CS W4111.001
Introduction to Databases
Fall 2021

Computer Science Department
Columbia University
Entity Relationship (ER) Model

- Locations
  - address: string
  - capacity: integer

- Employees
  - ssn: string
  - name: string
  - lot: string

- Works_In2

- Departments
  - did: string
  - dname: string
  - budget: real

- since: date
ER Model Basics

Same entity set could participate in different relationship sets, or in different “roles” in the same relationship set → we add “role indicators”
Constraints in ER Diagrams

- ER diagrams should express all known real-world “constraints” that the data should satisfy
- We’ll be able to express many of these constraints using ER model notation
- Some constraints cannot be expressed using ER model notation: we should still include them as plain-English annotations in the ER diagram, so that we can then remember to model them in SQL when we map the ER diagram to SQL
Key and Participation Constraints

- We have already discussed domain constraints (for attributes), and primary key and candidate key constraints (for entity sets).

- We will now discuss two kinds of constraints for relationship sets: *key constraints* and *participation constraints*.
Key Constraints

Key constraint for Manages relationship set: each department can have at most one manager
Key Constraints

Key constraint for Manages relationship set: each department can have at most one manager.

Also now: Each employee can manage at most one department; how do we update diagram?
Key Constraints in n-ary Relationship Sets with n>2

Restriction for n-ary relationship sets with n>2: can have at most one arrow (more on this next)
Key Constraints in n-ary Relationship Sets with n>2
Participation Constraints

Participation constraint for Works_In relationship set: each employee must work in at least one department.

The participation of Employees in Works_In is total.
Participation Constraints in n-ary Relationship with n>2

```
Locations
  address: string
  capacity: integer

Employees
  ssn: string
  name: string
  lot: string

Departments
  did: string
  dname: string
  budget: real

since: date
```

Diagram:
- Locations
  - address: string
  - capacity: integer
- Employees
  - ssn: string
  - name: string
  - lot: string
- Departments
  - did: string
  - dname: string
  - budget: real
- Works_In4
- since: date
Multiple Participation Constraints Allowed

Locations
- address: string
- capacity: integer

Employees
- ssn: string
- name: string
- lot: string

Departments
- did: string
- dname: string
- budget: real

Works_In5
- since: date
Key and Participation Constraints

Every employee works in at least one department?
Every department has exactly one manager?
Key and Participation Constraints

Every employee must report to someone?
Every employee reports to no more than one manager?
Weak Entity Sets

So far, the attributes of an entity set always include at least one key

But consider bank accounts:

<table>
<thead>
<tr>
<th>Accounts</th>
</tr>
</thead>
<tbody>
<tr>
<td>aid: string</td>
</tr>
<tr>
<td>name: string</td>
</tr>
<tr>
<td>balance: real</td>
</tr>
</tbody>
</table>

and their checks, with:

- number (100, 101, …)
- date check written
- amount check was written for

Attributes and key(s) of Checks entity set?
Modeling Checks and Accounts?

(Serious) problem with this diagram?
Weak Entity Sets

A weak entity is identified by its partial key (or discriminator attributes) together with the primary key of its identifying entity

- Weak entity set and identifying entity set must participate in a binary relationship set called identifying relationship set
- Weak entity set must have a key constraint and total participation in the identifying relationship set

identifying entity set  identifying relationship set  weak entity set
ISA Hierarchies (Inheritance)

- Each Hourly_Emp is an Employee
- Each Contract_Emp is an Employee
- ssn is the primary key of Hourly_Emps and Contract_Emps
- Attributes of Hourly_Emps: ssn, name, lot, hourly_wages
- Attributes of Contract_Emps: ssn, name, lot, contract_id

To specialize (or generalize) an entity set
To add attributes to specialized entity set that are not meaningful in more-general entity set
To define relationship sets with appropriate entity sets
Overlap Constraint

- **Overlap constraint:** Can an Employee be an Hourly_Emp as well as a Contract_Emp? **No** in diagram above (note type of arrow that connects the entity sets; to allow overlap, we must use separate arrows)
Overlap Constraint

- **Overlap constraint:** Can a Person be a Student as well as an Instructor? **Yes** in diagram above (note separate arrows that connect the entity sets)
Coverage (or Completeness) Constraint

Coverage (or completeness) constraint: Does every Employee have to be an Hourly_Emp or a Contract_Emp? No in diagram above (this is the default behavior)
Coverage (or Completeness) Constraint

- **Coverage (or completeness) constraint:** Does every Employee have to be an Hourly_Emp or a Contract_Emp? Does every Person have to be a Student or an Instructor? **Yes** in diagrams above (note annotation with “total” connected via dashed line)
Aggregation

“An employee monitors a specific department-project sponsorship”
Aggregation

- Useful to define relationship sets that involve other relationship sets
- The entity sets “inside” an aggregation can participate in other relationship sets, etc. normally (i.e., they are not “hidden” inside the aggregation)
Design Choices in ER Model

- Should a concept be modeled as an entity or an attribute?
- Should a concept be modeled as an entity or a relationship?
- Aggregation?
- ...

Entity vs. Attribute

Should **address** be an attribute of Employees or an entity set connected to Employees by a relationship?

**Option 1**

<table>
<thead>
<tr>
<th>Employees</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssn: string</td>
</tr>
<tr>
<td>name: string</td>
</tr>
<tr>
<td>lot: string</td>
</tr>
</tbody>
</table>

**Option 2**

<table>
<thead>
<tr>
<th>Employees</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssn: string</td>
</tr>
<tr>
<td>name: string</td>
</tr>
<tr>
<td>lot: string</td>
</tr>
</tbody>
</table>
Entity vs. Attribute

Should *address* be an attribute of Employees or an entity set connected to Employees by a relationship? Depends on intended use of addresses in database, and on data semantics:

- If address structure (city, street, etc.) is important (e.g., so that we can retrieve employees by city), address must be modeled as an entity set (attribute values are “atomic” in our version of ER model)
- If each employee can have multiple addresses, address must be an entity set (attributes cannot be set-valued in our version of ER model)
Can an employee work at the same department for more than one time period?
Entity vs. Descriptive Attribute

Employees
- ssn: string
- name: string
- lot: string

Works_In

Departments
- did: string
- dname: string
- budget: real
Relationship Set vs. Attribute?

Each employee works in exactly one department:

Can we then get rid of “Works_In” as follows?

```sql
Employees
  ssn: string
  name: string
  lot: string

Departments
  did: string
  dname: string
  budget: real

since: date
```
Relationship Set vs. Attribute?

Each employee works in exactly one department:

Can we then get rid of “Works_In” as follows?

No! Very bad modeling decision
did in Employees could now be any random string, not corresponding to any Department. Instead, we should model relationships explicitly as with Works_In above.
Summary of Conceptual Design

- Conceptual design follows requirement analysis, to yield a high-level description of data to be stored.
- ER model popular for conceptual design: ER constructs are expressive, close to how we think about applications.
- Basic ER constructs: entities, relationships, and attributes (of entities and relationships).
- Some additional ER constructs: weak entities, ISA hierarchies, and aggregation.

Note: There are many ER model variants. So that we know what you mean, please stick strictly to the version we covered in class for your homework and projects.