The chapter deals with practical database design:

- the construction of a relational schema from an E-R schema
- this stage of database design is sometimes called “logical design”, besides conceptual design (the construction of the E-R schema), physical design (the representation of relations as files on disk and the choice of index structures) depending on the specific DBMS chosen and a given mix of application programs

- CASE tools often
  - permit to build E-R schemas interactively
  - apply a version of the translation rules presented in this chapter

- The relational model is poorer than the E-R model: the translation from an E-R schema to a relational schema entails a loss in the quality of the correspondence between the database schema and the application domain

**From E-R to Relational Schemas: Summary**

1. **Suppressing generalizations from ER schemas**
   - Keeping the superentity
   - Keeping the subentity
   - Modeling by ordinary relationships

2. **ER-to-relational mapping**
   - Entities, weak entities
   - Relationships
   - Multi-valued attributes

3. **Direct mapping of generalizations to relations**

**Suppressing Generalizations from ER Schemas**

- The semantics of generalization to be preserved:
  - mechanism of property inheritance
  - is-a relationship (subentities are also instances of the superentity class)
- The solutions only approximate a faithful translation:
  - suppress subentity classes and express generalization semantics with the superclass
  - retain subentity classes only
  - model generalization as ordinary relationships
Transformation for the example:
- suppress subentities Graduate Student and Exchange Student
- propagate attributes and relationships of subentities to superentity Student, with optional participation (minimal cardinality of 0)
- add a new discriminating attribute (Rank) with values that refer to the former subclasses (for example, GradStud and ExchStud)

Constraints are necessary in the transformed schema to preserve specific properties (attributes and relationships) of subentities in the initial schema:
- all and only students with Rank = GradStud have an advising Professor and a Thesis Title
- only Students with Rank = ExchStud may visit another University
- all and only Students with Rank = ExchStud have an Option

Transformation Keeping the Superentity
- Properties of subentities are propagated to the superentity
  - attributes become optional
  - relationships become optional on the side of superentity
- A new (discriminating) attribute is added to the superentity with
  - cardinality (0,1) for generalization
  - cardinality (0,1) total, overlapping

Constraints are added between the discriminating attribute and
- attributes of subentities
- participation of subentities in relationships

Keeping the Superentity: Evaluation
- Drawbacks
  - optional attributes are introduced (null values)
  - operations concerning subentities now have to be expressed via the superentity: application programs become more complex
  - many constraints become explicit

Advantage: simple, always applicable
Keeping the Subentities: Example

Employee
SSN
Name
Uses
(1,n)
(0,n)
Supervision (0,1)
Skills (1,n)
Speciality
Word Processor
Manager
NbSuperv
WorksOn (0,1)
Project
Name
Budget
Keeping the Subentities: Translation of the Example

Secretary
Engineer
Manager
Skills (1,n)
Speciality
NbSuperv
WorksOn3
Supervision2
Supervision3
(0,1)
(0,1)
(0,n)
(0,n)
(0,1) Subordinate
Supervisor
Keeping the Subentities: Evaluation

Method: propagate to each subentity the attributes (and identifiers) and relationship links of the superentity

Drawbacks
- proliferation of essentially redundant attributes and relationships
- application programs become more complex (operations on the superentity must now access all subentities)
- inter-relation constraint (corresponding to identifier of superentity)

Only applicable to total-and-exclusive generalizations

Exercise:
- it is easy to transform any generalization into a total-and-exclusive generalization
- this may lead to creating entities that were not naturally identified in the first place

Attributes of Employee (SSN, Name), and relationships WorksOn and Supervision are made explicit for each subentity
SSN is an identifier for each of Secretary, Engineer, and Manager

Constraint: the values of SSN must be unique across all three entity types
Modeling with Relationships: Example

Modeling with Relationships: Translation of the Example

Modeling with Relationships: Evaluation
- New relationships are created (IsHwProj, ...): they are mandatory on subentities, optional on superentity
- Former subentities have become weak entities with the former superentity as identifying entity
- Old attributes and relationships are preserved
- Drawbacks:
  - redundancy (the new relationships have the same meaning)
  - constraints are needed to express the type of generalization (totality, exclusiveness)
  - some operations become more complex (e.g., insertion of a subtentity requires insertion of superentity)

ER-to-Relational Mapping
- Derivation of a relational schema equivalent to a given ER schema without generalizations
- Versions of the translation are realized by CASE tools for database design, that produce relational schemas in the DDL of specific DBMSs
- The basic translation can be realized with a few rules
Company: ER Schema

Company: Relational Representation

Rule 1: Entity Type
For each nonweak entity type E, create a relation R with
• all simple attributes of E
• the simple components of composite attributes of E (there is no notion of composite attribute in the relational model)
• keys of R corresponding to identifiers of E

• relations Employee, Department, Project represent entities (entity relations)
• relation Employee also represents one-to-many relationships WorksFor and Supervision, relation Department represents one-to-one relationship Manages, relation Project represents one-to-many relationship Controls
• relation DeptLocations represents multivalued attribute Locations of entity Department
• relation WorksOn represents the many-to-many relationship WorksOn between Employee and Project (relationship relation)
• relation Dependent merges relationship DependOf and weak entity Dependent

• relations Employee, Department, Project represent entities (entity relations)
• relation Employee also represents one-to-many relationships WorksFor and Supervision, relation Department represents one-to-one relationship Manages, relation Project represents one-to-many relationship Controls
• relation DeptLocations represents multivalued attribute Locations of entity Department
• relation WorksOn represents the many-to-many relationship WorksOn between Employee and Project (relationship relation)
• relation Dependent merges relationship DependOf and weak entity Dependent

• relations Employee, Department, Project represent entities (entity relations)
• relation Employee also represents one-to-many relationships WorksFor and Supervision, relation Department represents one-to-one relationship Manages, relation Project represents one-to-many relationship Controls
• relation DeptLocations represents multivalued attribute Locations of entity Department
• relation WorksOn represents the many-to-many relationship WorksOn between Employee and Project (relationship relation)
• relation Dependent merges relationship DependOf and weak entity Dependent

From Entity-Relationship to Relational Schemas, September 22, 2008 – 9

From Entity-Relationship to Relational Schemas, September 22, 2008 – 10
Rule 2: Weak Entity Type

For each weak entity type \( F \), create a relation \( R \) with
- all simple attributes of \( F \)
- simple components of composite attributes of \( F \)
- as foreign key of \( R \) (i.e., with referential integrity constraint), attribute(s) corresponding to the primary key of the relation (produced by Rule 1) representing the entity identifying \( F \)
- primary key of \( R \) = this foreign key + attributes of partial key of \( F \)

Dependent

<table>
<thead>
<tr>
<th>ESN</th>
<th>DependentName</th>
<th>Sex</th>
<th>BDate</th>
<th>Relationship</th>
</tr>
</thead>
</table>

Note that Rule 2 represents in relation \( R \) both the weak entity \( F \) and its relationship to the entity that contributes to identifying \( F \)

Remember that binary relationships \( R \) between \( E_1 \) and \( E_2 \) can be:
- **one-to-one** (1-1) if the maximal cardinality of both \( E_1 \) and \( E_2 \) in \( R \) is 1
- **one-to-many** (1-N) if the maximal cardinality of one of \( E_1 \) and \( E_2 \) in \( R \) is 1 and the other maximal cardinality is \( \geq 1 \)
- **many-to-many** (M-N) if the maximal cardinality of both \( E_1 \) and \( E_2 \) in \( R \) is \( \geq 1 \)

Basic ideas of the following rules for mapping relationships to relations:
- 1-1 and 1-N relationships can be represented by a foreign-key attribute (i.e., with referential integrity) in the relation representing the entity whose maximal cardinality in \( R \) is 1 (Rules 3 and 4)
- each M-N relationship is mapped to a relation (Rule 5)
- 1-1 and 1-N relationships can also be mapped to a relation (by Rule 5)
- each N-ary relationship is mapped to a relation (Rule 7)

Rule 3: 1-1 Binary Relationship

- Let \( R \) be a 1-1 relationship between entity types \( E_1 \) and \( E_2 \) (modeled as relations \( R_1 \) and \( R_2 \))
- Choose \( R_1 \) (or \( R_2 \)) to represent \( R \); include in \( R_1 \):
  - as foreign key (i.e., with referential integrity) the primary key of \( R_2 \)
  - the simple attributes and the simple components of the composite attributes of \( R \)
- If the minimal cardinality of \( E_1 \) in \( R \) is 0, there will be null values in the extension of \( R_1 \)

Department

| DMgr | MgrStartDate |

When choosing between \( R_1 \) and \( R_2 \) to represent relationship \( R \), it is preferable to choose the relation corresponding to an entity (\( E_1 \) or \( E_2 \)) that is total in \( R \) (minimal cardinality = 1) in \( R \); this choice minimizes null values in the chosen relation

Other possible representations for a 1-1 relationship:
- \( R \) could also be represented by a relation, using Rule 5 (this solution is sometimes advocated to minimize null values, when the minimum cardinality of both \( E_1 \) and \( E_2 \) in \( R \) is 0)
- \( R \), \( E_1 \), and \( E_2 \) could all be represented in a single relation with two keys, each corresponding to the identifier of \( E_1 \) and \( E_2 \)
Rule 4: 1-N Binary Relationship

- Let $R$ be a 1-N relationship between entity types $E_1$ and $E_2$ (modeled as relations $R_1$ and $R_2$), with maximal cardinality of $E_1 = 1$ and maximal cardinality of $E_2 = N$
- Include in $R_1$
  - as foreign key (i.e., with referential integrity) the primary key of $R_2$
  - the simple attributes and the simple components of the composite attributes of $R$

<table>
<thead>
<tr>
<th>Employee</th>
<th>...</th>
<th>SuperSSN</th>
<th>DNo</th>
</tr>
</thead>
</table>

Rule 5: M-N Binary Relationship

- Let $R$ be an M-N relationship between entity types $E_1$ and $E_2$ (modeled as relations $R_1$ and $R_2$)
- Relationship $R$ is represented as a relation $R$ with:
  - as foreign keys (i.e., with referential integrity) the primary keys of $R_1$ and $R_2$
  - a composite primary key made of those two foreign keys
  - attributes corresponding to the simple attributes and to the simple components of the composite attributes of relationship $R$
- This representation can also apply to 1-1 and 1-N relationships, thereby avoiding null values in relations
- Relations like $R$ are sometimes called relationships relations

<table>
<thead>
<tr>
<th>WorksOn</th>
<th>...</th>
<th>SuperSSN</th>
<th>DNo</th>
<th>Hours</th>
</tr>
</thead>
</table>

Rule 6: Multivalued Attribute

- Let $V$ be a multivalued attribute of an entity type $E$, and $R_1$ be the relation representing $E$
- Create a relation $R$ with an attribute $V$ and attribute(s) $K$ for the primary key of $R_1$ as foreign key of $R$ (i.e., with referential integrity)
- Primary key of $R = V \cup K$
- Problem: information about $E$ (Departments) is split in 2 relations $R$ and $R_1$

<table>
<thead>
<tr>
<th>DeptLocations</th>
<th>...</th>
<th>DNumber</th>
<th>DLocations</th>
</tr>
</thead>
</table>

What Multivalued Relational Attributes Could Look Like

<table>
<thead>
<tr>
<th>Department</th>
<th>DName</th>
<th>DNumber</th>
<th>DMgr</th>
<th>DLocations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Research</td>
<td>5</td>
<td>433445555</td>
<td>[Bellaire, Sugarland, Houston]</td>
<td></td>
</tr>
<tr>
<td>Administration</td>
<td>4</td>
<td>987654321</td>
<td>(Stafford)</td>
<td></td>
</tr>
<tr>
<td>Headquarters</td>
<td>1</td>
<td>886655555</td>
<td>[Houston]</td>
<td></td>
</tr>
</tbody>
</table>

- Multivalued attributes are not permitted in the relational model (“repeating groups” existed in its predecessors COBOL and CODASYL)
- This would be a natural extension, but with more complex definitions for
  - the DML (algebra, calculi, SQL)
  - functional dependencies
Rule 6 as Normalization of Multivalued Attributes into 1NF

<table>
<thead>
<tr>
<th>Department</th>
<th>DNumber</th>
<th>DMgr</th>
<th>DLocations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Research</td>
<td>5</td>
<td>333445555</td>
<td>Bellaire,Sugarland,Houston</td>
</tr>
<tr>
<td>Administration</td>
<td>4</td>
<td>987654321</td>
<td>Stafford</td>
</tr>
<tr>
<td>Headquarters</td>
<td>1</td>
<td>888665555</td>
<td>Houston</td>
</tr>
</tbody>
</table>

↓ 1NF Normalization

<table>
<thead>
<tr>
<th>DName</th>
<th>DNumber</th>
<th>DMgr</th>
<th>DLocations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Research</td>
<td>5</td>
<td>333445555</td>
<td>Bellaire,Sugarland,Houston</td>
</tr>
<tr>
<td>Administration</td>
<td>4</td>
<td>987654321</td>
<td>Stafford</td>
</tr>
<tr>
<td>Headquarters</td>
<td>1</td>
<td>888665555</td>
<td>Houston</td>
</tr>
</tbody>
</table>

The relational model, when it was defined in the early 1970’s, ruled out multivalued attributes to mark a clear simplification in comparison to the previously existing models (COBOL, Codasyl).

Multivalued attributes are natural and useful, they were present as “repeating groups” in the pre-relational era, and are available in all object models.

There is no good relational representation for E-R multivalued attributes:

- the two-relation solution separates information that naturally goes together in the real world (the Location information is clearly a property of Department)
- the idea of the single-relation solution is to keep together pieces of information that concern the original entity
  - the information about Department is represented in a single relation, the join of relations Department and DeptLocations in the two-relation representation
  - this introduces redundancy (several tuples for each department with more than one location)
  - the tuples of the join relation do not have a simple correspondence with the real-world perception: a tuple represents information about one department AND one of its locations (not just about one department)

Rule 7: N-ary Relationship (n > 2)

Create a relation Rel that represents the n-ary relationship R, with

- simple attributes of R and simple components of composite attributes of R
- as foreign keys (i.e., with referential integrity), the primary keys of relations that represent the entity types of R
- a composite primary key made of those foreign keys
  (except if an entity type E of R has maximum cardinality 1 in R, in which case the primary key of Rel is the primary key of the relation that represents E)
N-ary Relationship: Example

<table>
<thead>
<tr>
<th>Supplier</th>
<th>Supply</th>
<th>Project</th>
<th>Part</th>
</tr>
</thead>
<tbody>
<tr>
<td>SupName</td>
<td></td>
<td>ProjName</td>
<td>PartNo</td>
</tr>
<tr>
<td></td>
<td>Quantity</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- The composite key of relation *Supply* is made of *SupName*, *ProjName*, and *PartNo*
- There is a referential integrity constraint from each of those attributes to the corresponding attribute in relations *Supplier*, *Project*, and *Part*

Direct Relational Mapping of Generalization

Generalization can be mapped directly to relations, by combining the previous rules for generalization elimination and E-R to relational mapping, with 3 solutions:
- a relation for the superentity only
- relations for subentities only
- relations for both super- and subentities

Generalization: Single Relation for Superentity
- For a generalization with subclasses \(\{S_1, \ldots, S_m\}\) and a superclass \(S\) where the attributes of \(S\) are \(\{k, a_1, \ldots, a_m\}\) and \(k\) is an identifier of \(S\)
- Create a single relation \(L\) with
  - attributes \(\{k, a_1, \ldots, a_m\} \cup \text{attributes of } S_1 \cup \ldots \cup \text{attributes of } S_m \cup \{t\}\)
  - primary key = \(k\)
  - \(t\): new ("discriminating") attribute, keeping track of subentities (JobType)
  - null values possible for subclass attributes (with constraints via \(t\) values)

Employee:

<table>
<thead>
<tr>
<th>SSN</th>
<th>FName</th>
<th>TypingSpeed</th>
<th>TechGrade</th>
<th>EngType</th>
<th>JobType</th>
</tr>
</thead>
</table>

From Entity-Relationship to Relational Schemas, September 22, 2008 – 17

From Entity-Relationship to Relational Schemas, September 22, 2008 – 18
Generalization: Relations for Subentities Only

For a generalization with subclasses \( \{S_1, \ldots, S_m\} \) and a superclass \( S \) where the attributes of \( S \) are \( \{k, a_1, \ldots, a_m\} \) and an identifier of \( S \) is \( k \):

- Create a relation \( L_i \) for each subclass \( S_i \) with
  - attributes \( \{\text{attributes of } S_i\} \cup \{k, a_1, \ldots, a_m\} \)
  - primary key \( k \), with unicity constraint across all \( L_i \)’s

<table>
<thead>
<tr>
<th>Secretary</th>
<th>SSN</th>
<th>FName</th>
<th>MInit</th>
<th>LName</th>
<th>BirthDate</th>
<th>Address</th>
<th>TypingSpeed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technician</td>
<td>SSN</td>
<td>FName</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Engineer</td>
<td>SSN</td>
<td>FName</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Relations for Both Super- and Subentities: a Variant

- Add a (redundant) view for each subclass, by joining the subclass relation with the superclass relation

- Advantages
  - "materializing" attribute inheritance
  - saving a join in queries that involve inherited attributes

<table>
<thead>
<tr>
<th>Employee</th>
<th>SSN</th>
<th>FName</th>
<th>MInit</th>
<th>LName</th>
<th>BirthDate</th>
<th>Address</th>
<th>TypingSpeed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Secretary</td>
<td>SSN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Full-Secr</td>
<td>SSN</td>
<td>FName</td>
<td>MInit</td>
<td>LName</td>
<td>BirthDate</td>
<td>Address</td>
<td>TypingSpeed</td>
</tr>
</tbody>
</table>

Generalization: Relations for Both Super- and Subentities

For a generalization with subclasses \( \{S_1, \ldots, S_m\} \) and a superclass \( S \) where the attributes of \( S \) are \( \{k, a_1, \ldots, a_m\} \) and an identifier of \( S \) is \( k \):

- Create a relation \( L \) for \( S \) with attributes \( \{k, a_1, \ldots, a_m\} \) and primary key \( k \)
- Create a relation \( L_i \) for each \( S_i \) with attributes \( \{k\} \cup \{\text{attributes of } S_i\} \) and primary key \( k \)

<table>
<thead>
<tr>
<th>Employee</th>
<th>SSN</th>
<th>FName</th>
<th>MInit</th>
<th>LName</th>
<th>BirthDate</th>
<th>Address</th>
<th>TypingSpeed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Secretary</td>
<td>SSN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Technician</td>
<td>SSN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Engineer</td>
<td>SSN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Entity Relationship versus Relational: Summary

- ER schema is more compact (a connected graph) than relational schema (a collection of relations) \( \Rightarrow \) easier and more intuitive perception of information content
- ER model has two main constructs: entities and relationships
- Relational model only has relations, used as
  - "entity" relations (e.g., Employee, Department, Project, Dependent)
  - "relationship" relations (e.g., WorksOn)
  - "multivalued-attribute" relations (e.g., DeptLocations)
Loss of Information in the ER-to-Relational Translation

- Information loss relates to the quality of the correspondence between data model and the underlying real world
  - loss of precision: a single relational construct (relation) represents three ER constructs (entity, relationship, multivalued attribute)
  - the ER model was created after the relational model, to distinguish different types of relations (entity relations, relationship relations, multivalued attributes)

- Good correspondence between ER schema and relational schema depends on
  - constraints (multiple referential integrity constraints, constraints for cardinalities)
  - documentation: informal description of the links between relations and concepts perceived in the real world

Issues for Database Design and Efficiency

- 1-1 and 1-N relationship can be merged into the relation representing one of the participating entities or preserved as “relationship” relations (Rules 3 and 4)

- Pros of merging
  - fewer relations (simpler schemas)
  - fewer joins in queries

- Cons of merging
  - relationships as independent constructs essentially disappear
  - more complexity from asymmetry of relationship representation
  - reduced extensibility (difficulty to get cardinalities right directly)

- Representing multivalued attributes with two relations (Rule 6) is also a source of joins in queries