Database Design

- Goal: specification of database schema
- Methodology:
  - Use E-R model to get a high-level graphical view of essential components of the model and how they are related
  - Convert E-R diagram to DDL
- E-R Model is viewed as a set of
  - Entities
  - Relationships among entities
Entities

- **Entity**: an object that is involved in the enterprise
  - Ex: Mike, DBS
- **Entity Type**: set of similar objects
  - Ex: students, courses, professors
- **Attribute**: describes one aspect of an entity type
  - Ex: name, maximum enrollment, cno, etc.
Entity Type

- Entity type described by set of attributes
  - Person: $Id, Name, Address, Hobbies$

- Domain: possible values of an attribute
  - Value can be a set (in contrast to relational model)
    - (123456789, Mike, 123 College, {stamps, coins})

- Key: minimum set of attributes that uniquely identifies an entity

- Entity Schema: entity type name, attributes (and associated domain), key constraints
Entity Type cont’d

Graphical Representation in E-R diagram:

- Name
- Address
- SSN
- Hobbies

Set valued
Relationships

- **Relationship**: relates two or more entities
  - John *majors in* Computer Science
- **Relationship Type**: set of similar relationships
  - Student (entity type) related to Department (entity type) by MajorsIn (relationship type).

- **Distinction**:
  - *relation* (relational model) - set of tuples
  - *relationship* (E-R Model) – describes relationship between entities
    Both entity types and relationship types (E-R model) may be represented as relations (in the relational model)
Attributes and Roles

- **Attribute** of a relationship type describes the relationship
  - e.g., Mike majors in CS *since* 2004
    - Mike and CS are related
    - 2004 describes relationship - value of SINCE attribute of MajorsIn relationship type

- **Role** of a relationship type names one of the related entities
  - e.g., Mike is value of *Student* role, CS value of *Department* role of MajorsIn relationship type
  - (Mike, CS; 2004) describes a relationship
Graphical Representation

- Roles are edges labeled with role names (omitted if role name = name of entity set). Most attributes have been omitted.
Single-role Key Constraint

- If, for a particular participant entity type, each entity participates in *at most* one relationship, corresponding role is a key of relationship type
  - E.g., *Professor* role is unique in *WorksIn*
- Representation in E-R diagram: arrow

![Diagram](image-url)
Entity Type Hierarchies

- One entity type might be subtype of another
  - Freshman is a subtype of Student

- A relationship exists between a Freshman entity and the corresponding Student entity
  - e.g., Freshman John is related to Student John

- This relationship is called *IsA*
  - Freshman IsA Student
  - The two entities related by IsA are always descriptions of the same real-world object
Properties of IsA

- **Inheritance** - Attributes of supertype apply to subtype.
  - E.g., GPA attribute of Student applies to Freshman
  - Subtype *inherits* all attributes of supertype.
  - Key of supertype is key of subtype

- **Transitivity** - Hierarchy of IsA
  - Student is subtype of Person, Freshman is subtype of Student, so Freshman is also a subtype of Student
Advantages of IsA

- Can create a more concise and readable E-R diagram
  - Attributes common to different entity sets need not be repeated
  - They can be grouped in one place as attributes of supertype
  - Attributes of (sibling) subtypes can be different
IsA Hierarchy - Example

PERSON
  - Name
  - D.O.B.
  - SSN

EMPLOYEE
  - Department
  - Salary

STUDENT
  - GPA
  - StartDate

SECRETARY
  - Specialization

TECHNICIAN

FRESHMAN
  - Major

SOPHOMORE
  - Major

JUNIOR
  - Major

SENIOR
  - Advisor

Major

IsA

IsA
Participation Constraint

- If every entity participates in at least one relationship, a participation constraint holds:
  - e.g., every professor works in at least one department
Participation and Key Constraint

- If every entity participates in exactly one relationship, both a participation and a key constraint hold:
  - e.g., every professor works in exactly one department

E-R representation: thick line

Diagram:

```
Professor ----> WorksIn ----> Department
```
An entity type corresponds to a relation

Relation’s attributes = entity type’s attributes

- **Problem:** entity type can have set valued attributes, e.g.,
  Person: *Id, Name, Address, Hobbies*

- **Solution:** Use several rows to represent a single entity
  • (123456789, Mike, 123 College St, stamps)
  • (123456789, Mike, 123 College St, coins)

- **Problems with this solution:**
  • Redundancy
  • Key of entity type (Id) not key of relation
  • Hence, the resulting relation must be further transformed (we’ll see how later)
Representation of Relationship Types in the Relational Model

- Typically, a relationship becomes a relation in the relational model
- Attributes of the corresponding relation are
  - Attributes of relationship type
  - For each role, the primary key of the entity type associated with that role
- Example:

  - W2006Courses \((\text{CrsCode, SectNo, Enroll})\)
  - Professor \((\text{Id, DeptId, Name})\)
  - Teaching \((\text{CrsCode, SecNo, Id, RoomNo, TAs})\)
Representation of Relationship Types in the Relational Model

- Key of corresponding table = key of relation
  - Except when there are set valued attributes
  - Example: Teaching \((CrsCode, SectNo, Id, RoomNo, TAs)\)
    - Key of relationship type = \((CrsCode, SectNo)\)
    - Key of relation = \((CrsCode, SectNo, TAs)\)

<table>
<thead>
<tr>
<th>CrsCode</th>
<th>SectNo</th>
<th>Id</th>
<th>RoomNo</th>
<th>TAs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSC343</td>
<td>1</td>
<td>1234</td>
<td>BA1180</td>
<td>Joe</td>
</tr>
<tr>
<td>CSC343</td>
<td>2</td>
<td>1234</td>
<td>GB119</td>
<td>Mary</td>
</tr>
</tbody>
</table>

*Set valued*
Each role of relationship type produces a foreign key in corresponding relation
- Foreign key references table corresponding to entity type from which role values are drawn
CREATE TABLE WorksIn ( 
  Since DATE, -- attribute
  Status  CHAR (10), -- attribute
  ProfId  INTEGER, -- role (key of Professor)
  DeptId  CHAR (4), -- role (key of Department)
  PRIMARY KEY (ProfId), -- since a professor works in at most one department
  FOREIGN KEY (ProfId) REFERENCES Professor (Id),
  FOREIGN KEY (DeptId) REFERENCES Department )
CREATE TABLE Sold (  
  Price INTEGER, -- attribute  
  Date DATE, -- attribute  
  ProjId INTEGER, -- role  
  SupplierId INTEGER, -- role  
  PartNumber INTEGER, -- role  
  PRIMARY KEY (ProjId, SupplierId, PartNumber, Date),  
  FOREIGN KEY (ProjId) REFERENCES Project,  
  FOREIGN KEY (SupplierId) REFERENCES Supplier (Id),  
  FOREIGN KEY (PartNumber) REFERENCES Part (Number)  )
Representing Participation Constraints in the Relational Model

- **Inclusion dependency:** Every professor works in at least one dep’t.
  - in the relational model: (easy)
    - Professor (Id) references WorksIn (ProfId)
  - in SQL:
    - Simple case: *If ProfId is a key in WorksIn* (i.e., every professor works in exactly one department) then it is easy:
      - FOREIGN KEY  Id  REFERENCES  WorksIn (ProfId)
    - General case – *ProfId is not a key in WorksIn,* so can’t use foreign key constraint (not so easy):

```sql
CREATE ASSERTION  ProfsInDepts
    CHECK ( NOT EXISTS (  
        SELECT  *  FROM  Professor  P  
        WHERE  NOT EXISTS (  
          SELECT  *  FROM  WorksIn  W  
          WHERE  P.Id  =  W.ProfId ) ) )
```
Representing Participation Constraint in the Relational Model

- Example (can’t use foreign key in Professor if ProfId is not a candidate key in WorksIn)

<table>
<thead>
<tr>
<th>Professor</th>
<th>WorksIn</th>
</tr>
</thead>
<tbody>
<tr>
<td>1123</td>
<td>CSE</td>
</tr>
<tr>
<td>4100</td>
<td>AMS</td>
</tr>
<tr>
<td>3216</td>
<td>ECO</td>
</tr>
</tbody>
</table>

ProfId not a candidate key
Representing Participation and Key Constraint in SQL

- If both participation and key constraints apply, use foreign key constraint in entity table (but beware: if candidate key in entity table is not primary, presence of nulls violates participation constraint).

```sql
CREATE TABLE Professor (  
    Id   INTEGER,  
    ....
    PRIMARY KEY (Id),  -- Id can’t be null  
    FOREIGN KEY (Id) REFERENCES WorksIn (ProfId)  
    --all professors participate
)
```
Sometimes information can be represented as either an entity or an attribute.
Entity or Relationship?