Database Management Systems

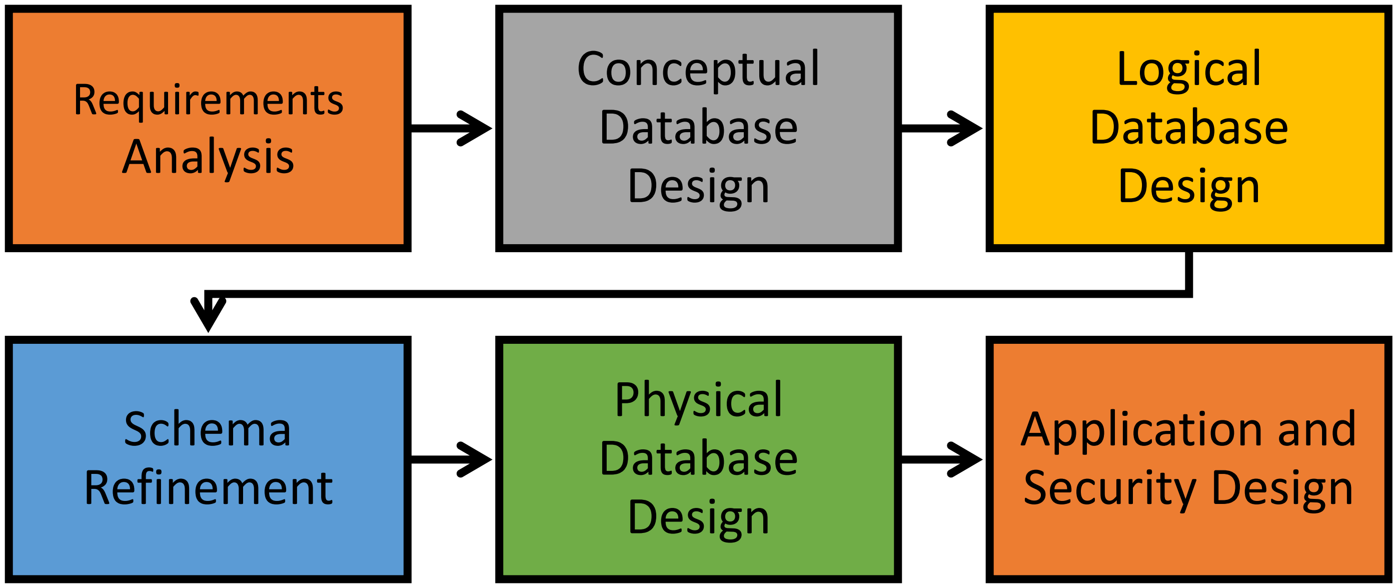
Ehsan Noei
e.noei@utoronto.ca

UNIVERSITY OF TORONTO

- Groups?
  - Send me an email (e.noei@utoronto.ca)
    - Your names and student#
    - Until Monday (May 13) 11:59pm
  - If not,
    - Random assignments

Overview of Database Design

- The database design process can be divided into six steps:



Requirements Analysis

- 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?

Conceptual Design

- The information gathered in the requirements analysis step is used to develop a high-level description of the data to be stored in the database, along with the constraints known to hold over this data.
- This step is often carried out using the **ER model**.

Logical Database Design

- Convert the conceptual database design into a database schema in the data model of the chosen DBMS.
  - Consider only relational DBMSs in this course.
  - Map an ER diagram into a relational schema.

Schema Refinement

- Normalization for desirable properties, eliminating redundancies from tables.

Physical Database Design

- Indexing, storage, etc. for performance

Application and Security Design

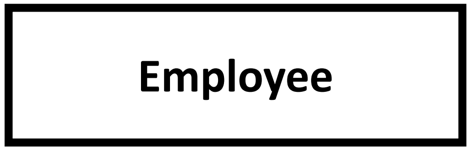
- We must identify the parts of the database that must be accessible and the parts of the database that must *not* be accessible.
- We must take steps to ensure that these access rules are enforced.

Entity-Relationship (ER) Data Model

- Allows us to describe the data involved in a real-world enterprise in terms of objects and their relationships.

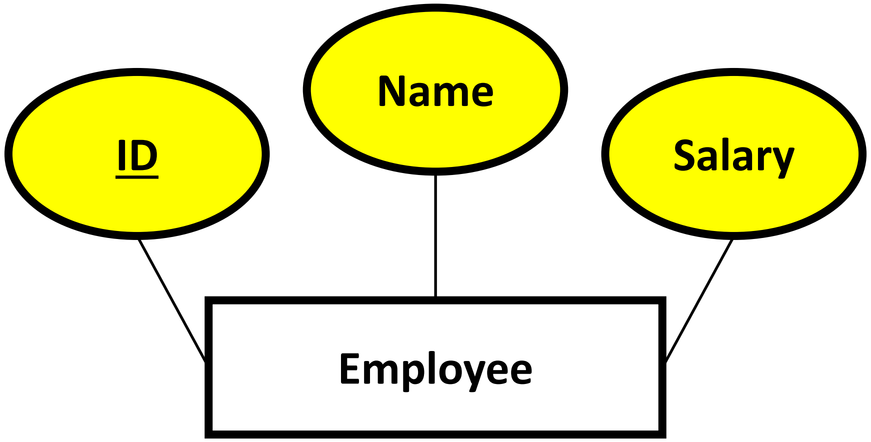
Entity

- Real-world object distinguishable from other objects. An entity is described (in DB) using a set of attributes
- e.g., employee, toy department, and manager of the toy department



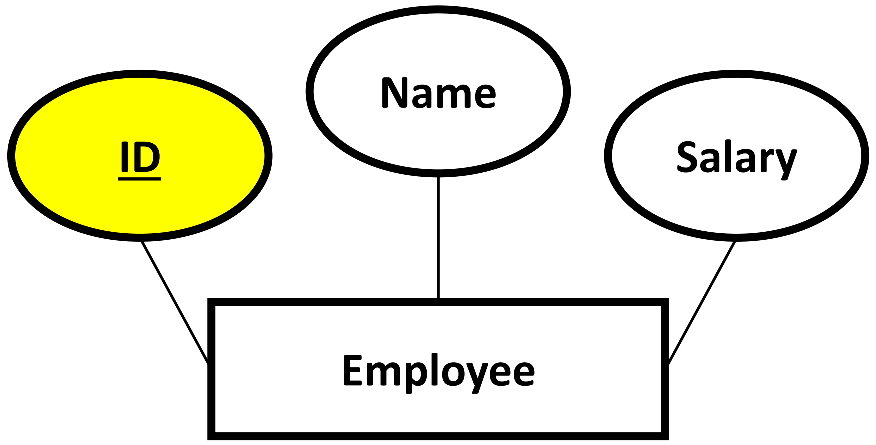
Attribute

- An entity is described (in DB) using a set of attributes
  - e.g., name, email, age, salary, position, etc.
- Domain is a set of possible values for an attribute.



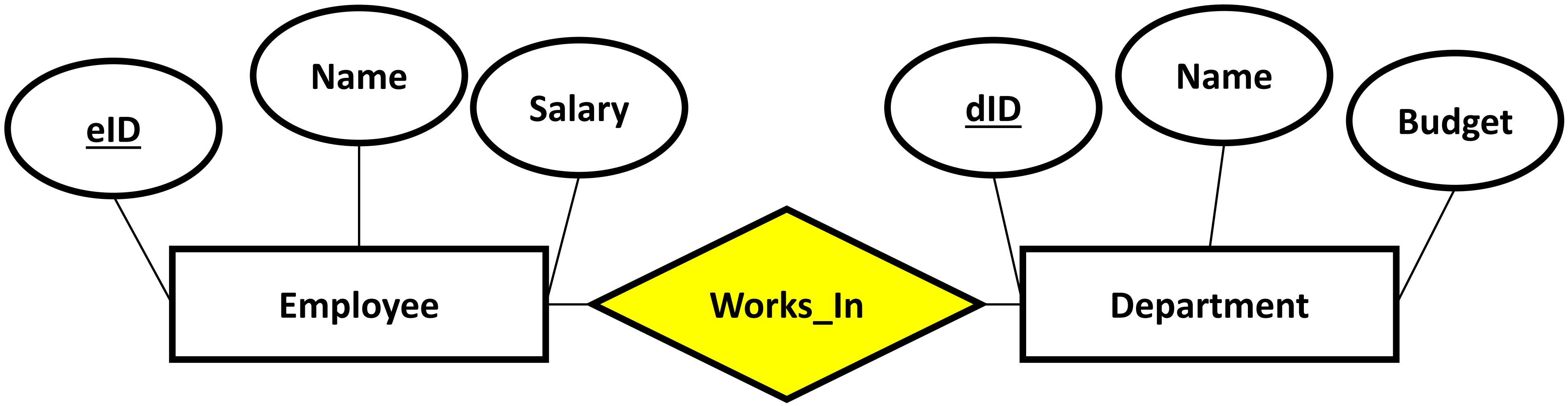
Key

- Minimal set of attributes whose values uniquely identify an entity
  - Many candidate key
    - Choose one of them as primary key
  - Assuming each entity set has a key



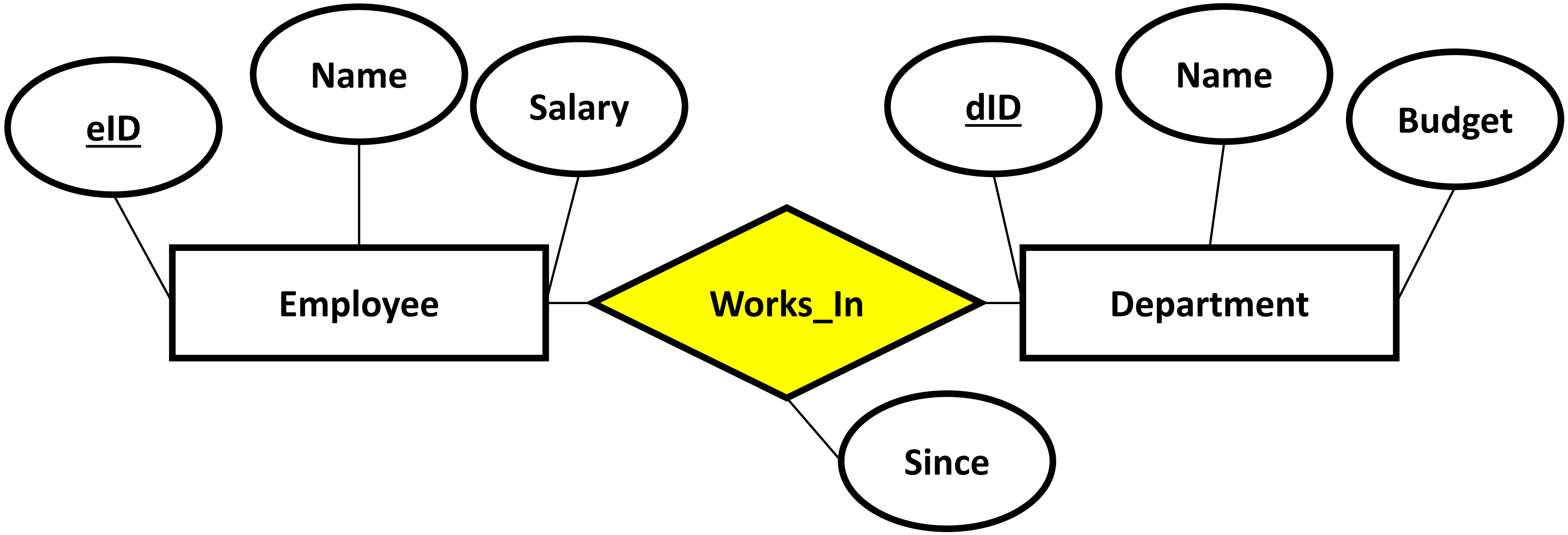
Relationship

- Association among two or more entities.
  - e.g., Steve works in the Toy Department

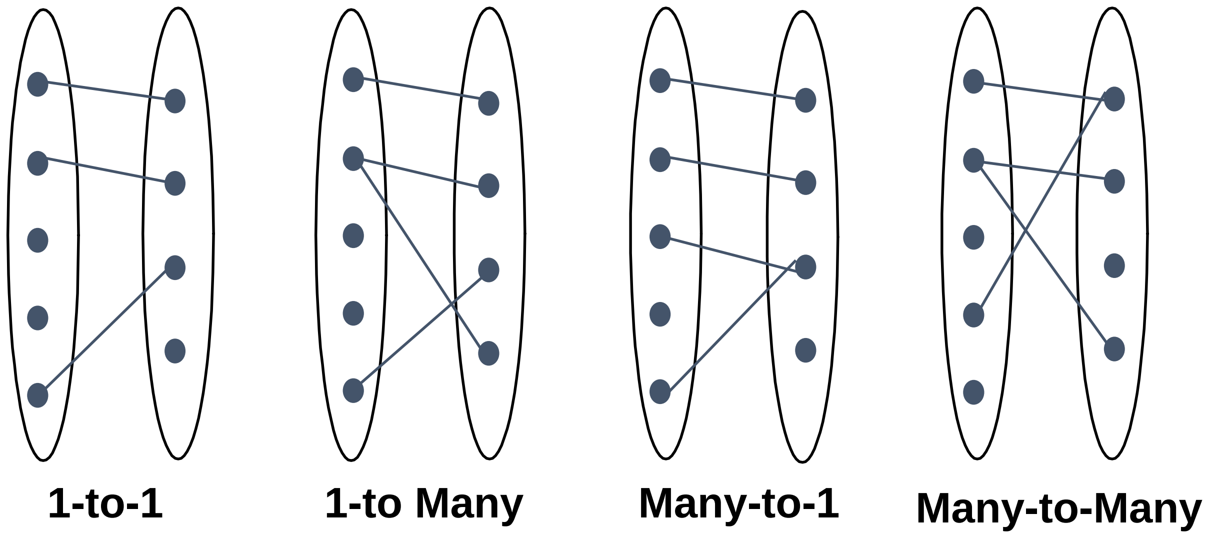


Relationship

- Relationship can also have attributes
  - Are used to record information about the relationship, rather than about any one of the participating entities

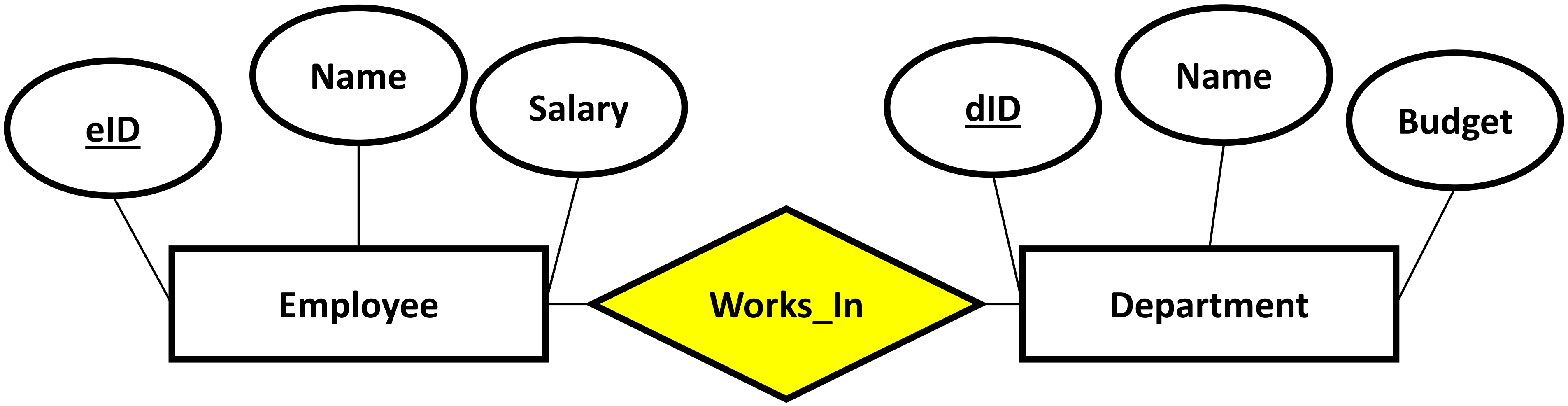


Type of Relationships



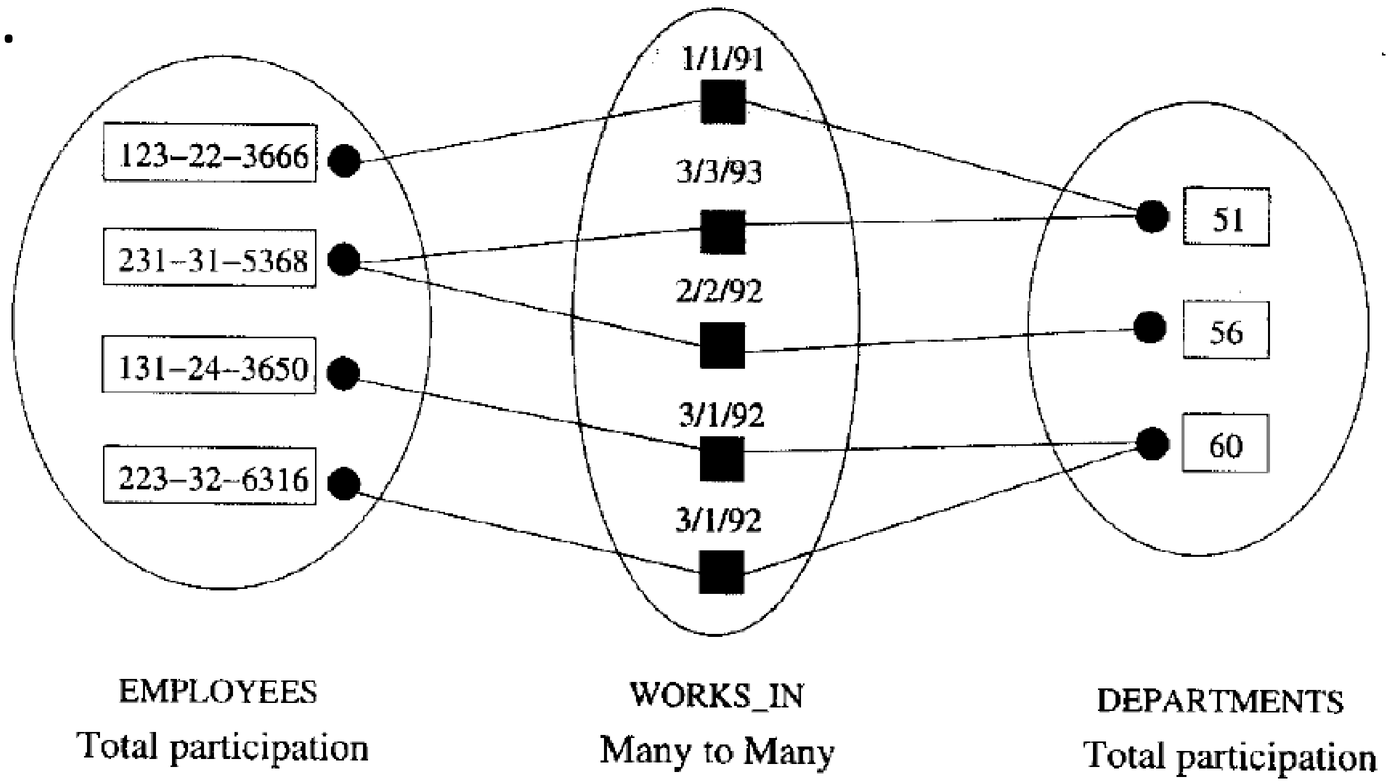
Relationship Key

- All keys involved.
  - e.g., (eID,dID)



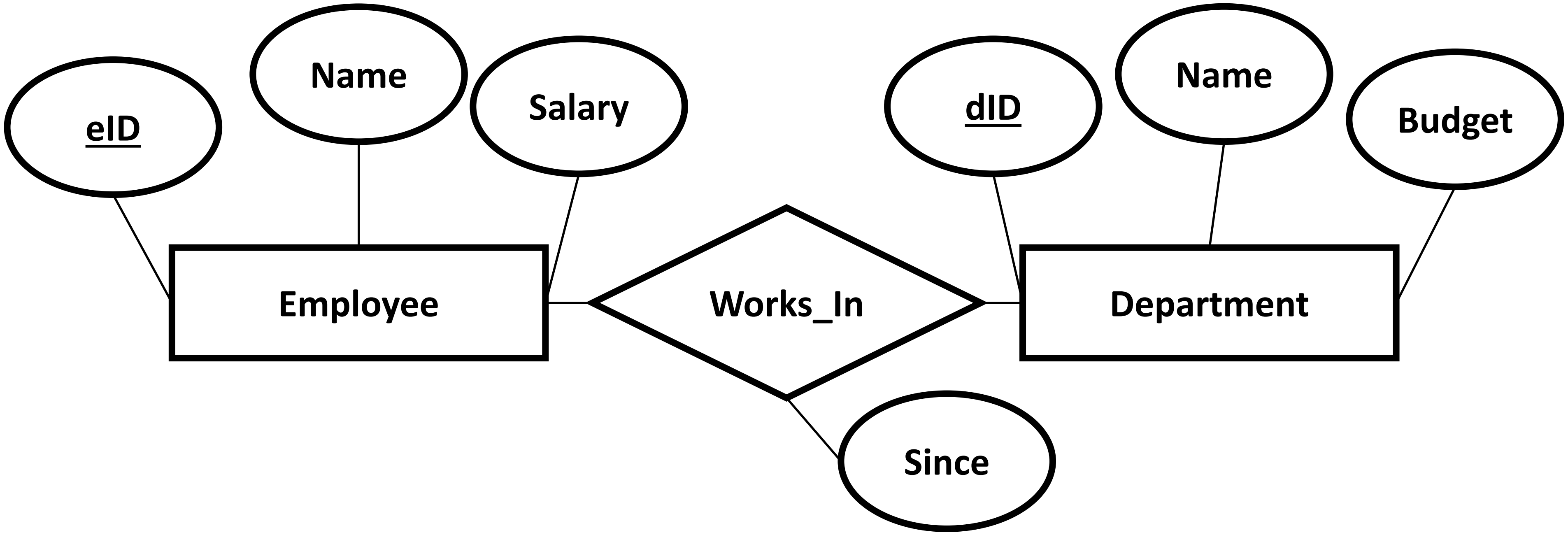
Instance of Relationship

- Snapshot of the relationship set at some instant in time.

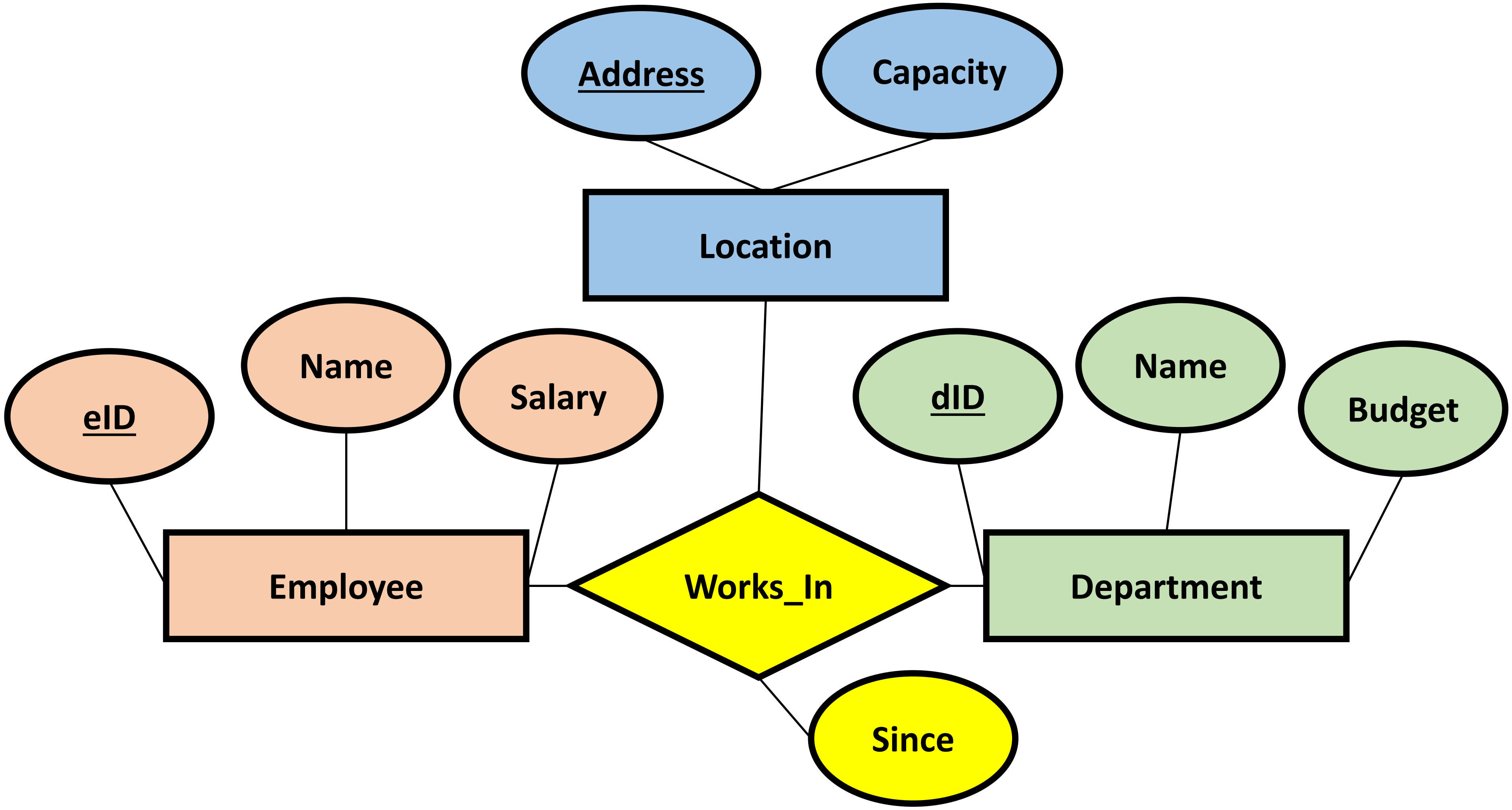


Relationship

- Suppose that each department has offices in several locations and we want to record the locations at which each employee works

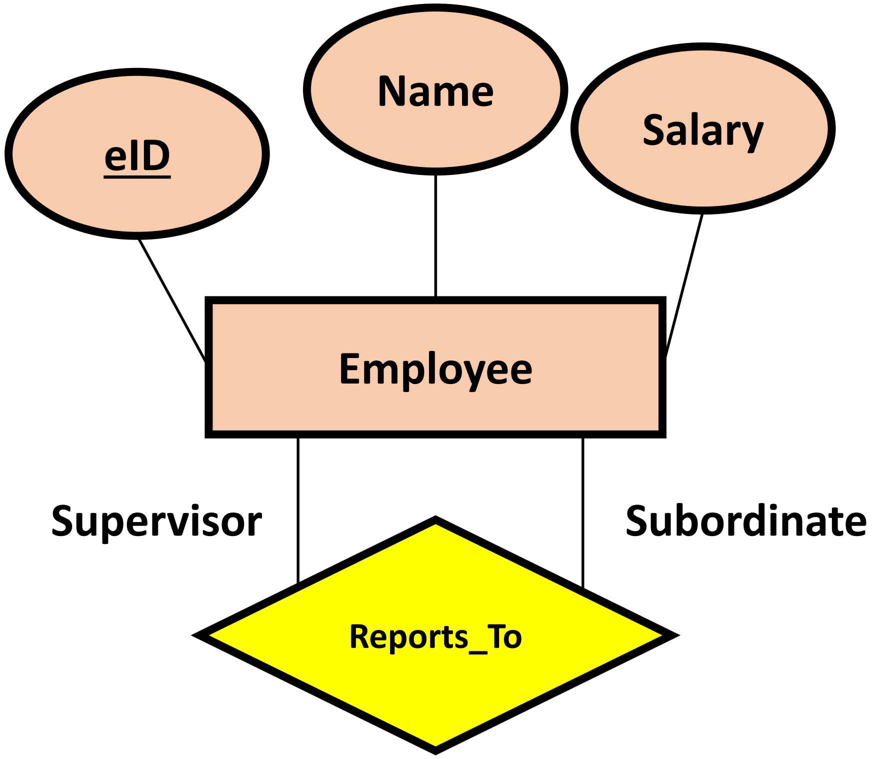


Relationship (ternary)



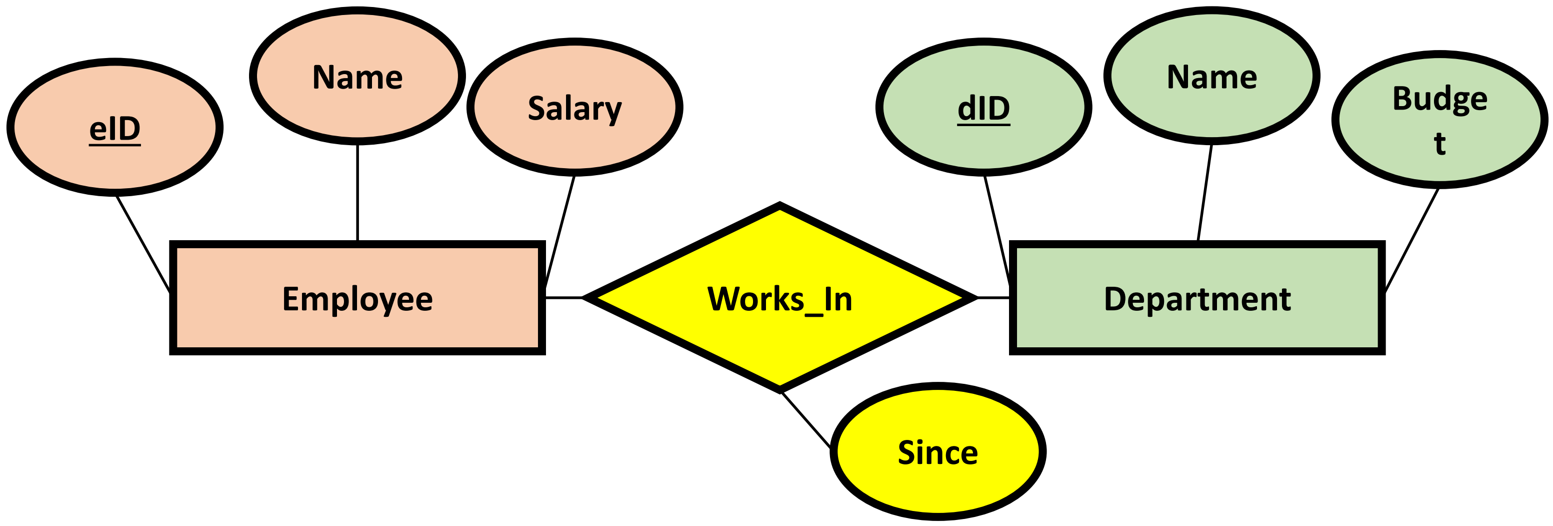
Relationship

- The entity that participate in a relationship set need not be distinct.



Key Constraints

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

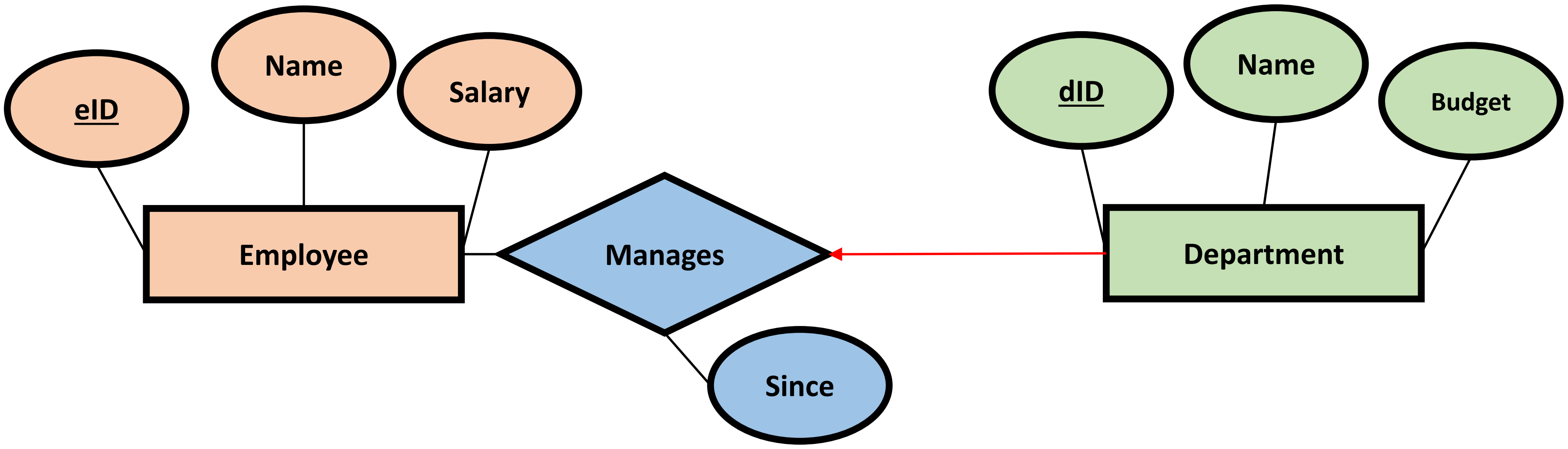


Key Constraints

- In contrast, each dept has at most one manager, according to the *key constraint* on Manages.
- Departments are keys: given a Department entity, we can uniquely determine the Manager relationship,
  - i.e., each department appear once in the Manages table
- One-to-many: one employee can manage many dep., but each dep. can have only one manager

Key Constraints

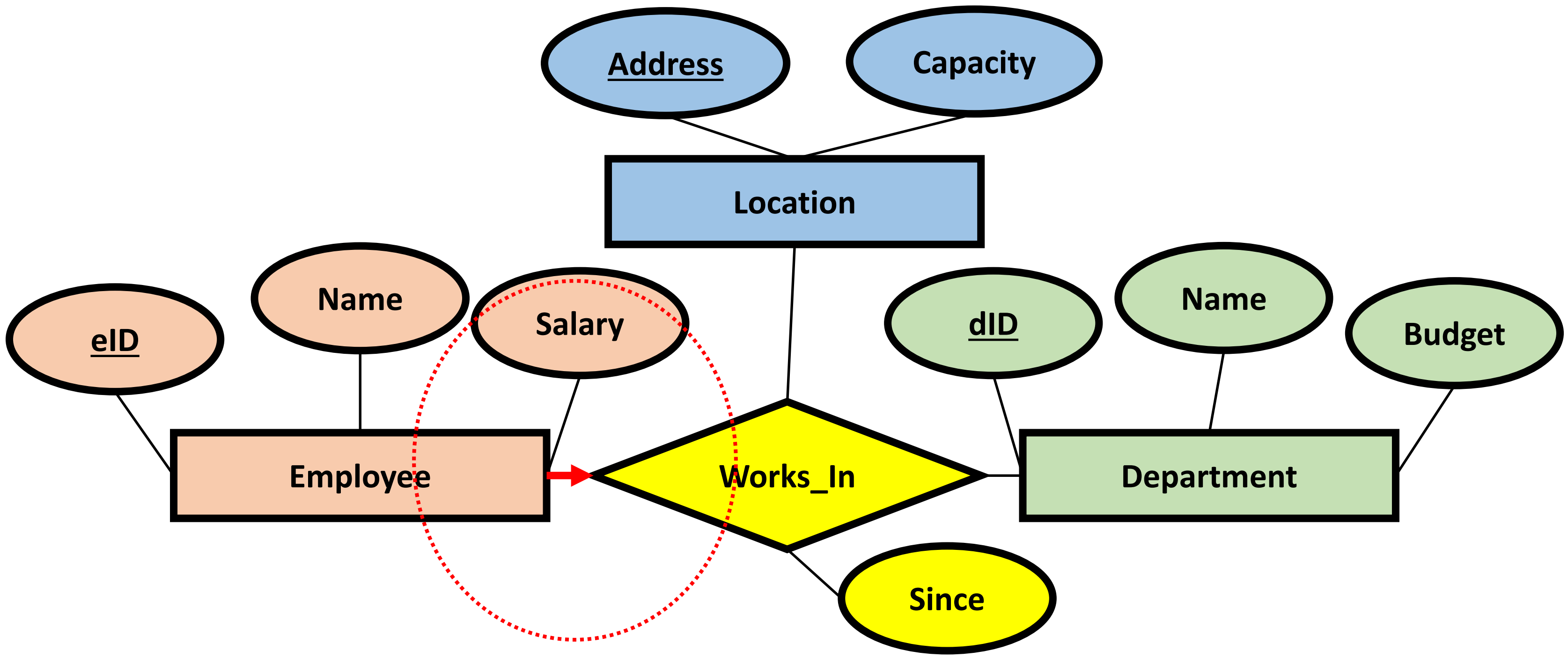
- One-to-many: one employee can manage many dep., but each dep. can have only one manager



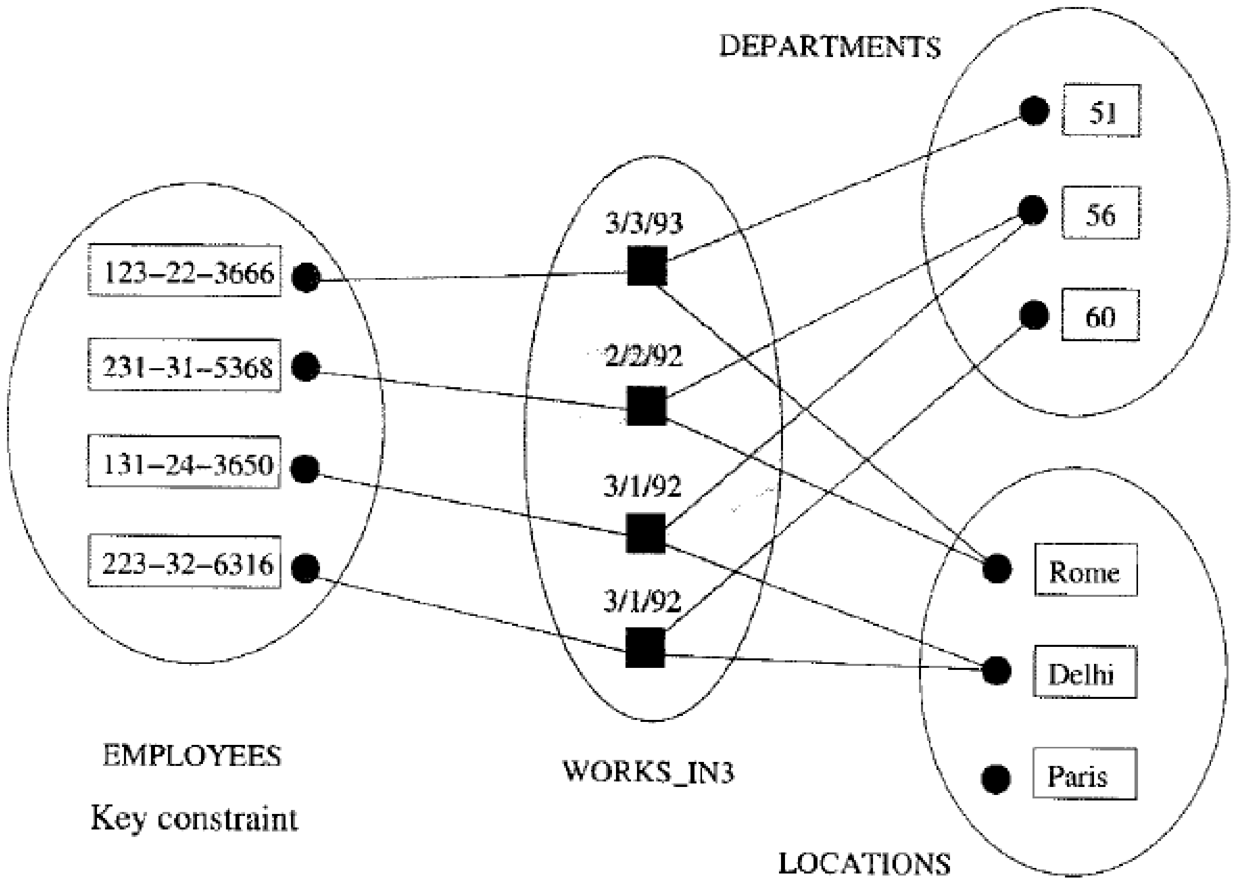
Key Constraints for Ternary Relationships

- If each employee works in at most one department and at a single location?

Key Constraints for Ternary Relationships



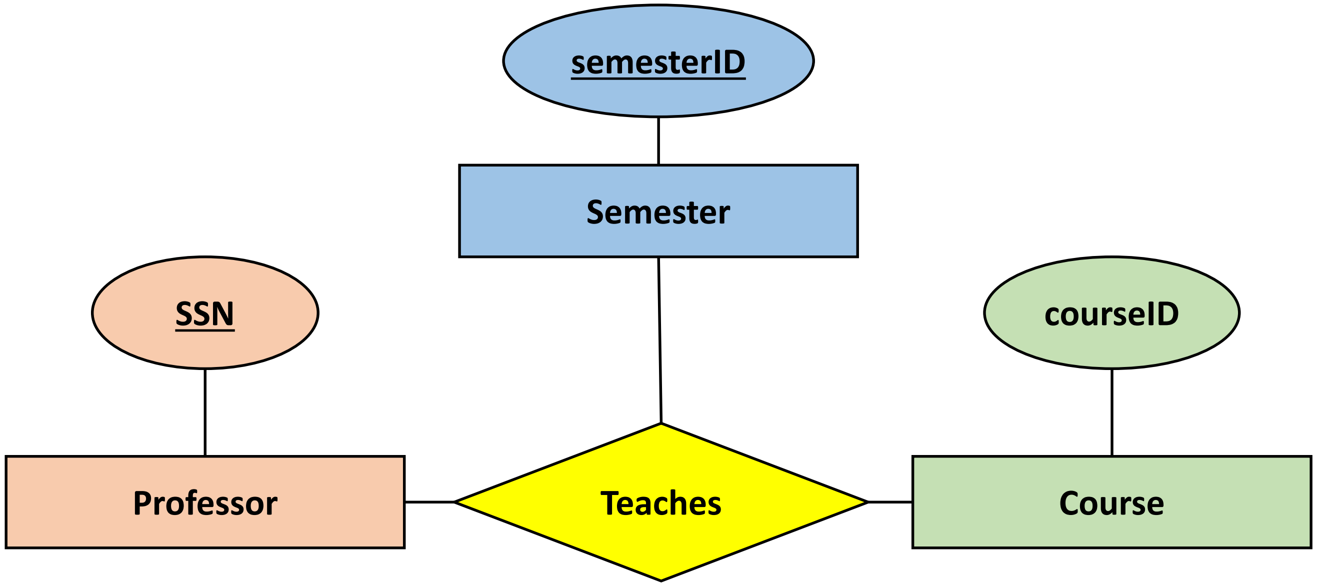
Key Constraints for Ternary Relationships



Exercise

- A university database contains information about professors (identified by social security number, or SSN) and courses (identified by courseID).
- Professors can teach the same course in several semesters, and each offering must be recorded.
- Draw an ER diagram that describes it.

Exercise

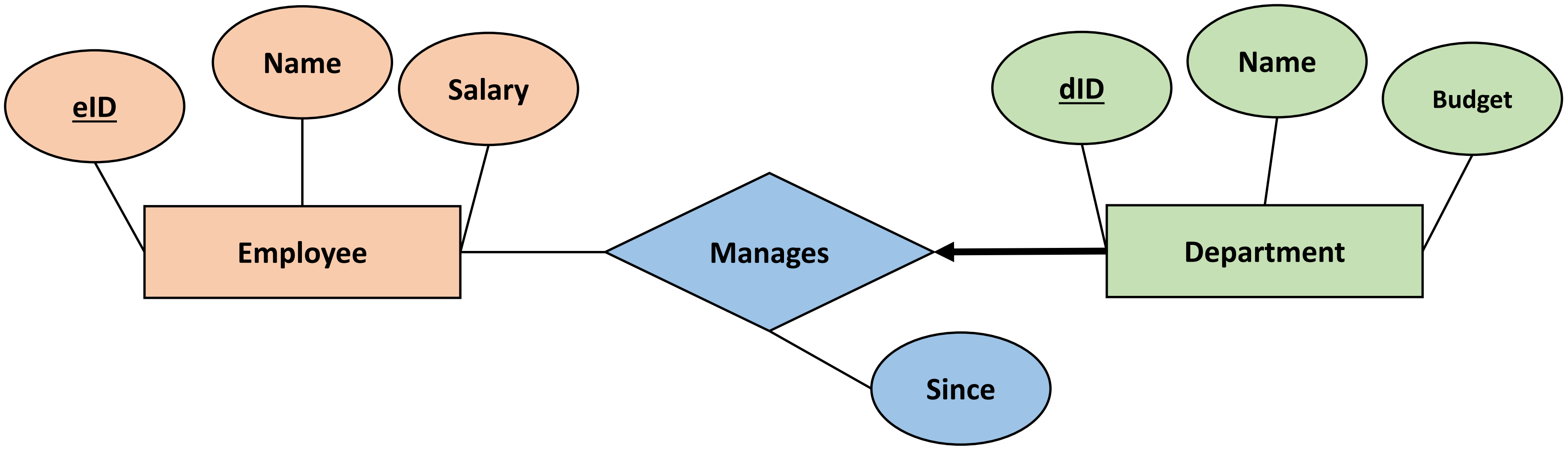

semesterID
Semester
SSN
courseID
Professor
Teaches
Course

Participation Constraints

- The key constraint tells us that a department has at most one manager.
- Does every department have a manager?
  - If so, this is a participation constraint: every department has to have such a relationship.
  - The participation of the entity Department in the relationship Manages is said to be total.

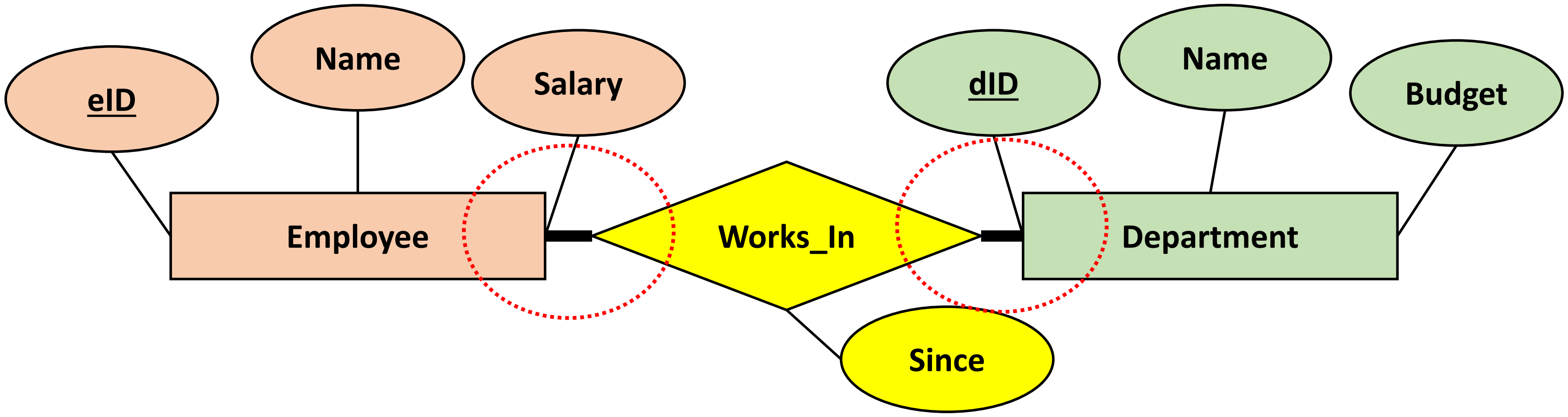
Total and Partial Participation

- The participation of the entity Department in the relationship Manages is said to be total.
- A participation that is not total is said to be partial.
    - not every employee gets to manage a department.

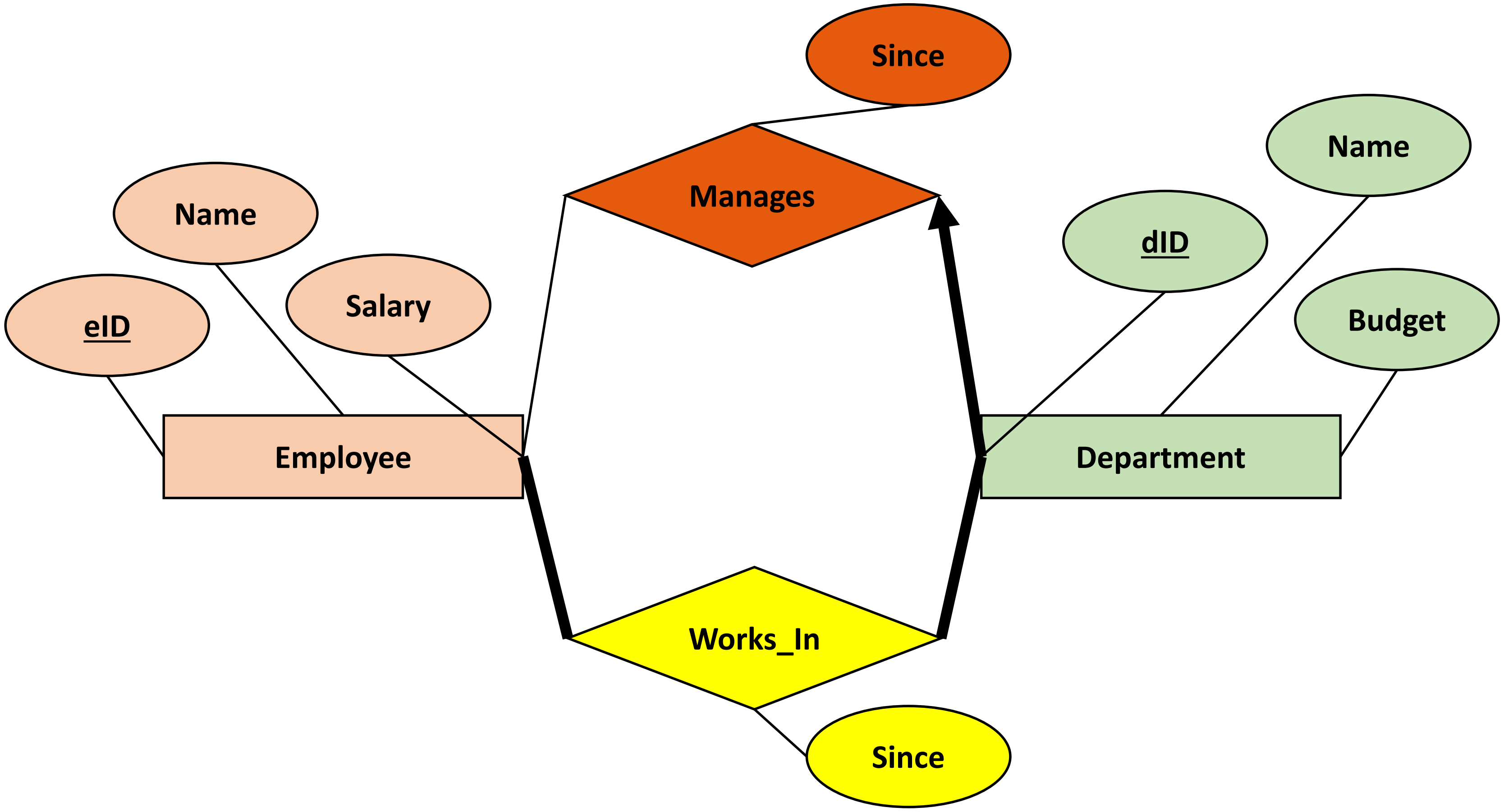


Participation Constraints

- Each employee works in at least one department and that each department has at least one employee.
- This means that the participation of both Employees and Departments in Works_In is total.



Participation/Key Constraints

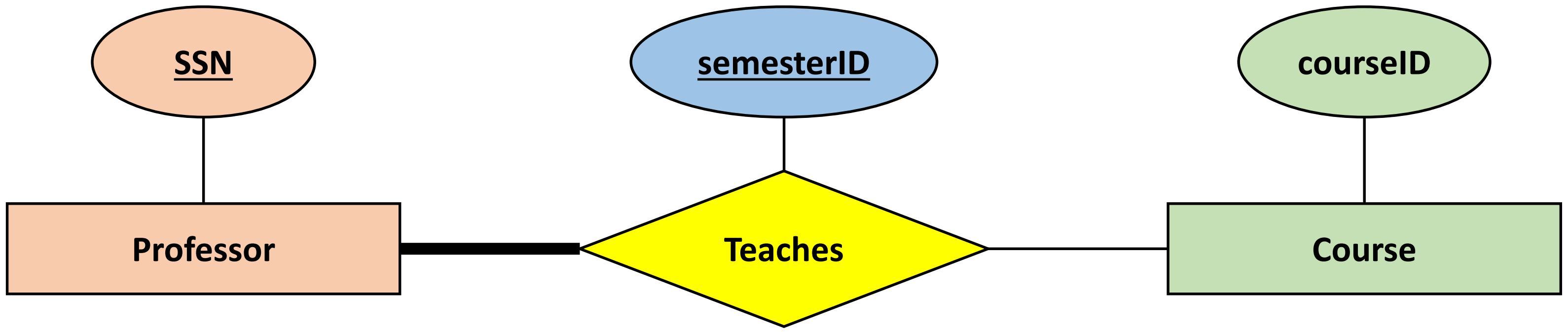


Example

- A university database contains information about professors (identified by social security number, or SSN) and courses (identified by courseID).
    1) Every professor must teach some course.
    2) Every professor teaches exactly one course (no more, no less).
    3) Every professor teaches exactly one course (no more, no less), and every course must be taught by some professor.
- Draw an ER diagram for each that describes it.

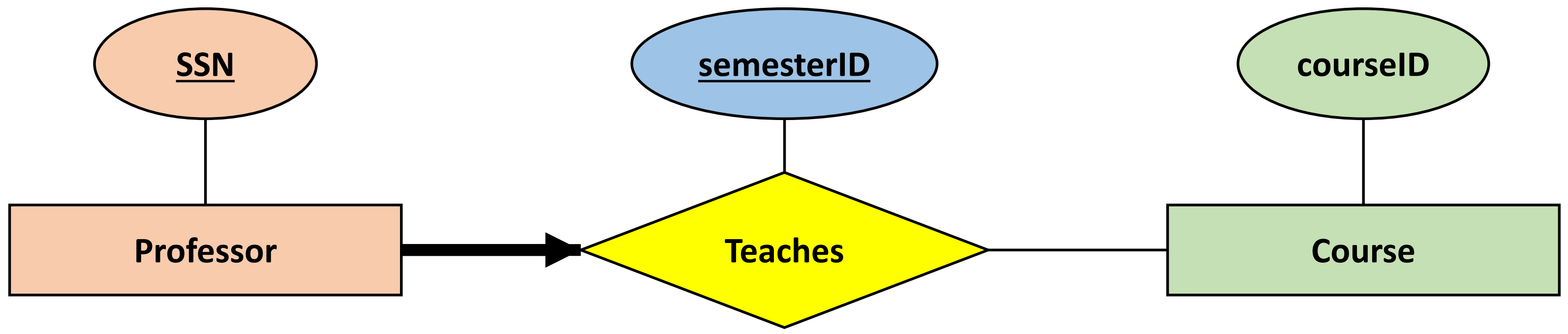
Example

- Every professor must teach some course.



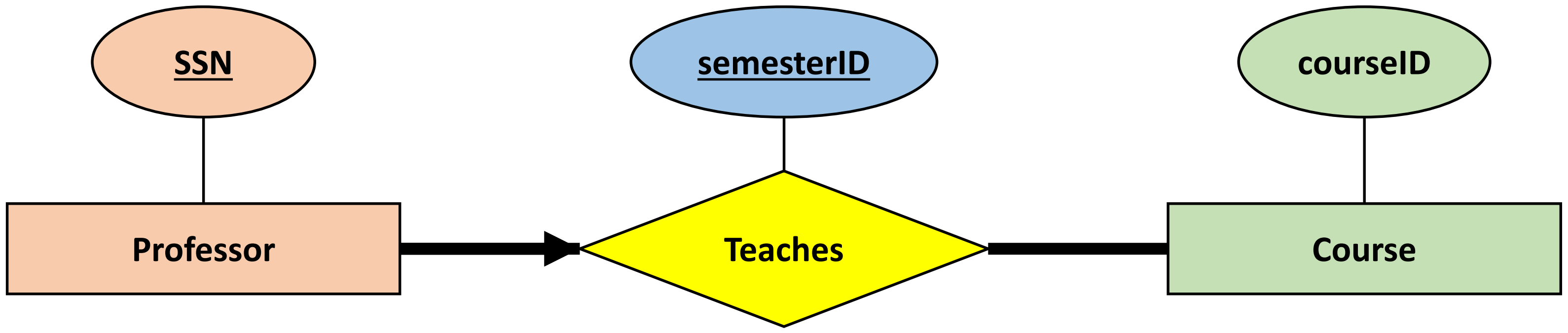
Example

- Every professor teaches exactly one course (no more, no less).



Example

- Every professor teaches exactly one course (no more, no less), and every course must be taught by some professor.



Weak Entities

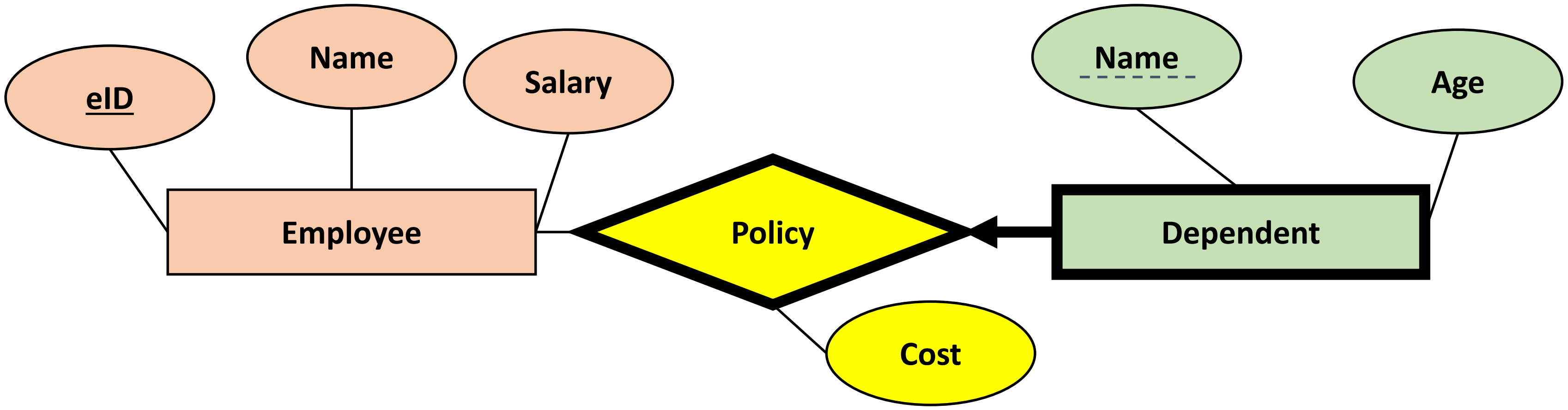
- A weak entity can be identified uniquely only by considering the primary key of another (owner) entity
  - Owner entity and weak entity must participate in a one-to-many relationship set (one owner, many weak entities)
  - Weak entity set must have total participation
  - Partial key: set of attributes of a weak entity that uniquely identify a weak entity for a given owner entity

Weak Entity Example

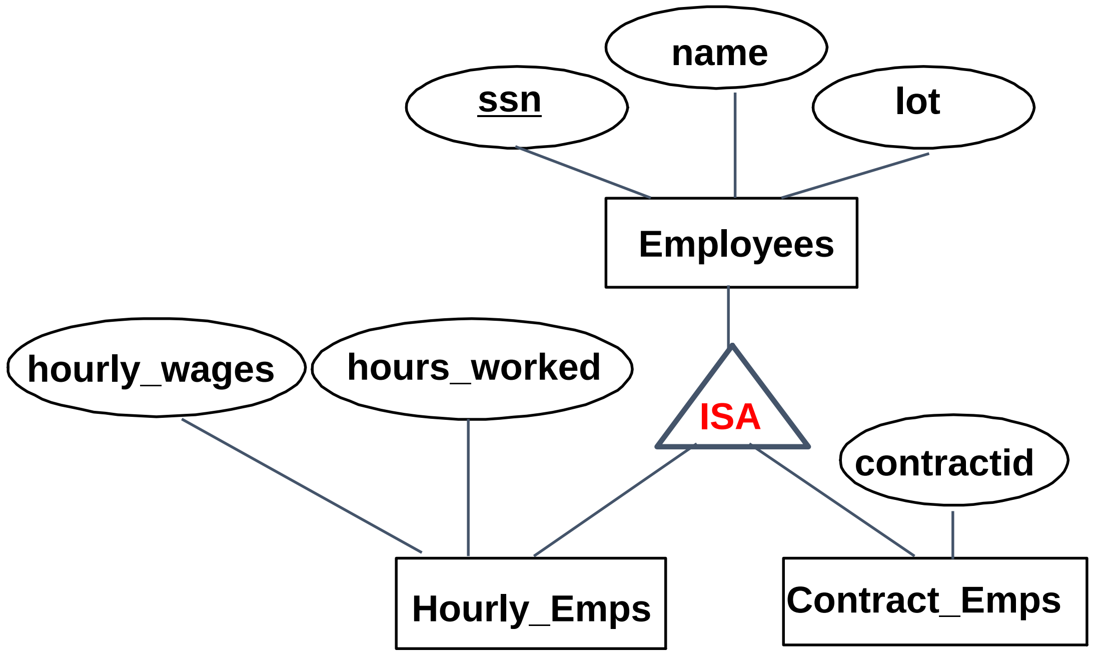
- Suppose that employees can purchase insurance policies to cover their <u>dependents</u>.
- We wish to record information about policies.

Weak Entity Example

- How to show? Thick borders and arrow
- Key?
  - Partial key



Class Hierarchies

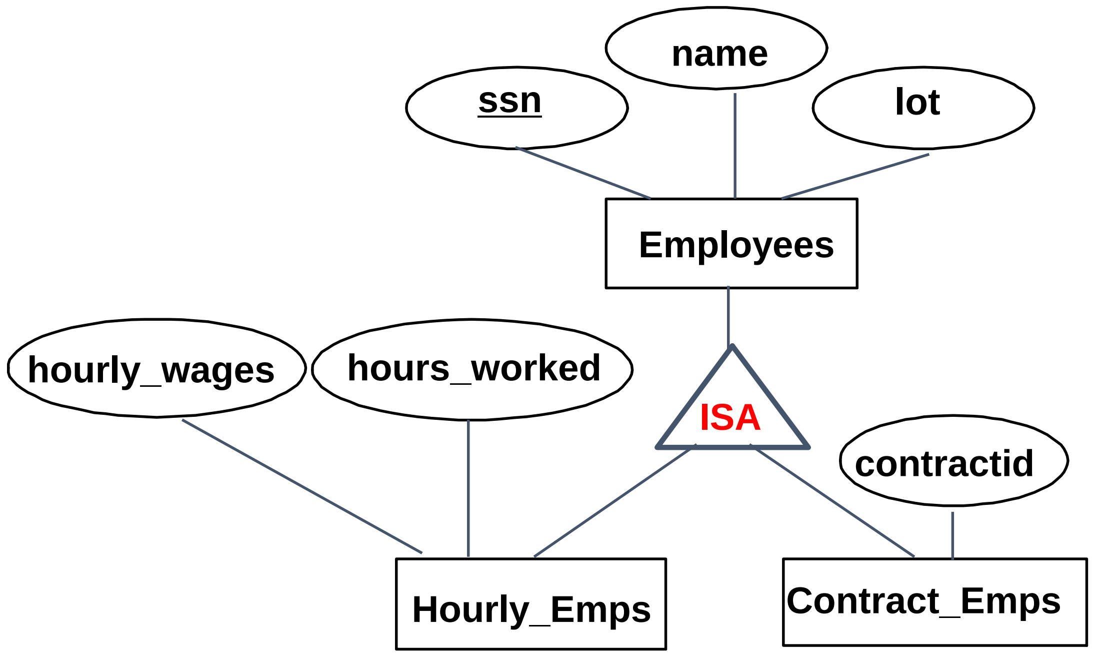


Class Hierarchies

- Attributes are inherited.
- If we declare A **ISA** B, every A entity is also considered to be a B entity.
- Constraints
  - *Overlap constraints*: Can Joe be an Hourly_Emps as well as a Contract_Emps entity?
    - No, unless stating "Contract_Emps OVERLAPS Hourly_Emps"
  - *Covering constraints*: Does every Employees entity also have to be an Hourly_Emps or a Contract_Emps entity?
    - No, unless stating "Hourly_Emps AND Contract_Emps COVER Employees"

Class Hierarchies

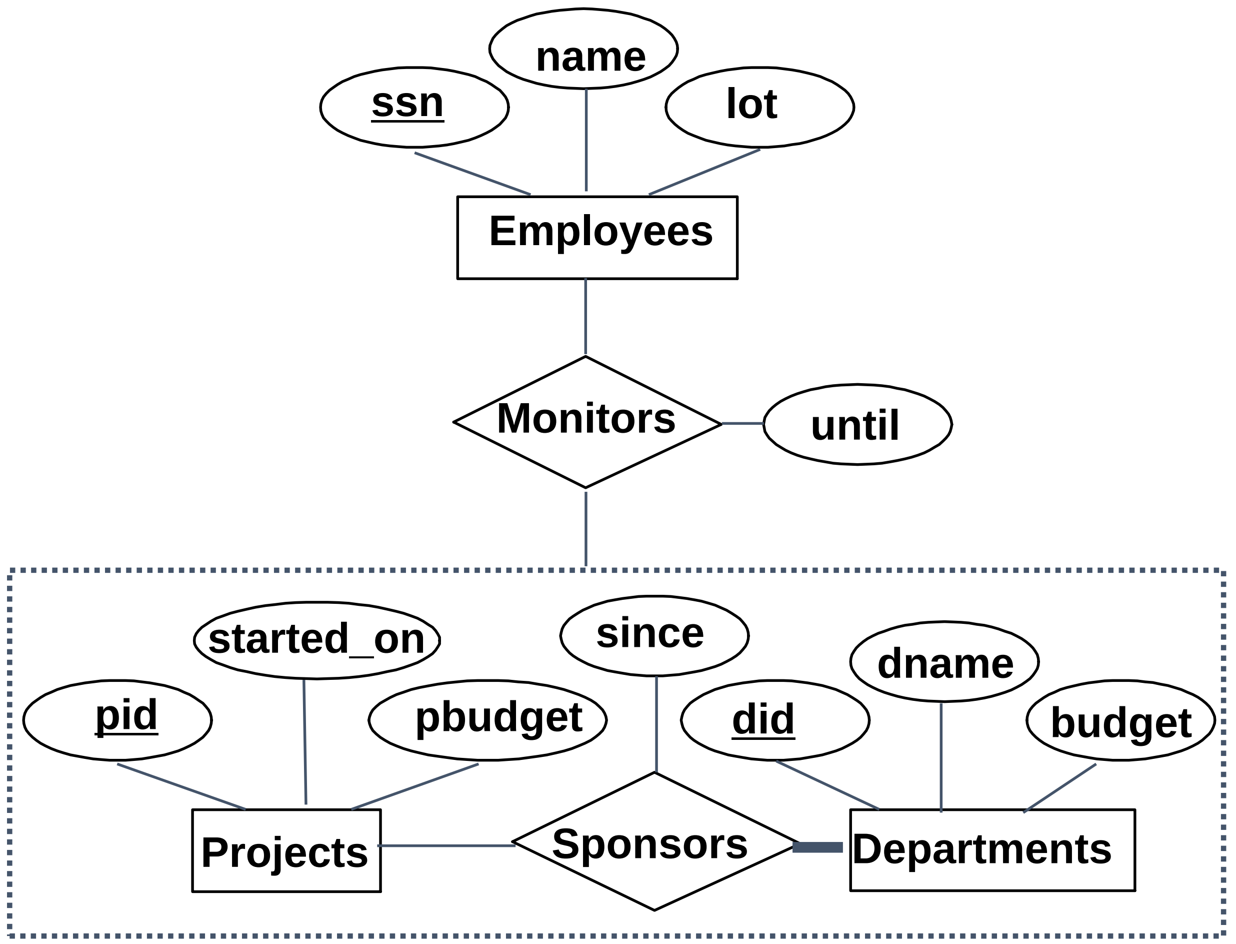
- Employees is specialized into subclasses.
- Hourly_Emps and Contract_Emps are generalized by Employees.



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 Example

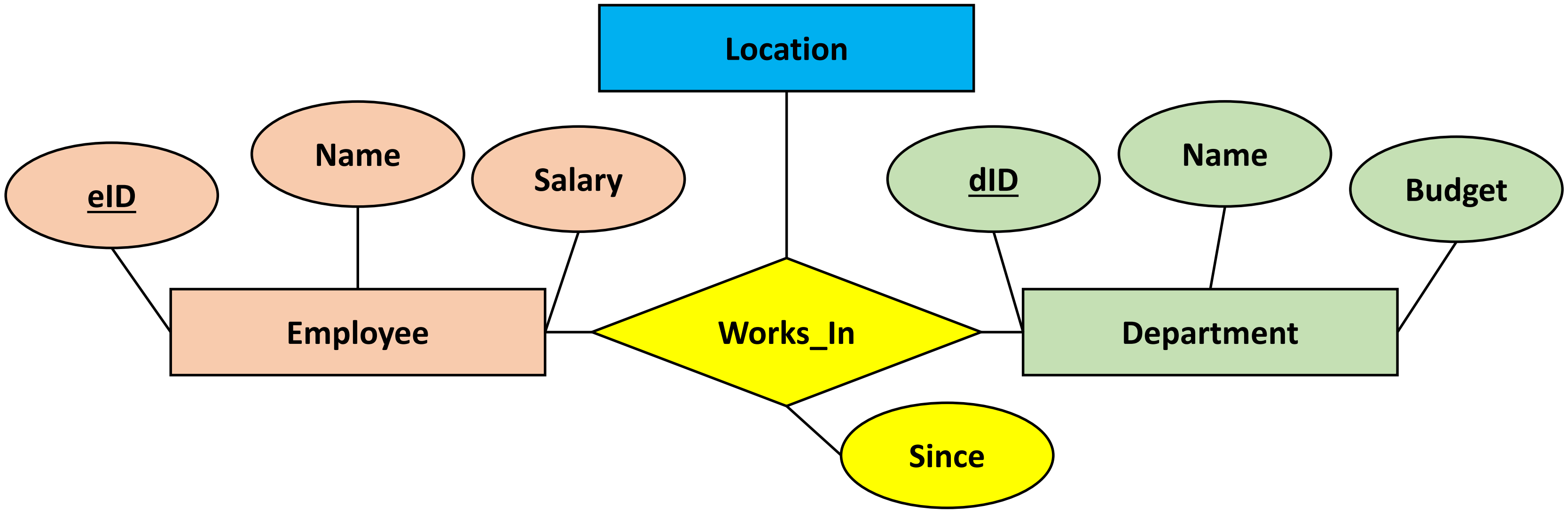


Conceptual Design Using the ER Model

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

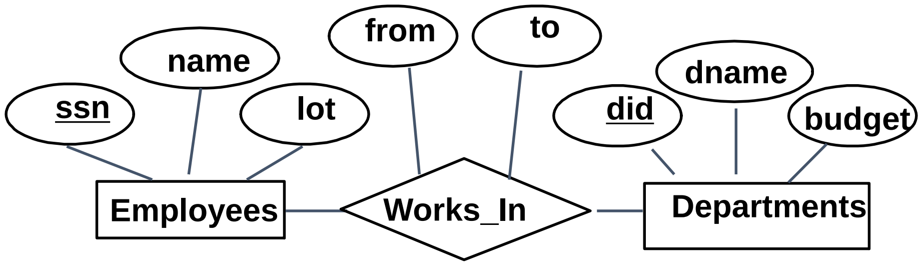
Entity vs. Attribute

- We have to record more than one address for an employee.
- Or, we want to capture the structure of an address in our ER diagram.

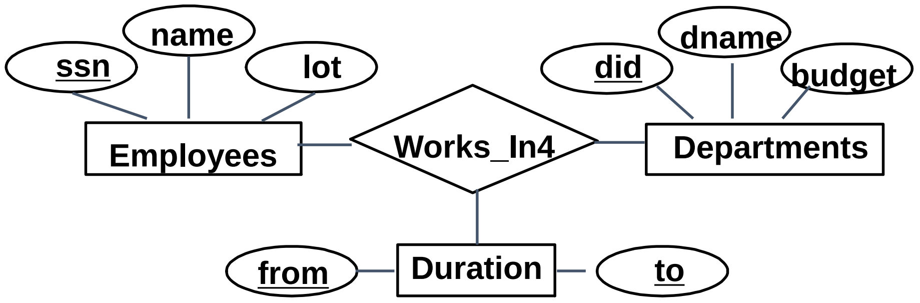


Entity vs. Attribute Example

- Works_In does not allow an employee to work in a department for two or more periods
  - A relationship is uniquely identified by the participating entities

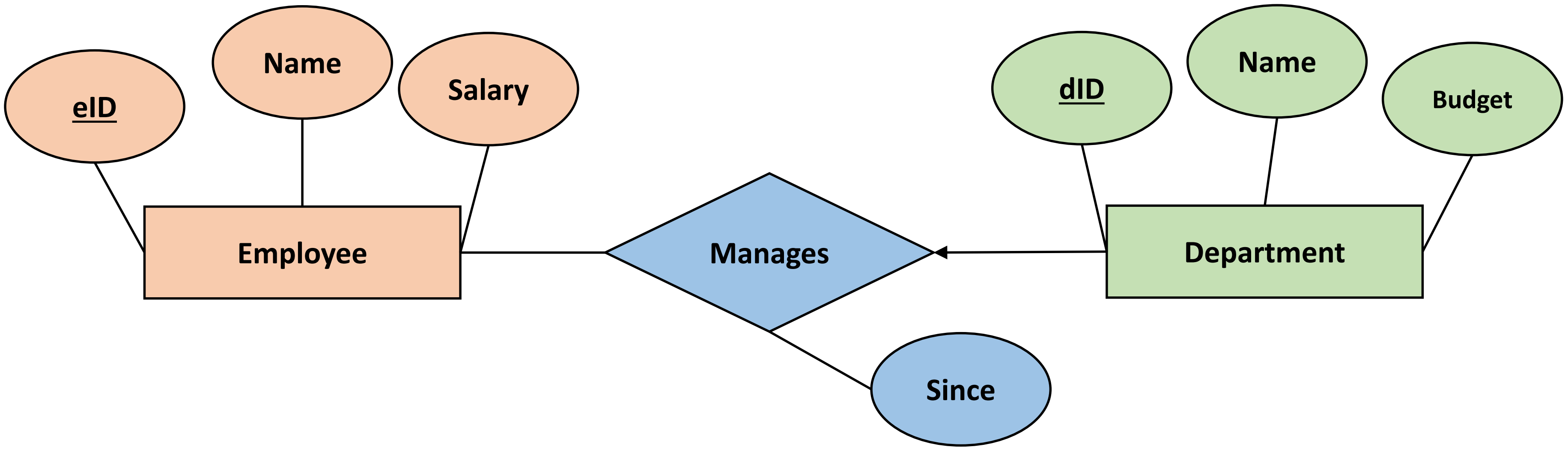


- Similar to the problem of wanting to record several addresses for an employee: We want to record several values of the descriptive attributes for each instance of this relationship.
  - Accomplished by introducing new entity set, Duration.
  - Set-valued, then attribute



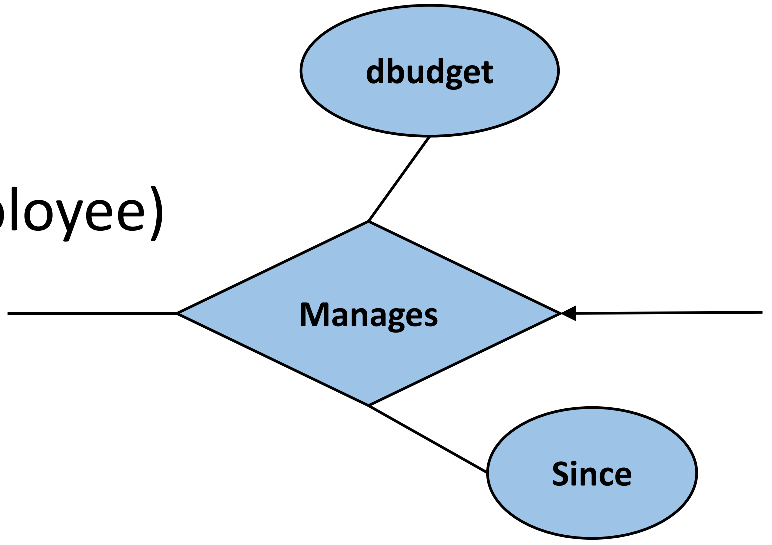
Entity vs. Relationship

- Consider the relationship set called Manages.
- Suppose that each department manager is given a discretionary budget *(dbudget).*



Entity vs. Relationship

- Add *dbudget to Manages?*
- What if the discretionary budget is a sum that covers *all* departments managed by that employee? (redundant storage)
- Misleading; it suggests that the budget is associated with the relationship, when it is actually associated with the manager.
- Solution?
  - New entity: Managers (ISA Employee)
    - dbudget attribute of Managers



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 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.