# Data model Entity-Relationship

Iztok Savnik, FAMNIT

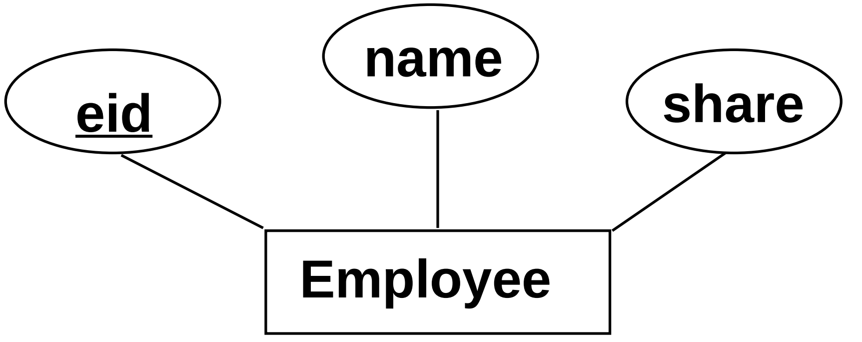
# Slides & Textbook

- Textbook:
  - Raghu Ramakrishnan, Johannes Gehrke, *Database Management Systems, McGraw-Hill, 3rd ed., 2007.*
- *Slides:*
  - *From „Cow Book“: R.Ramakrishnan, http://pages.cs.wisc.edu/~dbbook/*

# Overview of Database Design

- *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.

# Entities



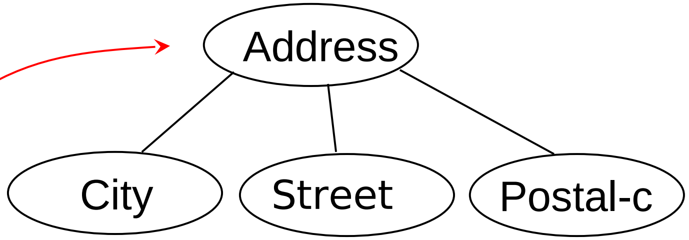

- *Entity:* object from the real world that can be separated from other objects
- Entity is represented in the database by a set of attributes (property).

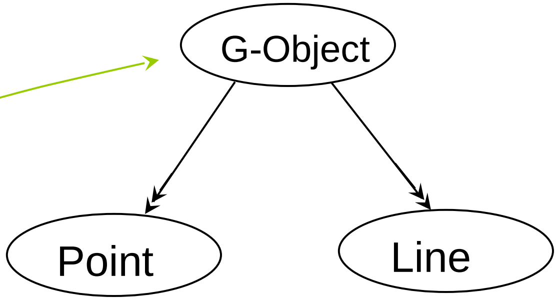
# Entities

- Entity sets: E, E1, E2
  - $E=\{e_1, e_2, \ldots, e_n\}$; $e_1, e_2, \ldots, e_n$ are entities
- Every entity has an identifier
  - Identifier is a set of attributes that uniquely identifies entity inside the entity set
    - Candidate keys uniquely identify n-tuple in a relation
    - We have more identifiers of entities from an entity set
  - Primary identifier is the decision of the designer
- Other attributes that are not part of the primary idetifier are called description attributes.

# Attributes

- Properties of entities are represented as attributes.
  - Attributed is one data element that describes a property
- Every attribute has range type:
  - Defines permitted values of an attribute
  - Atomic attributes: range type is a simple type as integer, string, etc.
  - Complex attributes: values are composed from simple values tkat can be of heterogenous types
  - Multi-valued attributes: values are sets of elements of the same type

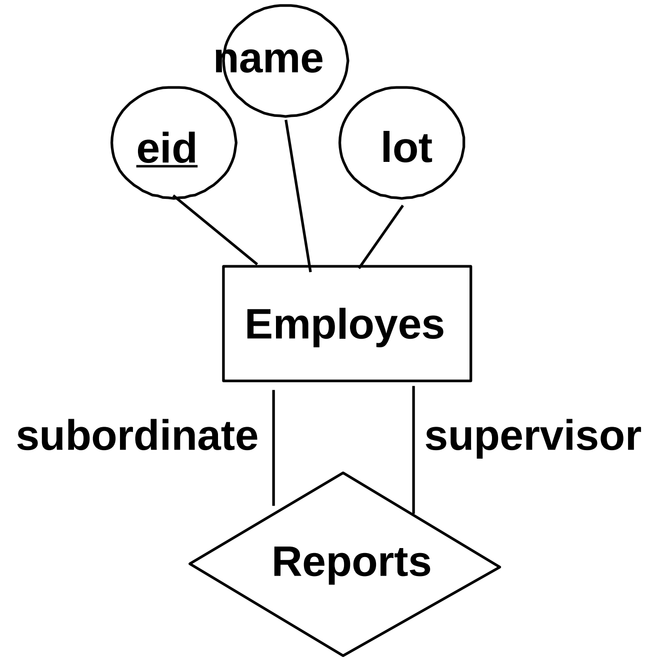


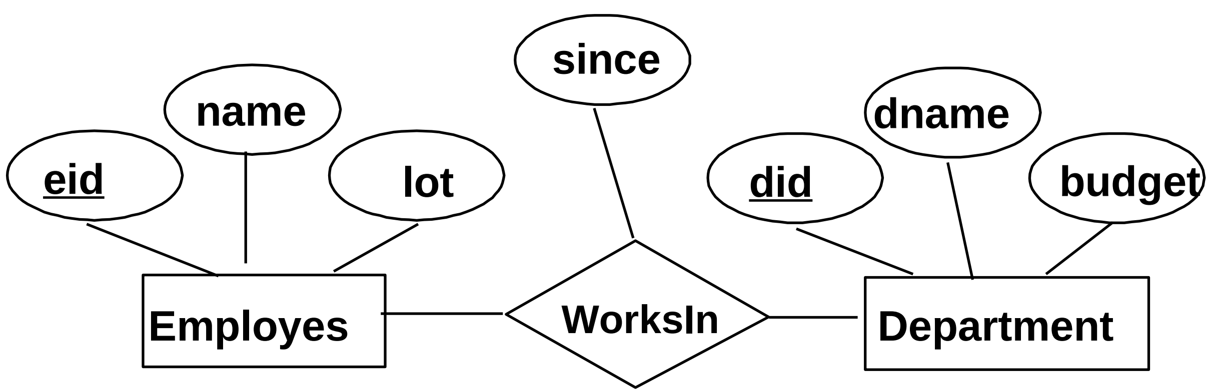



# Attributes

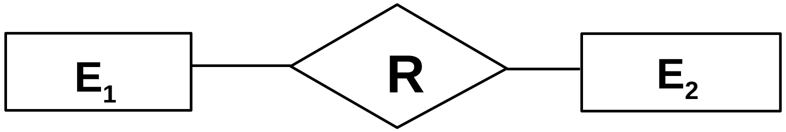
- Complex and multi-valued attributes allow for abstract representation of a property
  - Properties that would have to be represented by using several atomic attributes or by using a relationship, can be represented by a single concept
- Many practical (design) tools allow the use of complex attributes
  - Extended Entity-Relationship Model
  - SQL3 !

# Example ER schema



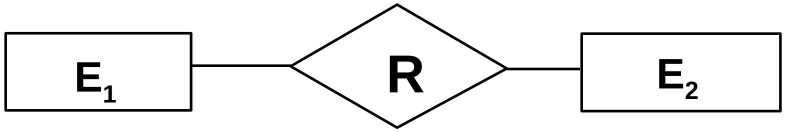



# Relationships



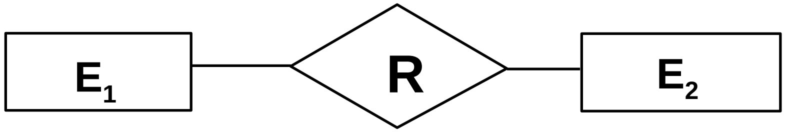

- Relationship defines the link between two or more entities that can belong to different entity sets
  - For instance,relationship Owner defines the link between the entity sets Customer and Account.
  - The concrete relationship can be described as a pair (s, r) where s is from the entity set Customer, and r the element of the entity set Account
- In general, the relationship is a n-tuple $(e_1, e_2, \ldots, e_n)$, where $e_i$ are entities

# Interpretation



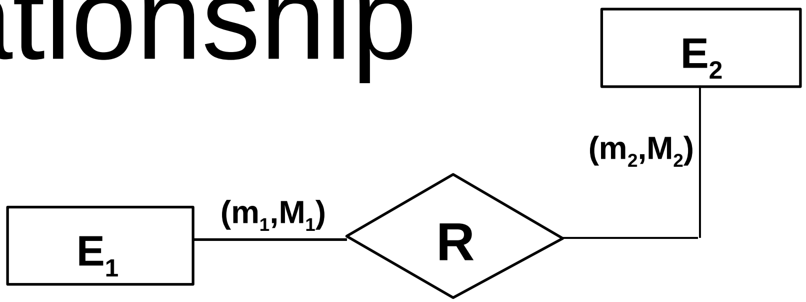

- Relationship set
  - Relationships are classified into sets that contain similar relationships
  - $R=\{(e_1,\ldots,e_n) \mid e_1 \in E_1, \ldots, e_n \in E_n\}$, where $E_i$ are entity sets
  - $R \subseteq E_1 \times \ldots \times E_n$
- The number of entities bound by a relationship is called degree
- Relationship betwen two entities is called binary relationship

# Relationships




- Entity set is represented graphically with a diamind connected with the entities in relationship.
  – Line between entity and relationship can be named.
  – The name represents the role of the entity in the relationship.
- Binary relationship that links two entities from the same entity set, is called recursive.
  – In recursive relationship we want to discern between two different roles of the relationship.
  – The following example presents a recursive relationship.

# Cardinality of a relationship




Relationship can be described more precisely with cardinality

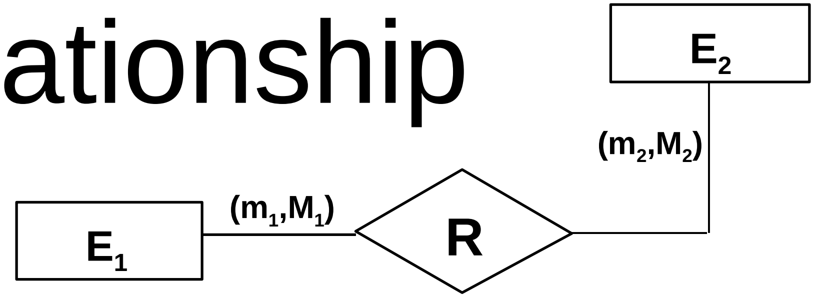
- Relationship has mapping constraints
- Cardinality of a mapping between the entity sets

Let $E_1, E_2, \ldots, E_n$ be entity sets connected with a relationship R

- Cardinality is defined for each particulr entity set taking part in the relartionship R.

Cardinality of an entity set $E_i$ in relationship R describes in how many different relationships an entity from R can paticipate

# Cardinality of a relationship



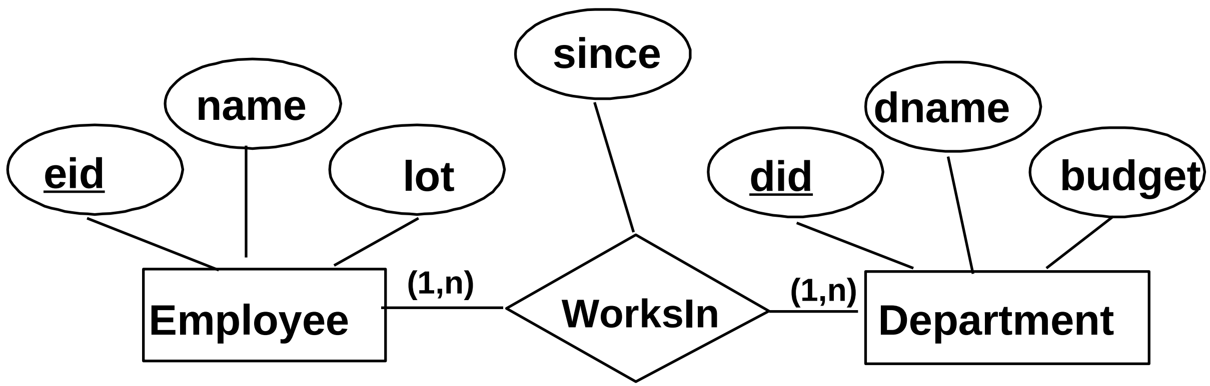

Cardinality of the entity set $E_i$ in the relationship R is a function:

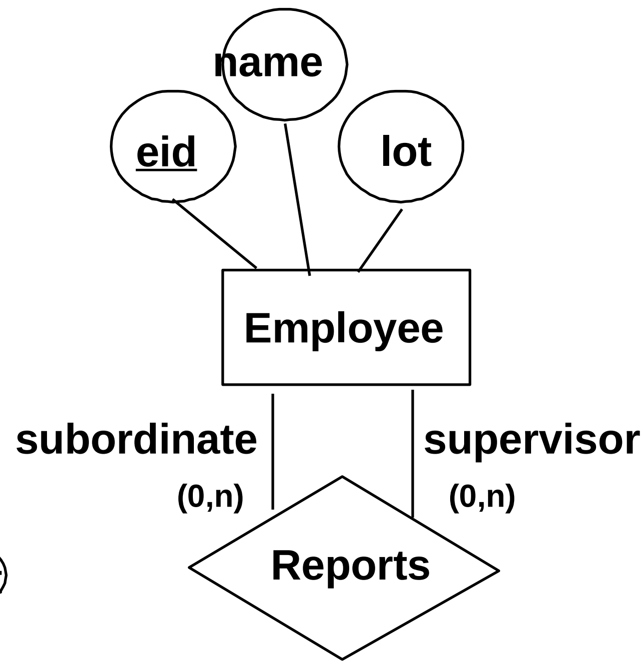
- card($E_i$,R) = (min, max)
- min is minimal cardinality of $E_i$ in relationship R
- max is maximal cardinality $E_i$ in R

Possible values of minimal and maximal cardinality:

- "0" (zero), "1" (one) "N" ("N" reads "many"; in general, "more than one").
- min-card(E,R) in max-card(E,R)

# Example




- Relationship role
  - Domain can be the same entity set

# Max-card() types of relationships

The classification of relationships is based on the maximal cardinality of entities in the relationship

Types of the entity set E roles in the relationship R:

- max-card(E,R) = 1 - E has single-valued role in the relationship R
- max-card(E,R) = N - E has multi-valued role in R

Binary relationship R between the entity sets E and F is denoted:

- N-N -- many-many – if E and F have multi-valued role in R
- 1-1 -- one-one -- if E and F have single-valued role in R
- 1-N (N-1) – one-many – if one entity set has single-valued and the other has multi-valued role in R

The classification into the relationship types “1-1”, “1-N” and “N-N” is based solely on max-card()!

# Min-card() types of relationships

Minimal cardinality serves as the second type of relationship classification: participation constraint

min-card(E,R) = 1

Each entity from the set E appears in at least one relationship instance of R

Entities from E are mandatory in the relationship R.

The function of R is total.

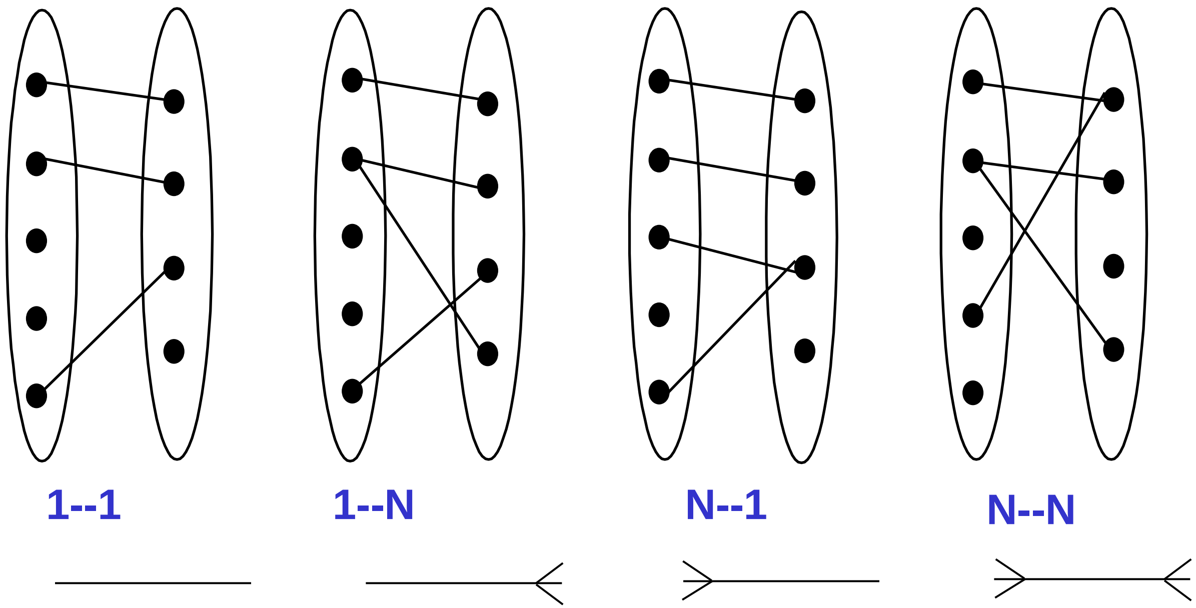
min-card(E,R) = 0

Some entities from E are not part of any relationship from R.

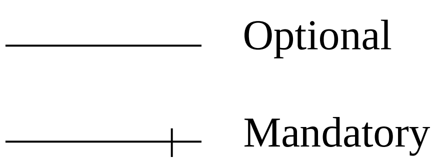
Entities from E are optional in the relationship R.

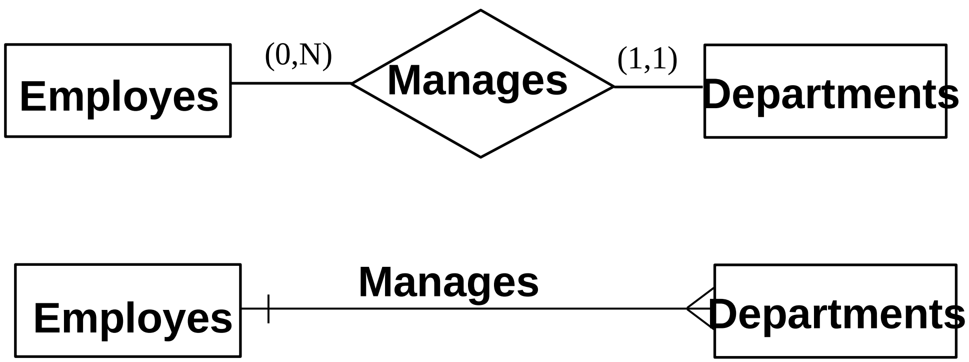
The function of R is partial.

# Types of binary relationships

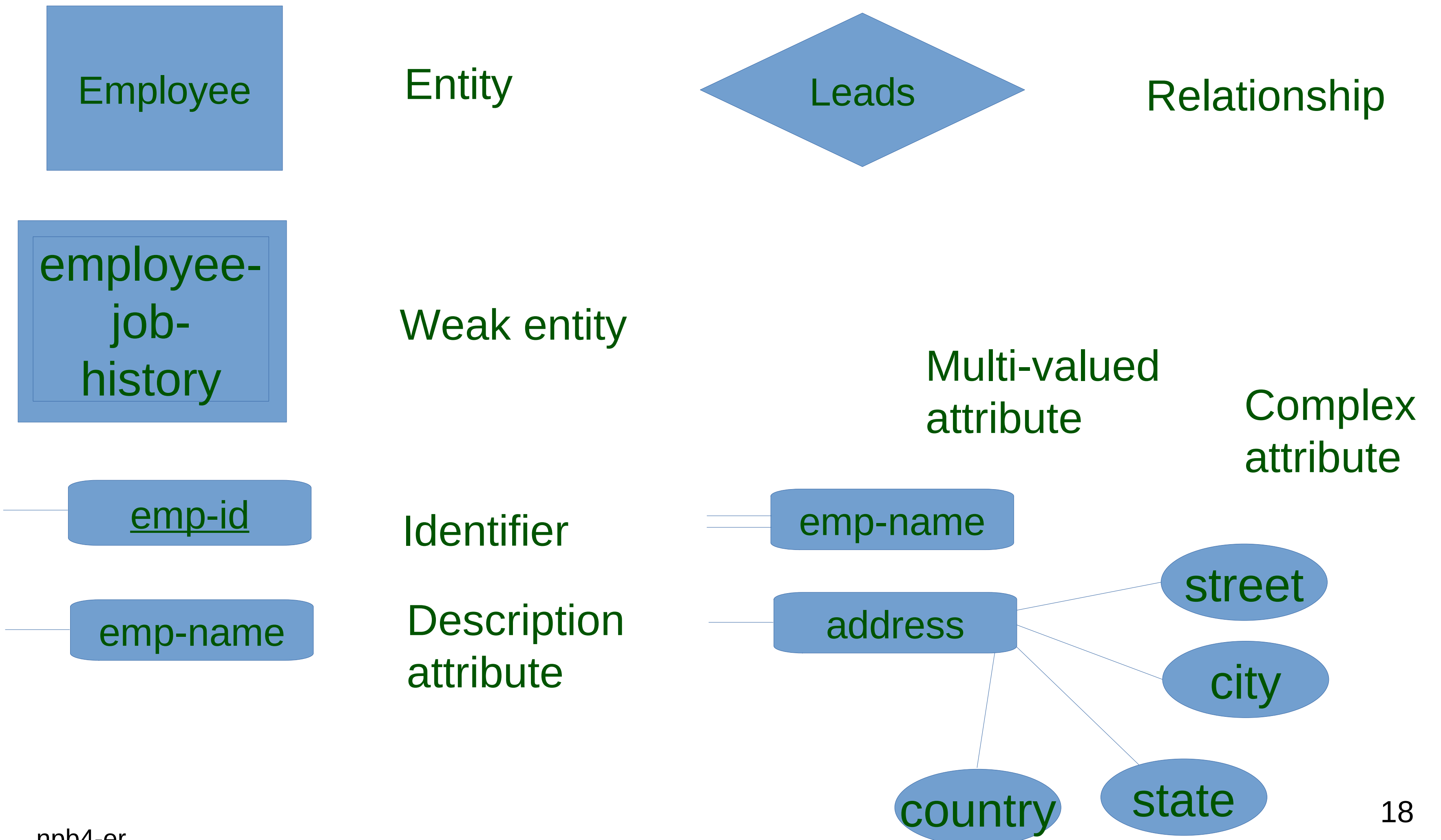



Crow's foot notation

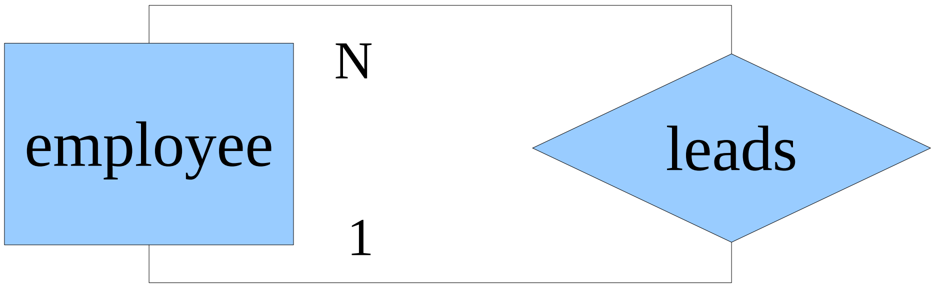


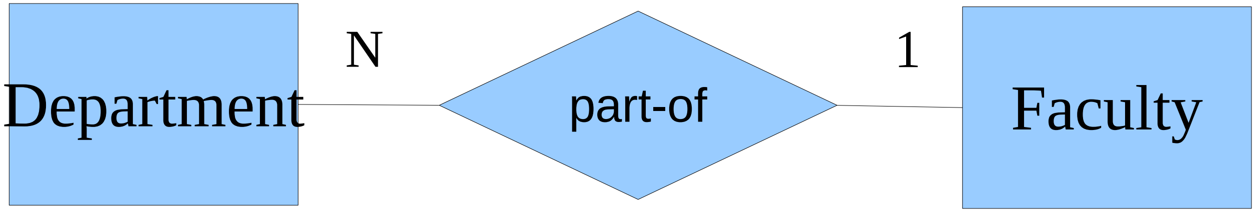
# Chen's notation



# Degree of relationship

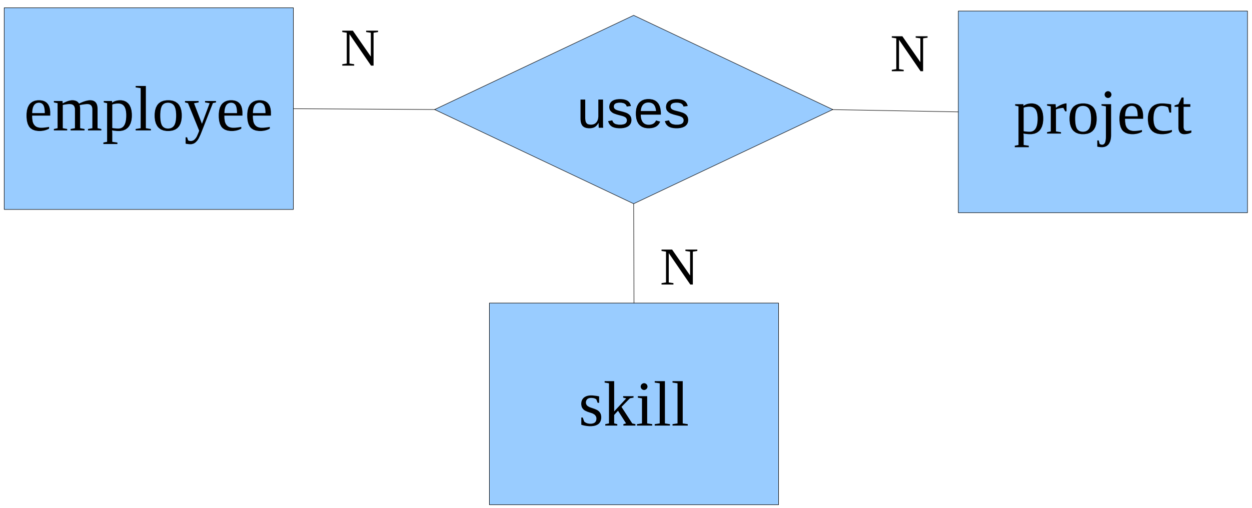



Recursive
binary relationship



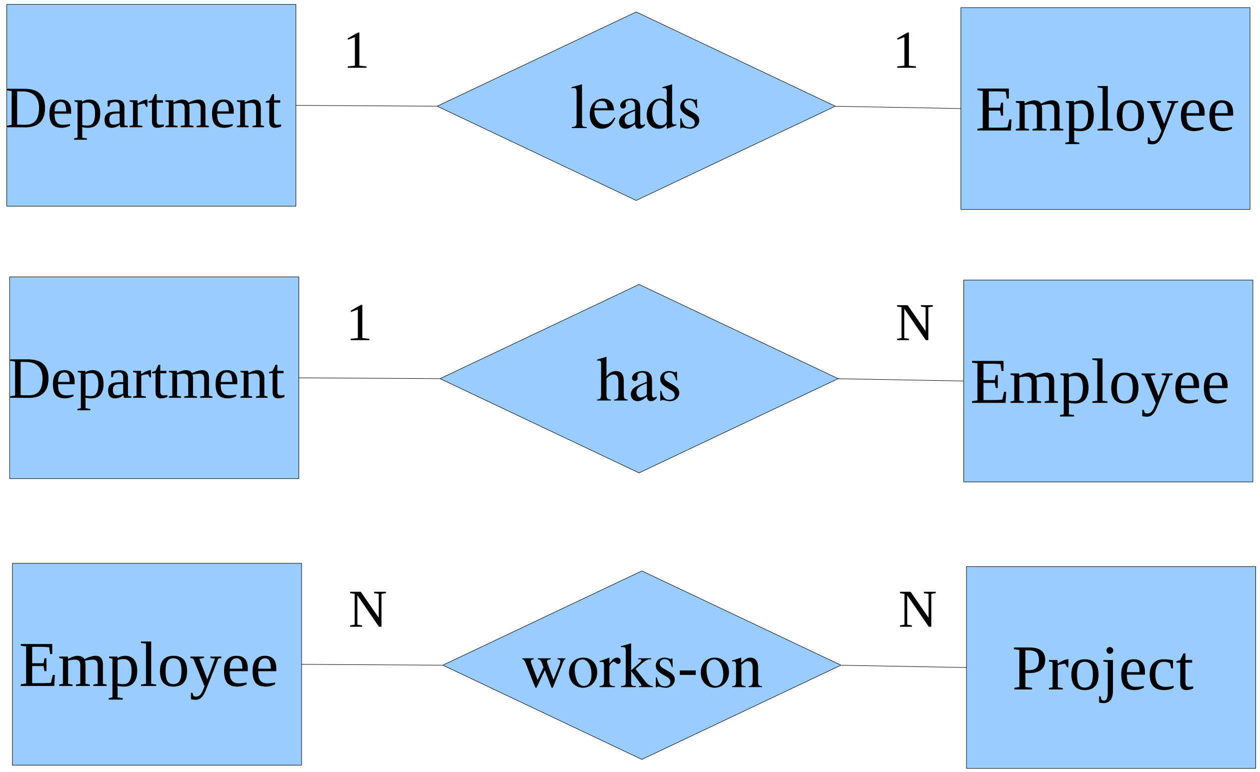

Binary relationship

# Degree of relationship




Ternary relationship

# Cardinality of relationship



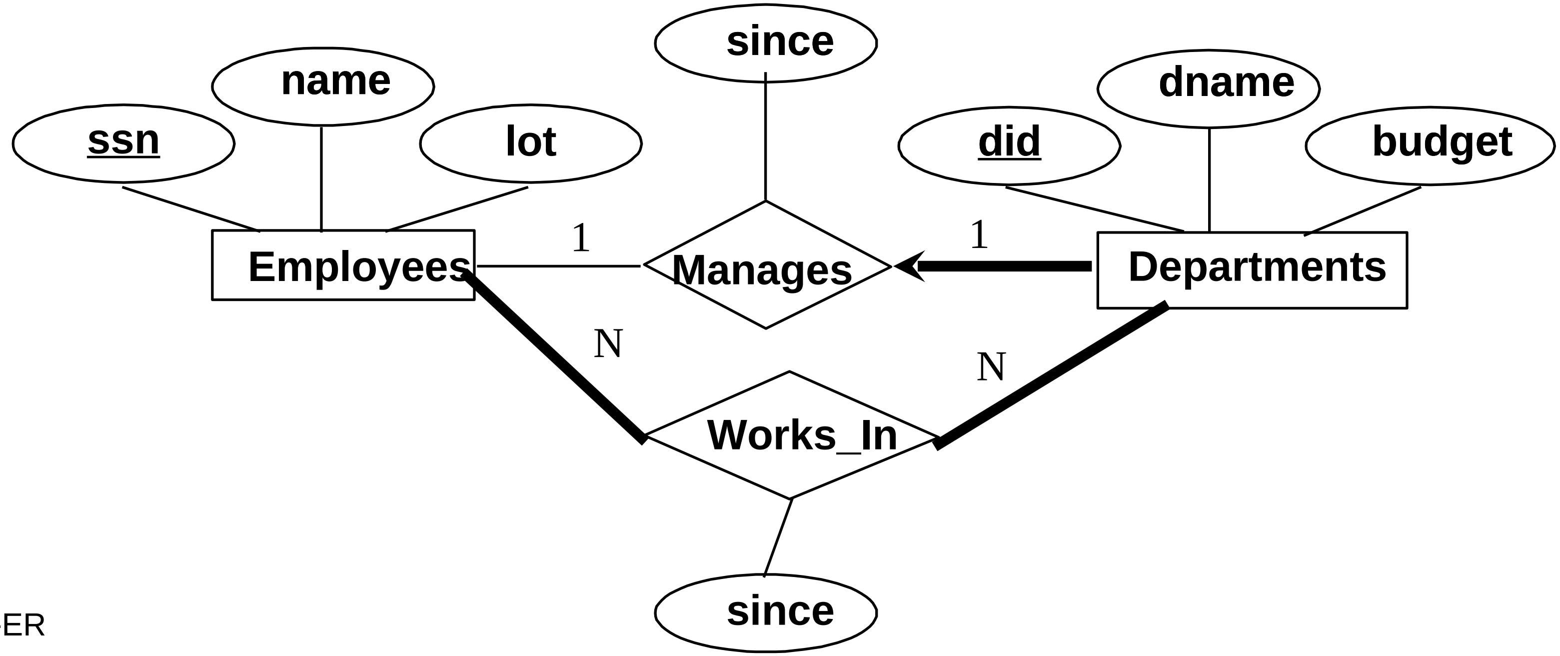

1-1

1-N

N-N

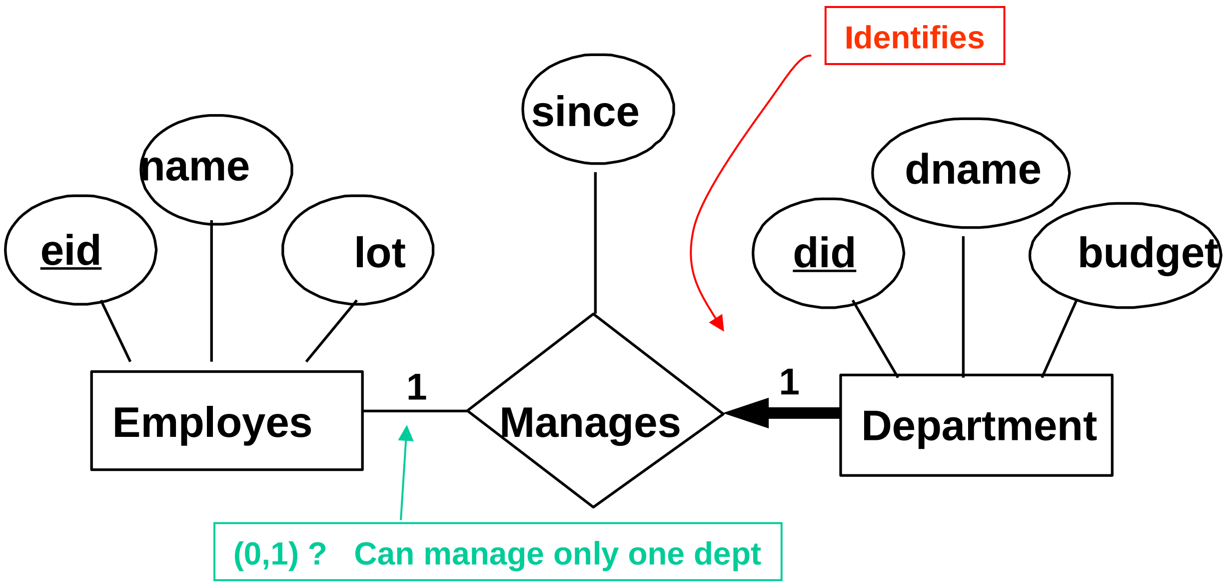
# Participation Constraints

- 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 Departments entity must appear in an instance of the Manages relationship.



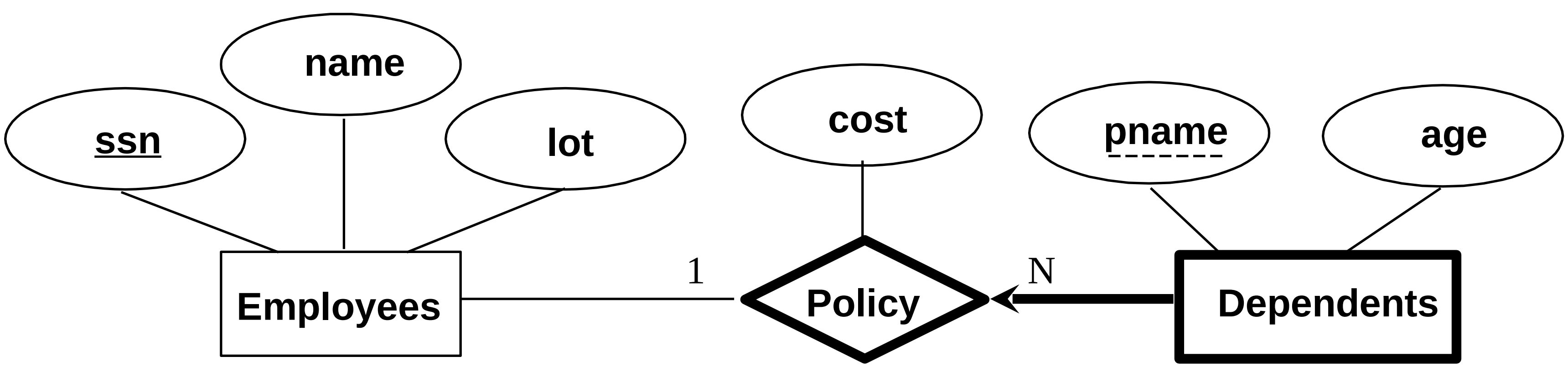
# Key (identifier) constraint

- Relationship Works_*In*:
  - Employee can work in more than one department
  - Department can have many employees
  - Keys?
- It is different to the relationship Manages
  - Every department has at most one manager
  - Entity Department identifies relationship Manages.

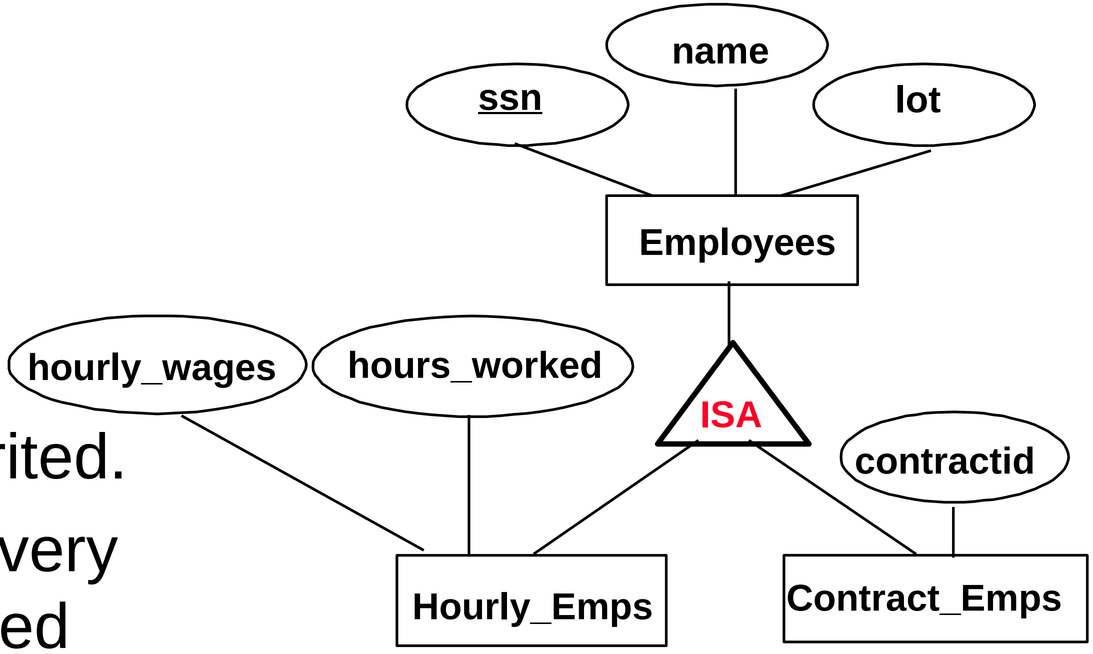


# Weak Entities

- 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.



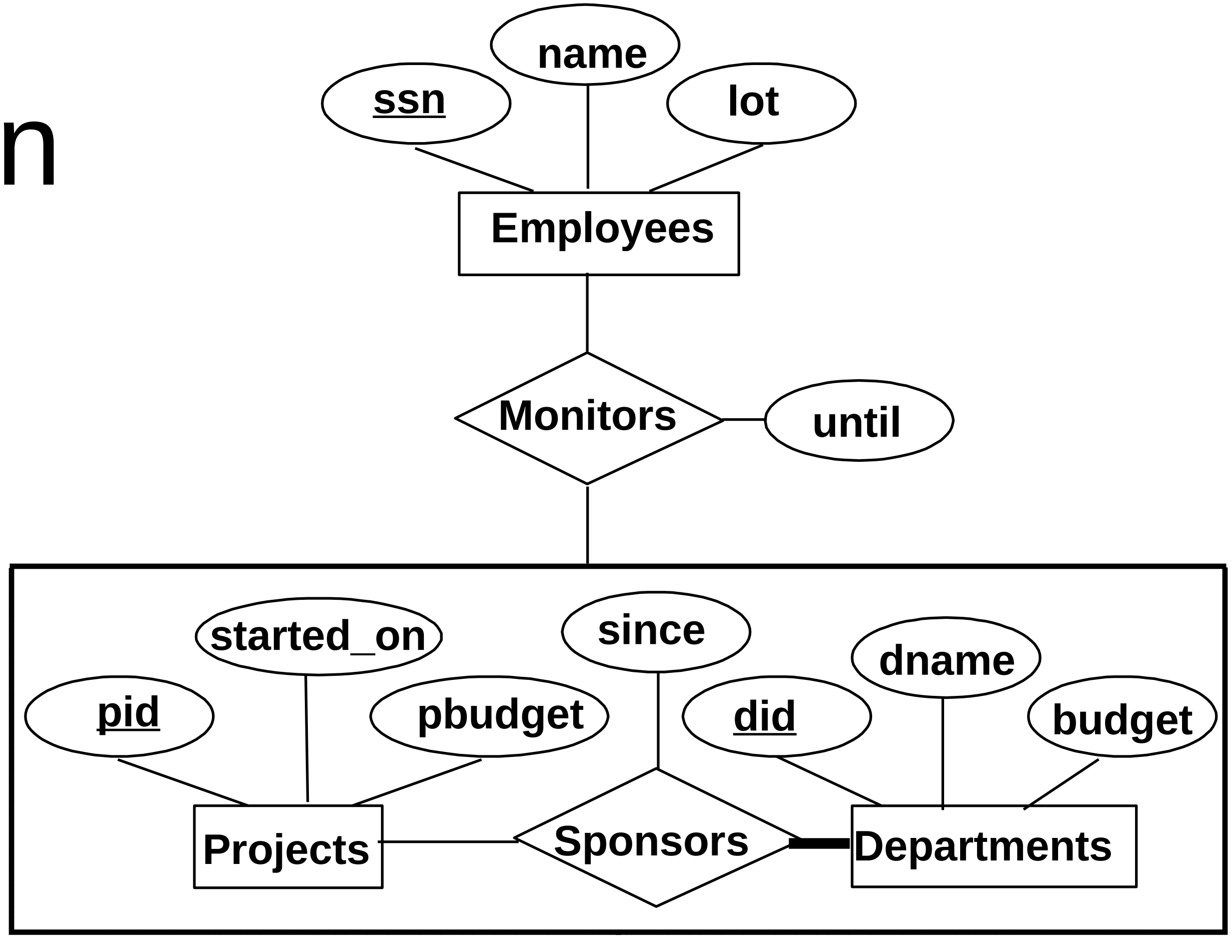
# ISA (`is a') Hierarchies




- As in C++, 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 (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 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.

# 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.