Introduction to Database Development and ER Model

The great successful men of the world have used their imaginations. They think ahead and create their mental picture. and then go to work materializing that picture in all its details, filling in here, adding a little there, altering this bit and that bit, but steadily building, steadily building.

Robert Collier

Data Model

- The Entity-Relationship *(ER) data model* allows us to describe the data involved in a real-world enterprise in terms of objects and their relationships.
  - Widely used to develop an initial database design.
- Move from informal description of what user wants to
- Precise description of what can be implemented in a DBMS

Steps in developing a database

- Requirements analysis
- Conceptual Database design
- Logical Database design
- Schema refinement
- Physical Database design
- Applications and security

Conceptual design: (ER Model is used at this stage.)

- What are the *entities* and *relationships* in the enterprise?
- What information about these entities and relationships should we store in the database?
- What are the *integrity constraints* or *business rules* that hold?
- A database `schema' in the ER Model can be represented pictorially (*ER diagrams*).
- Can map an ER diagram into a relational schema.

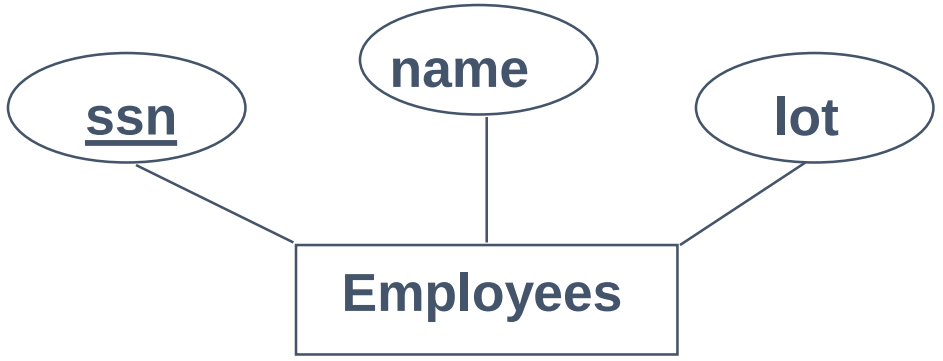
ER Model Basics

- *Entity:* Real-world object distinguishable from other objects. An entity is described using a set of *attributes*. Each attribute has a *domain*.

- *Entity Set*: A collection of similar entities. E.g., all employees.

  - All entities in an entity set have the same set of attributes. (Until we consider ISA hierarchies, anyway!)

  - Each entity set has a *key*.



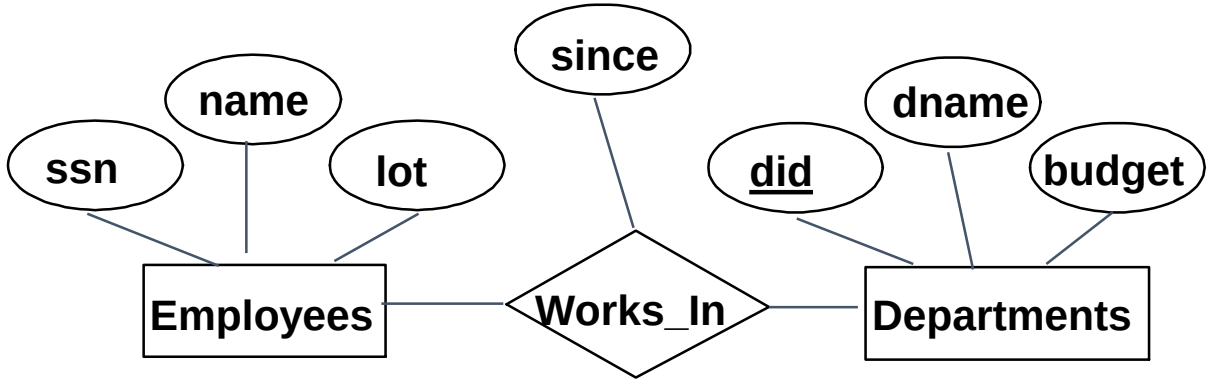
Keys

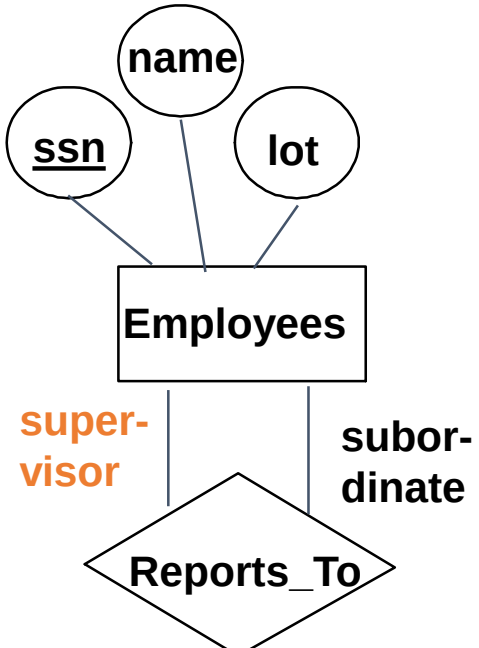
- Minimal set of attributes which uniquely identify an instance of a entity
- Many candidate keys choose one to be a primary keys
- SSN vs Name … key must be unique

ER Model Basics (Contd.)

- *Relationship*: Association among two or more entities. E.g., Attishoo works in Pharmacy department.

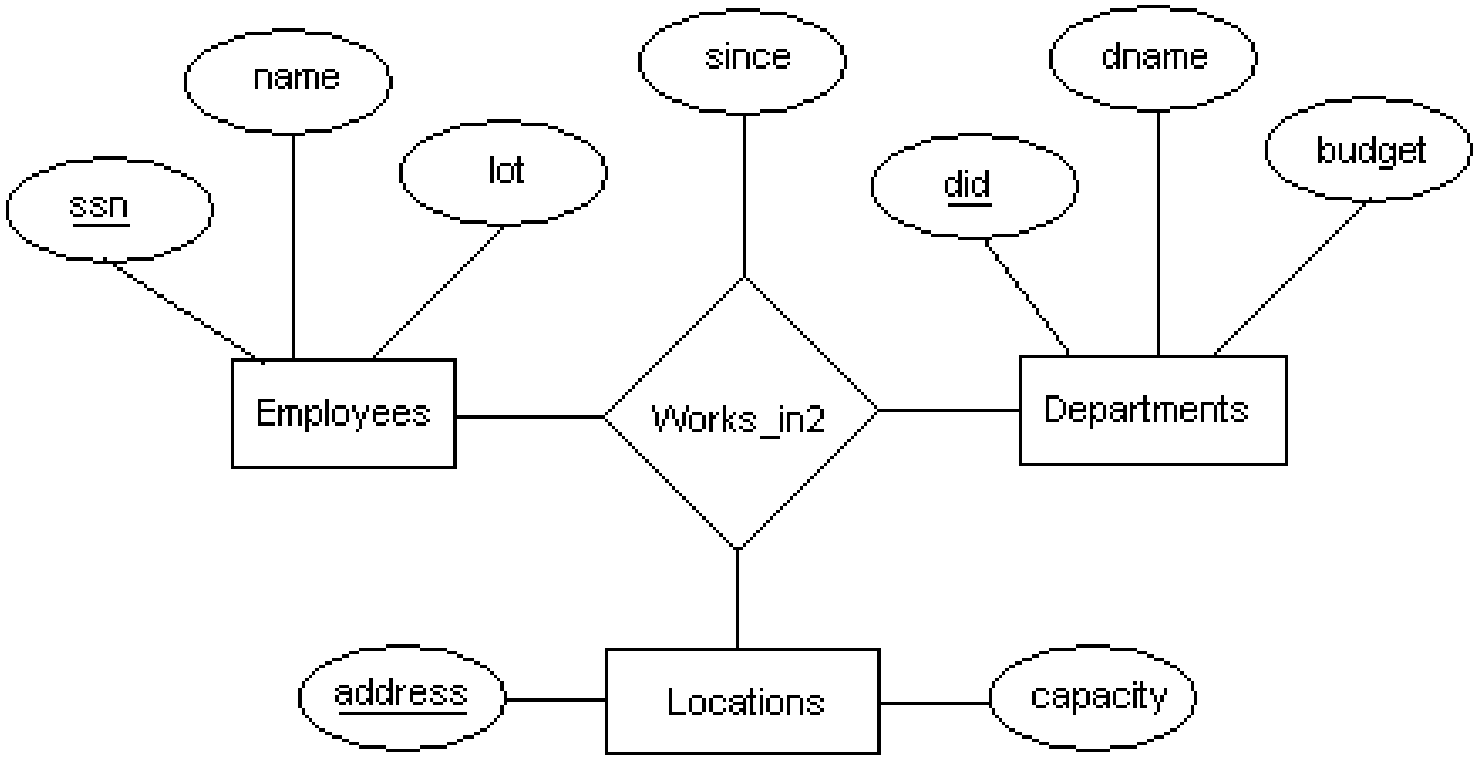
- *Relationship Set*: Collection of similar relationships.
  - An n-ary relationship set ***R*** relates ***n*** entity sets $E_1 \ldots E_n$;
  - each relationship in R involves entities $\{e_1 \in E_1, \ldots, e_n \in E_n\}$
  - Same entity set could participate in different relationship sets, or in different "roles" in same set.





Ternary Relationship Set

- Records association between Employees, Departments, Locations.



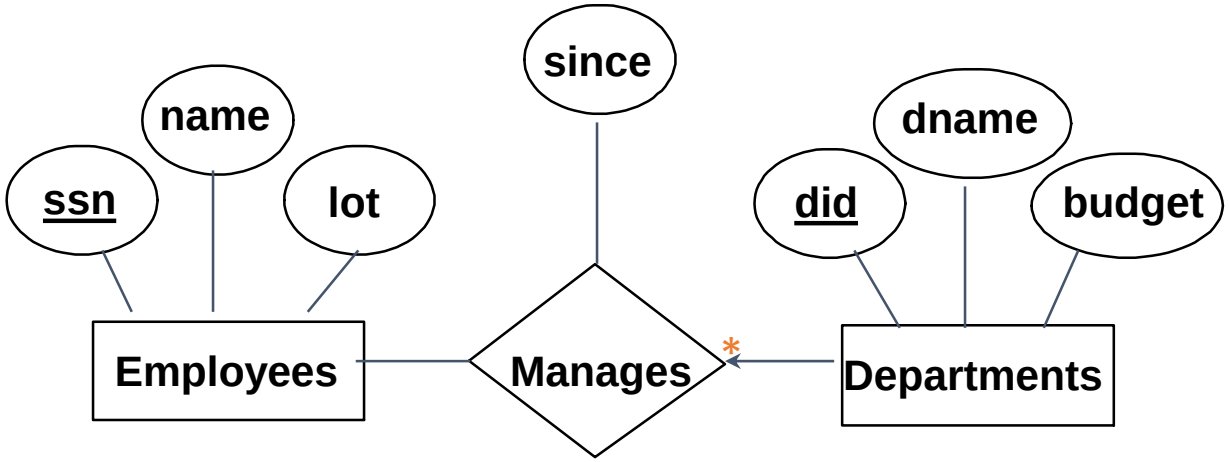
Key Constraints

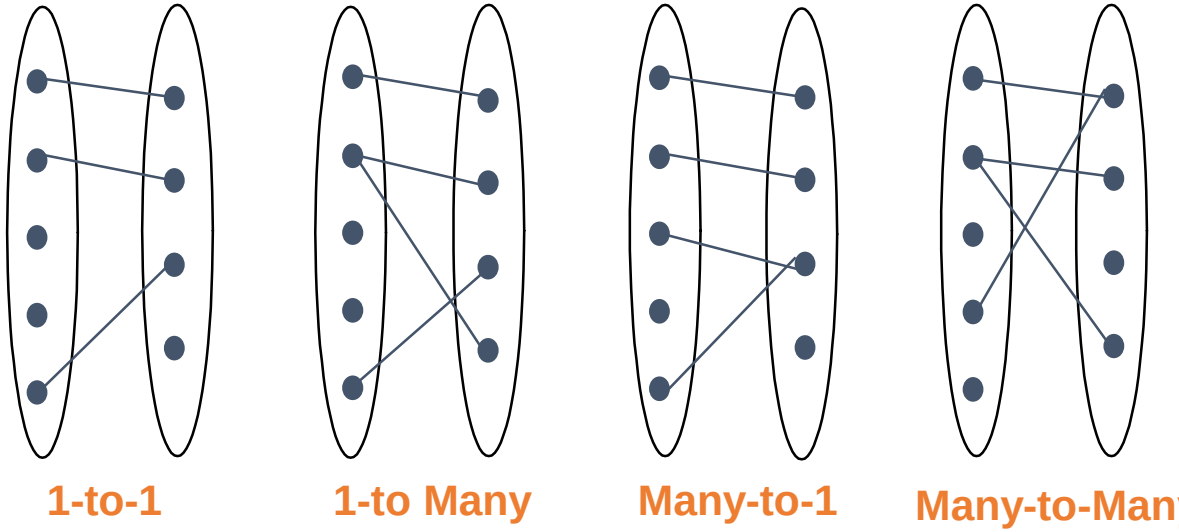
- Consider Works_In: An employee can work in many departments; a dept can have many employees.

So many to many

Key Constraints

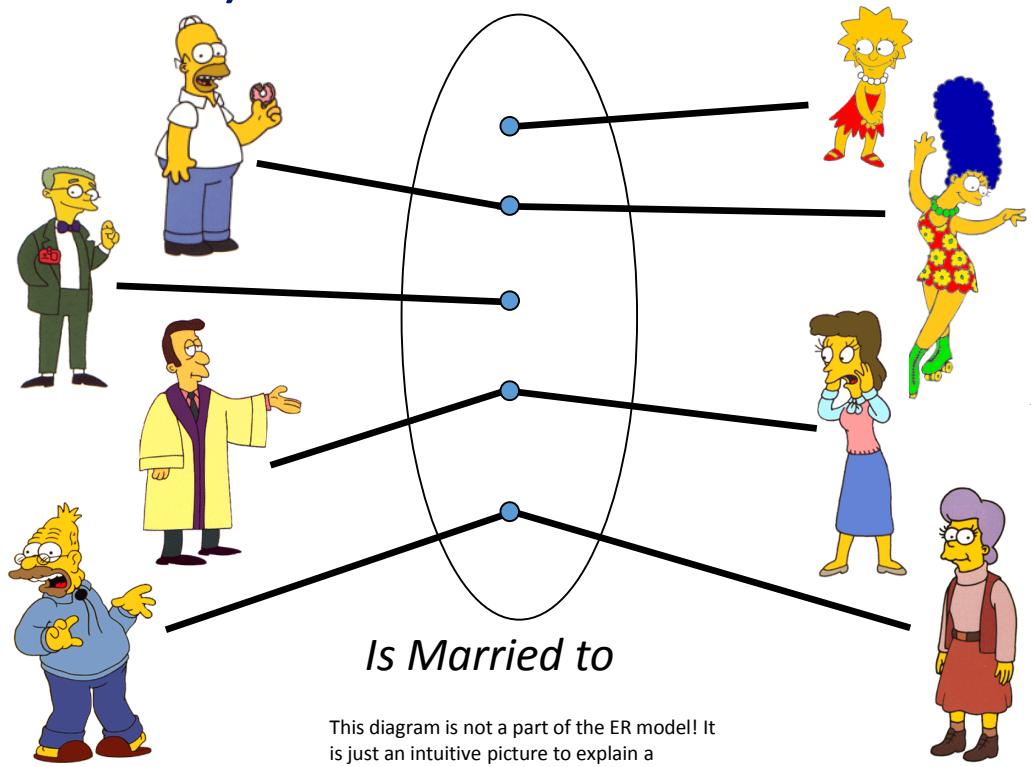
- In contrast, each dept has at most one manager, according to the *key constraint* on Manages.
- * arrow indicates that given a dept it uniquely determines the Manages relationship in which it appear.
- Intuitively, the arrow states that given a Departments entity, we can uniquely determine the Manages relationship in which it appears



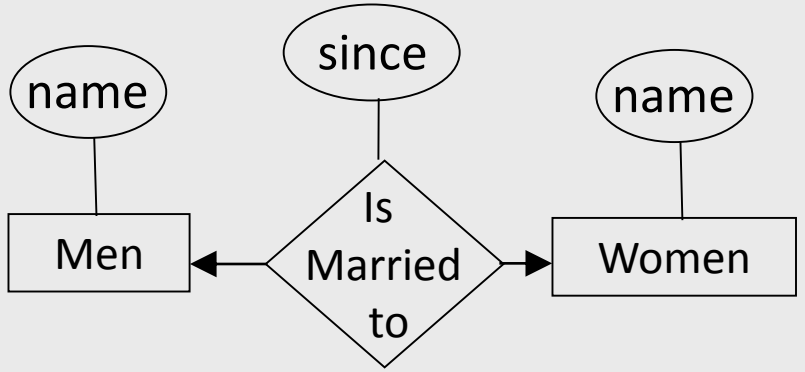


Key Constraints: Examples

- **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.
- A man may be married to at most one women, and woman may be married to at most one man (both men and women can be unmarried)

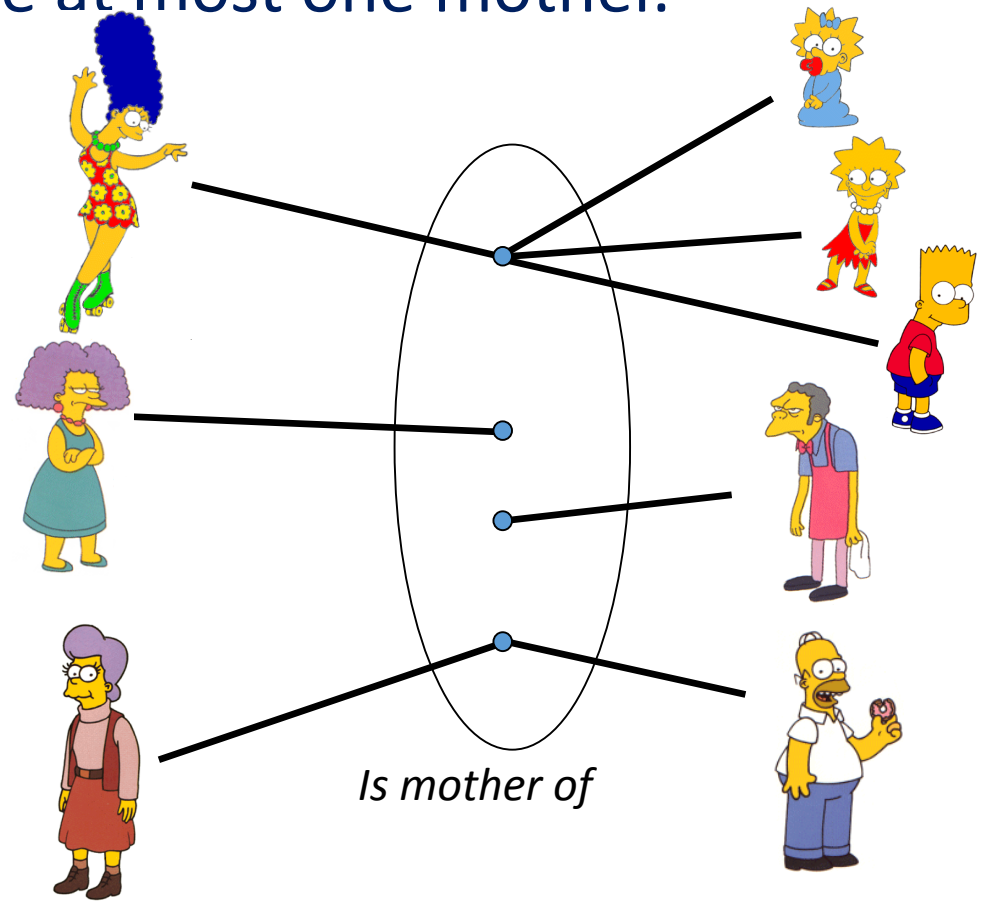


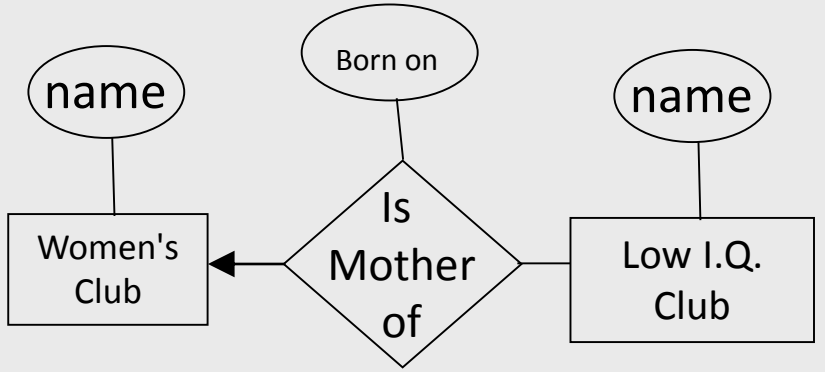
This diagram is not a part of the ER model! It is just an intuitive picture to explain a concept



Key Constraints: Examples

- **One-to-many**: An entity in A is associated with any number in B. An entity in B is associated with at most one entity in A.
- A women may be the mother of many (or no) children. A person may have at most one mother.

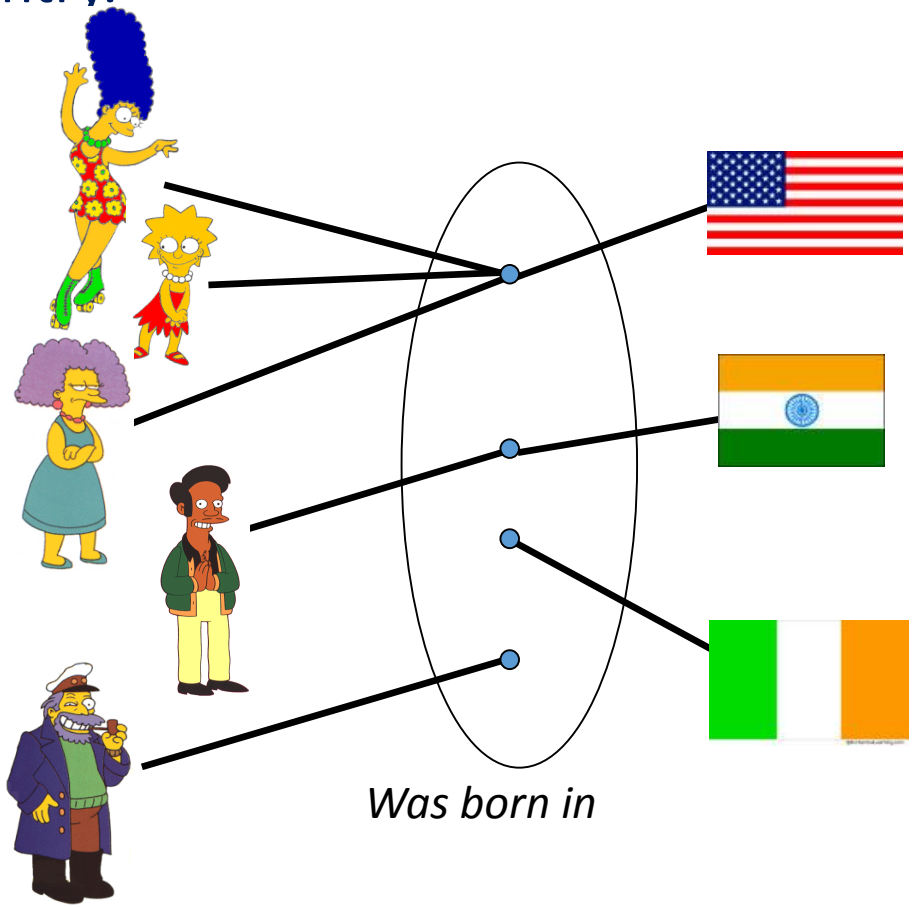


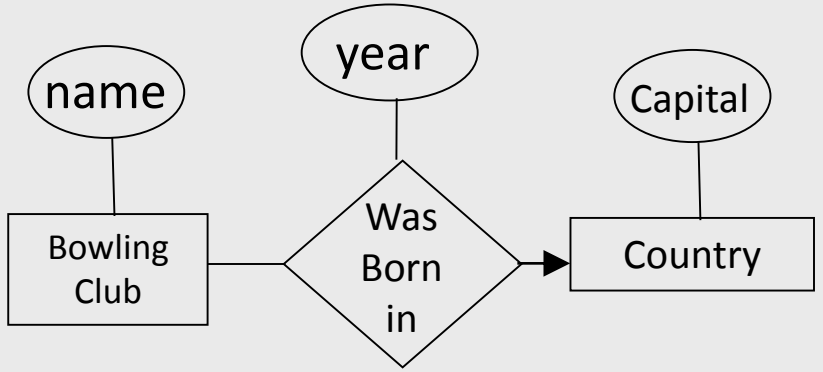


Note that this example is not saying that Moe does not have a mother, since we know as a biological fact that everyone has a mother.
It is simply the case that Moes mom is not a member of the Women's club.

Key Constraints: Examples

- **Many-to-one**: An entity in A is associated with at most one entity in B. An entity in B is associated with any number in A.
- Many people can be born in any county, but any individual is born in at most one country.

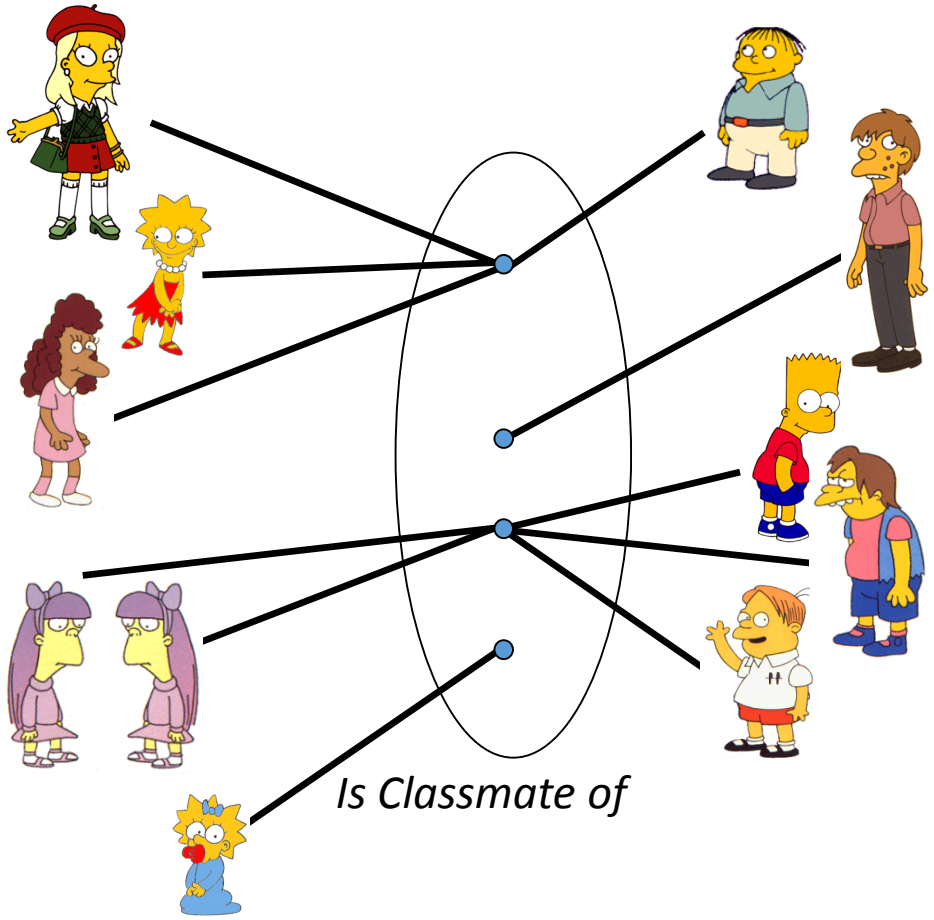


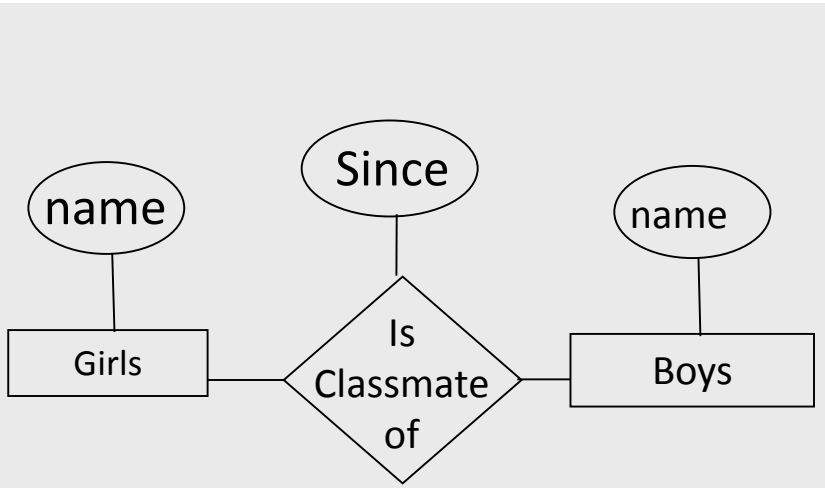


Note that we are not saying that the Sea Captain was not born in some country, he almost certainly was, we just don't know which country, or it is not in our Country entity set.
Also note that we are not saying that no one was born in Ireland, it is just that no one in

Key Constraints: Examples

- **Many-to-many**: Entities in A and B are associated with any number from each other.

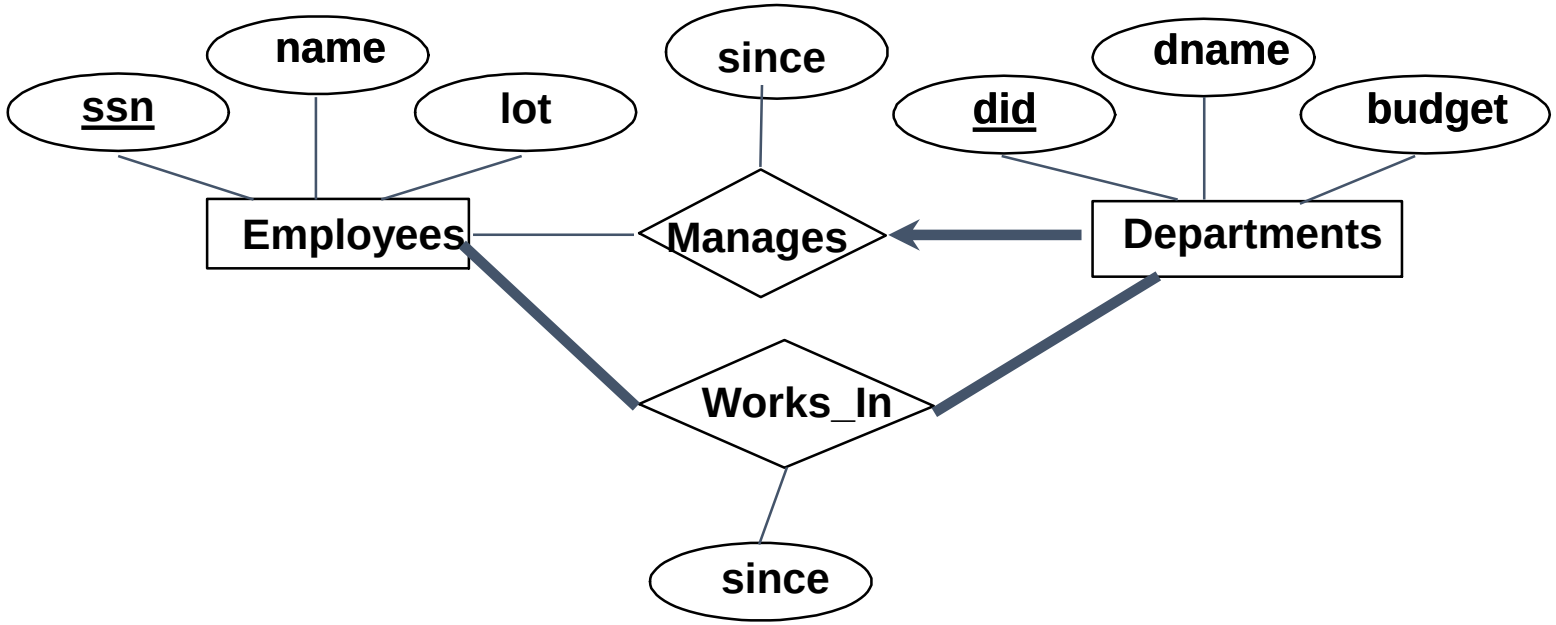




Participation Constraints

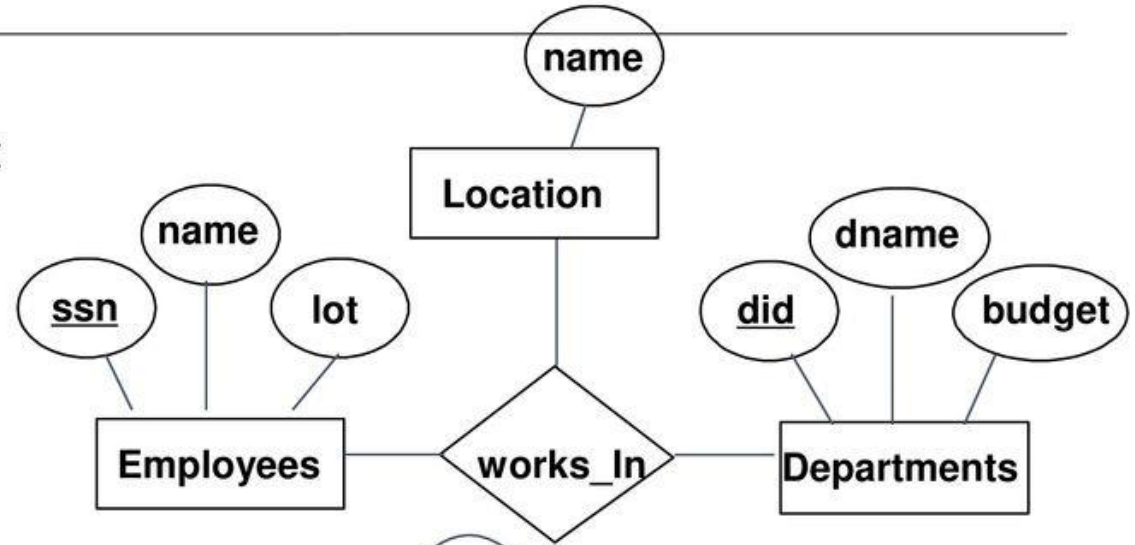
- Does every department have a manager?
  - If so, this is a *participation constraint*: the participation of Departments in Manages is said to be *total* (vs. *partial*).
- Every Department entity must appear in an instance of the relationship Works_In (have an employee) and every Employee must be in a Department

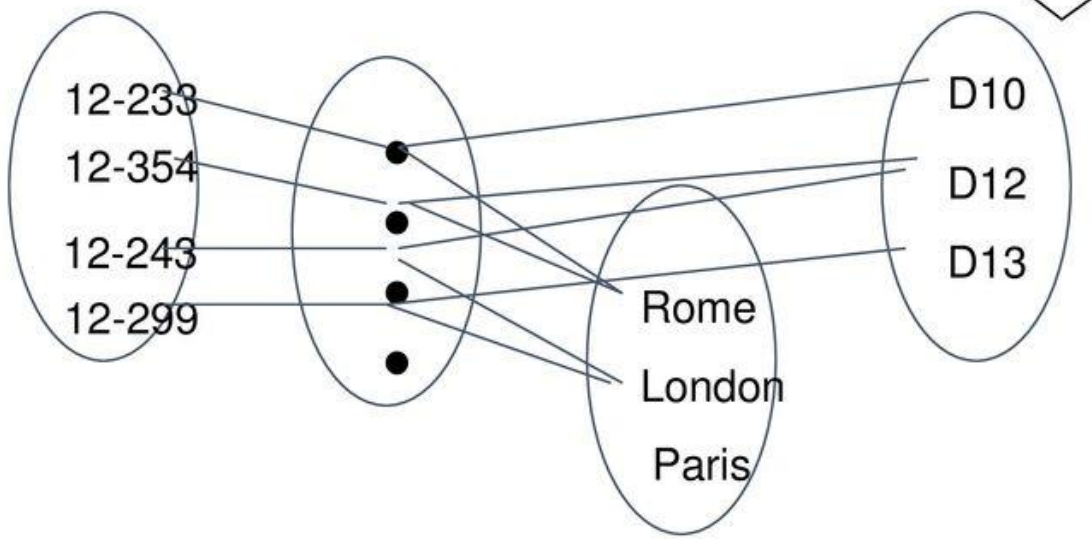
- Both Employees and Departments participate totally in Works_In



Key Constraints (ternary relationships)

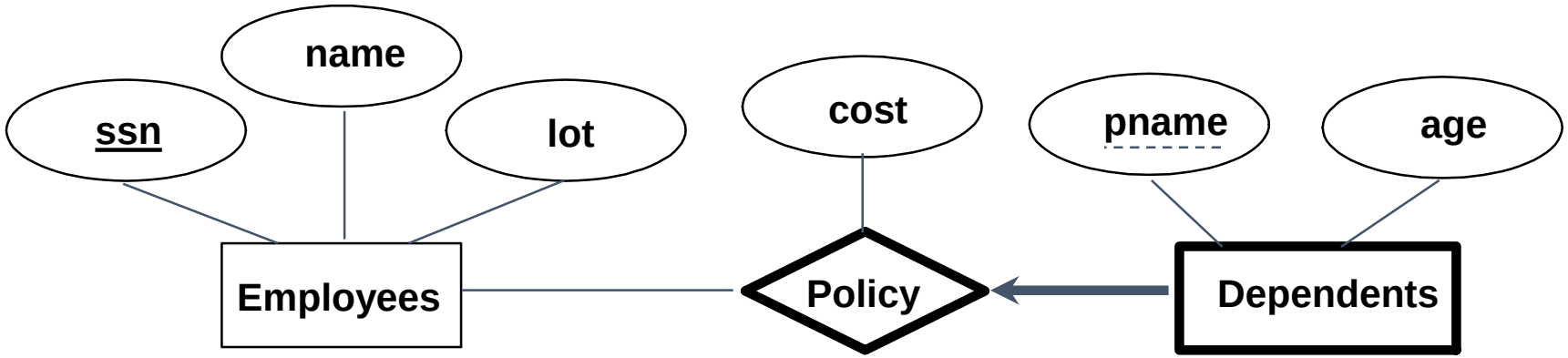
Each employee can work at most in one department at a single location





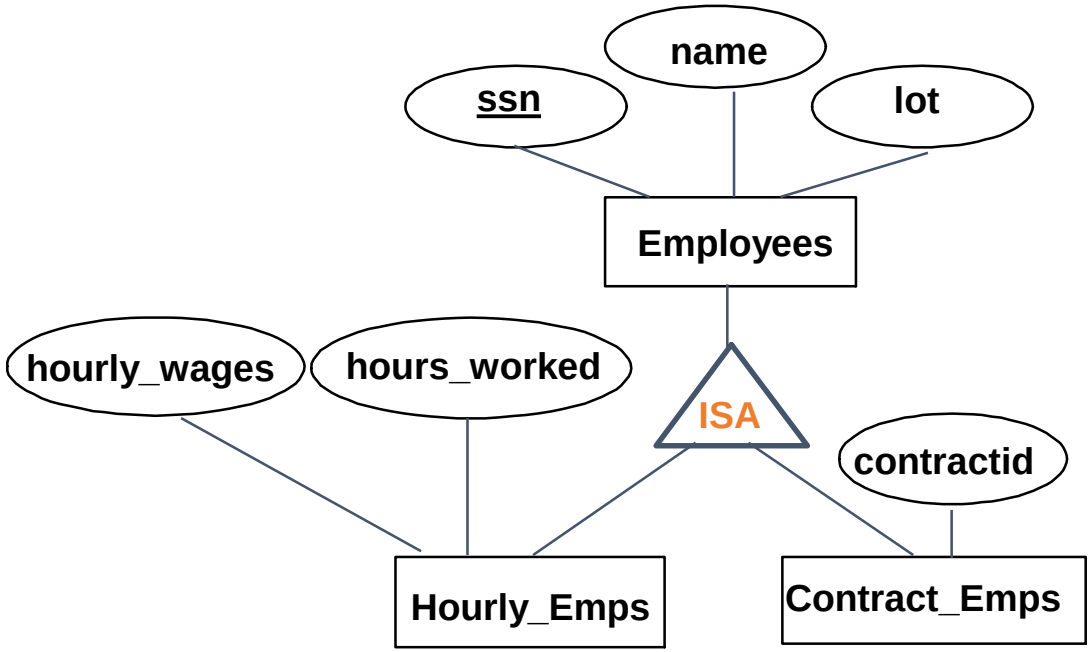
Weak Entities

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



ISA (`is a') Hierarchies

- As in C++, attributes can be inherited.
- If we declare A **ISA** B, every A entity is also considered to be a B entity.
- Upwards is generalization. Down is specialization



Constraints in ISA relation

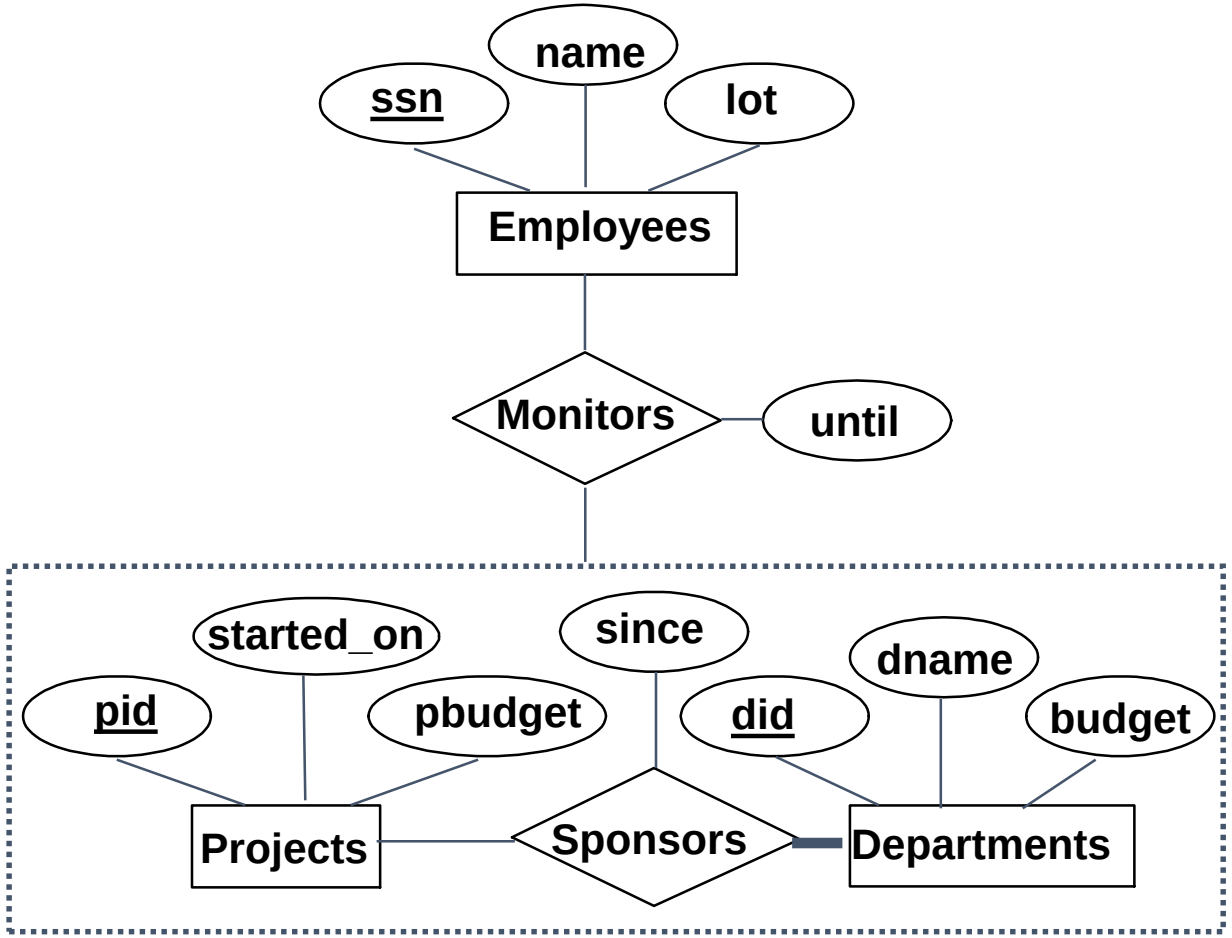
- *Overlap constraints*: Can Joe be an Hourly_Emps as well as a Contract_Emps entity? (*Allowed/disallowed*)

- *Covering constraints*: Does every Employees entity also have to be an Hourly_Emps or a Contract_Emps entity? *(Yes/no)*

- Reasons for using ISA:
  - To add descriptive attributes specific to a subclass.
  - To identify entitities that participate in a relationship.

Aggregation

- Used when we have to model a relationship involving (entitity sets and) a *relationship set*.



Aggregation allows us to treat a relationship set as an entity set for purposes of participation in (other) relationships.

Aggregation vs. ternary relationship:

- Monitors in last example is a distinct relationship, with a descriptive attribute.

- Also, can say that each sponsorship is monitored by at most one employee.

Conceptual Design Using the ER Model

- Design choices:
  - Should a concept be modeled as an entity or an attribute?
  - Should a concept be modeled as an entity or a relationship?
  - Identifying relationships: Binary or ternary? Aggregation?

- Constraints in the ER Model:
  - A lot of data semantics can (and should) be captured.
  - But some constraints cannot be captured in ER diagrams.

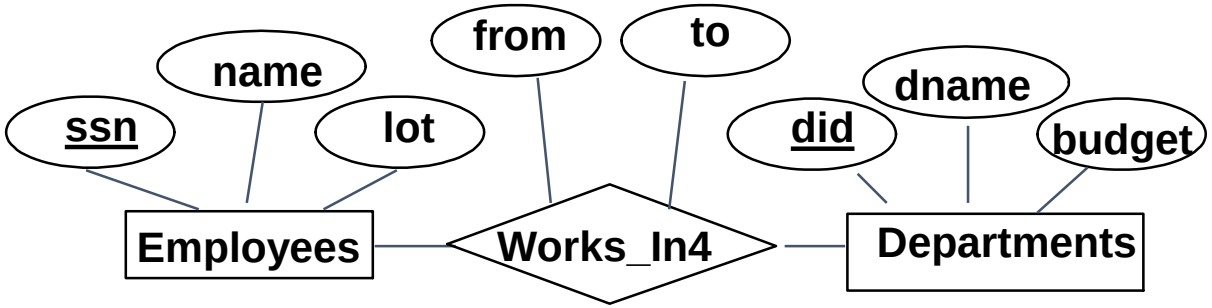
Entity vs. Attribute

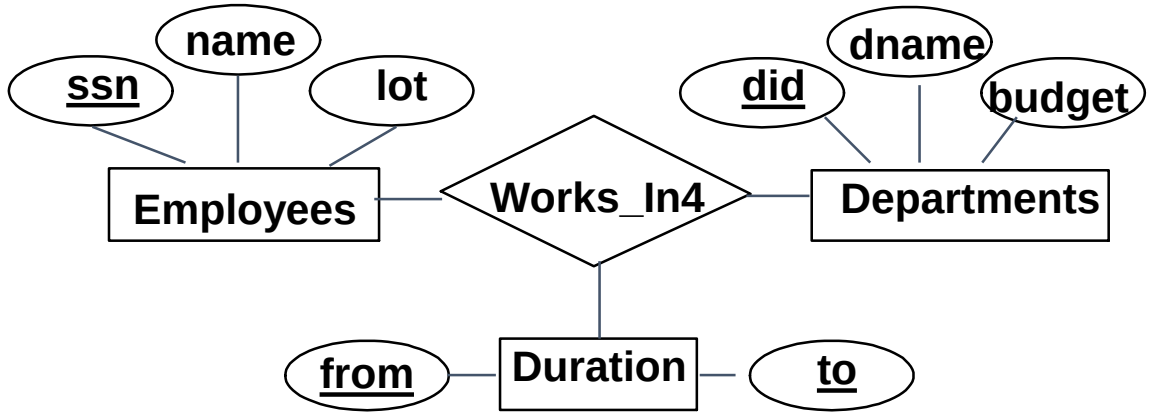
- Should *address* be an attribute of Employees or an entity (connected to Employees by a relationship)?
- Depends upon the use we want to make of address information, and the semantics of the data:
  - If we have several addresses per employee, *address* must be an entity (since attributes cannot be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

Entity vs. Attribute (Contd.)

- Works_In4 does not allow an employee to work in a department for two or more periods.

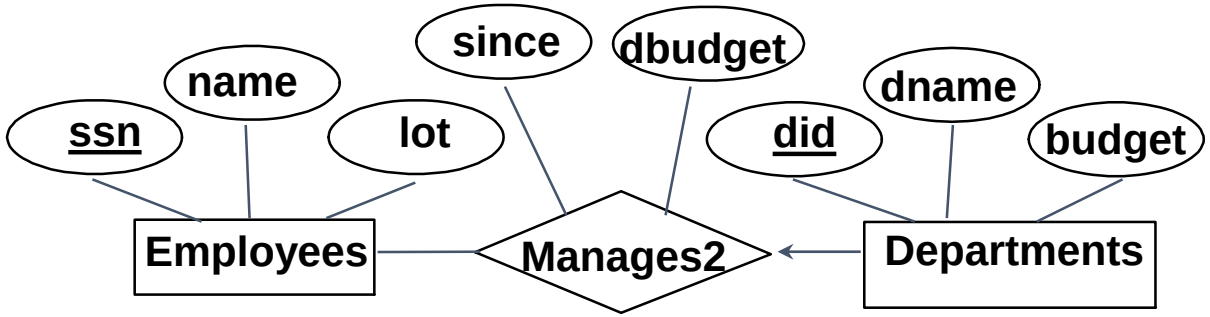
- Similar to the problem of wanting to record several addresses for an employee: We want to record *several values of the descriptive attributes for each instance of this relationship.* Accomplished by introducing new entity set, Duration.



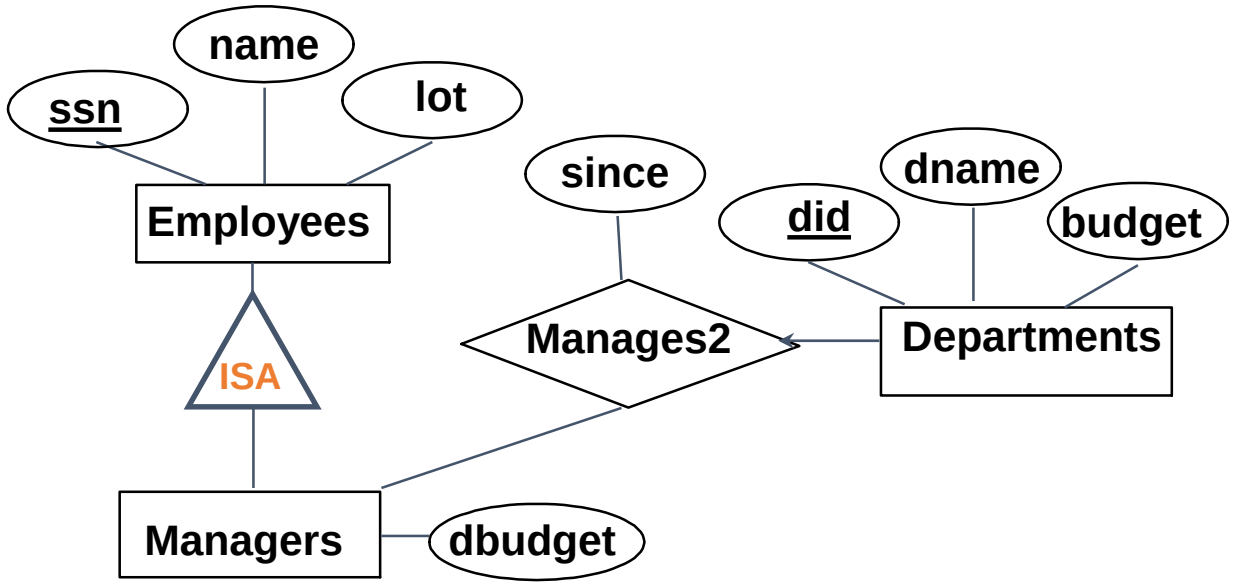


Entity vs. Relationship

- ER diagram OK if a manager gets a separate discretionary budget for each dept.
- What if a manager gets a discretionary budget that covers *all* managed depts?
  - Redundancy: *dbudget* stored for each dept managed by manager.
  - Misleading: Suggests *dbudget* associated with department-mgr combination.

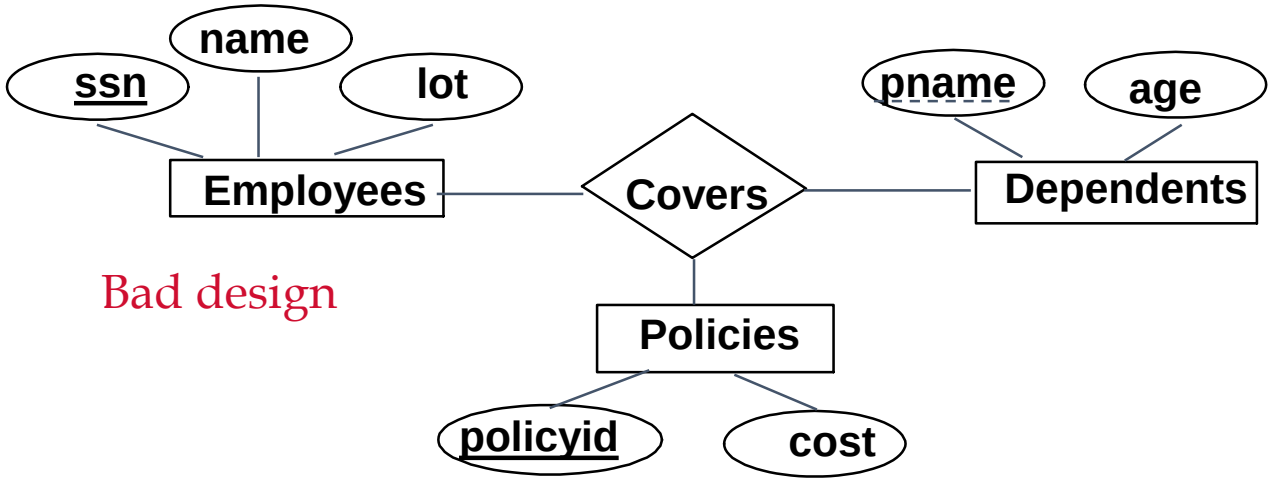


This fixes the problem!

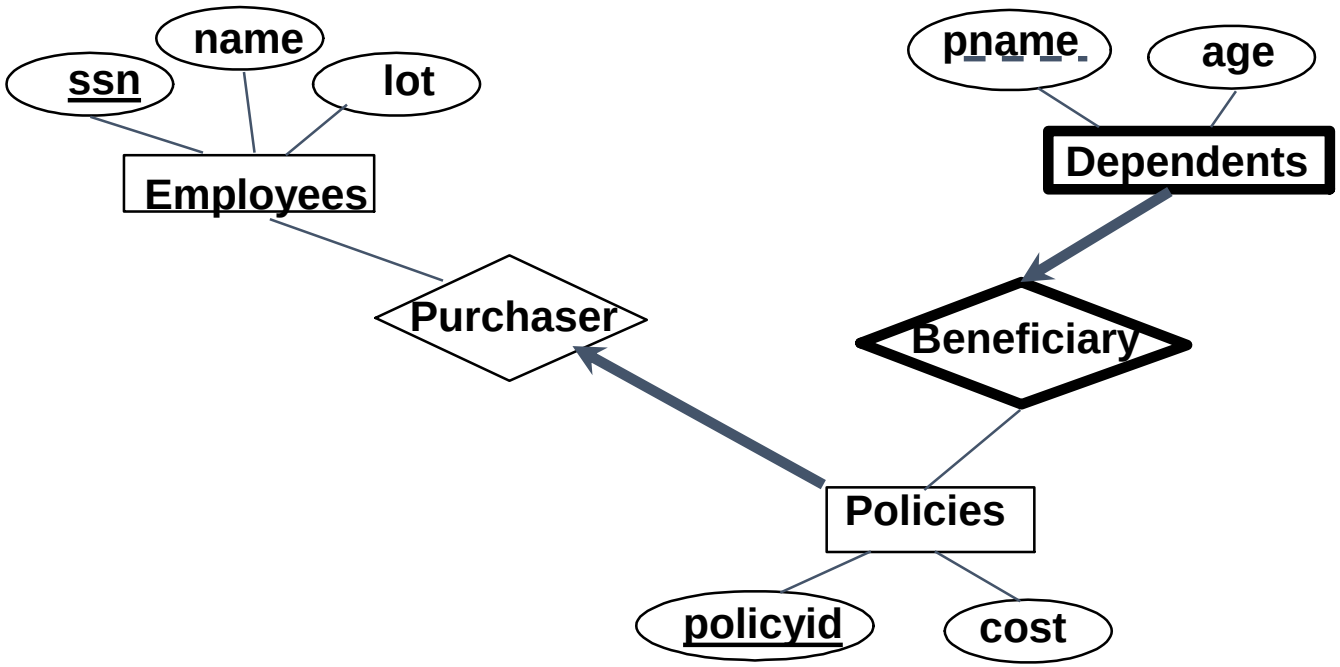


Binary vs. Ternary Relationships

- If each policy is owned by just 1 employee, and each dependent is tied to the covering policy, first diagram is inaccurate.
- What are the additional constraints do we need?



Better design



- Key constraint
- Total participation of policies in purchaser relationship

Binary vs. Ternary Relationships (Contd.)

- Previous example illustrated a case when two binary relationships were better than one ternary relationship.
- An example in the other direction: a ternary relation Contracts relates entity sets Parts, Departments and Suppliers, and has descriptive attribute *qty*. No combination of binary relationships is an adequate substitute:
  - S "can-supply" P, D "needs" P, and D "deals-with" S does not imply that D has agreed to buy P from S.
  - How do we record *qty*?

Summary of Conceptual Design

- *Conceptual design* follows *requirements analysis*,
  - Yields a high-level description of data to be stored
- ER model popular for conceptual design
  - Constructs are expressive, close to the way people think about their applications.
- Basic constructs: *entities*, *relationships*, and *attributes* (of entities and relationships).
- Some additional constructs: *weak entities*, *ISA hierarchies*, and *aggregation*.
- Note: There are many variations on ER model.

Summary of ER (Contd.)

- Several kinds of integrity constraints can be expressed in the ER model: *key constraints*, *participation constraints*, and *overlap/covering constraints* for ISA hierarchies. Some *foreign key constraints* are also implicit in the definition of a relationship set.
  - Some constraints (notably, *functional dependencies*) cannot be expressed in the ER model.
  - Constraints play an important role in determining the best database design for an enterprise.

Summary of ER (Contd.)

- ER design is *subjective*. There are often many ways to model a given scenario! Analyzing alternatives can be tricky, especially for a large enterprise. Common choices include:
  - Entity vs. attribute, entity vs. relationship, binary or n-ary relationship, whether or not to use ISA hierarchies, and whether or not to use aggregation.
- Ensuring good database design: resulting relational schema should be analyzed and refined further. FD information and normalization techniques are especially useful.

References

- Adhsakkdi Y Raghuram Krishnan and Johannes Gehrke, Database Management Systems, 3/e, TMH, 2007.
- Some of the figures and Examples are taken from various online sources.