Lecture 3: Data Modeling Using the Entity-Relationship Model.

Ref. Chapter 12

Prepared by L. Nouf Almujally & Aisha AlArfaj
Rev. by L. Fatimah Alhayan
The Process of Database Design

- Conceptual Design
- Logical Design (Relational Model)
- Physical Design
Conceptual Database Design

- ER Diagram is used.
- Does not include implementation details.
- Users and developers can easily understand and discuss.
- Used to ensure all user requirements are met.
- Enable designers to concentrate on specifying the properties of the data, without being concerned about storage details.
The **Entity-Relationship (ER) model** is a popular high-level conceptual data model.

Based on a perception of a real world that consists of a collection of basic objects, called **entities**, and of the **relationships** among these objects.

The ER model can be directly translated into relational tables.

The **Entity-Relationship Diagram (ERD)** is a graphical model for representing the conceptual model for the data.

E/R models the DB using three element types:
- Entities
- Attributes
- Relationships
Example of ER Diagram
Entities and Attributes

The ER Model describes data as entities, relationships and attributes.

1- Entity:

- The basic object that the ER model represents is an entity, which is a “thing” in the real world with an independent existence.
- Could be an object with a physical existence (student, car) or conceptual existence (company, course).
- Each entity has attributes.

Rectangles represent entities
Entity Types

- **Entity type**: set of entities of the same type with the same basic attributes are grouped or typed into an entity type.

- For example, the **Student entity type**

  ![Entity Diagram]

  - Student1
  - Student2
  - Student3

- **More Example:**

  - Course
  - book
  - Doctor
Entities with values of their attributes

An entity will have a value for each of its attributes; e.g. A specific student entity may have Name=’Nora”, Age = 21, ID = 1111 ...etc.

The attribute values that describe each entity become a part of the data stored in the database.
Entities and Attributes

- **2- Attribute:**
  - Property describing an entity.
  - E.g. Student name, age, ID, ...etc.

- **Oval** represent attributes

```
Student
```
```
st_name
```
Types of Attributes - Simple Versus Composite

• **Simple:**
  - An attribute that has a single atomic value with an independent existence.
    - E.g. ID, Salary, Gender

• **Composite:**
  - An attribute composed of multiple components, each of which represent more basic attributes with independent meaning.
  - Composite attribute indicated with sub ovals.
    - E.g. Name (FirstName, MiddleName, LastName).

[Diagram showing the relationship between the attributes representing the STUDENT table.]
Types of Attributes - Simple Versus Composite

- Composition may form a hierarchy where some components are themselves composite.
- The value of a composite attribute is the concatenation of the values of its simple attributes.
Types of Attributes - Single-Valued Versus Multivalued

- **Single-valued:**
  - An attribute that holds one single value for a single entity.
  - E.g. ID.

- **Multi-valued:**
  - An attribute that holds multiple values for the same entity.
  - use double line
  - E.g:
    - colors attribute for a car - \{Color\}
    - College degrees attribute for a person - \{college degree\}. 
Types of Attributes - Stored Versus Derived

- **Stored:**
  - E.g. Birth_date.

- **Derived:**
  - An attribute that represents a value that is *derivable* from the value of a related attribute or set of attributes.
  - derived attributes use dotted line

- **For Example:**
  - The *age* can be determined from the *current date* and the value of the *birth date* of a person, so:
    - *age* is a derived attribute and *birth_date* is a stored attribute.
  - *Total_cost* is derived from *quantity* *unit_price*
Some attribute values can be derived from related entities;
- e.g. an attribute `Number_of_employees` of a DEPARTMENT entity can be derived by counting the number of employees related to that department.
Keys of Entity

- **Candidate Key**: is a minimal set of attributes whose values uniquely identify an entity in the set. It cannot contain NULL.
- There could be more than one candidate key; if so, we select one of them as the primary key and the other will be an Alternate Key.

```
Element ( Symbol , name , atomic_no)
```

```
Primary Key
```

```
Alternate Key
```
Keys of Entity

- **Primary Key**
  - The candidate key that is selected to uniquely identify an entity type.
  - primary key attribute indicated with **underline**

- **Choice of Primary Key (PK) is based on:**
  - Attribute length
  - Number of attributes required
  - Certainty of uniqueness

- **PK’s may be**
  - **simple PK** (single attribute)
    - CUSTOMER (custno, ....)
    - STUDENT (ID, .....)
  - **composite PK** (multiple attributes)
    - ORDERLINE (orderno, prodno, quantity,...)
Drawing Entities and Attributes

- Rectangles represent entities
- Oval represent attributes
- derived attributes use dotted line
- multi-valued attributes use double line
- primary key attribute indicated with underline
- Composite attribute indicated with sub ovals.
Relationship

- Association between two (or more) entities
- Relationships represented by diamond

Examples
- Students Register for Subjects
- School Has Staff
- Driver Commits Traffic Offence
- Customer Orders Product
- Supplier Supplies Parts To Projects

- Relationship type (relationship set)
  - Set of similar associations among entity sets.
Attributes of Relationship Types

Relationship types can also have attributes.

- E.g. customer **orders** a product
  - quantity is attribute of **orders** relationship

```
  customer --orders-- product
     |   |        |
     v   v        v
      |   |        |
      v   v        v
    quantity
```

- E.g. Employee **WORKS-ON** Project
  - To record the number of hours per week that an employee works on a particular project. The attribute **Hours** can be included to the **WORKS-ON** relationship type.

```
  Employee --works-on-- Project
     |                     |
     v                     v
      |                     |
      v                     v
    Hours
```
Types of Relationships

- **Degree of relationship**: is the number of entity types that participate in it
  - **UNARY** – between entities from one entity set
  - **BINARY** – between entities from two entity sets
  - **N-ARY** – among entities from more than two entity sets
Unary(Recursive) Relationship

- **Unary relationship** is a relationship type where the same entity type participates more than once in a different role.
- It is a **Recursive relationship**.

![Employee Supervises Diagram](image-url)
Relationship Degree - Unary relationship

EMPLOYEE

SUPERVISION

Relationship Degree - Unary relationship

EMPLOYEE

SUPERVISION
Roles

- **Role** indicates the purpose that each participating entity type plays in a relationship. (e.g. prerequisite, requester)
Roles

- **Role** can be used when two entities are associated through more than one relationship to classify the purpose of each relationship.
A relationship of degree two (2 entity types) are binary.

ER Model
Relationship Degree - Binary relationship

EMPLOYEE    WORKS_FOR    DEPARTMENT

\[ e_1 \rightarrow r_1 \rightarrow d_1 \]
\[ e_2 \rightarrow r_2 \rightarrow d_1 \]
\[ e_3 \rightarrow r_3 \rightarrow d_2 \]
\[ e_4 \rightarrow r_4 \rightarrow d_2 \]
\[ e_5 \rightarrow r_5 \rightarrow d_3 \]
\[ e_6 \rightarrow r_6 \rightarrow \vdots \]
\[ e_7 \rightarrow r_7 \rightarrow \vdots \]
A relationship of degree three (3 entity types) are ternary. e.g. Supplying a part to a project by a supplier.
Relationship Degree - Ternary relationship

SUPPLIER

s₁
s₂
...

PART

p₁
p₂
p₃
...

SUPPLY

r₁
r₂
r₃
r₄
r₅
r₆
r₇
...

PROJECT

j₁
j₂
j₃
...

Graph showing the ternary relationship with nodes representing suppliers, parts, supplies, and projects, connected by edges indicating the relationship degrees.
Relationship Degree

**UNARY**: One entity set, recursive.

**BINARY**: Two entity sets linked (mostly used).

**TERNARY**: Three entity sets linked.
Example Relationships

- Supervisee
- Supervisor
- Staff
- Owner
- Owns
- Property_for_Rent
- Client
- SetsUp
- Interview

Diagram showing relationships between entities.
Constraints on Relationships

- There are two types of relationship constraints:
  - Cardinality ratio.
  - Participation.
Cardinality Ratio

**Cardinality ratio** Express the number of relationship an entity can participate in.

Relationships can be classified as:

- one-to-one (1:1)
- one-to-many (1:M)
- Many-to-one (M:1)
- many-to-many (M:N)
One-to-one Relationship (1:1)

Each professor manages at most one department; and each department is managed by only one professor.
One-to-Many Relationship (1:M)

Each professor teaches many sections; but each section is taught by only one professor (or at most one professor)
Many-to-One Relationship (M:1)

Each professor works in **only one** Department; but each Department has **many** professors (many professors work in one department)
Many-to-Many Relationship

Each student can enroll in many courses; and each course can be enrolled by many students.
Multiplicity

**Multiplicity** is the number (range) of possible entities that may relate to a single association through a particular relationship

- Takes the form (min#, max#)
Multiplicity

- Each Professor **may** manage 0 to 1 department,
- Each Department **should** be managed by only one Professor.
Each Professor may teach 0 to 5 Sections,
Each Section should be taught by only one Professor.
* = many .. Use it if the number is not specified
Multiplicity

- Each student should enroll in 1 to 7 Courses,
- Each Course may be enrolled by 0 to 100 students.
- * = many .. Use it if the number is not specified
Participation

- **Participation constraints** determine whether all or only some entities participate in a relationship.

- Two type of participation:
  - **Mandatory** (Total)
  - **Optional** (Partial)
Participation

- **Optional (partial) (0:*)** if an entity’s existence **does not require** a corresponding entity in a particular relationship. (Not existence-dependent).
  - eg, an employee *may* not have a spouse
  - *not every* staff manages a department

  Partial participation is represented by single line

- **Mandatory (total) (1:*)** if an entity’s existence **requires** the existence of an associated entity in a particular relationship (existence-dependent).
  - eg, *every* spouse *must* be that of an employee
  - *Every* department is managed by some staff.

  Mandatory participation is represented by double line
Examples

employee → 1 <-- has → 1 → spouse

CUSTOMER → 1 <-- place → N → ORDER

STUDENT → M <-- enrol → N → SUBJECT
Strong & Weak Entity Type

- A **strong entity** type is an entity that have a key attribute. (I.e. It exists **independently** of other entity types)

- A **weak entity** type is an entity that does not have a key attribute. (I.e. Its existence **depend** on some other entity type)

- The entity type which the weak entity type depends on is called the **identifying entity type** (**owner**). e.g EMPLOYEE

- Weak entities are identified by the combination of:
  - A **partial key** of the weak entity type.
  - The **primary key of** the owner entity type.
Representing Weak Entities and Relationships

- Weak entity is indicated by a double line rectangle, and weak relationship is represented by a double line diamond.

- Participation of weak entity in weak relationship is always mandatory.

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

The relationship between the weak entity and its owner entity is called *weak relationship*.

- Eg: Employee Has Dependent
  - Dependent can be regarded as a weak entity—only when the attributes of Dependent can not uniquely identify a dependent. Eg, Dependent has the attributes d_name and d_birthdate only, but two employees may have dependents with the same name and date of birth. However employee_no and d_name uniquely identify each dependent.
  - Employee is the owner entity
  - Has is the weak relationship
Weak Entity Type

EMPLOYEE

SSN

Salary

DEPENDENTS-OF

N

DEPENDENT

Name

Sex

BDate

Relationship
Modelling with E-R Diagram

- List the major entity sets in the system
- Represent the entity sets graphically by a rectangle
- Search for relationships between the entities and represent them graphically by a diamond
- Add attributes, underline PK attribute/s
- Model relationship cardinality and participation
Summary of Notation For ER-Diagrams

- **Symbol**
  - ENTITY TYPE
  - WEAK ENTITY TYPE
  - RELATIONSHIP TYPE
  - IDENTIFYING RELATIONSHIP TYPE
  - ATTRIBUTE
  - KEY ATTRIBUTE
  - MULTIVALUED ATTRIBUTE
  - COMPOSITE ATTRIBUTE
  - DERIVED ATTRIBUTE
  - TOTAL PARTICIPATION OF E2 in R
  - CARDINALITY RATIO 1:N FOR E1:E2 IN E
  - STRUCTURAL CONSTRAINT (min, max) ON PARTICIPATION OF E IN R
Example (Company)

The COMPANY database keeps track of a company’s employees, departments and projects.

The company is organized into departments. Each department has a unique number, name, several locations, and a particular employee who manages the department. We keep track of the start date when that employee began managing the department.

Each department may control many projects. However, Each project should be controlled by one department only. Project has a unique number, name, and a single location.
Example (Company)

We store each employee’s name, social security number, address, salary, sex, and birth date. An employee should be assigned to one department only but each department must have many employees.

Each employee should work on several projects and each project should run by many employees. We keep track of the number of hours per week that an employee works on each project.

We also keep track of the direct supervisor of each employee.

We want to keep track of the dependents of each employee for insurance purposes. We keep each dependent’s first name, sex, birth date, and relationship to the employee.
Tool For Drawing ER Diagrams In Oracle software

Microsoft Office Visio Viewer 11.0
Microsoft
Visualizes files created on Visio 2007 in an Internet Explorer 5.0 window.

KickStart 1.0
Westfaro Corporation
Kickstart is a scriptable RAD code generator for any programming language.

PERT Chart EXPERT 2.6
Critical Tools, Inc.
it is a project management software used to create PERT charts.

Springboard 0.8
Six Mile Creek Systems LLC
A fast, easy-to-use tool for drawing a storyboard for a movie production.
References