Database Design

• **Goal**: Create an accurate model of the data needs of an enterprise in the form of a database schema
• We get to the final specification of database schema using a multi-step **design process**

1. **Requirement Analysis**:  
   – Understand the requirements of the customer
2. **Conceptual Schema Design**:  
   – Capture the data requirements in a simple, easy to understand way  
   – At this stage representation of the data will be discussed between System Analyst and End User, and then communicated to Database Designer

Conceptual Database Design

• **Question**:  
  – If the implementation platform is a relational DBMS eventually relational tables will be created, so why not use SQL DDL in the conceptual design phase?
• **Answer**:  
  – SQL DDL is too low-level a representation to easily convey the data requirements of the enterprise
• **Requirement**:  
  – Concept tools to model the data needs of an enterprise at an easy to understand, high-level  
  – A methodology to map this high level model to relational tables
Database Design Using ER Methodology

1. Create an E-R model to get a high-level graphical view of essential components of enterprise and how they are related
   - *E-R Model*: enterprise is viewed as a set of
     - *Entities*
     - *Relationships among entities*
   - The System Analyst/Database Designer would need to decide:
     - 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?

Database Design Using ER Methodology (cont’d)

2. Convert E-R model to relations
   - Rules can be followed to carry out this conversion
3. Refine the relations defined in Step 2 using *relational normalization theory*
   - Rules can be followed to carry out this refinement
   - to be covered in Chapter 6
4. Write DDL for the relations
   - Tools are available that support the ER methodology
   - Many of the steps are automated by these tools

Comments on ER Methodology

• There are alternative approaches for modeling at the conceptual design stage
  – UML (Unified Modeling Language) is a popular alternative which has the benefit of standardization (unlike ER methodology)
• The ER approach is general enough and has been employed for designing not just database schemas but also for designing other types of systems
  – But in this course we are interested in database schemas

Topics

• Conceptual Database Design
• Entity & Relationship
  – Basic concepts
    – Describing a Relationship
    – Advanced Features
• Mapping from ER Diagrams to Relations
• Design Choices in ER Diagrams
Entities

- **Entity**: an object in the enterprise that is distinguishable from other objects
- **Examples**:
  - Math 112 classroom, Math Dept, Chairman of the Math Dept, Home address of the Chairman of the Math Dept
- Don’t take use of the word “object” above to think that terminology from object-oriented modeling is being used
- That being said, an Entity is very similar to an Object (in OO sense), except an Entity does not have methods

Entity Type

- **Entity Type**: set of similar entities
  - Example: Sets of class rooms, Departments, Employees, Addresses
- **Attribute**: describes one aspect of an entity type
  - An entity type typically will have many attributes
  - Example: For a Class room, the attributes could be: building name, room number, floor, capacity
- All entities in an entity type have the same attributes
  - Example: Person: SSN, Name, Address, Hobbies
  - We will revisit this point later

Entity Type (cont’d)

- **Domain**: possible values of an attribute
  - Single-valued attribute
  - Multi-valued attribute: value can be a set
- **Example**: Person: SSN, Name, Address, Hobbies
  - (333-22-4444, John, 123 Main St, {stamps, coins})
- **Key**: minimum set of attributes that uniquely identifies an entity (candidate key)
  - For now we require that an entity has at least one key.
  - We will revisit this point later
- **Entity Schema**: entity type name, attributes (and associated domain), key constraints

E-R Diagram for Entity Types

- An entity type represented by a rectangle
- Attributes represented by ovals
  - Key underlined
  - Set-valued attributes in double-ovals
Relationships

- **Relationship**: an association between 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 of an enterprise
  - 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., John majors in CS since 2000
    - John and CS are related
    - 2000 describes relationship - value of Since attribute of MajorsIn relationship type
- **Role**: The function an entity plays in a relationship is called the entity’s role
  - e.g., John plays the Student role, CS plays the Department role in the MajorsIn relationship that exists between John and CS

Relationship Type

- We describe a relationship type by the set of attributes and roles
  - e.g., MajorsIn: Student, Department, Since
- We have simply used the name of the entity type (Student, Department) as the name of the role
- This representation is clear in most cases, but …

Need for Naming Roles

- **Problem**: A relationship can be between entities of the same entity type
  - e.g., ReportsTo relationship type relates two elements of Employee entity type:
    - Bob reports to Mary since 2000
- We do not have distinct names for the roles so the relationship type would be described as:
  - ReportsTo: Employee, Employee, Since
Role Name

• **Solution:** the role name of relationship type need not be same as name of entity type from which participants are drawn
  – ReportsTo has roles Subordinate and Supervisor and attribute Since
  – Values of Subordinate and Supervisor both drawn from entity type Employee

• Role names may be used even when the entities types involved in the relationship type are distinct
  – If the the meaning of a relationship will be clarified by using the role name

Graphical Representation

• Roles are edges labeled with role names (omitted if role name = name of entity set). Most attributes have been omitted
• In most cases: nouns -> entities, verbs -> relationships

Schema of a Relationship Type

• **Role names**, $R_i$, and their corresponding entity sets
  – Roles are required to be single valued

• **Attribute names**, $A_j$, and their corresponding domains
  – Attributes may be set valued

• **Relationship:** $<e_1, \ldots e_n; a_1, \ldots a_k>$
  – $e_i$ is an entity, a value from $R_i$'s entity set
  – $a_j$ is a set of attribute values with elements from domain of $A_j$

Topics

• Conceptual Database Design
• Entity & Relationship
  – Basic concepts
  – Describing a Relationship
  – Advanced Features
• Mapping from ER Diagrams to Relations
• Design Choices in ER Diagrams
Describing a Relationship

- Does a relationship type have other important features (in addition to attributes and roles) that must be described?
- We will introduce a terminology for describing more features of a relationship type
- **Degree**: Number of roles (equivalently: entities associated) in the relationship
  - Binary = 2, Ternary = 3
  - Recursive binary: 2 roles but same entity appears in both the roles
  - Note: Degree already described by current definition of relationship type

Cardinality Constraints

- In many relationships there are constraints on the number of entities of one type that can be associated with another entity
- Example:
  - What is the maximum number of students that can enroll in a course? (max limit)
  - How many students must enroll in a course for it to be offered? (min limit)

Cardinality Constraints (cont’d)

- Let C be an entity type and A be a relationship type in which C appears in role R
- A cardinality constraint on the role R is a statement of form \( \text{min} \ldots \text{max} \) attached to R
  - Restricts the number of relationship instances of type A in which a single entity of type C can participate in role R to a number in the interval \( \text{min} \ldots \text{max} \) (with end points included)
  - The symbol * is used to denote that there is no max limit

Cardinality Constraints in a Binary Relationship

- For a binary relationship type R between entity types A and B, the cardinality constraints are described as one of:
  1. One-to-one
     - An entity in A is associated with at most one entity in B, and an entity in B is associated with at most one entity in A
     - Example: A *Husband* is married to a *Wife*
Cardinality Constraints in a Binary Relationship (cont’d)

2. One-to-Many:
   An entity in A is associated with any number (zero or more) of entities in B. An entity in B is associated with at most one entity in A
   • Example: If a college does not allow dual majors, then the relationship MajorsIn between Department and Student is one-to-many
   • Graphical Depiction:

   ![Graphical Depiction of One-to-Many Relationship]

3. Many-to-One:
   An entity in A is associated with at most one entity in B. An entity in B is associated with any number (zero or more) of entities in A
   • Example: If a college does not allow dual majors, then the relationship MajorsIn between Student and Department is many-to-one
   • Graphical Depiction:

   ![Graphical Depiction of Many-to-One Relationship]

4. Many-to-Many:
   An entity in A is associated with any number (zero or more) of entities in B. An entity in B is associated with any number of entities in A
   • Example: A Student can be a member of many Societies. A Society can have many Students as its members

   ![Graphical Depiction of Many-to-Many Relationship]

Key of a Relationship Type

• Key of a relationship type is the minimum set of roles and attributes that uniquely identify a relationship
• Can you think of what can or cannot be the key for binary relationships of different cardinality constraints?
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., Student role is unique in MajorsIn

- Alternate E-R representation:
  - Line with arrow = 0..1
  - Line without arrow = 0 .. *

Key of a Relationship Type (cont’d)

- For binary relationships with cardinality constraint:
  1. One-to-one:
     - The entity in either role can be the key
  2. One-to-many:
     - The entity in the role on the one side cannot be the key.
     - The entity in role on the many side can be the key
  3. Many-to-one:
     - Same as one-to-many
  4. Many-to-many:
     - Neither of the two entities can be a key by itself

Participation Constraint

- If every entity participates in at least one relationship, a participation constraint holds:
  - A participation constraint of entity type E having role ρ in relationship type R states that for e in E there is an r in R such that ρ(r) = e.
  - 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
Equivalent Graphical Representations:
left and right

0..*  

0..1  

1..*  

1..1 (or just 1)

Relationships of Higher Degree

• For relationships of higher degrees (than binary) one-to-one, one-to-many, etc., constraints refer to the relationship between pairs of entity types implied by the higher degree relationship

• Example:
  - An employee works in one dept at one location. A dept can have many emps and locations, and each location can be associated with many emps and depts

  ![Diagram of Employee, WorksIn, Department, Location with relationships]

Topics

• Conceptual Database Design
• Entity & Relationship
  – Basic concepts
  – Describing a Relationship
  – Advanced Features
• Mapping from ER Diagrams to Relations
• Design Choices in ER Diagrams

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
  – Every Freshman entity is also a Student entity
IsA Hierarchy - Example

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

- IsA hierarchy is also sometimes termed a Generalization/Specialization hierarchy

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
Constraints on Type Hierarchies

- Might have associated constraints:
  - Covering constraint: Union of subtype entities is equal to set of supertype entities
    - Employee is either a secretary or a technician (or both)
  - Disjointness constraint: Sets of subtype entities are disjoint from one another
    - Freshman, Sophomore, Junior, Senior are disjoint set

Weak Entities

- Assumption till now is that attributes associated with an entity set include a key
  - This assumption does not always hold
- Example:
  - Suppose employees can purchase policies to cover their dependents
  - If an employee quits, any policy owned by the employee is terminated and all info about the policy and dependents should be removed
  - There is thus no inherent interest in a dependent himself/herself, but only because of the employee
  - Instead of identifying a dependent by a SSN, we may identify it just by a name

Weak Entities & Identifying Owner

- Weak Entity:
  - Can be identified uniquely only by considering some of its attributes in conjunction with the primary key of another entity call identifying owner
- Identifying Relationship:
  - The relationship between a weak entity and its identifying owner
- Requirement:
  - Each weak entity must be related to exactly one identifying owner
  - Weak entities always participate in their identifying relationship
  - It though may also participate in additional relationship types

Representing a Weak Entity

- Weak entity types are represented using a double box and the identifying relationship type is depicted as a double diamond
  - The weak entity will always participate in one relationship instance
- Another view is to think of the identifying relationship as a part-of relationship where the weak entity is a part-of the identifying owner
Topics

• Conceptual Database Design
• Entity & Relationship
• Mapping from ER Diagrams to Relations
• Design Choices in ER Diagrams

Mapping of Entity Types to the Relational Model

• 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
    • (111111, John, 123 Main St, stamps)
    • (111111, John, 123 Main St, coins)
  – Problems with this solution:
    • Redundancy
    • Key of entity type (Id) not key of relation
    • Hence, the resulting relation must be further transformed (Chapter 6)

Mapping of Relationship Types to 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:

  Courses (CrsCode, SectNo, Enroll)
  Professor (Id, DeptId, Name)
  Teaching (CrsCode, SectNo, Id, RoomNo, TAs)

  TeachingCourses Professor
  DeptId
  Name
  RoomNo
  CrsCode
  Enroll
  SectNo
  Id
  TAs

  CSE305 1 1234 Hum 22 Joe
  CSE305 1 1234 Hum 22 Mary

Set valued
Representation in SQL

- 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
  - Furthermore all entity types involved in a relationship type much appear in a relationship instance, the attributes corresponding to foreign key references must have NOT NULL constraint
- This NOT NULL constraint would be implicit when the foreign keys appear in the primary key of the relationship

Example 1

```
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)  )
```

Example 2

```
CREATE TABLE WorksIn (                      
  Since DATE,                -- attribute
  Status CHAR (10),      -- attribute
  ProfId INTEGER,       -- role (key of Professor)
  DeptId CHAR (4) NOT NULL,       -- role (key of Department)
  PRIMARY KEY (ProfId),  
  FOREIGN KEY (ProfId) REFERENCES Professor (Id),  
  FOREIGN KEY (DeptId) REFERENCES Department )
```
Alternative Mapping for Single Role Key

- Instead of creating a separate table for the relationship, use foreign key constraint in entity table

```sql
CREATE TABLE Professor (  
  Id INTEGER,  
  Since DATE,  
  DeptId CHAR (4),  
  PRIMARY KEY (Id),  
  FOREIGN KEY (DeptId) REFERENCES Department  
)
```

- Only drawback: For professors not in a relationship with any department, added columns will be filled with NULL values

Representing Participation Constraints in the Relational Model

- Simple case: participation and single key role
  - Every professor works in exactly one department
  - `WorksIn` relationship is many-to-one from `Professor` to `Department`
  - Same as previous slides, but in addition create a NOT NULL constraint on the attribute corresponding to foreign key

Participation and Single Key Role Constraints

- Alternative mapping for the simple case: participation and single key role
  - Create table `WorksIn` with `Prof.Id` and `Department.DeptId` as foreign key and `WorksIn.ProfId` as primary key
  - Now in `Professor` specify that `Id` references the foreign key `WorksIn.ProfId`
  - This ensures participation and key constraint:
    - `Prof.Id` references `WorksIn`, so every `Prof.Id` has to appear in `WorksIn`

Example: DDL

```sql
CREATE TABLE prof(  
  profid integer,  
  name char(20),  
  primary key (profid));
create table dept(  
  deptid integer, primary key (deptid));
create table works_in(  
  profid references prof (profid),  
  deptid references dept (deptid),  
  primary key (profid));
ALTER TABLE prof ADD CONSTRAINT worksin_fk
  FOREIGN KEY (profid) REFERENCES works_in (profid)
  INITIALLY DEFERRED;
-- also need to add a NOT NULL constraint on WorksIn.DeptId
```
General Participation Constraint
- General participation constraint: every professor must work in a department
- What if we specify that Professor.Id references the foreign key WorksIn.ProfId?
  - That would ensure that every Professor appears at least once in WorksIn
- Can’t do that: ProfId is not a candidate key in WorksIn

Representing Participation Constraint in the Relational Model
- Conclusion:
  - There are participation constraints that cannot be modeled by simple foreign key constraints
  - Enforcing such constraints requires the use of Assertions or Check constraints or Triggers

Mapping Type Hierarchies to the Relational Model
- Supertypes and subtypes can be realized as separate relations
- Recall every subtype entity is also a supertype entity
  - Need a way of identifying subtype entity with its (unique) related supertype entity
  - Choose a candidate key and make it an attribute of all entity types in hierarchy

General Representation
- Translated by adding the primary key of supertype to all subtypes. Plus foreign key from subtypes to the supertype.
**General Representation (cont’d)**

- Example:
  - A person who is both an employee and a student have the Name and DOB attributes stored once in Person relation with additional attributes in Employee and Student relations

<table>
<thead>
<tr>
<th>Person</th>
<th>Employee</th>
<th>Student</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSN</td>
<td>Name</td>
<td>DOB</td>
</tr>
<tr>
<td>1234</td>
<td>Mary</td>
<td>1950</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SSN</th>
<th>Department</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234</td>
<td>Accounting</td>
<td>35000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SSN</th>
<th>GPA</th>
<th>StartDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234</td>
<td>3.5</td>
<td>1997</td>
</tr>
</tbody>
</table>

**Alternative for Representing Disjoint IsA Relationships**

- Create a single relation whose attribute set is the union of attribute sets of all the entities involved
  - Add another attribute to indicate the original entity type of each tuple in the relation
  - Value of attributes that do not belong to the original entity type will be null in the corresponding tuples in the relation

| SSN GPA StartDate Major Status |
|-------------------------------|----------------|-------------|
| 111 3.6 08/28/04 Freshman     |
| 222 3.3 08/31/02 CSCI Junior  |

**Alternative for Mapping Covering IsA Relationships**

- Create relations for the subtypes
  - Don’t create relation for the supertype at all
  - Pro: Data for a subtype not distributed across multiple tables
  - Con: If the IsA Relationship is covering but not disjoint, redundant data is stored

<table>
<thead>
<tr>
<th>Employee</th>
<th>Student</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSN Name DOB</td>
<td>Department</td>
</tr>
<tr>
<td>1234 Mary 1950</td>
<td>Accounting</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SSN Name GPA StartDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234 Mary 3.5 1997</td>
</tr>
</tbody>
</table>

**Mapping Weak Entity Types**

- Weak entity types always:
  - Participate in many-to-one relationship
  - With key constraint
  - And total participation

- Use the approach for similar relationships
  - Create a table for the weak entity type
  - Don’t create a table for the identifying relationship type
  - Store the primary key of the identifying owner as a foreign key in the weak entity
    + Foreign key must have NOT NULL constraint
    - ON DELETE CASCADE
Example: Mapping Weak Entity Type

```
CREATE TABLE Dependent (  
  Name char(50),  
  ... -- other attributes  
  EmpId integer NOT NULL,  
  PRIMARY KEY (EmpId, Name),  
  FOREIGN KEY (EmpId) REFERENCES Employee(Id)  
  ON DELETE CASCADE;
```

Topics

- Conceptual Database Design
- Entity & Relationship
- Mapping from ER Diagrams to Relations
- Design Choices in ER Diagrams

Design Choices

- There is considerable freedom in deciding whether a datum should be an entity, a relationship of an attribute
- Once that issue is settled, there are still different choices in expressing relationships

Entity or Attribute?

- Sometimes information can be represented as either an entity or an attribute.
Entity or Attribute? (cont’d)

- To a certain extent, it is a matter of taste
- But generally keeping the number of entity types and relations small is a good idea
- Think if the datum has an internal structure of its own
  - E.g., Semester may have its own attributes, such as Start_date, End_date, Holidays
  - If yes, creating an entity type makes sense. Otherwise, model this datum as an attribute

Entity or Relationship?

- Again some of the comments for the case of Entity/Attribute decision are relevant here
- If an entity type $E$ has total participation and key role in a relationship type $R$, it might be possible to remove $R$, move its attributes to $E$ and make $E$ a relationship between the remaining entity types in $R$
  - Not always possible since $E$ may be involved in other relationship types in addition to $R$

(Non-) Equivalence of Diagrams

- Transformations between binary and ternary relationships.
Example: Information Loss

- From the 3 binary relationship, we cannot infer that Acme doesn’t supply Hammer to the Garage Project

<table>
<thead>
<tr>
<th>Sold (as ternary relationship)</th>
<th>Note: Attributes Date and Price not shown</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProjId</td>
<td>SupplierId</td>
</tr>
<tr>
<td>Garage</td>
<td>Acme</td>
</tr>
<tr>
<td>Garage</td>
<td>Harry’s</td>
</tr>
<tr>
<td>Deck</td>
<td>Acme</td>
</tr>
</tbody>
</table>

Sold

<table>
<thead>
<tr>
<th>ProjId</th>
<th>SupplierId</th>
</tr>
</thead>
<tbody>
<tr>
<td>Garage</td>
<td>Acme</td>
</tr>
<tr>
<td>Garage</td>
<td>Harry’s</td>
</tr>
<tr>
<td>Deck</td>
<td>Acme</td>
</tr>
</tbody>
</table>

Uses

<table>
<thead>
<tr>
<th>ProjId</th>
<th>PartNum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Garage</td>
<td>Screw</td>
</tr>
<tr>
<td>Garage</td>
<td>Hammer</td>
</tr>
<tr>
<td>Deck</td>
<td>Hammer</td>
</tr>
</tbody>
</table>

Supplies

<table>
<thead>
<tr>
<th>ProjId</th>
<th>SupplierId</th>
</tr>
</thead>
<tbody>
<tr>
<td>Garage</td>
<td>Acme</td>
</tr>
<tr>
<td>Garage</td>
<td>Harry’s</td>
</tr>
<tr>
<td>Deck</td>
<td>Acme</td>
</tr>
</tbody>
</table>

Summary

- In this chapter we looked at the Entity-Relationship methodology for conceptual design of database schemas
- ER diagramming techniques capture the conceptual model at a high level
- Rules were introduced to go from an ER model to DDL for the table
- However, not all the needed information may not be captured in an ER model and the tables that we come up with may have certain deficiencies
  - We will study functional dependencies and normalization to address these points

Information Loss

- The new diagram results in information loss
  - In the original diagram for a given combination of Project and Part, we can determine the Supplier
  - In the second diagram, this is no longer possible
- This problem is called a navigation trap
  - Starting with a given entity and moving along the triangle formed by the 3 relationships we might end up with a different entity