The Entity-Relationship (ER) Model

Module 2

Professor Alex Brodsky

Database Systems

Purposes of DBMS

- Provide support for “easy-to-use” data
  - Data model (data)
  - Transaction model (operation)
- Provide efficient storage and access of the data in terms of the data model and transactional model.

Data Models

- Tools to obtain data *abstraction*.
- Necessary to be *general* and *intuitive.*
- Data model: A class of mathematical structures, with description and operations
- Conceptual data model: Just structural description

Overview of Database Design

- *Conceptual design*
  - Use *ER Model:* E- *Entities* and R-*Relationships*
  - Decide the *entities* and *relationships* in the enterprise.
  - Decide what information about these entities and relationships should we store in the database.
  - Decide the *integrity constraints* or *business rules.*
- *Implementation*
  - Map an ER model into a relational schema.

ER Model Basics

- ***Entity:*** A real-world object distinguishable from other objects.
- Distinguishable via its description (*data*)
- *Attribute*: a mapping that maps an object to a value (called the *attribute value).* E.g.: Age is an attribute of students objects.
- An entity is described (in DB) using a set of *attributes values.*
- *Entity Set*: A collection of *similar* entities. E.g., all employees.
  - Similar: All entities in an entity set have the same set of attributes.

ER Diagram: Entity Set & Example

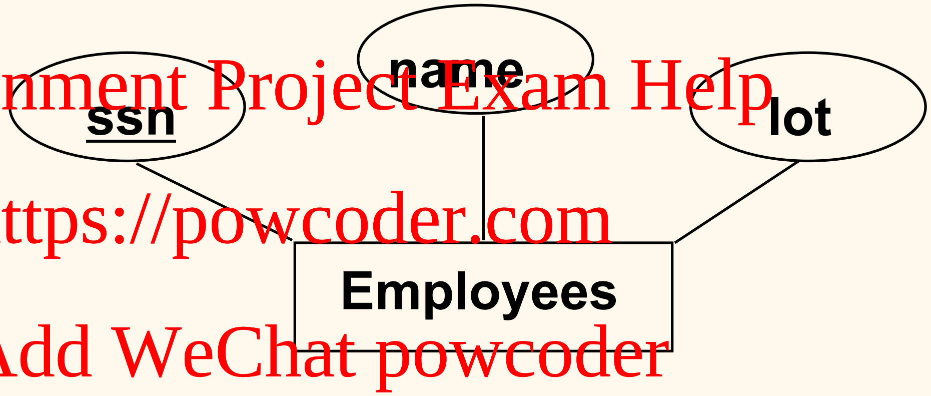


Diagram rule:

Entity set: Box
Attribute: “bubble”
Primary key: underlined

Keys of Entity Sets

- A *superkey* of an entity set is a (sub)set of the attributes such that no two entities in the set is *allowed* to have the same values on all these (key) attributes.

- Allowed?
  - *Designer's choice!*

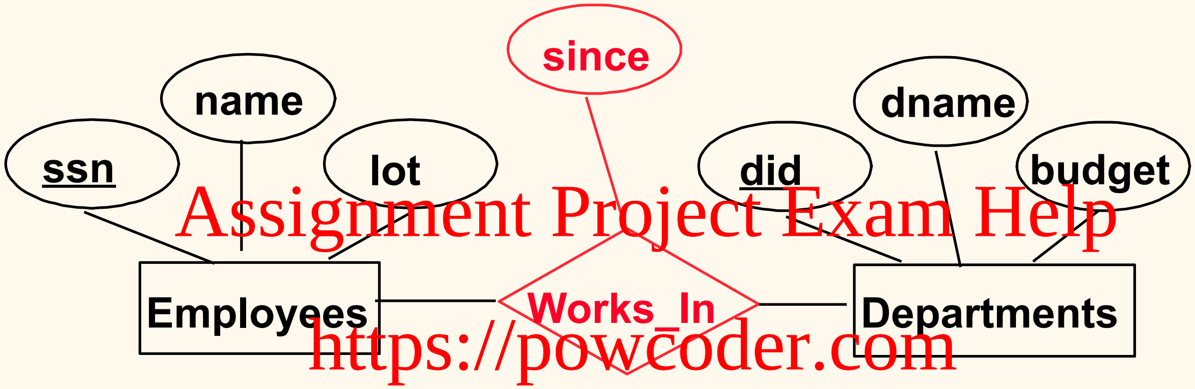
- *Candidate key* = A superkey that does not have a "redundant" attribute, i.e., if any attribute is removed, the set is not a superkey anymore.

- *Primary key* = One of the candidate keys *designated* to be so.
  - *Designated? By whom?*
- Every entity set must have a key.

ER Model Basics (Contd.)

- *Relationship*: Association among two or more entities. E.g., Gandalf *works* in the Pharmacy department.
- *Relationship Set*: Collection of *similar* relationships.
- Similarity: is in terms of entity sets where the entities are from.
- E.g.: A person (from employees entity set) *works* in a department (from *Departments* entity set).
- An *n-ary* relationship set R relates *n* entity sets E_1 ... E_n; each relationship in R involves entities e_1 *in* E_1,...,e_n *in* E_n
- Same entity set could participate in different relationship sets, or in different "roles" in same set.

Relationship Set Example

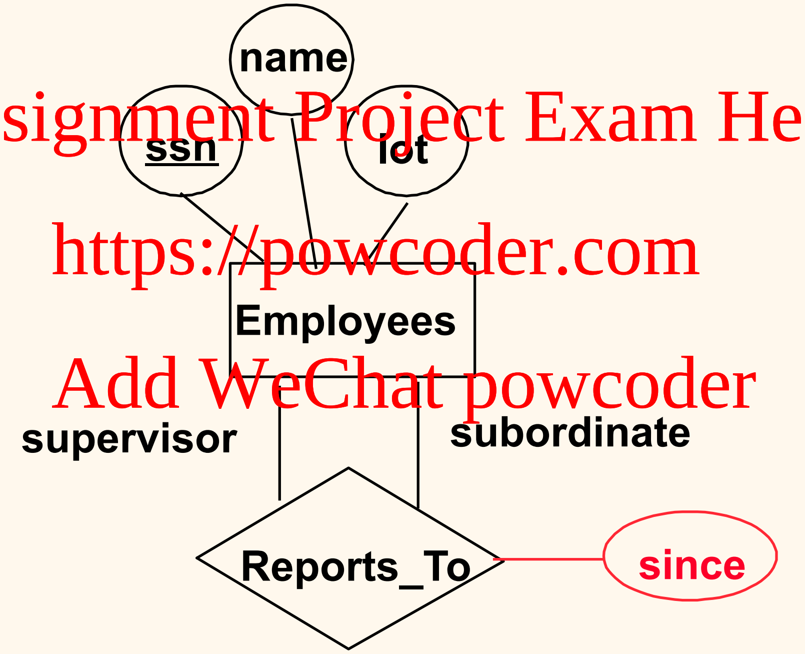


Relationship set: Works_In

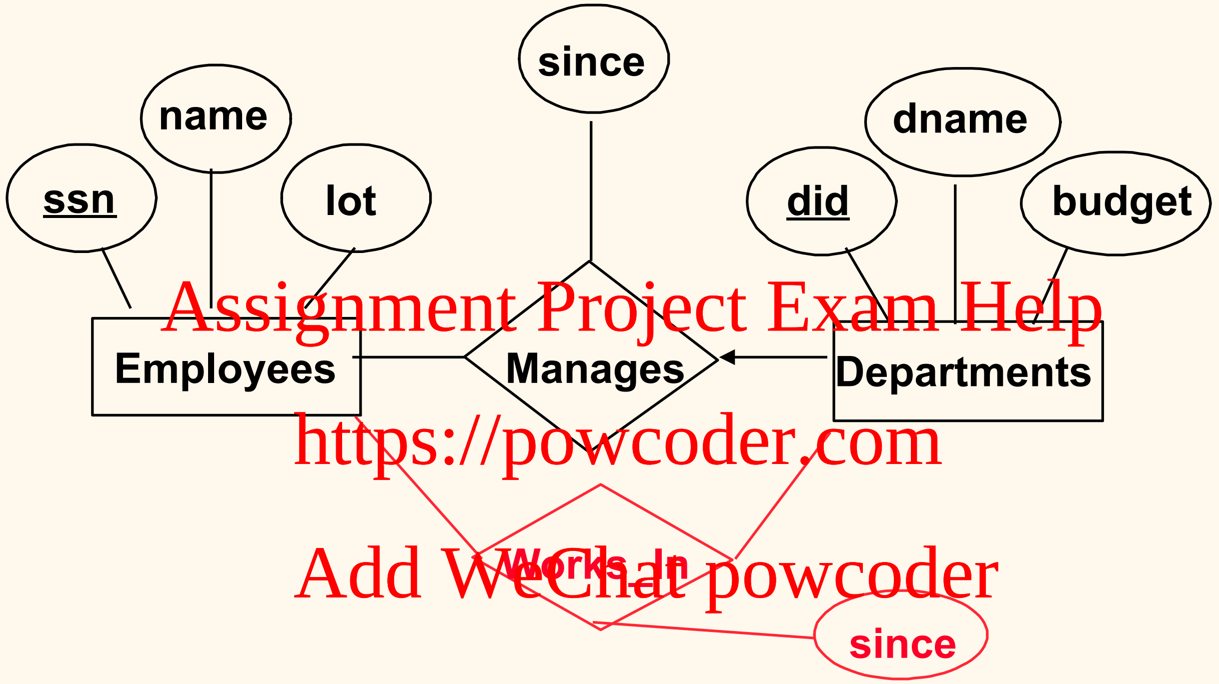
Descriptive Attributes

- Relationships can have attributes
- These attributes are called “descriptive” attributes, because they only “describe” relationships, but do not “distinguish” relationships.
- A relationship can only be distinguished by the participating entities.
- Therefore, there can’t be more than one relationship involving the same entities.

Another Relationship Set

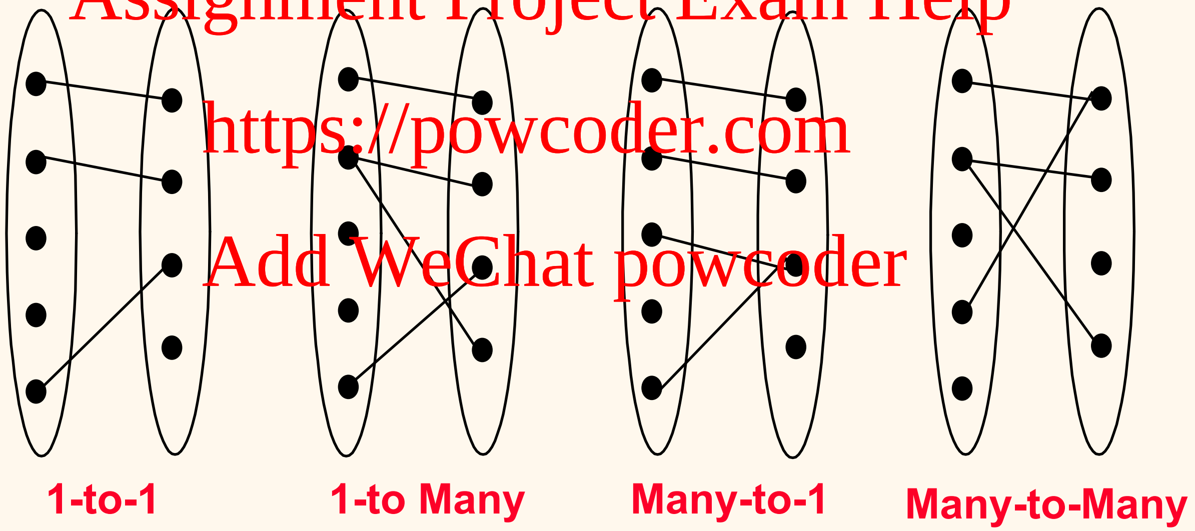


Key Constraints



- Consider Works_In: An employee can work in many departments; a dept can have many employees.
- In contrast, each dept has at most one manager, according to the *key constraint* on Manages.

Types of Binary Relationship Sets

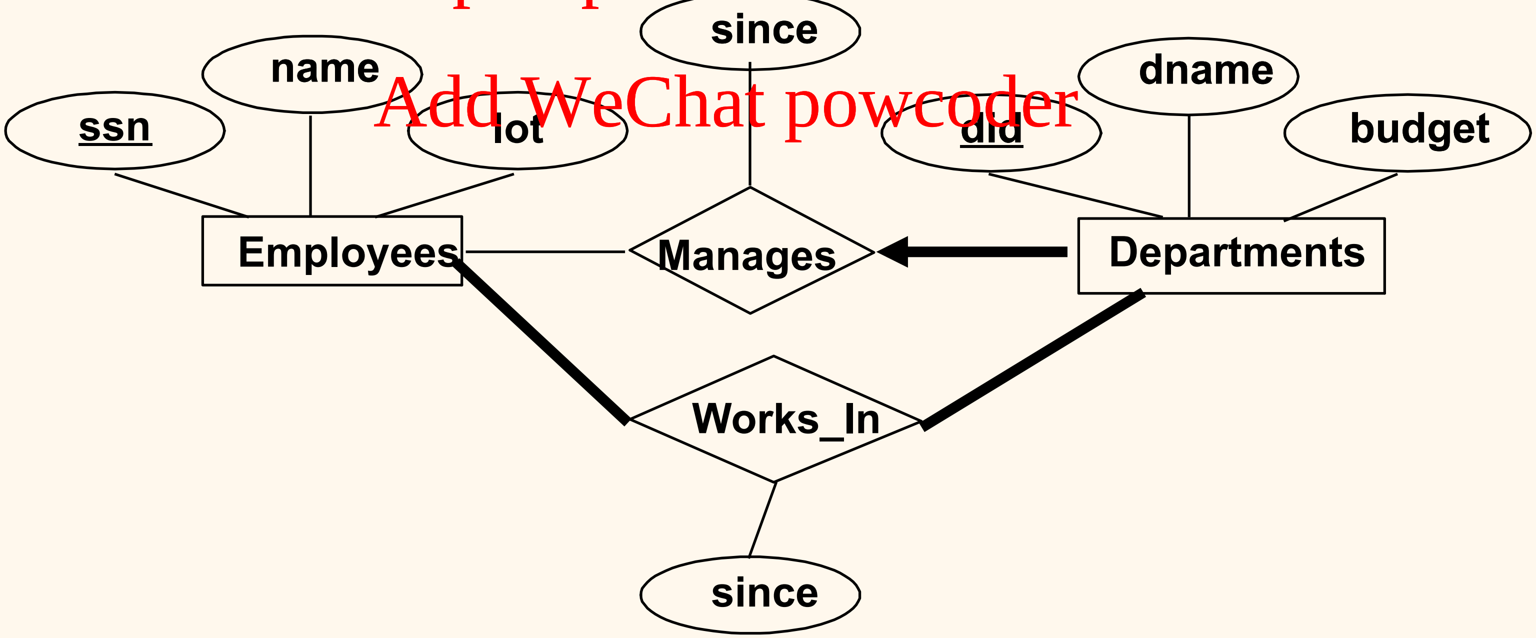


Key Constraint

- An entity set may participate in a relationship set as a "key" participant.
- What it means is that each entity of the "key" entity set can only participate *at most once* in the relationship set.
- More than one relationship set can be key participant (e.g. one-to-one relationship set).

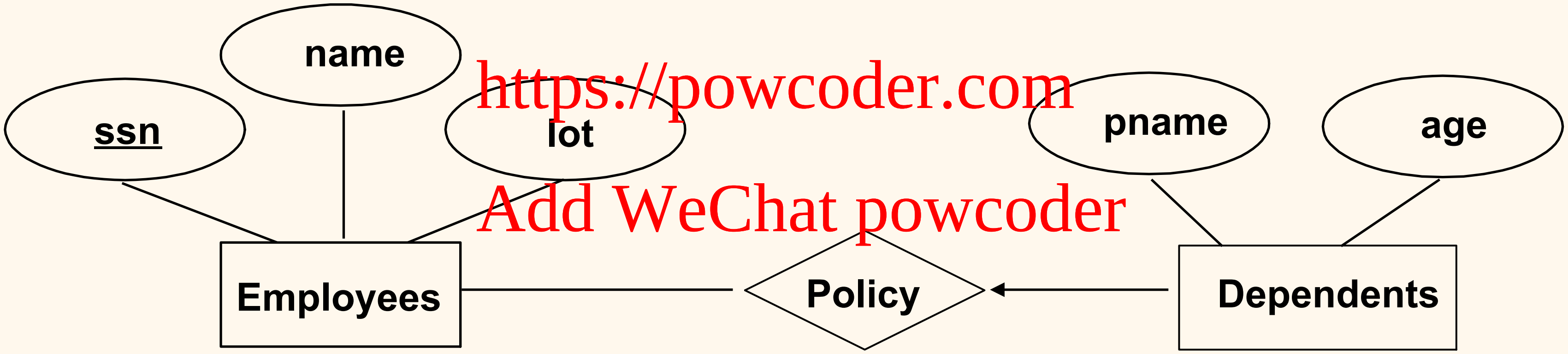
Participation Constraints

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



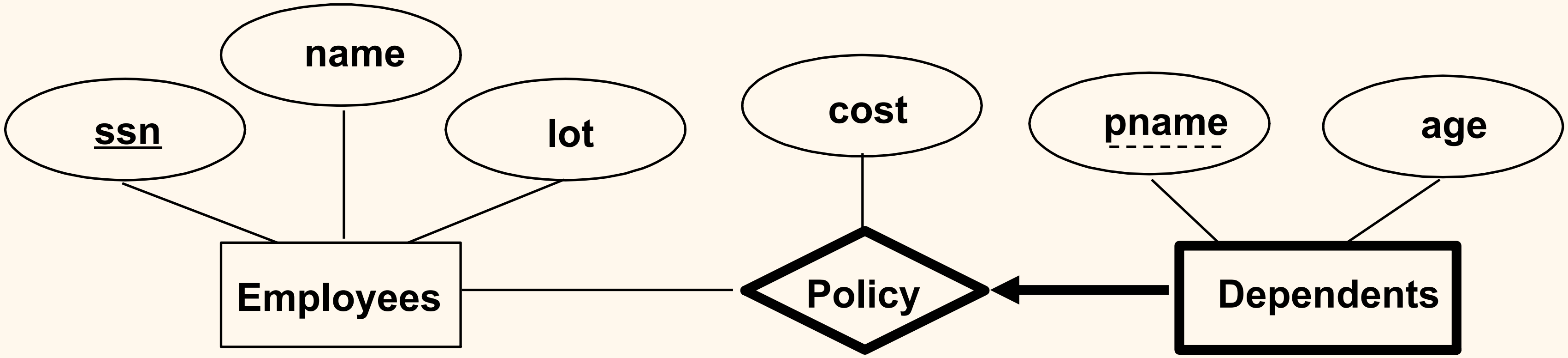
Weak Entities

- Consider the following situation:

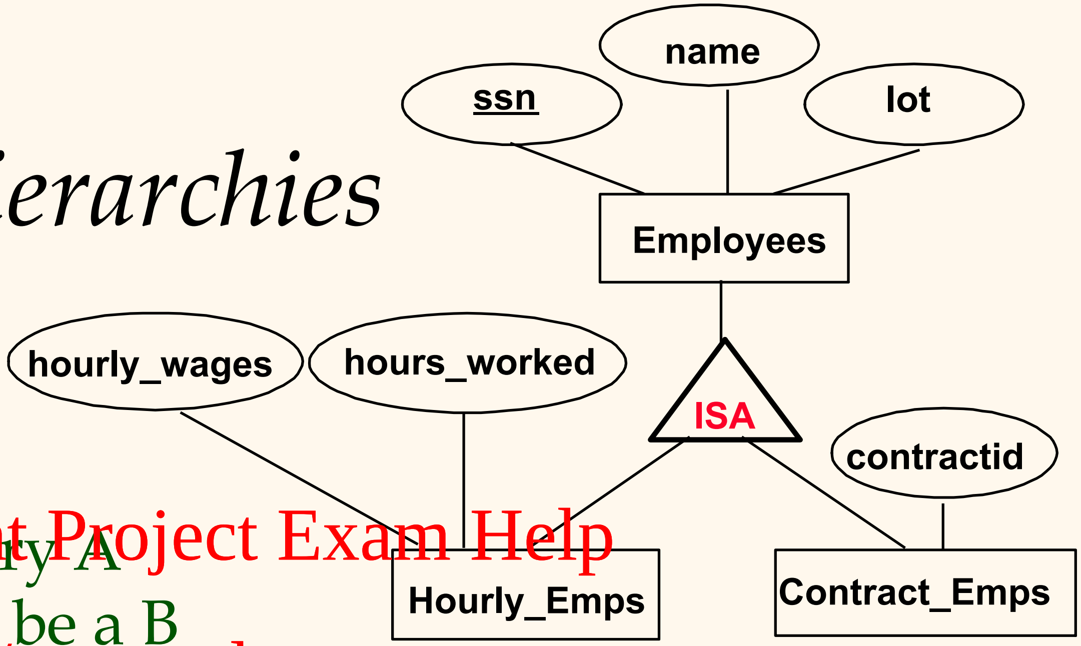


Weak Entity Sets

- 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

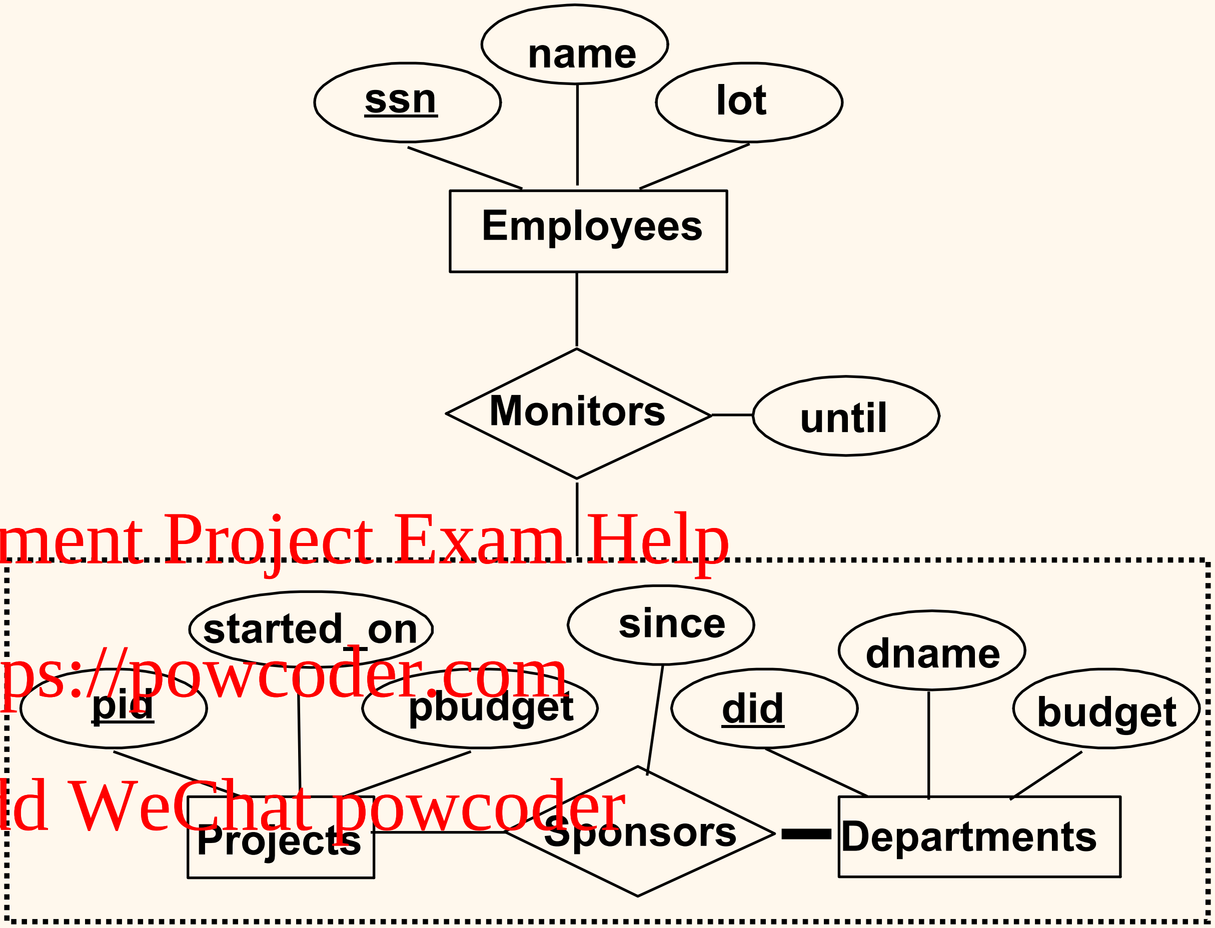


- As in C++, or other PLs, attributes are inherited.
- If we declare A ISA B, every A entity is also considered to be a B entity.

- *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 (entity 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 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

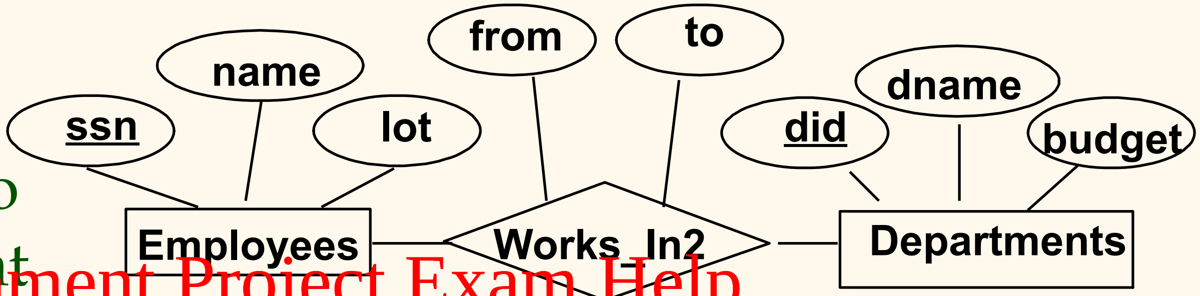
- Design choices:
  - Should a concept be modeled as an entity or an attribute?
  - Should a concept be modeled as an entity or a relationship?
  - Identifying relationships: Binary or ternary? Aggregation?
- Constraints in the ER Model:
  - A lot of data semantics can (and should) be captured.
  - But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

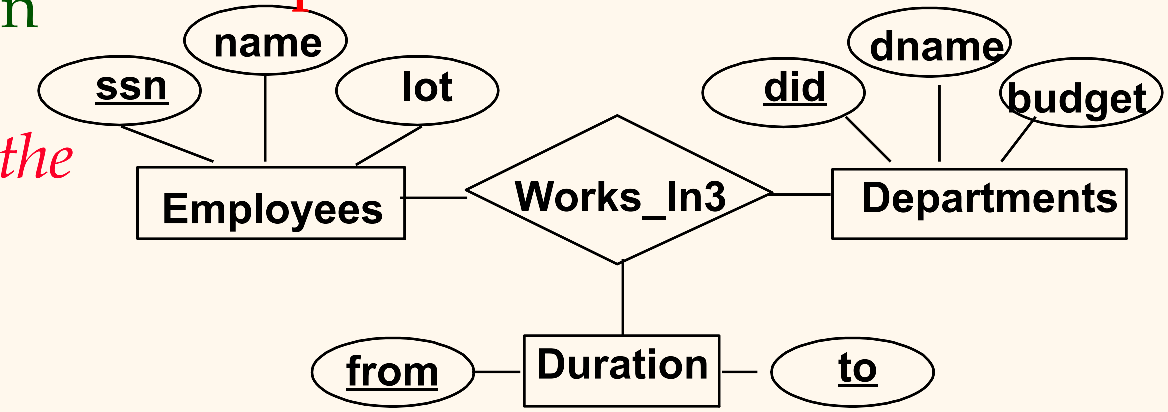
Entity vs. Attribute

- 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.)

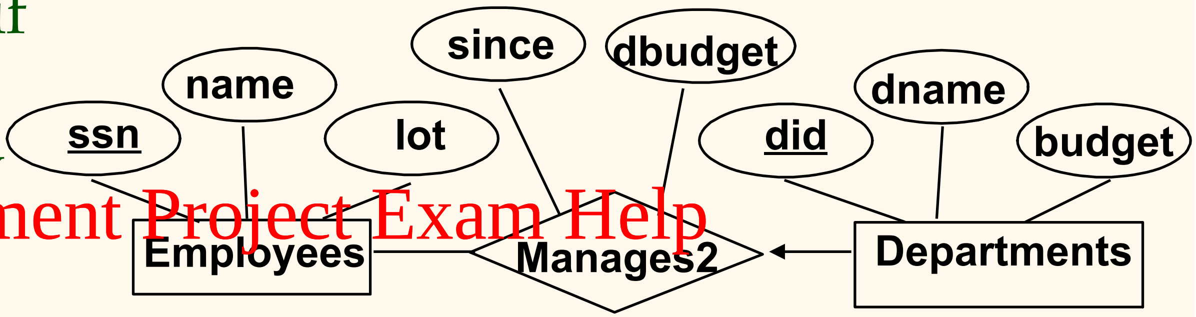
- Works_In2 does not allow an employee to work in a department for two or more periods.
- Similar to the problem of wanting to record several addresses for an employee: we want to record *several values of the descriptive attributes for each instance of this relationship.*





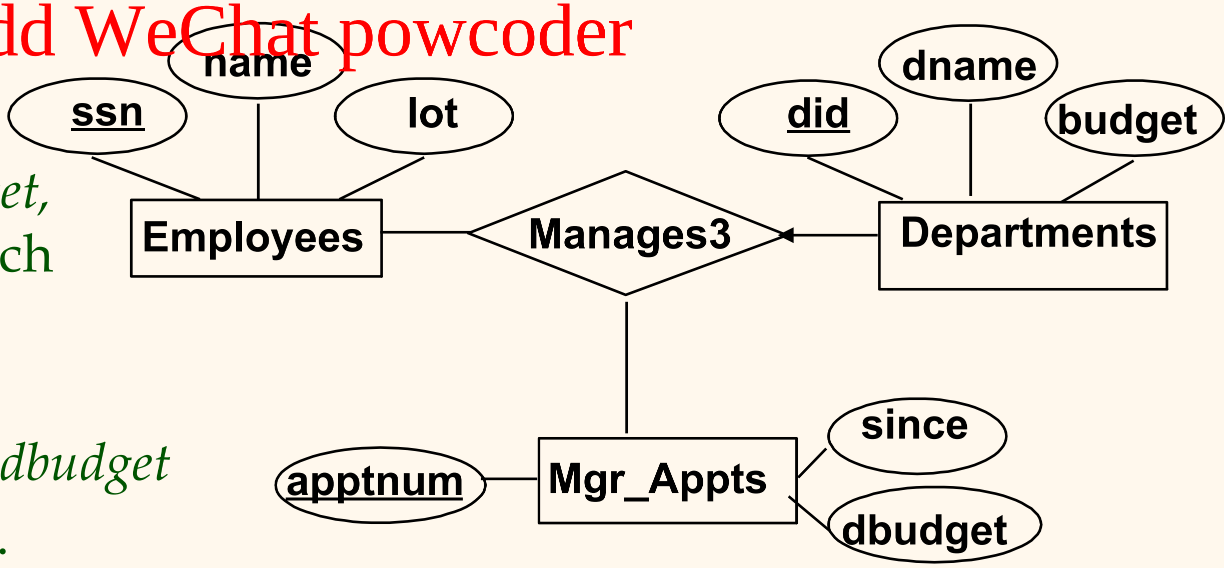
Entity vs. Relationship

- First ER diagram OK if a manager gets a separate discretionary budget for each dept.



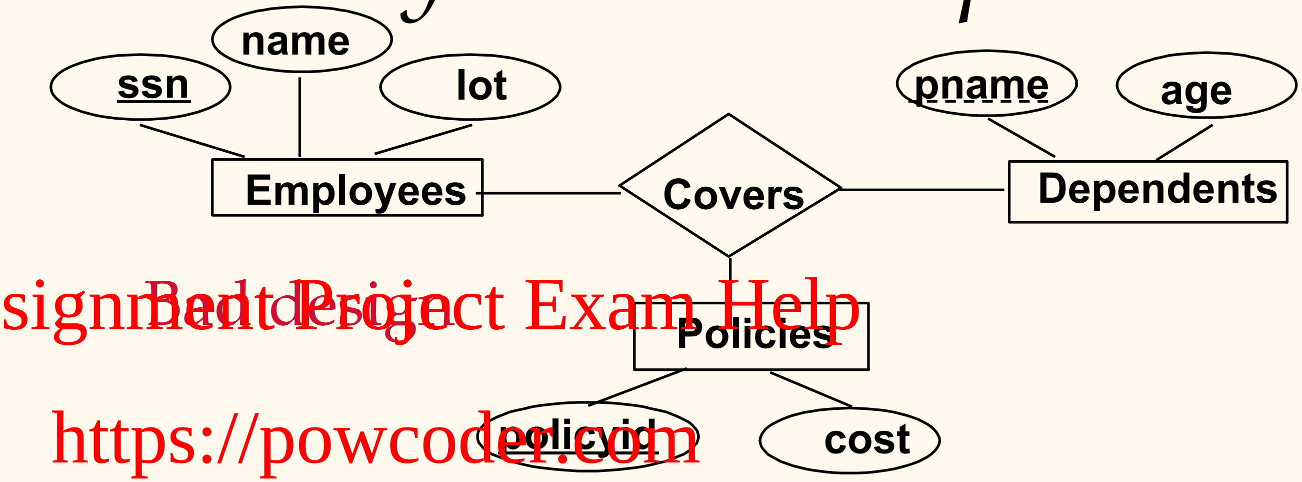
- What if a manager gets a discretionary budget that covers *all* managed depts?
  - Redundancy of *dbudget*, which is stored for each dept managed by the manager.

    Misleading: suggests *dbudget* tied to managed dept.

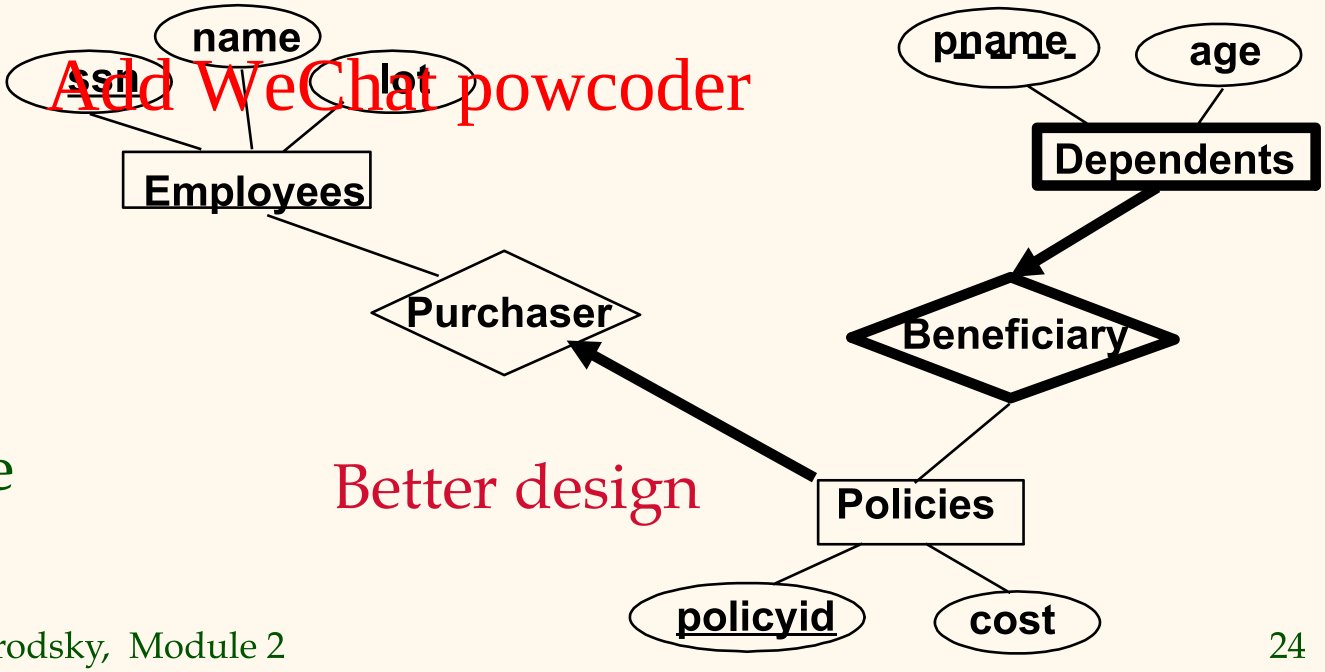


Binary vs. Ternary Relationships

- If each policy is owned by just 1 employee:
  - Key constraint on Policies would mean policy can only cover 1 dependent!
- What are the additional constraints in the 2nd diagram?



Bad design



Better design

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.