Chapter Three

Conceptual Database Design
(ER modeling)
Agenda (Chapter Three)

- Overview - database design
- Conceptual Design (E-R Modeling)
- Structural Constraints
- EER - Generalization and Specialization
- Reducing E-R Model to Table (logical Design)
Objective

- Learn skills and methods of Modeling organizational data using Entity relationship diagram
- Implementing the rules of relational data model through the process of normalization
Overview-database design

- Database design is the process of coming up with different kinds of specification for the data to be stored in the database.

- The database design part is one of the middle phases we have in information systems development where the system uses a database approach.
Design is the part on which we would be engaged to describe

- how the data should be perceived at different levels and
- finally how it is going to be stored in a computer system
Continued...

- Information System development with Database application consists of several tasks which include:
  - Planning of Information systems Design
  - Requirements Analysis,
  - **Database Design (Conceptual, Logical and Physical Design)**
    - Interface, program etc designs are also there
  - Implementation
  - Operation and Support
From these different phases, the prime interest of this course will be the Database Design part which is again sub divided into other three sub-phases.

These sub-phases are:

- **Conceptual Design**
- **Logical Design**, and
- **Physical Design**
In general, one has to go back and forth between these tasks to refine a database design, and decisions in one task can influence the choices in another task.

In developing a good design, one should answer such questions as:

- What are the relevant Entities for the Organization
- What are the important features of each Entity
- What are the important Relationships
- What are the important queries from the user
- What are the other requirements of the Organization and the Users
Conceptual Database Design

- Conceptual design revolves around discovering and analyzing organizational and user data requirements.
- The important activities are to identify:
  - Entities
  - Attributes
  - Relationships
  - Constraints
- And based on these components develop the ER model using ER diagram components.
Designing conceptual model for the database is not a one linear process but an iterative activity where the design is refined again and again.

Important steps then could be

- Identification of components
- Use of notations to model
- Review, refine and validate
Before working on the conceptual design of the database, one has to know and answer the following basic questions

- What are the *entities* and *relationships* in the enterprise?

- What *information* about these entities and relationships should we store in the database?
Cont...

- What are the *integrity constraints* that hold?
  - Constraints on each data with respect to update, retrieval and store

- Represent this information pictorially in *ER diagrams*, then map ER diagram into a relational schema.
Modeling tools

- Be professional - use the right tool symbols are languages
- There are many ER diagramming tools
  - Rational Rose
  - Microsoft Visio
  - Oracle Designer
  - Power Designer
E-R Model Constructs (recall)

- **Entities:**
  - Entity instance—person, place, object, event, concept (often corresponds to a row in a table)
  - Entity Type—collection of entities (often corresponds to a table)

- **Relationships:**
  - Relationship instance—link between entities (corresponds to primary key-foreign key equivalencies in related tables)
  - Relationship type—category of relationship...link between entity types

- **Attribute**—property or characteristic of an entity or relationship type (often corresponds to a field in a table)
To identify the entities, attributes, relationships, and constraints on the data, there are different set of methods used during the analysis phase.

These include information gathered through

- *Interviewing end users individually and in a group*
- *Questionnaire survey*
- *Direct observation*
- *Examining different documents*

Generally understand the business rules
Cont...

- Business Rules
  - Statements that define or constrain some aspect of the business
  - Assert business structure
  - Control/influence business behavior
  - Expressed in terms familiar to end users
  - Automated through DBMS software
Cont...

- A Good Business Rule is:
  - Declarative—what, not how
  - Precise—clear, agreed-upon meaning
  - Atomic—one statement
  - Consistent—internally and externally
  - Expressible—structured, natural language
  - Distinct—non-redundant
  - Business-oriented—understood by business people
Cont...

- A Good Data Name is:
  - Related to business, not technical, characteristics
  - Meaningful and self-documenting
  - Unique
  - Readable
  - Composed of words from an approved list
Describing the business in terms of data (Data Definitions; modeling)

- Explanation of a term or fact
  - Term—word or phrase with specific meaning
  - Fact—association between two or more terms

- Guidelines for good data definition
  - Gathered in conjunction with systems requirements
  - Accompanied by diagrams
  - Iteratively created and refined
  - Achieved by consensus
Constructs Identification-ERD

Entity; Attribute; Relationship
Steps: Creating an ERD

- Here are the steps you may follow to create an entity-relationship diagram.

- **Identify Entities:**
  - These are typically the nouns and noun-phrases in the descriptive data produced in your analysis.
  - Do not include entities that are irrelevant to your domain.

- **Find Relationships:**
  - Discover the semantic relationships between entities.
  - These are usually the verbs that connect the nouns.
  - Not all relationships are this blatant, you may have to discover some on your own.
  - The easiest way to see all possible relationships is to build a table with the entities across the columns and down the rows, and fill in those cells where a relationship exists between entities.
Cont...

- **Draw Rough ERD**
  - Draw the entities and relationships that you have discovered.

- **Fill in Cardinality**
  - Determine the cardinality of the relationships. You may want to decide on cardinality when you are creating a relationship table.

- **Define Primary Keys**
  - Identify attribute(s) that uniquely identify each occurrence of that entity.

- **Draw Key-Based ERD**
  - Now add them (the primary key attributes) to your ERD.
  - Revise your diagram to eliminate many-to-many relationships, and tag all foreign keys.
• **Identify Attributes**
  • Identify all entity characteristics relevant to the domain being analyzed.

• **Map Attributes**
  • Determine to which entity each characteristic belongs.
  • Do not duplicate attributes across entities.

• **Draw fully attributed ERD**
  • Now add these attributes.
  • The diagram may need to be modified to accommodate necessary new entities.

• **Check Results**
  • Is the diagram a consistent and complete representation of the domain.
Cont...

- In this all tasks
  - Keep in mind – Business rule
  - Business functions are expressed in terms of business rules
Example business function-to-data entity matrix

<table>
<thead>
<tr>
<th>Business Functions</th>
<th>Customer</th>
<th>Product</th>
<th>Raw Material</th>
<th>Order</th>
<th>Work Center</th>
<th>Work Order</th>
<th>Invoice</th>
<th>Equipment</th>
<th>Employee</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business Planning</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Product Development</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Materials Management</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Order Fulfillment</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Order Shipment</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Sales Summarization</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Production Operations</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Finance and Accounting</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

X = data entity (column) is used within business function (row)
Sample E-R Diagram - what it looks like
What Should an Entity Be?

• SHOULD BE:
  • An object that will have many instances in the database
  • An object that will be composed of multiple attributes
  • An object that we are trying to model

• SHOULD NOT BE:
  • A user of the database system
  • An output of the database system (e.g., a report)
Example of inappropriate entities

- **TREASURER**
  - Receives
  - Manages

- **EXPENSE REPORT**
  - Summarizes

- **ACCOUNT**
  - Is_charged

- **EXPENSE**
  - Is_charged

**System user**

**System output**

**Inappropriate entities**

**Appropriate entities**
Attributes

- Attribute—property or characteristic of an entity or relationship type
- Classifications of attributes:
  - Required versus Optional Attributes
  - Simple versus Composite Attribute
  - Single-Valued versus Multivalued Attribute
  - Stored versus Derived Attributes
  - Identifier Attributes
Identifiers (Keys)

- **Identifier (Key)**—An attribute (or combination of attributes) that uniquely identifies individual instances of an entity type
- **Simple** versus **Composite** Identifier
- **Candidate Identifier**—an attribute that could be a key…satisfies the requirements for being an identifier
- Will not be null
A composite attribute

An attribute broken into component parts

Entity with multivalued attribute (Skill) and derived attribute (Years_Employed)

Multivalued
an employee can have more than one skill

Derived
from date employed and current date
Simple and composite identifier attributes

(a) Simple identifier attribute

(b) Composite identifier attribute

The identifier is boldfaced and underlined
More on Relationships

• Relationship Types vs. Relationship Instances
  • The relationship type is modeled as lines between entity types...the instance is between specific entity instances

• Relationships can have attributes
  • These describe features pertaining to the association between the entities in the relationship

• **Associative Entity** — combination of relationship and entity
Relationship types and instances

a) Relationship type

b) Relationship instances
Degree of relationships

- One entity related to another of the same entity type
- Entities of two different types related to each other
- Entities of three different types related to each other
Examples of relationships of different degrees

a) Unary relationships
Examples of relationships of different degrees (cont.)

b) Binary relationships

- One-to-one relationship: EMPLOYEE to PARKING SPACE
- One-to-many relationship: PRODUCT LINE to PRODUCT
- Many-to-many relationship: STUDENT to COURSE
Examples of relationships of different degrees (cont.)

c) Ternary relationship

Note: a relationship can have attributes of its own
Structural Constraints
(Cardinality Constraints)

- Cardinality Constraints - the number of instances of one entity that can or must be associated with each instance of another entity
  - **Minimum Cardinality**
    - If zero, then optional
    - If one or more, then mandatory
  - **Maximum Cardinality**
    - The maximum number of tuples taking part in the relationship
Cont...

- One-to-One
  - Each entity in the relationship will have exactly one related entity

E.g.: Relationship *Manages* between *EMPLOYEE* and *BRANCH*

The multiplicity of the relationship is:
- One branch can only have one manager
- One employee could manage either one or no branches
One-to-Many

An entity on one side of the relationship can have many related entities, but an entity on the other side will have a maximum of one related entity.

E.g.: Relationship *Leads* between *EMPLOYEE* and *PROJECT*

The multiplicity of the relationship:

- One employee may Lead no, one or more project(s)
- One project is Lead by one staff
Cont...

- Many-to-Many

  - Entities on both sides of the relationship can have many related entities on the other side

E.g.: Relationship *Teaches* between *INSTRUCTOR* and *COURSE*

*The multiplicity of the relationship*

- One Instructor Teaches one or more Course(s)
- One Course Thought by no, one or more Instructor(s)
Examples of cardinality constraints

a) Mandatory cardinalities

A patient history is recorded for one and only one patient

A patient must have recorded at least one history, and can have many
b) One optional, one mandatory

A project must be assigned to at least one employee, and may be assigned to many.

An employee can be assigned to any number of projects, or may not be assigned to any at all.
Examples of cardinality constraints (cont.)

a) Optional cardinalities

A person is married to at most one other person, or may not be married at all.
Examples of multiple relationships
Employees and departments

Entities can be related to one another in more than one way
Multivalued attributes can be represented as relationships and entity

<table>
<thead>
<tr>
<th>ATTRIBUTE</th>
<th>RELATIONSHIP &amp; ENTITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>COURSE</td>
<td>Has_prerequisites</td>
</tr>
<tr>
<td>Course_ID</td>
<td>Pre-Req_Course_ID</td>
</tr>
<tr>
<td>Course_Title</td>
<td>Is_prerequisite_for</td>
</tr>
</tbody>
</table>

**simple**

**composite**
Identifying relationship

Strong entity

Weak entity

Employee_ID
Employee_Name
Associative Entities

• An **Entity**—which has attributes

• A **relationship**—links entities together

• When should a *relationship with attributes* instead be an *associative entity*?
  • *All relationships for the associative entity should have many cardinality*
  • *The associative entity could have meaning independent of the other entities*
  • *The associative entity preferably has a unique identifier, and should also have other attributes*
  • *The associative entity may participate in other relationships other than the entities of the associated relationship*
  • *Ternary relationships should be converted to associative entities*
What if we want to keep the **date** the employee completed the course
A binary relationship with an attribute

Here, the date completed attribute pertains specifically to the employee’s completion of a course…it is an attribute of the relationship

relationship
An associative entity (CERTIFICATE)

**Associative entity is like a relationship with an attribute**, but it is also considered to be an entity in its own right.

Note that the *many-to-many* cardinality between entities will be replaced by *two one-to-many* relationships with the associative entity.
An associative entity – bill of materials structure

This could just be a relationship with attributes… it’s a judgment call
Ternary relationship as an associative entity

Business Rules

1. Each vendor can supply many parts to any number of warehouses, but need not supply any parts.

2. Each part can be supplied by any number of vendors to more than one warehouse, but each part must be supplied by at least one vendor to a warehouse.

3. Each warehouse can be supplied with any number of parts from more than one vendor, but each warehouse must be supplied with at least one part.
Notation Standards

ER Diagramming
Related diagramming convention techniques:

- *Peter Chen notation*
- *Crow’s Foot Notation*
- Bachman notation
- EXPRESS
- IDEF1X[4]
- Martin notation
- (min, max)-notation of Jean-Raymond Abrial in 1974
- UML class diagrams
Basic E-R notation

Entity symbols

A special entity that is also a relationship

Relationship degrees specify number of entity types involved

Relationship cardinalities specify how many of each entity type is allowed

Relationship symbols

Attribute symbols

Entity types

Strong

Weak

Associative

Attribute symbols

ENTITY NAME
Identifier
Partial identifier
Optional
[Derived]
{Multivalued}
Composite( , )

Attribute symbols

Relationship symbols

Relationship degrees

Unary

Binary

Ternary

Relationship cardinality

Mandatory one

Mandatory many

Optional one

Optional many
Crow’s Foot Notation

**Chen**

```
Person ----> N ------ Birthplace ------+ 1 ----> Location
```

**IDEFIX**

```
Person      ----> Location
```

**Bachman**

```
Person ----> Birthplace of ----> Location
```

**Martin / IE / Crow's Foot**

```
Person ----> Birthplace of ----> Location
```

**Min-Max / ISO**

```
Person (1,1) Born in Birthplace of (0,N) ----> Location
```

**UML**

```
<<Entity>> Person 0..N < Birthplace of > 1 <<Entity>> Location
```

---

**Artists**

```
Artist Performed Song
```

---

**58**
Symbols

<table>
<thead>
<tr>
<th>Entity</th>
<th>Relationship line</th>
<th>Relationship</th>
<th>Option symbol</th>
<th>One (1) symbol</th>
<th>Many (M) symbol</th>
<th>Composite entity</th>
<th>Weak entity</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Chen" /></td>
<td><img src="image2" alt="Chen" /></td>
<td><img src="image3" alt="Chen" /></td>
<td><img src="image4" alt="Chen" /></td>
<td><img src="image5" alt="Chen" /></td>
<td><img src="image6" alt="Chen" /></td>
<td><img src="image7" alt="Chen" /></td>
<td><img src="image8" alt="Chen" /></td>
</tr>
<tr>
<td><img src="image1" alt="Crow's Foot" /></td>
<td><img src="image2" alt="Crow's Foot" /></td>
<td><img src="image3" alt="Crow's Foot" /></td>
<td><img src="image4" alt="Crow's Foot" /></td>
<td><img src="image5" alt="Crow's Foot" /></td>
<td><img src="image6" alt="Crow's Foot" /></td>
<td><img src="image7" alt="Crow's Foot" /></td>
<td><img src="image8" alt="Crow's Foot" /></td>
</tr>
<tr>
<td><img src="image1" alt="Rein85" /></td>
<td><img src="image2" alt="Rein85" /></td>
<td><img src="image3" alt="Rein85" /></td>
<td><img src="image4" alt="Rein85" /></td>
<td><img src="image5" alt="Rein85" /></td>
<td><img src="image6" alt="Rein85" /></td>
<td><img src="image7" alt="Rein85" /></td>
<td><img src="image8" alt="Rein85" /></td>
</tr>
<tr>
<td><img src="image1" alt="IDEF1X" /></td>
<td><img src="image2" alt="IDEF1X" /></td>
<td><img src="image3" alt="IDEF1X" /></td>
<td><img src="image4" alt="IDEF1X" /></td>
<td><img src="image5" alt="IDEF1X" /></td>
<td><img src="image6" alt="IDEF1X" /></td>
<td><img src="image7" alt="IDEF1X" /></td>
<td><img src="image8" alt="IDEF1X" /></td>
</tr>
</tbody>
</table>
The University of Toronto has several departments. Each department is managed by a chair, and has at least one professor working for it. Professors must be assigned to one, but possibly more departments. At least one professor teaches each course, but a professor may be on sabbatical and not teach any course. Each course may be taught more than once by different professors. We know of the department name, the professor name, the professor employee id, the course names, the course schedule, the term/year that the course is taught, the departments the professor is assigned to, the department that offers the course.
The University of Toronto has several departments. Each department is managed by a chair, and has at least one professor working for it. Professors must be assigned to one, but possibly more departments. At least one professor teaches each course, but a professor may be on sabbatical and not teach any course. Each course may be taught more than once by different professors. We know of the department name, the professor name, the professor employee id, the course names, the course schedule, the term/year that the course is taught, the departments the professor is assigned to, the department that offers the course.
Solution

• **Identify Entities**
  
  • These are typically the nouns and noun-phrases in the descriptive data produced in your analysis. Do not include entities that are irrelevant to your domain.
  
  • The entity candidates are *departments, chair, professor, course, and course section*.
  
  • Since there is only one instance of the University of Toronto, we exclude it from our consideration.
### Finding Relationships

<table>
<thead>
<tr>
<th>department</th>
<th>chair</th>
<th>professor</th>
<th>course</th>
</tr>
</thead>
<tbody>
<tr>
<td>department</td>
<td>managed by</td>
<td>is assigned</td>
<td>offers</td>
</tr>
<tr>
<td>chair</td>
<td>manages</td>
<td>is a</td>
<td></td>
</tr>
<tr>
<td>professor</td>
<td>assigned to</td>
<td></td>
<td>teaches</td>
</tr>
<tr>
<td>course</td>
<td>offered by</td>
<td>taught by</td>
<td></td>
</tr>
</tbody>
</table>
Draw Rough ERD

Draw the entities and relationships that you have discovered.
Fill in Cardinality: Determine the cardinality of the relationships. You may want to decide on cardinality when you are creating a relationship table.
Cont...

- Here we must eliminate many-to-many relationships, and collapse one-to-one relationships where it makes sense.
- For example, the chair, without any behaviours, is really just an attribute of a department. So we can remove it as an entity and later add it as an attribute.
Cont...

- **Define Primary Keys**
  - Identify attribute(s) that uniquely identify each occurrence of that entity.

<table>
<thead>
<tr>
<th>department</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>dept-prof</td>
<td>dept. name</td>
</tr>
<tr>
<td></td>
<td>employee id</td>
</tr>
<tr>
<td>professor</td>
<td>employee id</td>
</tr>
<tr>
<td>prof-course</td>
<td>employee id</td>
</tr>
<tr>
<td></td>
<td>course name</td>
</tr>
<tr>
<td>course</td>
<td>course name</td>
</tr>
</tbody>
</table>
Cont...

- **Draw Key-Based ERD**
  - Now add them (the primary key attributes) to your ERD.
  - Revise your diagram to
  - eliminate many-to-many relationships, and tag all foreign keys
Cont...

- **Identify Attributes**
  - Identify all entity characteristics relevant to the domain being analyzed.
  - Excluding those keys already identified:
    - Schedule, Term, Professor name, Department Chair (which is an employee ID, a foreign key to Professor)

- **Map Attributes**
  - Determine to which entity each characteristic belongs.
  - Do not duplicate attributes across entities.
  - If necessary, contain them in a new, related, entity. Schedule -- Prof-Course, Term -- Prof-Course, Chair -- Department
Cont...

• **Draw fully attributed ERD**
  • Now add these attributes
  • The diagram may need to be modified to accommodate necessary new entities.
Cont...

- Check Results
  - Is the diagram a consistent and complete representation of the domain.
  - If not - fix it.
Example 2

ERD
A company has several departments. Each department has a supervisor and at least one employee. Employees must be assigned to at least one, but possibly more departments. At least one employee is assigned to a project, but an employee may be on vacation and not assigned to any projects. The important data fields are the names of the departments, projects, supervisors and employees, as well as the supervisor and employee number and a unique project number.
Conceptual Database Design
(EER modeling)
Cont...

- **Enhanced E-R (EER) Models**
  - Object-oriented extensions to E-R model
  - EER is important when:
    - Some tuples have special attribute to be described with
    - A relationship between two entities is partial

- **EER Concepts**
  - Generalization…...Superclass/Supertype
  - Specialization………Subclass/Subtype
  - Attribute Inheritance
    - Subtype entities inherit values of all attributes of the super-type
    - An instance of a subtype is also an instance of the super-type
  - Constraints on specialization and generalization
Supertypes and Subtypes

- **Subclass/Subtype**
  - An entity type whose tuples have attributes that distinguish its members from tuples of the generalized or Superclass entities.
  - Subclasses can be either mutually exclusive (disjoint) or overlapping (inclusive).
  - A single subclass may inherit attributes from two distinct superclasses.
  - A mutually exclusive category/subclass is when an entity instance can be in only one of the subclasses.
    - E.g.: An EMPLOYEE can either be SALARIED or PART-TIMER but not both.
  - An overlapping category/subclass is when an entity instance may be in two or more subclasses.
    - E.g.: A PERSON who works for a university can be both EMPLOYEE and a STUDENT at the same time.
Supertypes and Subtypes...

- **Superclass / Supertype**
  - An entity type whose tuples share common attributes.
  - Attributes that are shared by all entity occurrences (including the identifier) are associated with the supertype.
  - Is the generalized entity
Relationship of Superclass and Subclass

- An instance can not only be a member of a subclass. i.e. Every instance of a subclass is also an instance in the Superclass.
- A member of a subclass is represented as a distinct database object, a distinct record that is related via foreign key attribute to its super-class.
- The relationship between a subclass and a Superclass is an “IS A” or “IS PART OF” type.
  - Subclass IS PART OF Superclass
  - Manager IS AN Employee
Notation

- All subclasses or specialized entity sets should be connected with the superclass using a line to a circle where there is a subset symbol indicating the direction of subclass/superclass relationship.

![Diagram showing subclass relationships]

Figure 1
Basic notation for supertype/subtype notation

a) EER notation

- **SUPERTYPE**
  - Attributes shared by all entities (including identifier)
  - Relationships in which all instances participate

- **SUBTYPE 1**
  - Attributes unique to subtype 1
  - Relationships in which only specialized versions participate

- **SUBTYPE 2**
  - Attributes unique to subtype 2
  - Relationships in which only specialized versions participate

- General entity type
- Specialized versions of supertype
- and so forth
Cont...

- **Attribute Inheritance**
  - An entity that is a member of a *subclass inherits all the attributes* of the entity as a member of the superclass.
  - The entity also *inherits all the relationships* in which the superclass participates.
  - An entity may have *more than one subclass* categories.
  - All subclasses of a superclass *share a common unique identifier* attribute (primary key). i.e.
Consider the **EMPLOYEE** supertype entity shown above.

This entity can have several different subtype entities (for example: **HOURLY** and **SALARIED**), each with distinct properties not shared by other subtypes.

But whether the employee is **HOURLY** or **SALARIED**, same attributes (**EmployeeId**, **Name**, and **DateHired**) are shared.

The Supertype **EMPLOYEE** stores all properties that subclasses have in common.

And **HOURLY** employees have the unique attribute **Wage** (hourly wage rate), while **SALARIED** employees have two unique attributes, **Pension** and **Salary**.
Generalization and Specialization

- **Generalization**: The process of defining a more general entity type from a set of more specialized entity types.  
  *BOTTOM-UP*

- **Specialization**: The process of defining one or more subtypes of the supertype and forming supertype/subtype relationships.  
  *TOP-DOWN*
Example of generalization

a) Three entity types: CAR, TRUCK, and MOTORCYCLE

All these types of vehicles have common attributes
Example of generalization (cont.)

b) Generalization to VEHICLE supertype

So we put the shared attributes in a supertype

Note: no subtype for motorcycle, since it has no unique attributes
Example of specialization

a) Entity type PART

- **Part_No**
- **Description**
- **Qty_on_Hand**
- **Location**
- **Routing_Number**
- **{Supplier (Supplier_ID, Unit_Price)}**

- Only applies to manufactured parts
- Applies only to purchased parts
Example of specialization (cont.)

b) Specialization to MANUFACTURED PART and PURCHASED PART

Created 2 subtypes
Example of specialization (cont.)

b) Specialization to MANUFACTURED PART and PURCHASED PART

Note: multivalued attribute was replaced by an associative entity relationship to another entity

Created 2 subtypes
ERD-to- Relation Mapping
Cont...

- Why
- ER is
  - very high level closer to end users understanding
  - Not directly possible to implement
  - Needs transformation to fit to all business rules required
Cont...

- How
  - The first step is converting the conceptual design to a form suitable for relational logical model, which is in a form of tables- mapping
  - The next is Normalization (applying the rules in relational data model)- Chapter 4
ER Diagrams into Relations

Mapping Regular Entities to Relations/Tables

Every Entity will be a Table

1. Simple attributes: E-R attributes map directly onto the relation
2. Composite attributes: Use only their simple, component attributes
3. Multivalued Attribute—Becomes a separate relation with a foreign key taken from the superior entity
Mapping a regular entity

(a) CUSTOMER entity type with simple attributes

(b) CUSTOMER relation
Mapping a composite attribute

(a) CUSTOMER entity type with composite attribute

(b) CUSTOMER relation with address detail
Mapping an entity with a multivalued attribute

Multivalued attribute becomes a separate relation with foreign key

One-to-many relationship between original entity and new relation
Mapping Weak Entities

- Becomes a separate relation with a foreign key taken from the superior/Strong entity
- Primary key composed of:
  - Partial identifier of weak entity and/or
  - Primary key of identifying relation (strong entity)
Example of mapping a weak entity

a) Weak entity DEPENDENT
Example of mapping a weak entity (cont.)

b) Relations resulting from weak entity

NOTE: the domain constraint for the foreign key should NOT allow null value if DEPENDENT is a weak entity.
Mapping Relationships

The cardinality will be the basis for mapping relationship

- **One-to-One:**
  - Primary key of one of the side can be taken into the other as a foreign key.
  - Recommendation: Primary key on the optional side becomes a foreign key on the mandatory side

- **One-to-Many:**
  - Primary key on the one side becomes a foreign key on the many side

- **Many-to-Many:**
  - Create a *new relation* with the primary keys of the two entities becomes the foreign key in the new table. The foreign keys will as well be part of the primary key in the new table.
Example of mapping a 1:M relationship

a) Relationship between customers and orders

Note the mandatory one

b) Mapping the relationship

Again, no null value in the foreign key…this is because of the mandatory minimum cardinality
Example of mapping an M:N relationship

a) Many to Many relationship (M:N)

The *Completes* relationship will need to become a separate relation
Example of mapping an M:N relationship (cont.)

b) Three resulting relations

- EMPLOYEE
  - Employee_ID
  - Employee_Name
  - Birth_Date

- CERTIFICATE
  - Employee_ID
  - Course_ID
  - Date_Completed

- COURSE
  - Course_ID
  - Course_Title

Composite primary key
Foreign key
New intersection relation

106
Example of mapping a binary 1:1 relationship

a) One to one relationship (1:1)

Often in 1:1 relationships, one direction is optional.
Example of mapping a binary 1:1 relationship (cont.)

b) Resulting relations

Foreign key goes in the relation on the optional side, Matching the primary key on the mandatory side
Example of mapping an associative entity

a) An associative entity
Example of mapping an associative entity (cont.)

b) Three resulting relations

Composite primary key formed from the two foreign keys
Example of mapping an associative entity with an identifier

a) SHIPMENT associative entity
Example of mapping an associative entity with an identifier (cont.)

b) Three resulting relations

![Diagram showing three resulting relations between CUSTOMER, SHIPMENT, and VENDOR entities]

Primary key differs from foreign keys
ER Diagrams into Relations
Mapping Unary Relationships

- **One-to-One:**
  - Recursive foreign key in the same relation

- **One-to-Many:**
  - Recursive foreign key in the same relation

- **Many-to-Many:**
  - Create a new relation and the primary key of the entity will be taken as a foreign key twice playing both roles in the recursive relationship. (N.B: make sure that the two foreign key names are different from one another even though they refer to same primary key value in the main table)
Mapping a unary 1:N relationship

(a) EMPLOYEE entity with unary relationship

(b) EMPLOYEE relation with recursive foreign key
Mapping a unary M:N relationship

(a) Bill-of-materials relationships (M:N)

(b) ITEM and COMPONENT relations
ER Diagrams into Relations

*Mapping Ternary (and n-ary) Relationships*

- Create a new relation and post the primary key of all participating entities into the new table.
- Include if there are any attributes that would be meaningful only if the relationship takes place.
a) PATIENT TREATMENT Ternary relationship with associative entity

Mapping a ternary relationship

![ER Diagram](image)
Mapping a ternary relationship (cont.)

b) Mapping the ternary relationship PATIENT TREATMENT

Remember that the primary key MUST be unique.

This is why treatment date and time are included in the composite primary key.

But this makes a very cumbersome key…

It would be better to create a surrogate key like Treatment#.
EER Diagrams into Relations

Mapping Supertype/Subtype Relationships

- *A separate relation for supertype and for each subtype*
- Supertype attributes (including the subtype discriminator) go into supertype relation
- Subtype attributes go into each subtype; primary key of supertype relation also becomes primary key of subtype relation
Supertype/subtype relationships

- EMPLOYEE
  - Employee_Number
  - Employee_Name
  - Address
  - Date_Hired
  - Employee_Type
  - Employee_Type =
    - "H" (HOURLY Employee)
    - "S" (SALARIED Employee)
    - "C" (CONSULTANT)
  - Hourly_Rate
  - Annual_Salary
  - Stock_Option
  - Contract_Number
  - Billing_Rate
Mapping Supertype/subtype relationships to relations

These are implemented as one-to-one relationships
Solution

- Convert the ERD into Relation
- The simple process is

1. *Create a table/relation for each entity from the ERD.*
2. *The entity attributes become the attributes/columns for the table/relation.*
3. *Potentially, include some foreign keys to represent relationships shown in the ERD.*
Cont...

- Create a table for each ERD entity
  - There are 6 entities in the above Figure so there are six tables with the same names:
    - Department,
    - Supervisor,
    - EmployeeDepartment,
    - Employee,
    - EmployeeProject,
    - Project.
• **Entity attributes become table attributes**
  - **DEPARTMENT**( DepartmentName )
  - **SUPERVISOR**( SupervisorNumber, SupervisorName )
  - **EMPLOYEEDEPARTMENT**( DepartmentName, EmployeeNumber )
  - **EMPLOYEE**( EmployeeNumber, EmployeeName )
  - **EMPLOYEEPROJECT**( EmployeeNumber, ProjectNumber )
  - **PROJECT**( ProjectNumber )
Cont...

- **Include some foreign keys to represent relationships**
- Let's list the relationships and take each in turn
  - **Run By**
    - Can be represented by adding SupervisorNumber to the Department table
    - **DEPARTMENT** (DepartmentName, SupervisorNumber )
  - **Is assigned (employee-department)**
    - Is already represented in the EmployeeDepartment table
Cont...

- **Involves**
  - Also represented in the EmployeeDepartment table

- **Works on**
  - Represented in the EmployeeProject table

- **Is assigned (employ-project)**
  - Also represented by the EmployeeProject table.
This means no other foreign keys are really needed.

This leaves the Project table with only a primary key.

But

- The primary key is meant to uniquely identify instances of an entity.

The only data being stored in this table is the primary key, it has nothing to identify.

The choices are

- Remove the table. It's not needed if it only has the primary key.
- Add more attributes to the table.

So the Project table becomes

- **PROJECT** (ProjectNo, ProjectName, Budget, EndDate)
Add more required attributes to other table

- EMPLOYEEDEPARTMENT
  (DepartmentName, EmployeeNumber, StartDate)

- EMPLOYEEDPROJECT
  (EmployeeNumber, ProjectNumber, StartDate)

Normally, this sort of process would not be required.
Exercise
End of Chapter Three