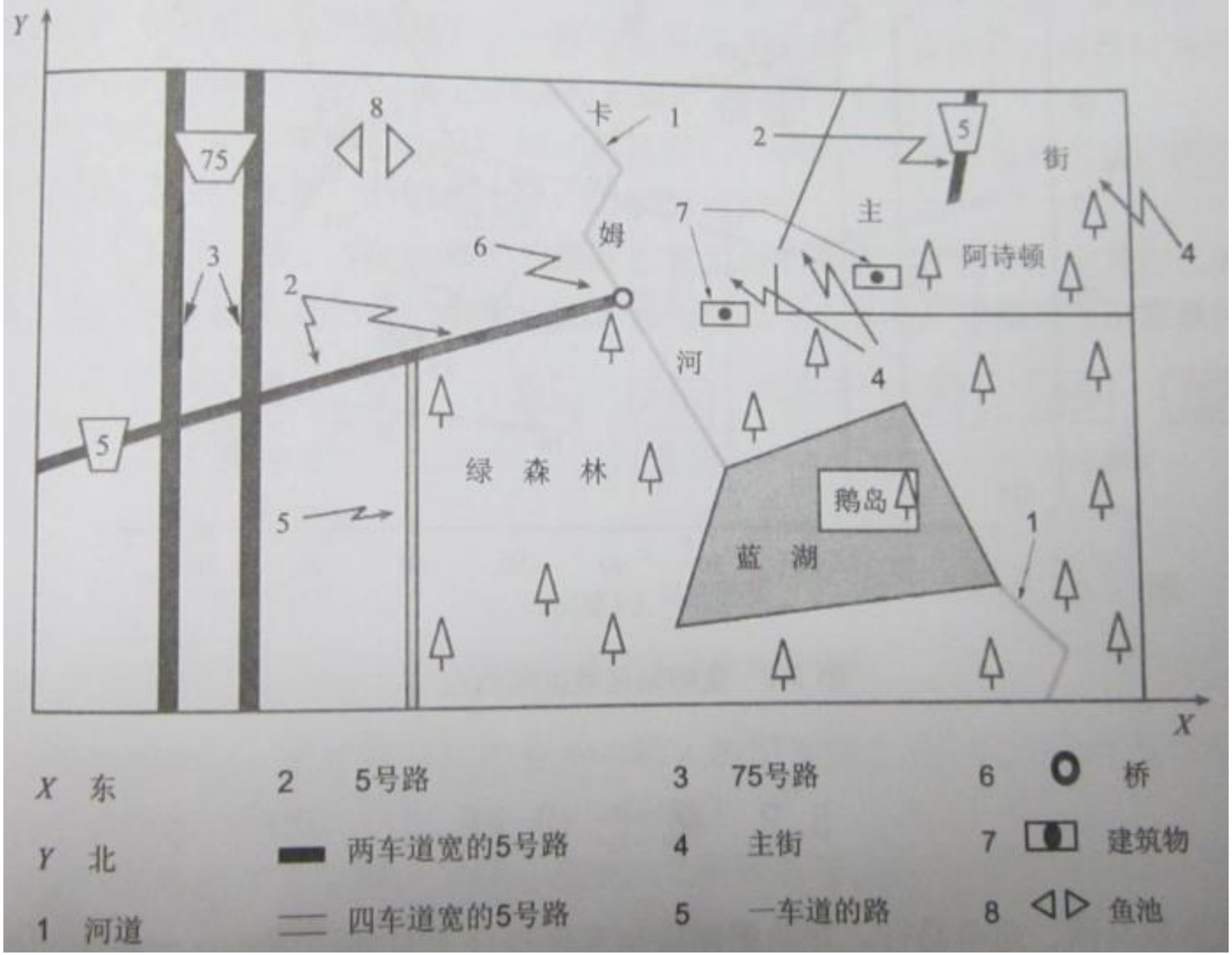
- 空间数据库设计、创建和查询



OGC SFA Part 2 SQL Option P72-111 Conformance tests

5.5 空间数据库设计实例

- 蓝湖区域地图



5.5 空间数据库设计实例

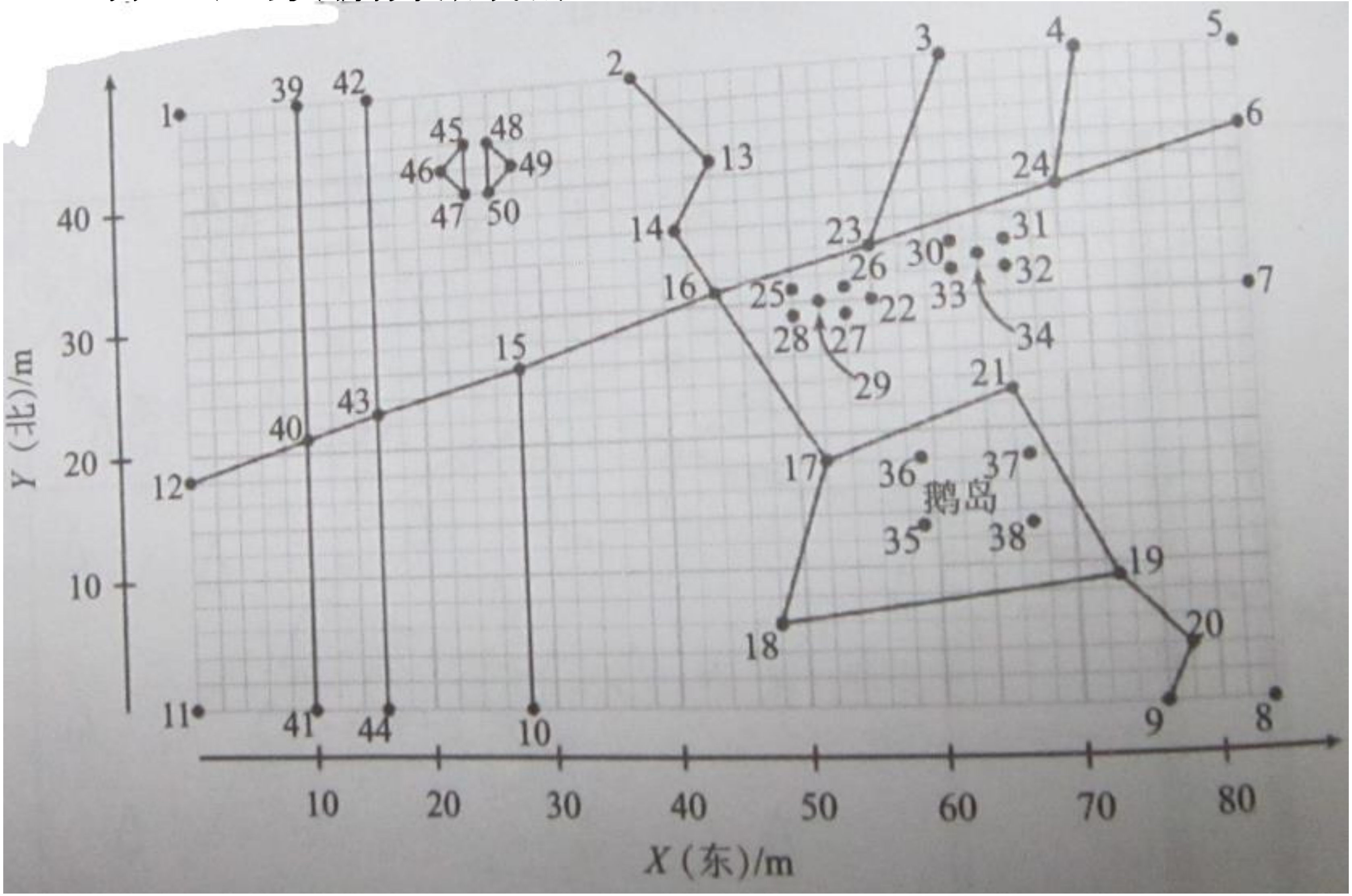
- 图示区域为地球上通用横轴墨卡托投影(Universal Transverse Mercator,UTM)坐标下的一个矩形。水平坐标系统32214号。注意WGS72/UTM 14带东伪偏移值为500000m，单位为m
- “鹅岛”的“蓝湖”是该地区的重要要素
- 有一条从北到南的水系。从上面注入湖的部分叫“卡姆河”，从湖下面出来的那部分没有名字
- 这里有个区域叫“阿诗顿”
- 州属森林管理的区域包括湖和阿诗顿的一部分。形成了州属森林的边界。其中，“绿森林”等于种树森林减去湖

5.5 空间数据库设计实例

- 5号路延伸出了地图
- 组合的75号路高速路用粗的双黑线表示，每条线是被分离高速路的一部分。这两条路被视为多线
- 跨越卡姆河的桥叫“卡姆桥”，被视为点对象
- 与5号路共享一段路的主街总是四车道宽的
- 沿着主街有两个建筑物，他们被视为点，或者被视为矩形区域
- 一车道的路形成了周树森林边界的一部分，如带黑边的灰色区域
- 这里有两个鱼池，他们不是独立的，而是一起的，故是多多边形

5.5 空间数据库设计实例

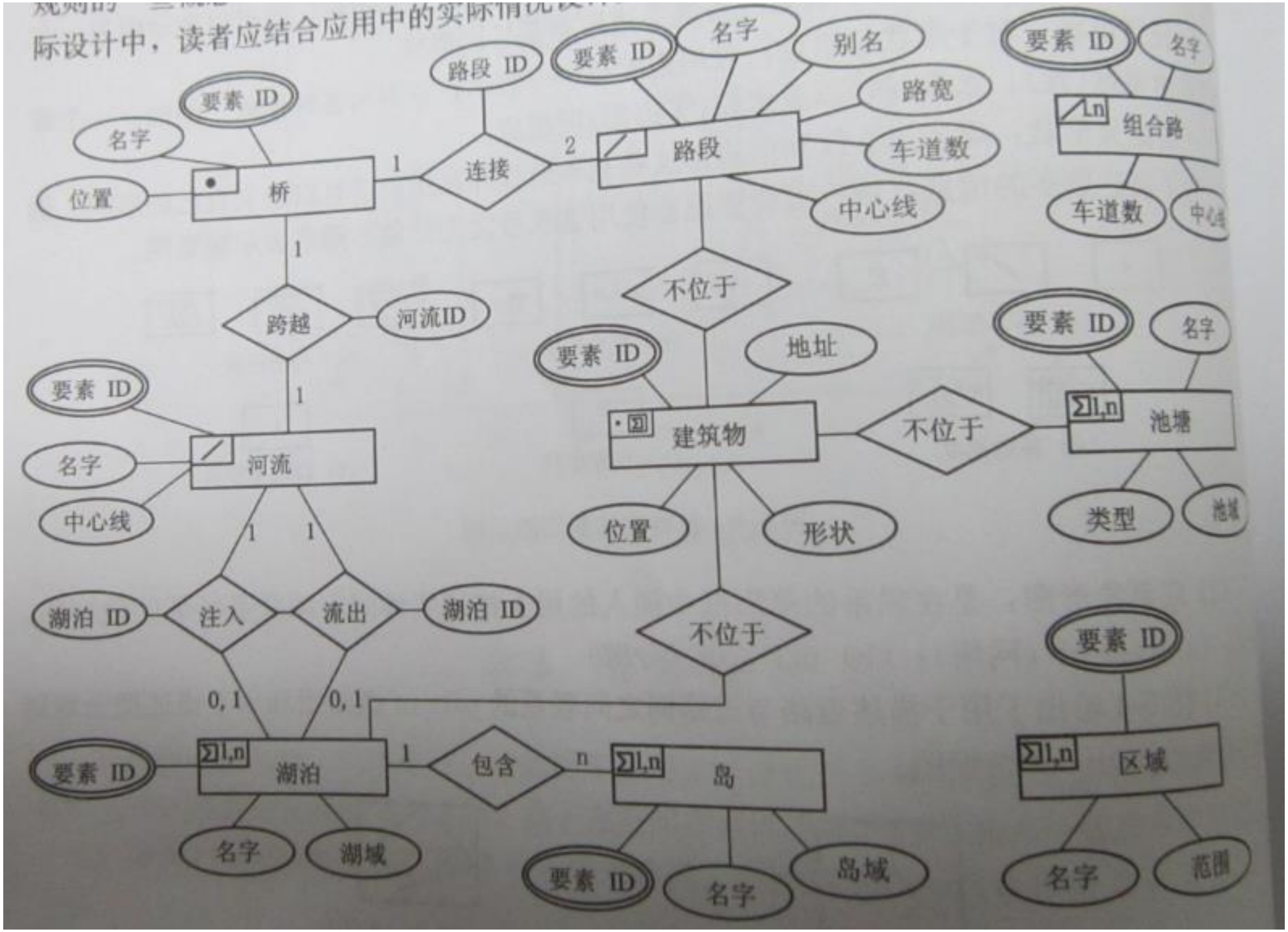
- 蓝湖地区数据集的点



蓝湖区域概念设计

- 空间对象抽象 (9类)
  - 桥，河流，湖泊，岛，路段，组合路，建筑物，池塘，区域
  - 桥 – 点
  - 路段和河流 – 线
  - 组合路由两条分类的路组合成的一条路 – 多边形
  - 建筑物 – 点或多边形（备选状态）
  - 池塘 – 多边形
  - 湖泊，岛和区域 – 多多边形
- 同时具有空间和非空间属性，称为“要素”

蓝湖区域概念设计



际设计中，读者应结合应用中的实际情况设

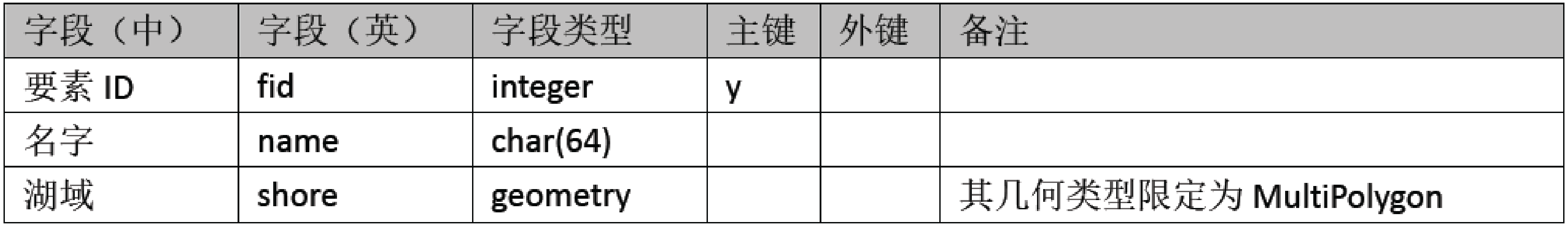
蓝湖区域概念设计

- 同一道路不同路段拥有不同的“别名”和“车道数”，路段作为道路的最小建模单位
- 联系 [要素的行为规则]
  - 1个桥连接2条路段
  - 1个桥跨越1条河流
  - 1条河流可能注入0到1个湖泊，或从0到1个湖泊流出
  - 建筑物不能位于湖泊或池塘中，也不能位于路段上

  - 上述联系仅限于本例，不一定适用与其他情况

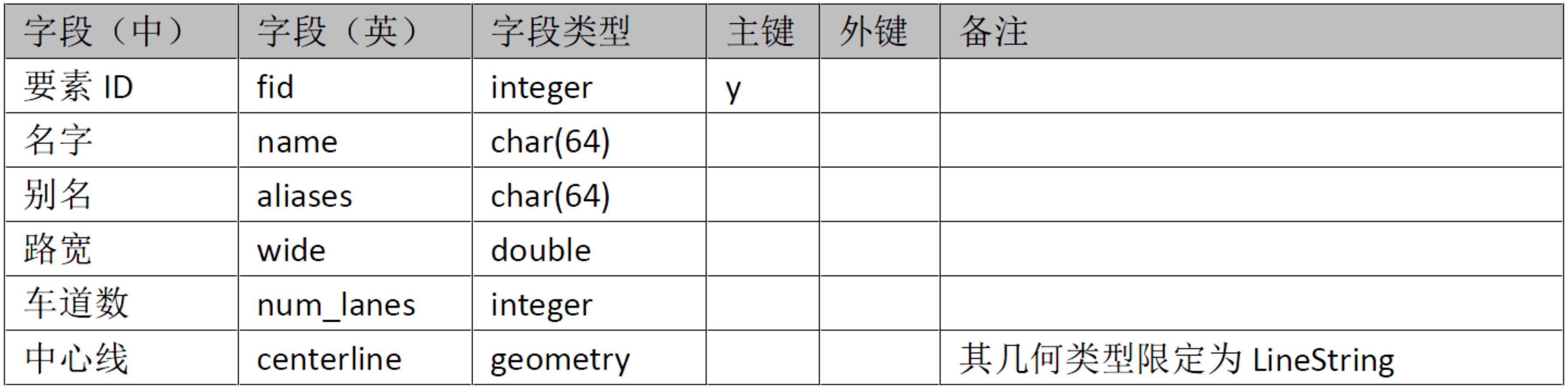
蓝湖区域逻辑结构设计

- 基于E/R图，将每个要素都转换为带有几何类型扩展的二维要素表



字段（中）	字段（英）	字段类型	主键	外键	备注
要素 ID	fid	integer	y		
名字	name	char(64)			
湖域	shore	geometry			其几何类型限定为 MultiPolygon

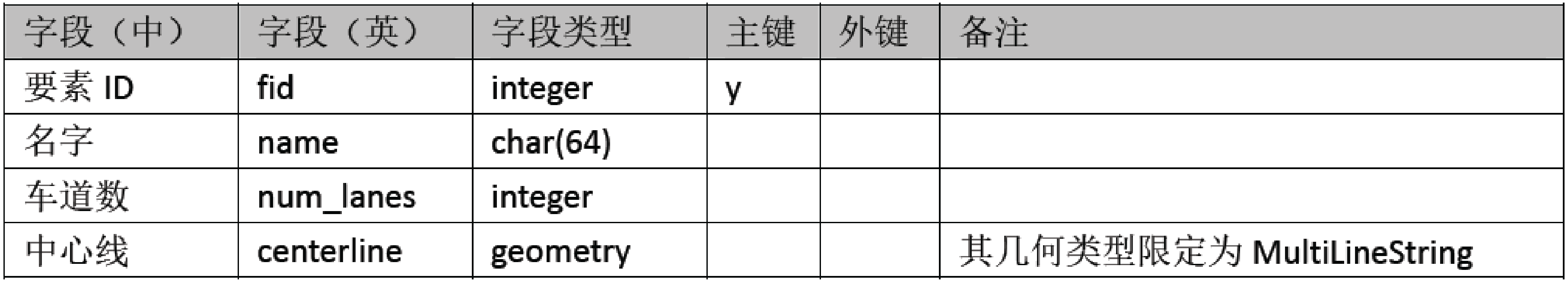
5- 1 湖泊



字段（中）	字段（英）	字段类型	主键	外键	备注
要素 ID	fid	integer	y		
名字	name	char(64)			
别名	aliases	char(64)			
路宽	wide	double			
车道数	num_lanes	integer			
中心线	centerline	geometry			其几何类型限定为 LineString

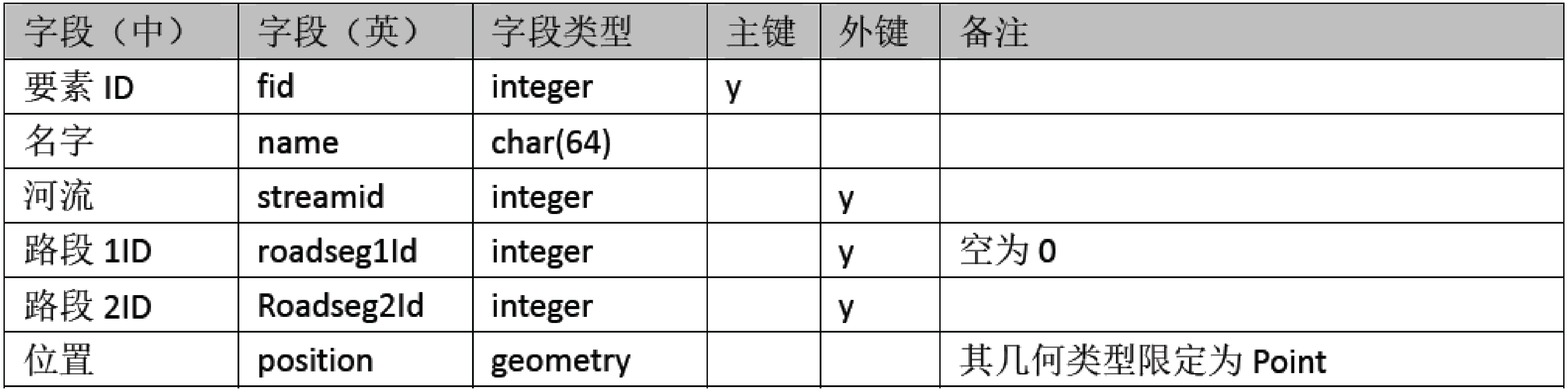
5- 2 路段

蓝湖区域逻辑结构设计



字段（中）	字段（英）	字段类型	主键	外键	备注
要素 ID	fid	integer	y		
名字	name	char(64)			
车道数	num_lanes	integer			
中心线	centerline	geometry			其几何类型限定为 MultiLineString

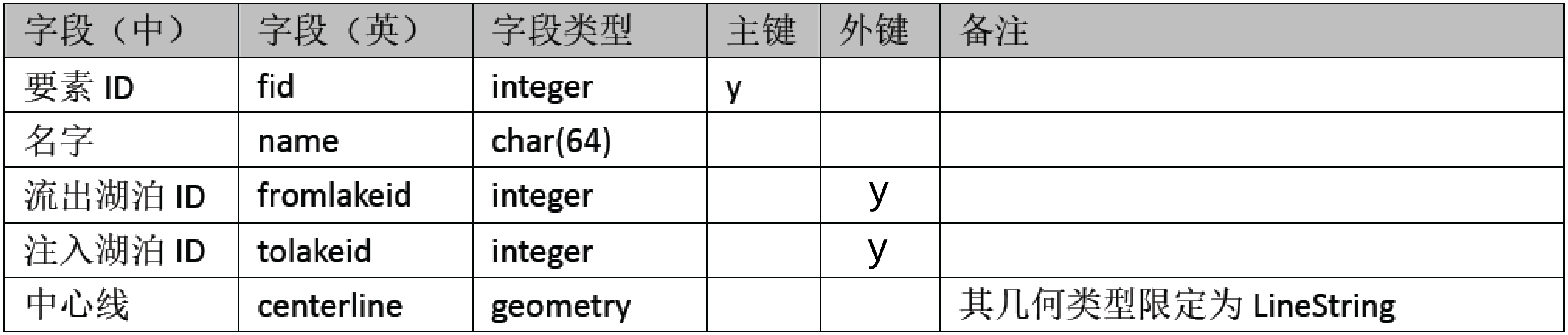
5- 3 组合路段



字段（中）	字段（英）	字段类型	主键	外键	备注
要素 ID	fid	integer	y		
名字	name	char(64)			
河流	streamid	integer		y	
路段 1ID	roadseg1Id	integer		y	空为 0
路段 2ID	Roadseg2Id	integer		y	
位置	position	geometry			其几何类型限定为 Point

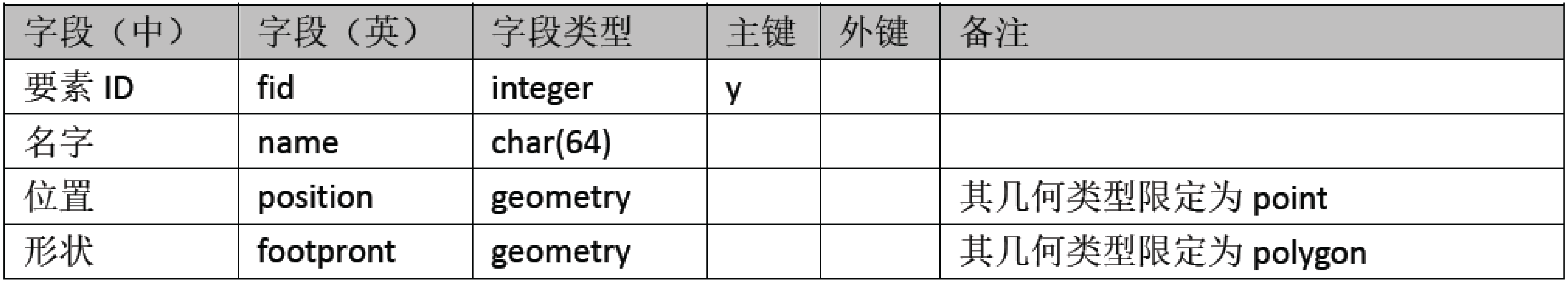
5- 4 桥

蓝湖区域逻辑结构设计



字段（中）	字段（英）	字段类型	主键	外键	备注
要素 ID	fid	integer	y		
名字	name	char(64)			
流出湖泊 ID	fromlakeid	integer		y	
注入湖泊 ID	tolakeid	integer		y	
中心线	centerline	geometry			其几何类型限定为 LineString

5- 5 河流



字段（中）	字段（英）	字段类型	主键	外键	备注
要素 ID	fid	integer	y		
名字	name	char(64)			
位置	position	geometry			其几何类型限定为 point
形状	footpront	geometry			其几何类型限定为 polygon

5- 6 建筑物

蓝湖区域逻辑结构设计

字段（中）	字段（英）	字段类型	主键	外键	备注
要素 ID	fid	integer	y		
名字	name	char(64)			
类型	type	char(64)			
池域	shores	geometry			其几何类型限定为 Mutipolygon

5- 7 池塘

字段（中）	字段（英）	字段类型	主键	外键	备注
要素 ID	fid	integer	y		
名字	name	char(64)			
所在湖泊 ID	lakeID	Integer		y	
形状	footpront	geometry			其几何类型限定为 Mutipolygon

5- 8 岛

字段（中）	字段（英）	字段类型	主键	外键	备注
要素 ID	fid	integer	y		
名字	name	char(64)			
边界	boundary	geometry			其几何类型限定为 Mutipolygon

5- 9 区域

蓝湖区域逻辑结构设计

- 如果空间数据库的几何类型足够丰富，要素表中的空间属性应定义为相应的几何类型
- 取决于具体的空间数据库支持的类型
- Geometry，不同要素表中的几何类型的限定，由用户在应用层进行限制
- 对于具有备选形状的建筑物，为其建立“位置”和“形状”两个geometry字段，其中位置用于存放点状形式表达的建筑物，而形状用于存放多边形形式表达的建筑物

蓝湖区域逻辑结构设计

- 对于图中的“连接”、“跨越”、“流出”、“包含”的关系，在一定程度上分别通过“桥”表中的“路段1ID”和“路段2ID”，“桥”表中的河流ID，河流表中的“注入湖泊ID”，岛中的“所在湖泊ID”进行了一定的限制，但空间关系的限制还要通过触发器或开发者在应用程序中实现。如“不位于”关系只能通过触发器或开发者应用程序中实现
- 逻辑设计表和列名建议改为英文

蓝湖区域物理设计 (课堂练习)

- 选择PostgreSQL/PostGIS实现，在pgAdmin 4中完成建表、插入数据和查询任务
- 建表前，检查空间数据库SPATIAL_REF_SYS系统表中，是否存在WGS72/UTM14带的空间参考信息
- 若不存在，需要创建该空间参考系
- 建表SQL语句
- 建表后，在SPATIAL_REF_SYS和GEOMETRY_COLUMNS系统表中查看建表信息

蓝湖区域物理设计 (课堂练习)

- 空间数据插入
- 空间数据查询
  - 简单空间查询
  - 复杂空间查询
  - Spatial Join 通过空间关系进行连接

第五章 空间扩展E/R图

- 5.1 Database Design
- 5.2 E/R Basic: Entities & Relations
- 5.3 E/R Design Considerations
  - 5.3.1 Relations: multiplicity, multi-way
  - 5.3.2 Design considerations
  - 5.3.3 Conversion to SQL
- 5.4 Advanced E/R Concepts
  - 5.4.1 Subclasses & connection to OO
  - 5.4.2 Constraints
  - 5.4.3 Weak entity sets
- 5.5 Spatial Database Design Example

参考教材：
数据库系统概念 7.1-7.10
空间数据库管理系统概论 5.1-5.4