Conceptual and Logical Database Design
Abdussalam Alawini
Lesson 1

Learning Objectives

After this lecture, you will be able to:

• Design conceptual models for real-world applications using the Entity-Relationship (ER) Diagrams and the Unified Modeling Language (UML)
• Determine the cardinality of relationships from the application requirements
• Design multiway relationships and convert them to binary relationships.
• Model Weak Entities and Subclasses in ER and UML
• Develop conceptual models based on ER Design principles
• Translate ER/UML diagrams to relational schema

1. Overview of Database Design

The database design process consists of four steps.

1. Conceptual design The database design process starts by modeling the application data we want to store in the database using the entity-relationship diagram ¹ or the Unified Modeling Language ² a modeling tool used for designing software systems. The main focus of this step is to identify real-world entities and relationships among them.

2. Logical design: The second step in the design process is to transform the ER or UML design to a relational schema. The result of this step is a set of DDL SQL commands creating the database


4. Physical Database Design and Tuning: Consider typical workloads; (sometimes) modify the database design; select file types and indexes.

In this lesson, we will focus on the first two steps.

¹ ER Model:
• Proposed by Peter Chen in 1976
• Gives us a language to specify
• What information the database must hold
• How the bits of information relate to one another

² UML Model:
• UML is a standard language for designing software systems, but also used for DB design
• created by the Object Management Group (OMG)
• UML 1.0 specification draft was proposed to the OMG in early 1997.
2. Conceptual DB Design using ER & UML

Entity-relationship diagrams consist of entity sets connected via relationship sets.

- **Entity**: Real-world object distinguishable from other objects. An entity is described using a set of attributes.
- **Entity Set**: A collection of similar entities. E.g., all employees. (often referred to as just entity, which blurs the distinction between type and collection)
- **Relationship**: Association among two or more entities. E.g., Kristin’s home department is Research & Development.
- **Relationship Set**: Collection of similar relationships. E.g., Home (often referred to as just relationship)

Figure 1 shows an example of an entity-relationship diagram that models the design of a company database with entity sets, such as Employee, Department and Project, and relationship sets, such as Home.

![Figure 1: An example of an entity relationship diagram of a company following the original ER syntax](image)

### 2.1 RELATIONSHIPS

**Definition 1** Relationships. A relationship between two sets $A$ and $B$ is defined as a subset of the product of $A$ and $B$ ($A \times B$).

In ER and UML, a relationship between two or more entities is defined as the subset of the product of the participating entities. Figure 2 shows an example sponsor relationship between the Project and Department entities.
2.1.1 Multiplicity of Relationships

Multiplicity of relationships defines how many entities can participate in a relationship. A relationship can be one-one, many-one, or many-many.

- **One-one:** Every entity on the left-hand side (LHS) is connected to at most one entity on the right-hand side (RHS) and every entity on the RHS is connected to at most one entity on the LHS.
- **Many-one:** Every entity on the left-hand side (LHS) is connected to at most one entity on the right-hand side (RHS).
- **Many-many:** Every entity on the left-hand side (LHS) is connected zero or more entities on the right-hand side (RHS) and every entity on the RHS is connected to zero or more entities on the LHS.

Figure 3 shows the UML version of the entity-relationship diagram shown in Figure 1. In UML, entities are called classes and each class has a set of attributes. A relationship is represented by a labeled line that connects two classes. Multiplicity of relationships is represented by \( n.m \) where \( n \) is the minimum and \( m \) is the maximum number of entities that can participate in a relationship. For example, in Figure 4 0..1 from the Department side of home relationship indicates that an Employee can have zero or one home department. The

\(^3\) Multiplicity can be shown with arrows. An arrow means at most 1. A special case of a constraint on the degree of the relation (e.g., at most 10)

\(^4\) UML only supports binary relationships.
0..* from the Employee side indicates that a department can be home to zero to any number of employees.

Relationships in ER and UML can have attributes that describe the relationship. For instance, the home relationship can have start-date as an attribute that stores the date in which an employee joined a department.

We can also add role names to relationship sets. The manager relationship in Figure ?? has two role names: manages and managed-by. The managed-by role can be read as follows: A Department is managed by exactly one manager.

2.1.2 Modeling Constraints
Finding constraints is part of the modeling process. ER and UML tools can model relational constraints, such as
• **Keys**: attributes that identify entities in an entity set. E.g., social security number uniquely identifies a person.

• **Referential integrity constraints**: relationship-based constraints, e.g. if you work for a company, it must exist in the database.

2.1.3 *Multiway Relationships*

Multiway relationships are relationships that connect three or more entities. For example, the relationship *Purchase* in Figure 8 connects *Product*, *Store* and *Person* entities. Multiway relationships are modeled mathematically as subset of the product of the participating entities.
Motivation

Exercise 1

What does the arrow from Rental to Movie in Figure ?? mean?
2.2 Weak Entity Sets

**Definition 2** Weak Entity Set. Entity set $E$ is weak if in order to identify entities of $E$ uniquely, we need to follow one or more many-one relationships from $E$ and include the key of the related entity sets.

Occasionally, entities of an entity set need “help” to identify them uniquely. A weak entity set is an entity that cannot be uniquely identified by its attributes. The team entity in Figure 9 represents an example of a weak entity because a team cannot be identified by the sports name and the team number. For instance, "football one" or "tennis two" are not enough to identify a team. We have to follow the affiliation relationships to the university entity to identify a team uniquely. We use the key of the university entity to be able to identify the team. "University" is a "supporting entity set" for "Team."

2.2.1 Multiway to Binary Relationships

The process of converting a multiway relationship to binary starts by first converting the multiway relationship set to an entity set. Sec-
ond, we create a binary relationship between the new entity set and each entity set that participated in the multiway relationship. Third, the new entity set will be defined as a weak entity to all participating entities. Figure 10 version of the multiway relationship shown in Figure 8.

2.3 Subclasses in ER and UML

Subclasses are used to model inheritance between entities. Figure 11 shows an example of subclass relationships. The Educational Product and Software Products are both subclasses to the superclass Product. In this example, we see that all products share a set of common attributes (price, name and category). Thus, we can define a superclass (called Product) and include all the common attributes among products. Then we can define subclasses, such as Educational Product, to include the attributes that are unique to the subclass (e.g., age-group). We indicate a subclasses relationship by using the is-a triangle between the subclass and the superclass. Triangle should always point to the super class.
Subclasses

- “Isa” triangles indicate the subclass relationship and point to the superclass.
- Subclasses form a tree. I.e., no “multiple inheritance”.
- Why subclasses? Unnecessary to add redundant properties to the root entity set that don’t apply to many of the entities.

Logical Design: Converting ER/UML to Relational Schema

Step 1: Translate each entity set into a table

Entity sets can be represented as a table in the relational model. The key for the entity becomes a key for the table. Figure 12 shows the DDL SQL code for creating a table for the Employee entity set.

CREATE TABLE Employee
(SSN CHAR(11) NOT NULL,
E-Name CHAR(20),
Office INTEGER,
PRIMARY KEY (SSN))

Step 2: Translate each many-to-many relationship set into a table

Figure 11: An example of subclass relationships

Figure 12: Translate each entity set into a table
Create a table for each many-many relationship. The attributes for this table are foreign keys to the primary keys of the participating entities. Below is the relational schema for the many-many Assignment relationship shown in Figure 13.

**Assignment(P-Number, SSN)**

- P-Number is a foreign key for Project
- SSN is a foreign key for Employee

**Project(P-Number, P-Due-Date)**

**Employee(SSN, E-Name, Office)**

![Figure 13: Many-many relationships are translated into a single table](image)

**Step 3: Create a foreign key for a 1-to-many relationship set**

Create a foreign key constraint for each one-many relationship set. Below is the relational schema for the many-one Manager relationship shown in Figure 14.

**Project(P-number, P-name, Due-Date, MgrSSN)**

**Employee(SSN, E-Name, Office)**

**MgrSSN** is a foreign key (referencing the Employee relation) We can also convert the many-one relationship into a table.

**Project(P-number, P-name, Due-Date)**

**Employee(SSN, E-Name, Office)**

**Manager(P-number, SSN)**

![Figure 14: Many-one relationships are translated into a single table or defined as a foreign key constraint](image)
**Step 4**: Translate each weak entity set along with its supporting relationship set into a table

Weak entity sets and supporting relationship sets are translated into a single table. Must include key of supporting entity set, as a foreign key. Here’s the DDL SQL code for converting the weak entity set shown in Figure 15:

```sql
CREATE TABLE Insurance_Policy (  
  dep-name CHAR(20),  
  cost REAL,  
  ssn CHAR(11) NOT NULL,  
  PRIMARY KEY (dep-name, ssn),  
  FOREIGN KEY (ssn) REFERENCES Employee,  
  ON DELETE CASCADE)
```

**Step 5**: Converting Subclass Structures to Relations

There are three approaches for converting subclass entity sets to relational schema.

1. **Follow the E/R viewpoint**. For each entity set E in the hierarchy, create a relation that includes the key attributes from the root and any attributes belonging to E.

2. **Treat entities as objects belonging to a single class**. For each possible subtree that includes the root, create one relation, whose schema includes all the attributes of all the entity sets in the subtree.
3. **Use null values.** Create one relation with all the attributes of all the entity sets in the hierarchy. Each entity is represented by one tuple, and that tuple has a null value for whatever attributes the entity does not have. 

Translating ER to a relational schema

1. Create a table for each entity set.
2. Create a table for each many-many relationship. The attributes for this table are foreign keys to the primary keys of the participating entities.
3. Create a foreign key constraint for each one-many relationship set.
4. Create a table for each weak entity and its supporting relationship set. You must include the key of the supporting entity as part of the key for this table.
5. Convert subclass structures to relationships using ER, Object or Null approach.

Read textbook section 4.6 for examples and comparison of approaches