Databases

Cap. 2 Data Modeling. ER Model. Relational Model

Textbook: Ramakrishnan, Gehrke, “Database Management Systems”, McGraw Hill, 2003

2020 UPT Conf.Dr. Dan Pescaru

DB development

- DB development steps:
  1. Requirement analysis
  2. Conceptual design – ER model
  3. Logical DB design – relational model
  4. DB Schema refinement – normalization
  5. Physical DB design – indexes etc.
  6. Client applications design
  7. Applications implementation
  8. DB deployment

Requirement analysis

1. Result - requirements documentation including answers to the following questions
   - What are the entities and relationships from the enterprise?
   - What specific information about these entities have to be stored to the database? What are the domains of these information?
   - What integrity constraints and domain rules that have to be considered?
   - What data processing is necessary to support the business. What the users roles?

Database related models

1. **Conceptual models** are used to analyze and understand application data
   - The **ER** model is a de-facto standard in DB field. Alternative: the **UML**
2. **Data models** are used to describe data
   - A **schema** is a description of a particular collection of data, using the a given data model
   - The relational model is the most widely used data model today
3. **Physical models** are representation of a data design which considers the facilities and constraints of a given **DBMS**

Conceptual model

1. A high-level description of a business informational needs
2. A conceptual model identifies the general relationships between the different entities
3. Characteristics of conceptual data model
   - Include information of all important entities and the relationships among them
   - No data organization is specified
   - Just some constraints are specified

The ER model

1. For basic applications the DB schema results directly from requirements analysis
2. For complex applications conceptual design is required
3. ER model could be used for a graphical representation (ER diagrams) of DB schema
4. There is a direct mapping of ER diagrams into relational data modes

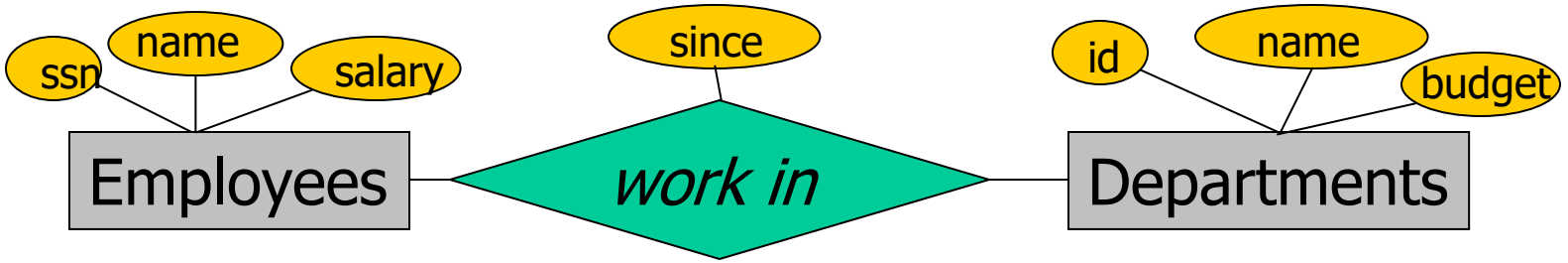
The ER model. Definitions (I)

1. Entity = Real-world entity (object) distinguishable from other entities. An entity is described using a set of attributes
2. Entity set = A collection of similar entities
   - E.g., all employees from a company
   - ER representation: a rectangle
   - All entities in an entity set have the same set of attributes
   - Each entity set has a key
   - Each attribute has a domain (type)

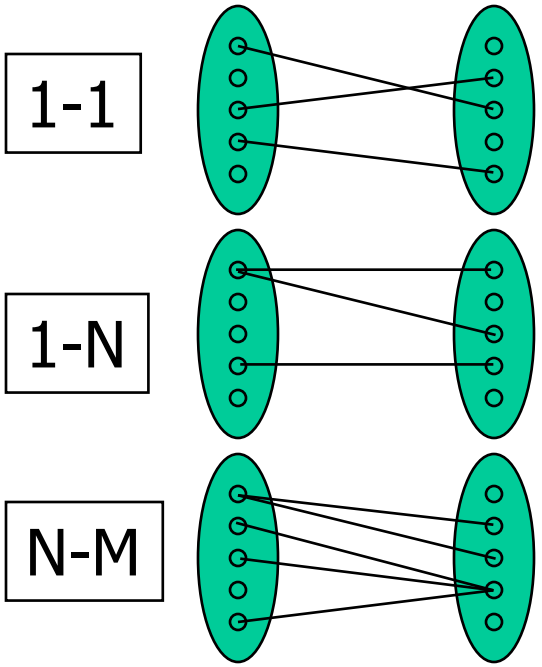
The ER model. Definitions (II)

1. Relationship = association among two or more entities
   - E.g., Anton *works in* Financial department
2. Relationship set = collection of similar relationships
   - ER representation: a diamond
   - 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 entity set (e.g. works-in, manages)

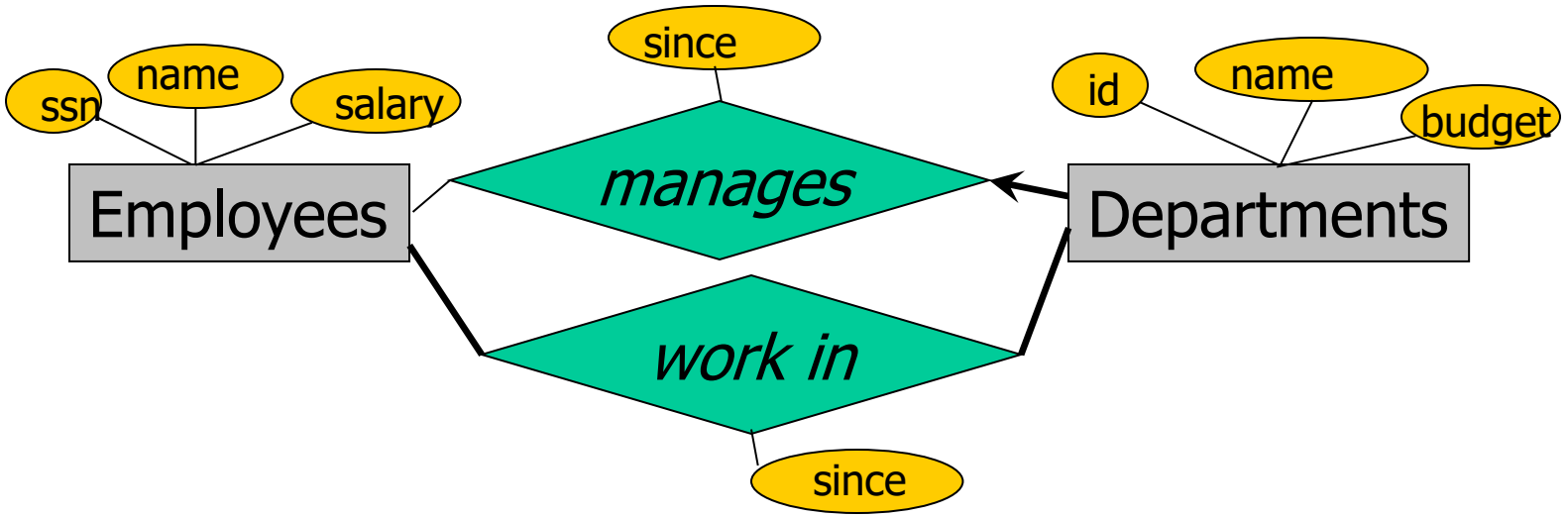
Types of relationships



1. 1-1 relationship. One Department has exactly one General Manager (*key constraint – an arrow to relationship)*
2. 1-N relationship. One Department could have more than one Project managers. A Project manager works in one Department
3. N-M relationship. An Employee could work in more than one Department. One Department could have more than one Employee



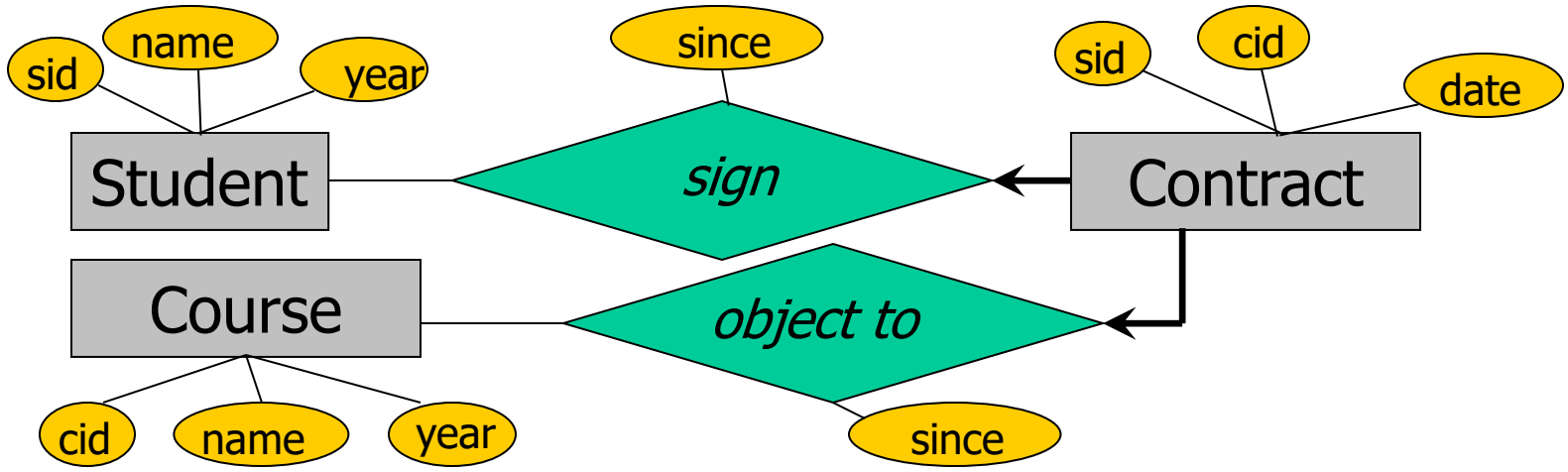
Participation constraints



1. Does every Department have a manager? (in relationship with Employees) ?
2. Partial participation in the relationship – thin line (e.g. just some Employees are also managers)
3. Total participation in the relationship – thick line (e.g. all Departments have to have a corresponding manager in Employees)

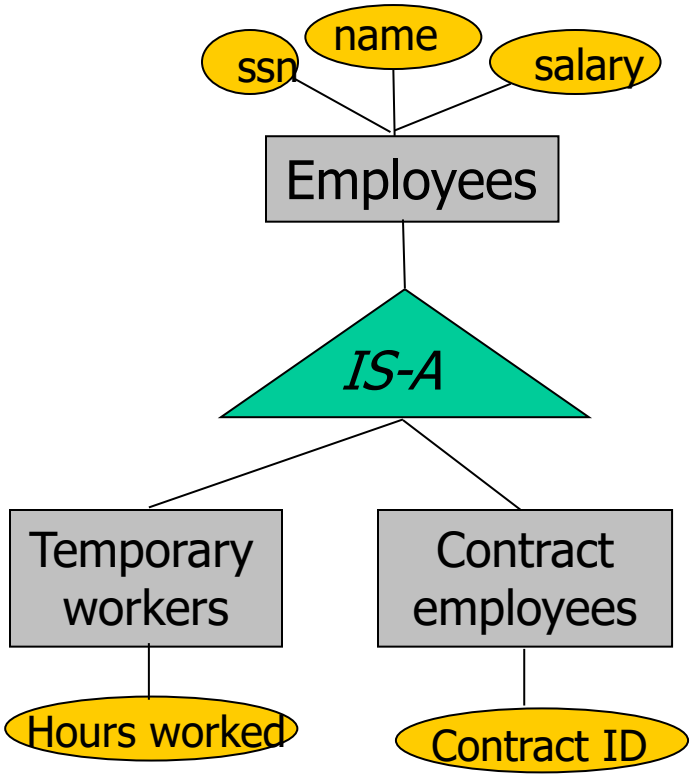
Weak entity

1. Weak entity set = a set of entities that 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



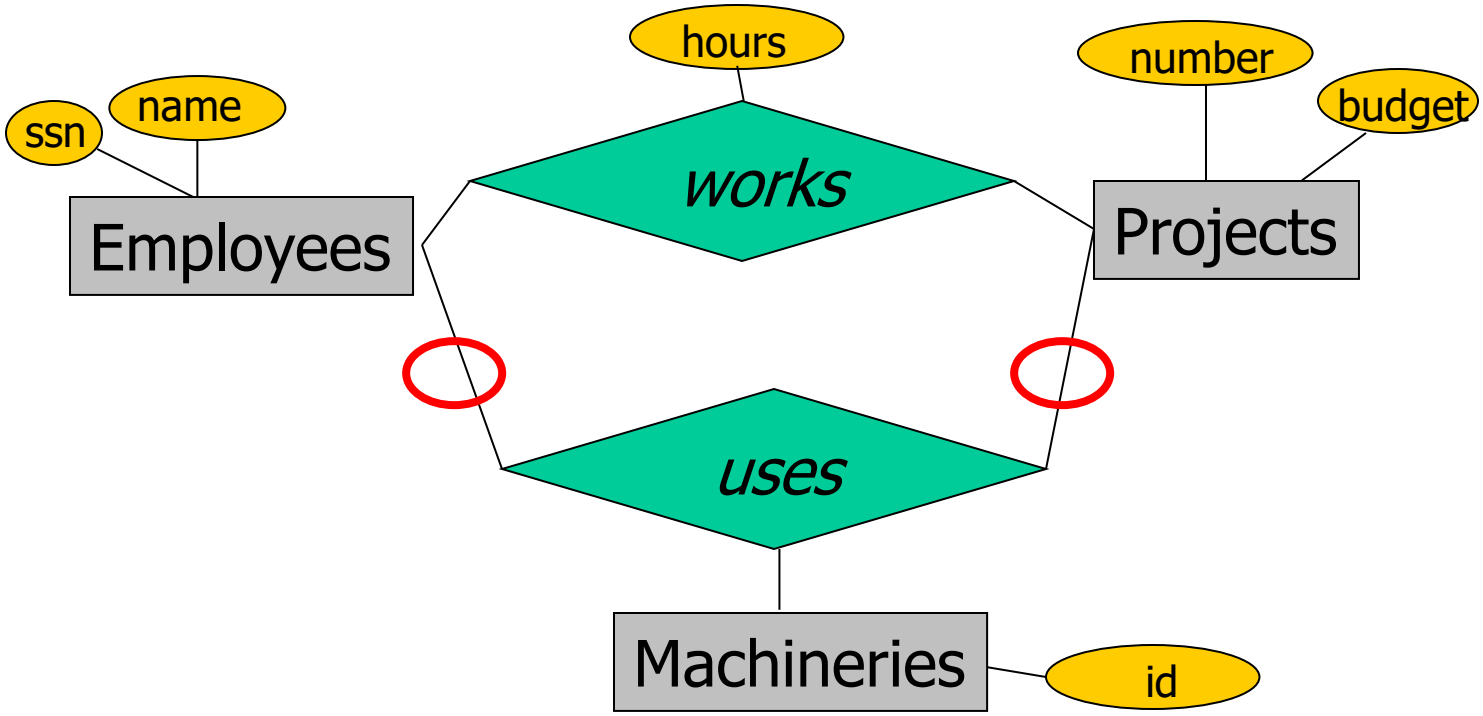
IS-A hierarchies

1. IS-A imply attributes inheritance. It allows adding descriptive attributes specific to a subclass
2. If A *IS-A* B, every A entity is also considered to be a B entity
3. **Overlap constraints**: A *IS-A* C and B *IS-A* C, e∈A and e∈B (allowed/disallowed)
4. **Covering constraints**: does every C to be a A or B (yes/no)



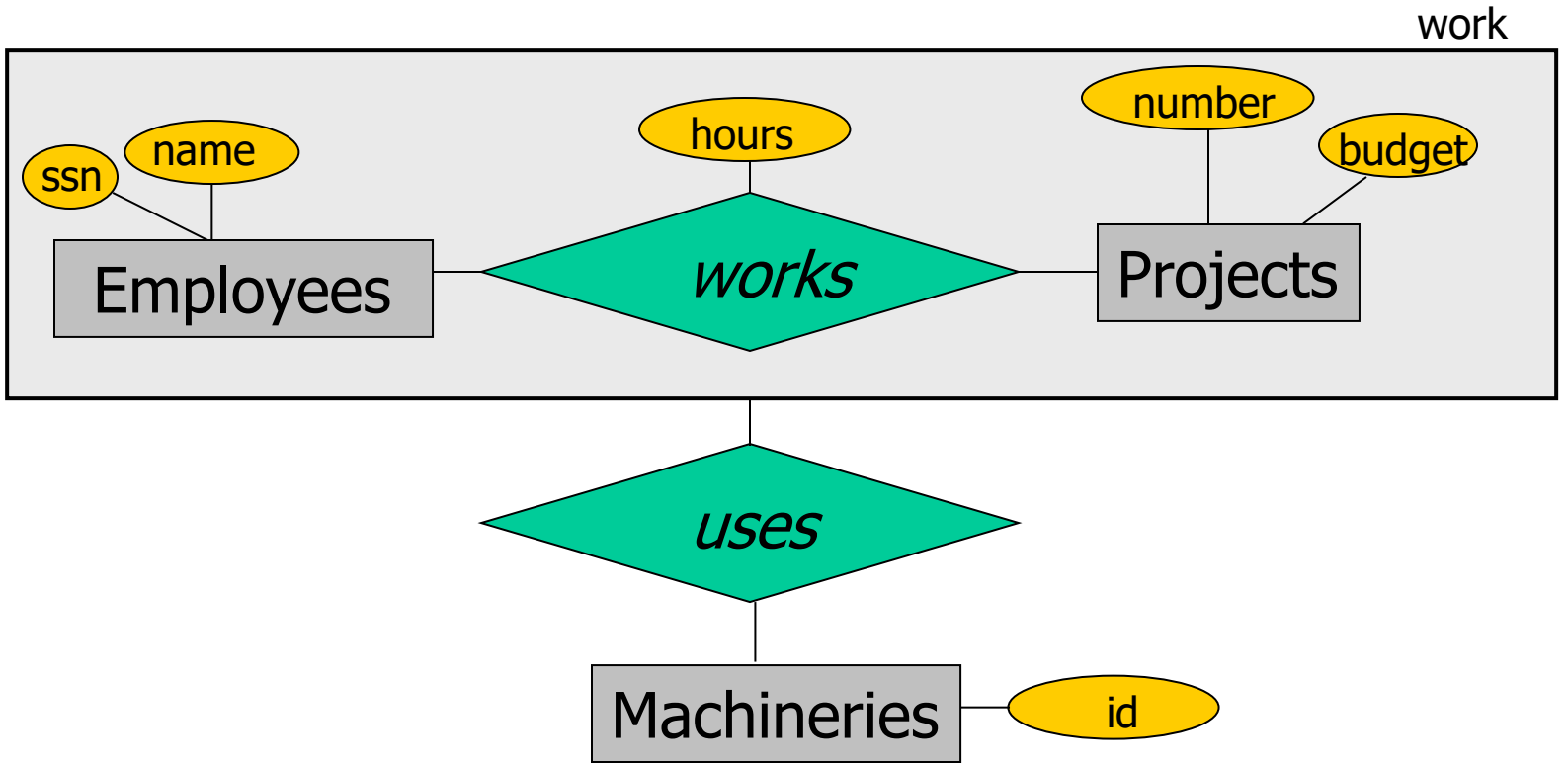
Redundant relationships

1. The E-R model cannot express directly relationships among relationships. This could determine redundant relationships



Aggregation

1. An abstraction through which relationships are treated as higher-level entities
   - E.g. the relationship set work and the entity set employee and project as a higher-level entity set "**work**"



Conceptual Design Using the ER Model

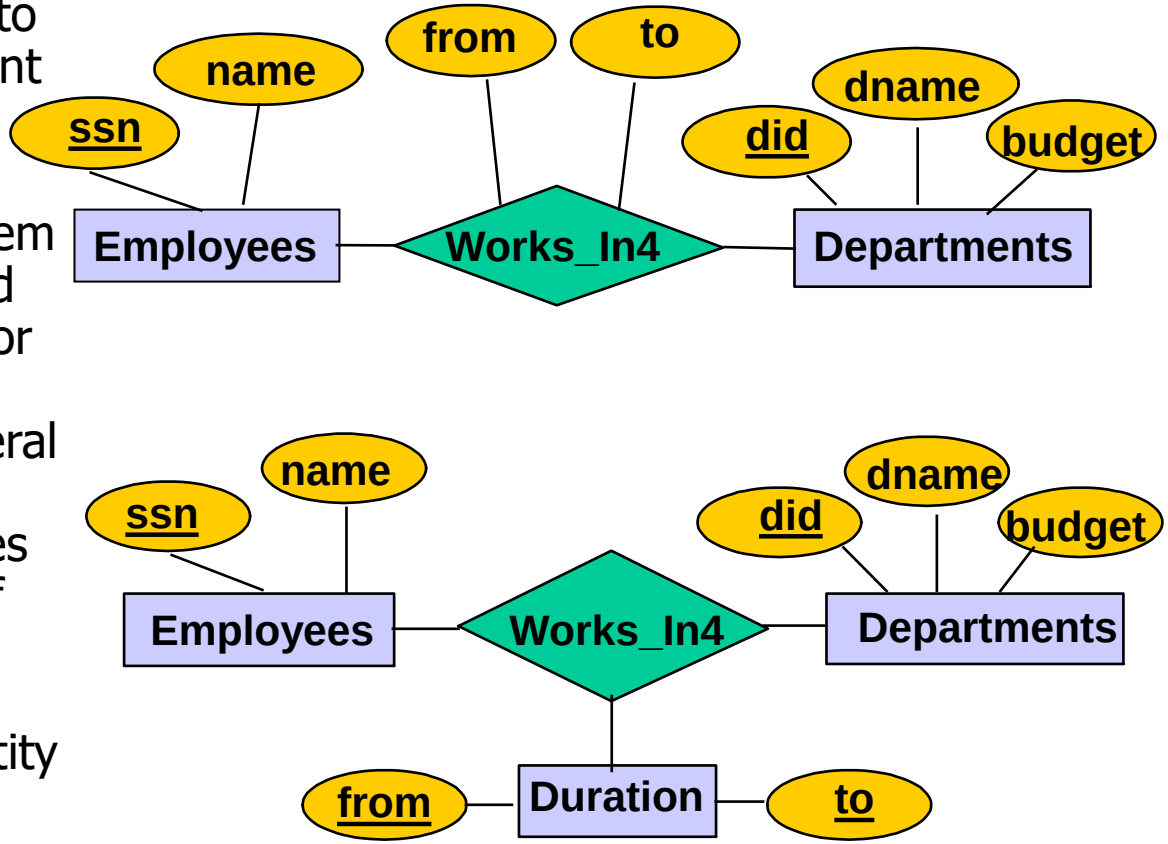
1. Design choices:
   - Should a concept be modeled as an entity or an attribute?
   - Identifying relationships: Binary or ternary? Aggregation?
2. Constraints in the ER Model
   - A lot of data semantics can (and should) be captured
   - Some constraints cannot be captured in ER diagrams

Entity vs. Attribute

1. Should a concept be modeled as an entity or an attribute?
2. 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)

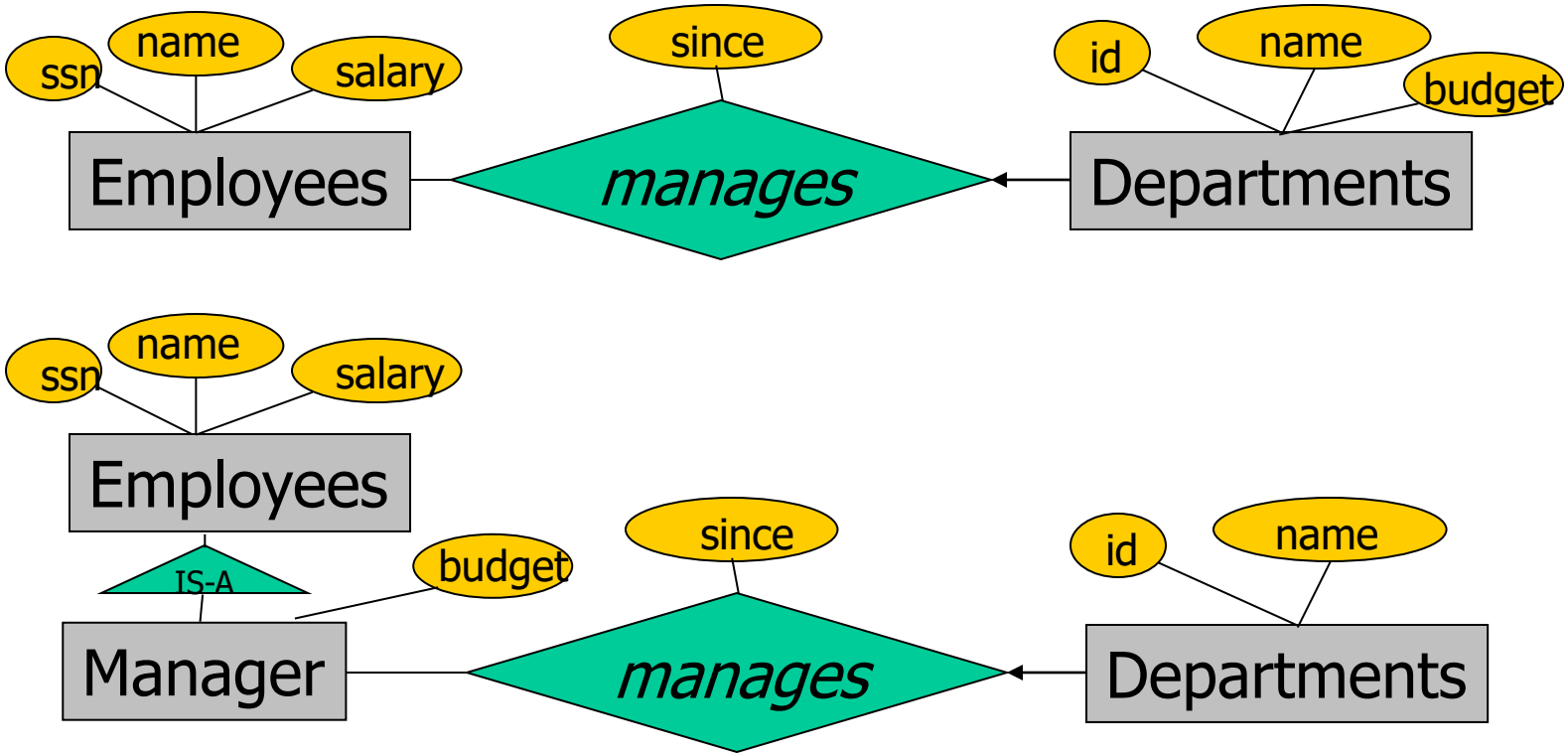
Entity vs. Attribute

1. Works_In4 does not allow an employee to work in a department for two or more periods
2. 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"



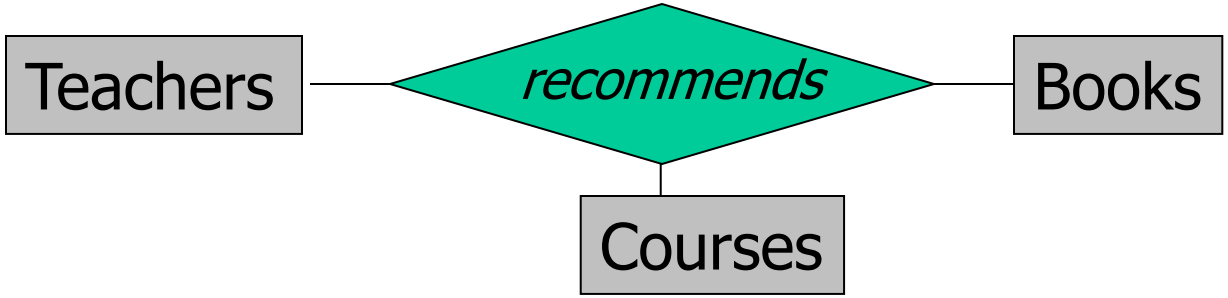
Entity vs. Relationship

1. Should a concept be modeled as an entity or a relationship?
   - Has each Department its own budget?
   - Has each Manager its own budget (for all Depts)?



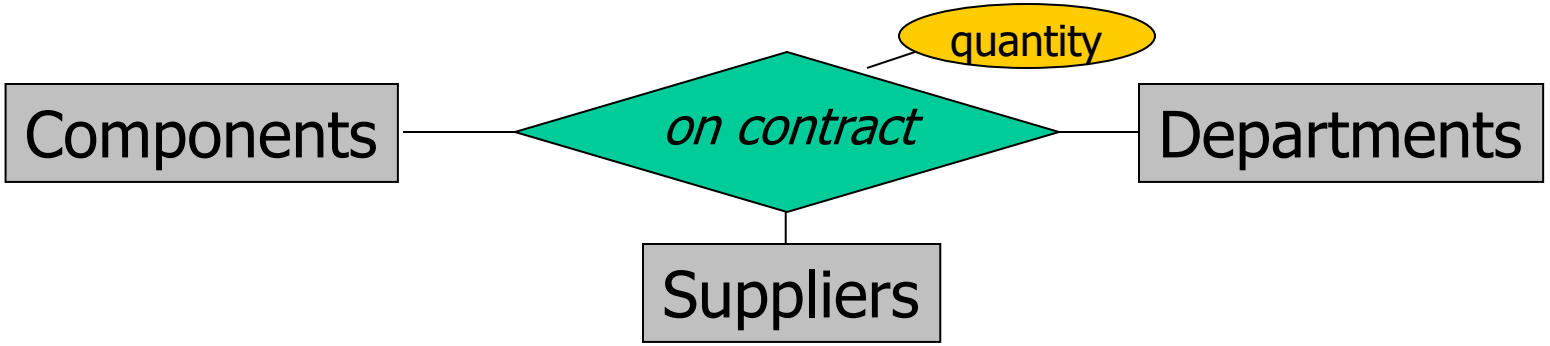
Binary vs. Ternary Relationships

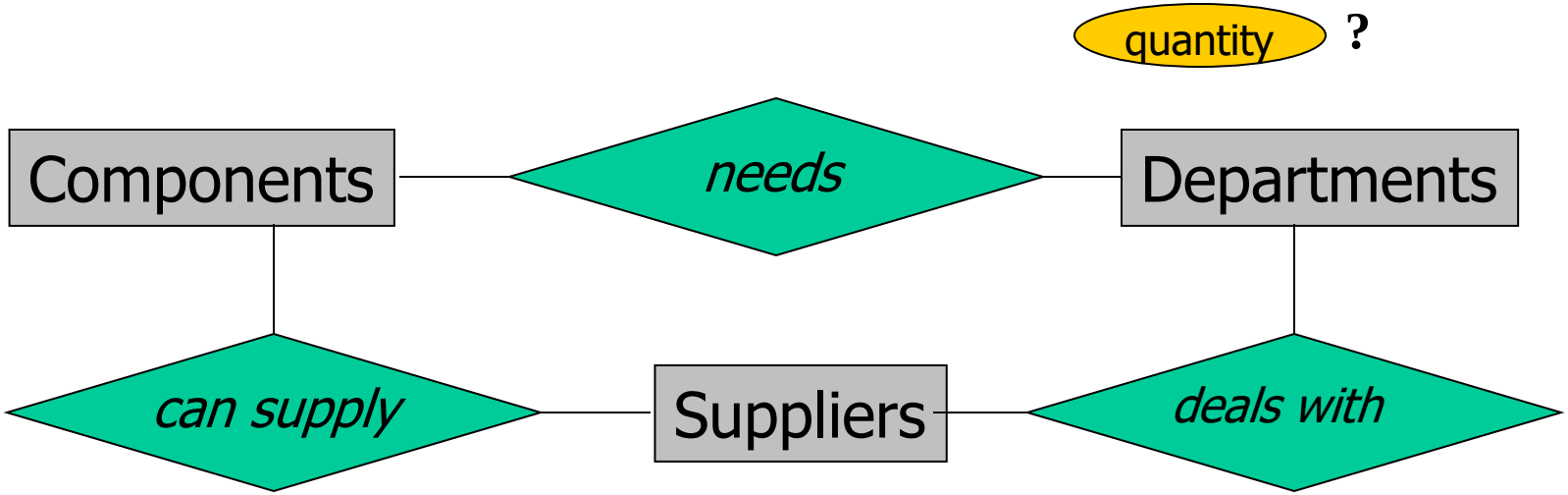
1. Binary relationship: a link between two entities
2. Ternary relationship: a link between three entities
3. Binary relations are less complex (easy to be represented and easy to be understand)
4. Sometimes it is not possible to replace a ternary relationship with two corresponding binary ones
5. Always a ternary relationship could be replaced by an entity (through a "verb to noun" transform).



Binary vs. ternary relationships

1. Where to place the quantity attribute?





The relational data model

1. First introduced in 1970 by E.F. Codd at IBM
2. Two early research projects on relational model
   - IBM System R (prototype RDBMS)
   - UC Berkeley's INGRES (academic project)
3. Today's dominant technology for DBMS
4. Hundreds of RDBMS products from PC to powerful servers

Definitions

1. Relational database: a set of **relations**
2. A **relation** is described by two parts:
   - **Instance**: a table, with rows and columns. (#Rows = cardinality, #fields = degree)
   - **Schema**: specifies name of relation, plus name and type of each attribute
3. E.g.: *Students* (sid: string, name: string, grade:real)
4. Can think of a relation as a set of rows or tuples $\{a_1, a_2, \ldots a_n\}$
   - All rows have to be distinct

Primary Key

1. A set of fields is a superkey for a relation if no two distinct tuples can have same values in all its fields (is unique)
2. A set of fields is a key if any subset of it is not unique
3. **Primary key** is a key selected by DBA to identify the relation
4. E.g.: SSN is a key for relation Person

Foreign Key

1. A **foreign key** is set of fields in one relation that is used to refer to a tuple in another relation
   - Must correspond to primary key of the second relation
   - Is like a logical pointer – implements logical links between two relations
2. **Referential integrity constraints** – prevent broken links
   - all foreign key constraints are enforced

Advantages of the relational model

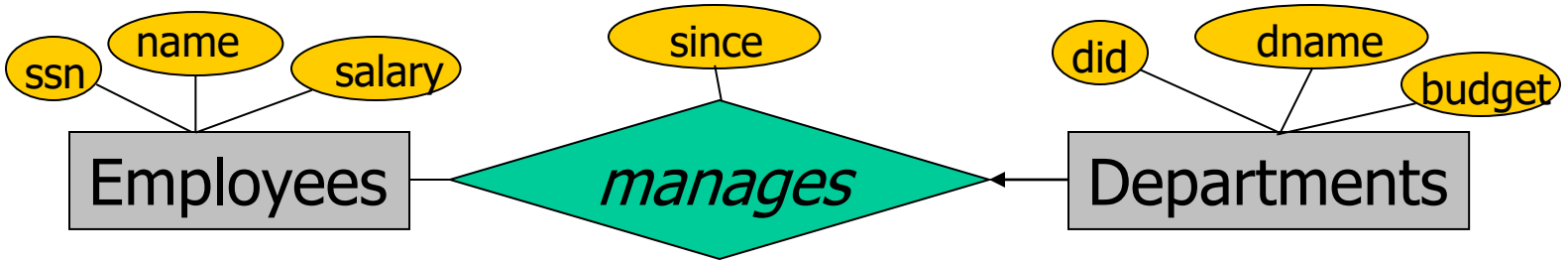
1. Simple and intuitive – currently the most widely used
2. Integrity constraints can be specified based on application semantics. DBMS checks for violations
3. Primary and foreign keys constraints plus domain constraints ensures data consistency
4. Powerful and natural query languages (e.g. SQL, QBE)
5. Rules to translate ER to relational model

Transforming ER into relational model

1. General steps:
   - Entities: each entity will be converted directly to a relation
   - Attributes of the entity: become the attributes of the relation
   - Identifier of the entity becomes a key in the relation
   - Relationships will be mapped on relations or as Foreign Keys

ER to Relational: key constraints (I)

1. E.g.: each dept has at most one manager, according to the key constraint on *manages*



ER to Relational: key constraints (II)

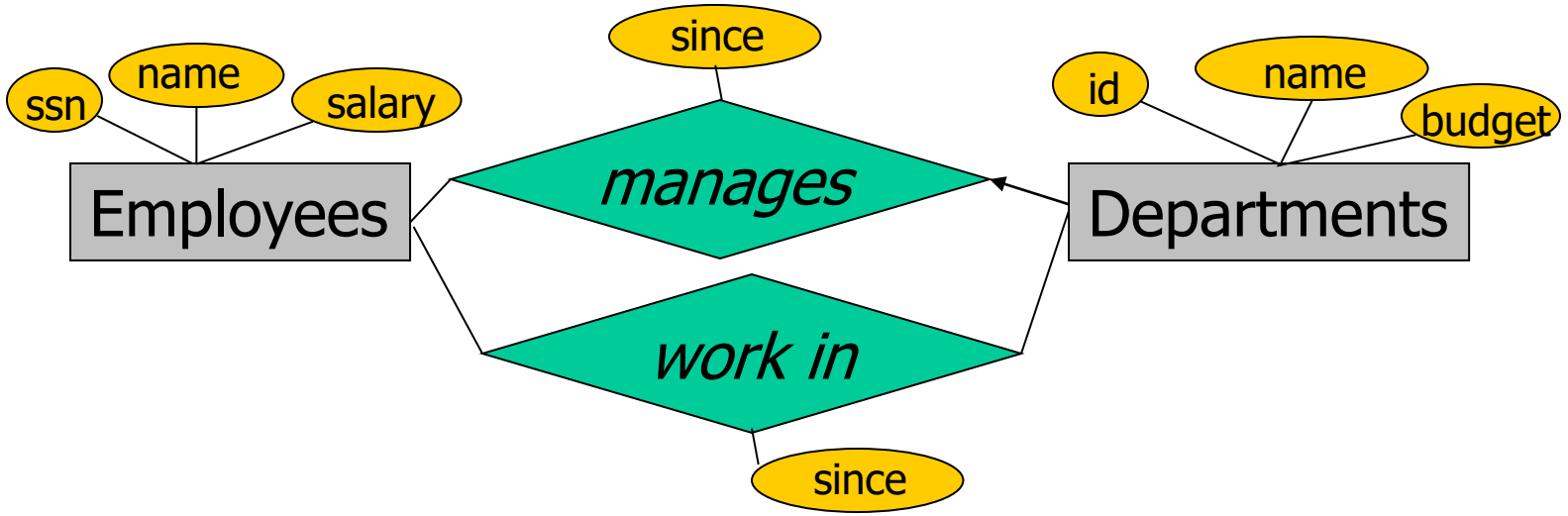
1. Map relationship to a table:
   - Note that *did* is the key now!
   - Separate tables for *Employees* and *Departments*
2. Since each department has a unique manager, we could instead combine Manages and Departments into a single relation

```
CREATE TABLE Manages (
  ssn CHAR(11),
  did INTEGER,
  since DATE,
  PRIMARY KEY (did),
  FOREIGN KEY (ssn) REFERENCES Employees,
  FOREIGN KEY (did) REFERENCES Departments
)
```

```
CREATE TABLE Departments (
  did INTEGER,
  dname CHAR(20),
  budget REAL,
  ssn CHAR(11),
  since DATE,
  PRIMARY KEY (did),
  FOREIGN KEY (ssn) REFERENCES Employees
)
```

ER to Rel.: participation constraints (I)

1. 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!)



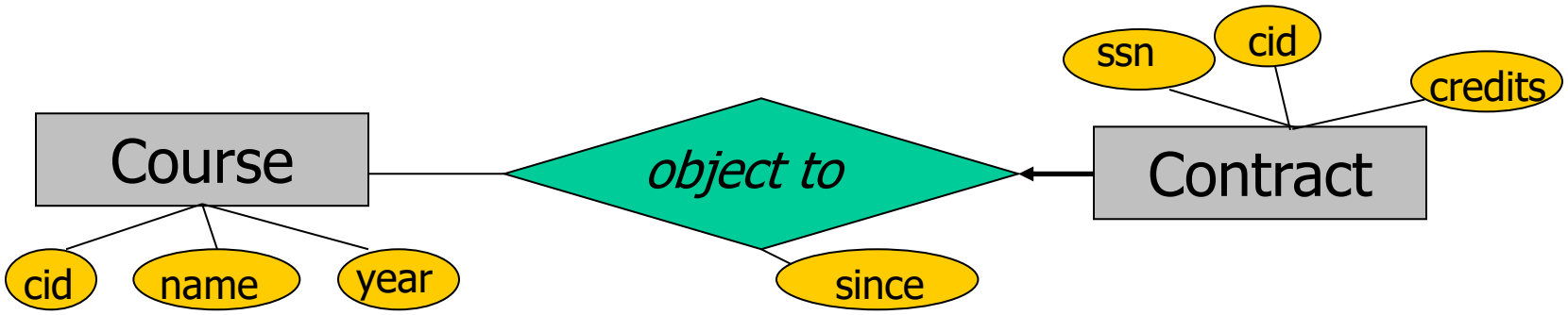
ER to Rel.: participation constraints (II)

1. We can capture participation constraints involving one entity set in a binary relationship, but little else (only through additional assertions)
   - ON DELETE NO ACTION = cannot delete the manager of an existing department from Employees. Solution?

```
CREATE TABLE  Departments (
  did  INTEGER,
  dname  CHAR(20),
  budget  REAL,
  ssn  CHAR(11) NOT NULL,
  since  DATE,
  PRIMARY KEY  (did),
  FOREIGN KEY  (ssn) REFERENCES Employees, ON DELETE NO ACTION
)
```

ER to Rel.: weak entities (I)

1. Weak entity set = a set of entities that 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



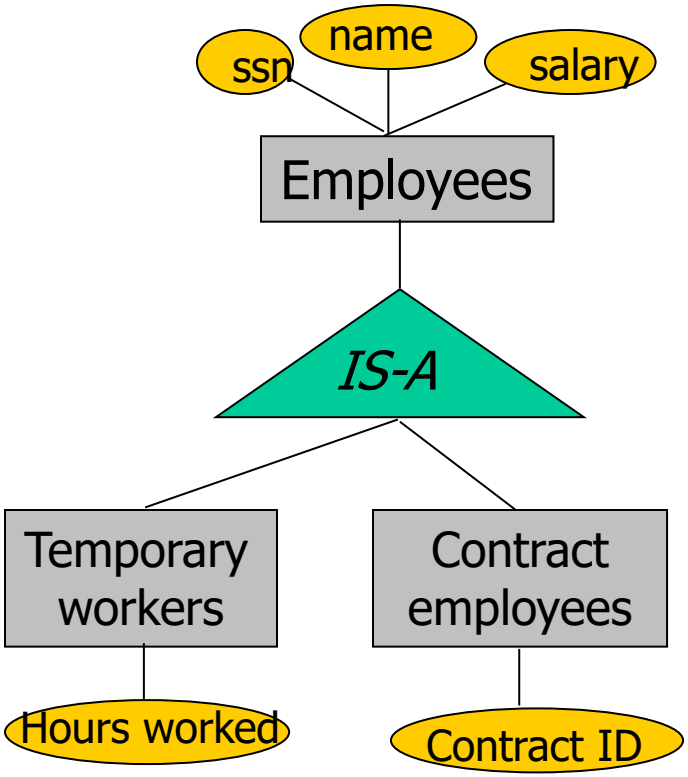
ER to Rel.: weak entities (II)

1. Weak entity set and identifying relationship set are translated into a single table
   - When the owner entity is deleted, all owned weak entities must also be deleted
   - ON DELETE CASCADE – when a referenced parent table row is removed all children are removed automatically

```
CREATE TABLE  Contract (
   cid  CHAR(3) NOT NULL,
   ssn  INTEGER,
   credits INTEGER,
   since  DATE,
   PRIMARY KEY  (cid,ssn),
   FOREIGN KEY  (cid) REFERENCES Courses, ON DELETE CASCADE
)
```

ER to Rel.: IS-A hierarchies (I)

1. IS-A imply attributes inheritance. It allows adding descriptive attributes specific to a subclass
2. If A *IS-A* B, every A entity is also considered to be a B entity
3. **Overlap constraints**: A *IS-A* B and A *IS-A* C simultaneously (allowed/disallowed)
4. **Covering constraints**: does every A to be a B or A to be a C (yes/no)



ER to Rel.: IS-A hierarchies (II)

1. Two strategies:
   i. Map each entity sets to relations: Employees, TemporaryWorkers and ContractEmployees. The last two have to contain the primary key of the superclass – *ssn* (as primary key and foreign key in the child table). The rest of Employees attributes are "inherited" (stored in the superclass table). ON DELETE CASCADE is necessary for all children.
      - Queries involving all attributes are slowed down (join ops are needed).

   ii. Map just leafs as relations: TemporaryWorkers and ContractEmployees. The Employees' attributes will be duplicated.
      - Problems if other kinds of employees exists (covering: no) or if overlapped is allowed.