Chapter 5

Relational Data Model and ER/EER-to-Relational Mapping

Nguyen Thi Ai Thao
thaonguyen@cse.hcmut.edu.vn

Spring- 2016
<table>
<thead>
<tr>
<th></th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Relational Data Model</td>
</tr>
<tr>
<td>2</td>
<td>Main Phases of Database Design</td>
</tr>
<tr>
<td>3</td>
<td>ER-/EER-to-Relational Mapping</td>
</tr>
</tbody>
</table>
Relational Data Model

- Basic Concepts: relational data model, relation schema, domain, tuple, cardinality & degree, database schema, etc.

- Relational Integrity Constraints
  - key, primary key & foreign key
  - entity integrity constraint
  - referential integrity

- Update Operations on Relations
Basic Concepts

- The relational model of data is based on the concept of a relation
- A relation is a mathematical concept based on the ideas of sets
- The model was first proposed by Dr. E.F. Codd of IBM in 1970 in the following paper: "A Relational Model for Large Shared Data Banks," Communications of the ACM, June 1970
Basic Concepts

- **Relational data model**: represents a database in the form of *relations* - 2-dimensional table with rows and columns of data. A database may contain one or more such tables. A relation schema is used to describe a relation.

- **Relation schema**: $R(A_1, A_2, \ldots, A_n)$ is made up of a relation name $R$ and a list of *attributes* $A_1, A_2, \ldots, A_n$. Each attribute $A_i$ is the name of a role played by some domain $D$ in the relation schema $R$. $R$ is called the *name* of this relation.
Basic Concepts

- The **degree of a relation** is the number of attributes \( n \) of its relation schema.

- **Domain** \( D \): \( D \) is called the domain of \( A_i \) and is denoted by \( \text{dom}(A_i) \). It is a set of atomic values and a set of integrity constraints
  
  - `STUDENT(Name, SSN, HomePhone, Address, OfficePhone, Age, GPA)`
  - `Degree = ??`
  - `\text{dom}(GPA) = ??`
Basic Concepts

- **Tuple**: row/record in a table
- **Cardinality**: number of tuples in a table
- **Database schema** $S = \{R_1, R_2, \ldots, R_m\}$

Schema diagram for the COMPANY relational database schema
Basic Concepts

A relation \( r \) (or relation state, relation instance) of the relation schema \( R(A_1, A_2, \ldots, A_n) \), also denoted by \( r(R) \), is a set of \( n \)-tuples \( r = \{ t_1, t_2, \ldots, t_m \} \).

- Each \( n \)-tuple \( t \) is an ordered list of \( n \) values \( t = <v_1, v_2, \ldots, v_n> \), where each value \( v_i, i=1..n \), is an element of \( \text{dom}(A_i) \) or is a special \text{null} value. The \( i^{\text{th}} \) value in tuple \( t \), which corresponds to the attribute \( A_i \), is referred to as \( t[A_i] \).
Basic Concepts

Relational data model
Database schema
Relation schema
Relation
Tuple
Attribute
Basic Concepts

- A relation can be conveniently represented by a table, as the example shows.
- The columns of the tabular relation represent attributes.
- Each attribute has a distinct name, and is always referenced by that name, never by its position.
- Each row of the table represents a tuple. The ordering of the tuples is immaterial and all tuples must be distinct.
Basic Concepts

Relational Data Model & ER/EER-to-Relational Mapping

<table>
<thead>
<tr>
<th>Relation name</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>STUDENT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Name</td>
</tr>
<tr>
<td></td>
<td>Benjamin Bayer</td>
</tr>
<tr>
<td></td>
<td>Katherine Ashly</td>
</tr>
<tr>
<td></td>
<td>Dick Davidson</td>
</tr>
<tr>
<td></td>
<td>Charles Cooper</td>
</tr>
<tr>
<td></td>
<td>Barbara Benson</td>
</tr>
</tbody>
</table>
## Basic Concepts

### Alternative Terminology for Relational Model

<table>
<thead>
<tr>
<th>Formal Terms</th>
<th>Informal Terms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relation</td>
<td>Table</td>
</tr>
<tr>
<td>Attribute</td>
<td>Column Header</td>
</tr>
<tr>
<td>Domain</td>
<td>All possible Column Values</td>
</tr>
<tr>
<td>Tuple</td>
<td>Row</td>
</tr>
<tr>
<td>Schema of a Relation</td>
<td>Table Definition</td>
</tr>
<tr>
<td>State of the Relation</td>
<td>Populated Table</td>
</tr>
</tbody>
</table>
Relational Integrity Constraints

- Constraints are **conditions** that must hold on **all** valid relation instances. There are three main types of constraints:
  - Key constraints
  - Entity integrity constraints
  - Referential integrity constraints

- But …
Relational Integrity Constraints

- **Null value**
  - Represents value for an attribute that is currently unknown or inapplicable for tuple
  - Deals with incomplete or exceptional data
  - Represents the absence of a value and is not the same as zero or spaces, which are values
Relational Integrity Constraints - Key Constraints

- **Superkey** of R: A set of attributes SK of R such that no two tuples in any valid relation instance r(R) will have the same value for SK. That is, for any distinct tuples t1 and t2 in r(R), t1[SK] ≠ t2[SK]

- **Key** of R: A "minimal" superkey; that is, a superkey K such that removal of any attribute from K results in a set of attributes that is not a superkey
Relational Integrity Constraints - Key Constraints

Example: The CAR relation schema:
CAR(State, Reg#, SerialNo, Make, Model, Year)
has two keys
Key1 = {State, Reg#}
Key2 = {SerialNo}, which are also superkeys.
{SerialNo, Make} is a superkey but not a key

- If a relation has several candidate keys, one is chosen arbitrarily to be the primary key. The primary key attributes are underlined.
Relational Integrity Constraints - Key Constraints

The CAR relation, with two candidate keys: License_Number and Engine_Serial_Number

<table>
<thead>
<tr>
<th>License_number</th>
<th>Engine_serial_number</th>
<th>Make</th>
<th>Model</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Texas ABC-739</td>
<td>A69352</td>
<td>Ford</td>
<td>Mustang</td>
<td>02</td>
</tr>
<tr>
<td>Florida TVP-347</td>
<td>B43696</td>
<td>Oldsmobile</td>
<td>Cutlass</td>
<td>05</td>
</tr>
<tr>
<td>New York MPO-22</td>
<td>X83554</td>
<td>Oldsmobile</td>
<td>Delta</td>
<td>01</td>
</tr>
<tr>
<td>California 432-TFY</td>
<td>C43742</td>
<td>Mercedes</td>
<td>190-D</td>
<td>99</td>
</tr>
<tr>
<td>California RSK-629</td>
<td>Y82935</td>
<td>Toyota</td>
<td>Camry</td>
<td>04</td>
</tr>
<tr>
<td>Texas RSK-629</td>
<td>U028365</td>
<td>Jaguar</td>
<td>XJS</td>
<td>04</td>
</tr>
</tbody>
</table>
Relational Integrity Constraints - Entity Integrity

- **Relational Database Schema**: A set $S$ of relation schemas that belong to the same database. $S$ is the name of the database: $S = \{R_1, R_2, \ldots, R_n\}$

- **Entity Integrity**: primary key attributes $PK$ of each relation schema $R$ in $S$ cannot have null values in any tuple of $r(R)$ because primary key values are used to identify the individual tuples: $t[PK] \neq \text{null}$ for any tuple $t$ in $r(R)$
  
  - Note: Other attributes of $R$ may be similarly constrained to disallow null values, even though they are not members of the primary key
Relational Integrity Constraints - Referential Integrity

- A constraint involving two relations (the previous constraints involve a single relation)
- Used to specify a relationship among tuples in two relations: the referencing relation and the referenced relation
- Tuples in the referencing relation $R_1$ have attributes FK (called foreign key attributes) that reference the primary key attributes PK of the referenced relation $R_2$. A tuple $t_1$ in $R_1$ is said to reference a tuple $t_2$ in $R_2$ if $t_1[FK] = t_2[PK]$
- A referential integrity constraint can be displayed in a relational database schema as a directed arc from $R_1$.FK to $R_2$
Relational Integrity Constraints - Referential Integrity

<table>
<thead>
<tr>
<th>DEPARTMENT</th>
<th>DNUMBER</th>
<th>MGRSSN</th>
<th>MGRSTARTDATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Research</td>
<td>5</td>
<td>333445555</td>
<td>1988-05-22</td>
</tr>
<tr>
<td>Administration</td>
<td>4</td>
<td>987654321</td>
<td>1995-01-01</td>
</tr>
<tr>
<td>Headquarters</td>
<td>1</td>
<td>888665555</td>
<td>1981-06-19</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EMPLOYEE</th>
<th>MINIT</th>
<th>LNAME</th>
<th>SSN</th>
<th>BDATE</th>
<th>ADDRESS</th>
<th>SEX</th>
<th>SALARY</th>
<th>SUPERSSN</th>
<th>DNO</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>B</td>
<td>Smith</td>
<td>123456789</td>
<td>1965-01-09</td>
<td>731 Fondren, Houston, TX</td>
<td>M</td>
<td>30000</td>
<td>333445555</td>
<td>5</td>
</tr>
<tr>
<td>Franklin</td>
<td>T</td>
<td>Wong</td>
<td>333445555</td>
<td>1955-12-08</td>
<td>638 Voss, Houston, TX</td>
<td>M</td>
<td>40000</td>
<td>888665555</td>
<td>5</td>
</tr>
<tr>
<td>Alicia</td>
<td>J</td>
<td>Zelaya</td>
<td>999887777</td>
<td>1968-07-19</td>
<td>3321 Castle, Spring, TX</td>
<td>F</td>
<td>25000</td>
<td>987654321</td>
<td>4</td>
</tr>
<tr>
<td>Jennifer</td>
<td>S</td>
<td>Wallace</td>
<td>987654321</td>
<td>1941-06-20</td>
<td>291 Berry, Bellaire, TX</td>
<td>F</td>
<td>43000</td>
<td>888665555</td>
<td>4</td>
</tr>
<tr>
<td>Ramesh</td>
<td>K</td>
<td>Narayan</td>
<td>666884444</td>
<td>1962-09-15</td>
<td>975 Fire Oak, Humble, TX</td>
<td>M</td>
<td>38000</td>
<td>333445555</td>
<td>5</td>
</tr>
<tr>
<td>Joyce</td>
<td>A</td>
<td>English</td>
<td>453453453</td>
<td>1972-07-31</td>
<td>5631 Rice, Houston, TX</td>
<td>F</td>
<td>25000</td>
<td>333445555</td>
<td>5</td>
</tr>
<tr>
<td>Ahmad</td>
<td>V</td>
<td>Jabbar</td>
<td>987987987</td>
<td>1969-03-29</td>
<td>980 Dallas, Houston, TX</td>
<td>M</td>
<td>25000</td>
<td>987654321</td>
<td>4</td>
</tr>
<tr>
<td>James</td>
<td>E</td>
<td>Borg</td>
<td>888665555</td>
<td>1937-11-10</td>
<td>450 Stone, Houston, TX</td>
<td>M</td>
<td>55000</td>
<td>null</td>
<td>1</td>
</tr>
</tbody>
</table>
Relational Integrity Constraints

Referential Integrity

Statement of the constraint

- The value in the foreign key column (or columns) FK of the the referencing relation $R_1$ can be either:
  - (1) a value of an existing primary key value of the corresponding primary key PK in the referenced relation $R_2$, or
  - (2) a NULL

- In case (2), the FK in $R_1$ should **not** be a part of its own primary key
Referential integrity constraints displayed on the COMPANY relational database schema
Relational Integrity Constraints

Other Types of Constraints

- Semantic Integrity Constraints:
  - based on application semantics and cannot be expressed by the model per se
  - E.g., “the max. no. of hours per employee for all projects he or she works on is 56 hrs per week”
  - A constraint specification language may have to be used to express these
  - SQL-99 allows triggers and ASSERTIONS to allow for some of these

- State/static constraints (so far)

- Transition/dynamic constraints: e.g., “the salary of an employee can only increase”
Update Operations on Relations

- INSERT a tuple
- DELETE a tuple
- MODIFY a tuple

Integrity constraints should not be violated by the update operations
Update Operations on Relations

- **Insertion**: to insert a new tuple \( t \) into a relation \( R \). When inserting a new tuple, it should make sure that the database constraints are not violated:
  - The value of an attribute should be of the correct data type (i.e. from the appropriate domain).
  - The value of a prime attribute (i.e. the key attribute) must not be null.
  - The key value(s) must not be the same as that of an existing tuple in the same relation.
  - The value of a foreign key (if any) must refer to an existing tuple in the corresponding relation.

- **Options if the constraints are violated**: Homework !!
Update Operations on Relations

- **Deletion**: to remove an existing tuple \( t \) from a relation \( R \). When deleting a tuple, the following constraints must not be violated:
  - The tuple must already exist in the database
  - The referential integrity constraint is not violated

- **Modification**: to change values of some attributes of an existing tuple \( t \) in a relation \( R \)
Update Operations on Relations

In case of integrity violation, several actions can be taken:

• Cancel the operation that causes the violation (REJECT option)
• Perform the operation but inform the user of the violation
• Trigger additional updates so the violation is corrected (CASCADE option, SET NULL option)
• Execute a user-specified error-correction routine

Again, homework!!
Contents

1  Relational Data Model
2  Main Phases of Database Design
3  ER-/EER-to-Relational Mapping
Main Phases of Database Design

- Three main phases
  - Conceptual database design
  - Logical database design
  - Physical database design
A simplified diagram to illustrate the main phases of database design.
Main Phases of Database Design

- Conceptual database design
  - The process of constructing a model of the data used in an enterprise, independent of all physical considerations
  - Model comprises entity types, relationship types, attributes and attribute domains, primary and alternate keys, structural and integrity constraints
Main Phases of Database Design

- Logical database design
  - The process of constructing a model of the data used in an enterprise based on a specific data model (e.g. relational), but independent of a particular DBMS and other physical considerations
  - ER- & EER-to-Relational Mapping
  - Normalization
Main Phases of Database Design

- Physical database design
  - The process of producing a description of the implementation of the database on secondary storage; it describes the base relations, file organizations, and indexes design used to achieve efficient access to the data, and any associated integrity constraints and security measures.
The ERD for the COMPANY database
Result of mapping the COMPANY ER schema into a relational schema

Relational Data Model & ER/EER-to-Relational Mapping
### Contents

1. Relational Data Model
2. Main Phases of Database Design
3. ER-/EER-to-Relational Mapping
ER- & EER-to-Relational Mapping

**ER-**
- Step 1: Mapping of Regular Entity Types
- Step 2: Mapping of Weak Entity Types
- Step 3: Mapping of Binary 1:1 Relationship Types
- Step 4: Mapping of Binary 1:N Relationship Types
- Step 5: Mapping of Binary M:N Relationship Types
- Step 6: Mapping of Multivalued attributes
- Step 7: Mapping of N-ary Relationship Types

**EER-**
- *Step 8: Options for Mapping Specialization or Generalization.*
- *Step 9: Mapping of Union Types (Categories)*
ER-to-Relational Mapping

- Step 1: Mapping of Regular (strong) Entity Types
  - Entity --> Relation
  - Attribute of entity --> Attribute of relation
  - Primary key of entity --> Primary key of relation
  - **Example:** We create the relations EMPLOYEE, DEPARTMENT, and PROJECT in the relational schema corresponding to the regular entities in the ER diagram. SSN, DNUMBER, and PNUMBER are the primary keys for the relations EMPLOYEE, DEPARTMENT, and PROJECT as shown.
The ERD for the COMPANY database

Strong Entity Types
Step 2: Mapping of Weak Entity Types

- For each weak entity type W in the ER schema with owner entity type E, create a relation R and include all simple attributes (or simple components of composite attributes) of W as attributes of R.
- In addition, include as foreign key attributes of R the primary key attribute(s) of the relation(s) that correspond to the owner entity type(s).
- The primary key of R is the combination of the primary key(s) of the owner(s) and the partial key of the weak entity type W, if any.
- **Example:** Create the relation DEPENDENT in this step to correspond to the weak entity type DEPENDENT. Include the primary key SSN of the EMPLOYEE relation as a foreign key attribute of DEPENDENT (renamed to ESSN).

The primary key of the DEPENDENT relation is the combination {ESSN, DEPENDENT_NAME} because DEPENDENT_NAME is the partial key of DEPENDENT.

- Note: CASCADE option as implemented.
The ERD for the COMPANY database

Owner’s PK

PK

Weak Entity Types

Partial key

Relational Data Model & ER/EER-to-Relational Mapping
Result of mapping the COMPANY ER schema into a relational schema
ER-to-Relational Mapping

- ER-
  - Step 1: Mapping of Regular Entity Types
  - Step 2: Mapping of Weak Entity Types
  - Step 3: Mapping of Binary 1:1 Relationship Types
  - Step 4: Mapping of Binary 1:N Relationship Types
  - Step 5: Mapping of Binary M:N Relationship Types
  - Step 6: Mapping of Multivalued attributes
  - Step 7: Mapping of N-ary Relationship Types

Transformation of binary relationships - depends on functionality of relationship and membership class of participating entity types
**Mandatory** membership class

- For two entity types E1 and E2: If E2 is a mandatory member of an N:1 (or 1:1) relationship with E1, then the relation for E2 will include the prime attributes of E1 as a foreign key to represent the relationship.

- 1:1 relationship: If the membership class for E1 and E2 are both mandatory, a foreign key can be used in either relation.

- N:1 relationship: If the membership class of E2, which is at the N-side of the relationship, is **optional** (i.e. partial), then the above guideline is not applicable.
Assume every module must be offered by a department, then the entity type MODULE is a mandatory member of the relationship OFFER. The relation for MODULE is:

\[ \text{MODULE}(\text{MDL-NUMBER}, \text{TITLE}, \text{TERM}, \ldots, \text{DNAME}) \]
The ERD for the COMPANY database

Relationships Types
Result of mapping the COMPANY ER schema into a relational schema

**Employee**
- FNAME
- MINIT
- LNAME
- SSN
- BDATE
- ADDRESS
- SEX
- SALARY
- SUPERSSN
- DNO

**Department**
- DNAME
- DNUMBER
- MGRSSN
- MGRSTARTDATE

**Dept_locations**
- DNUMBER
- DLOCATION

**Project**
- PNAME
- PNUMBER
- PLOCATION
- DNUM

**Works_on**
- ESSN
- PNO
- HOURS

**Dependent**
- ESSN
- DEPENDENT_NAME
- SEX
- BDATE
- RELATIONSHIP
Optional membership classes

- If entity type E2 is an optional member of the N:1 relationship with entity type E1 (i.e. E2 is at the N-side of the relationship), then the relationship is usually represented by a new relation containing the prime attributes of E1 and E2, together with any attributes of the relationship. The key of the entity type at the N-side (i.e. E2) will become the key of the new relation.

- If both entity types in a 1:1 relationship have the optional membership, a new relation is created which contains the prime attributes of both entity types, together with any attributes of the relationship. The prime attribute(s) of either entity type will be the key of the new relation.
One possible representation of the relationship:
BORROWER(BNUMBER, NAME, ADDRESS, ...)
BOOK(ISBN, TITLE, ..., BNUMBER)

A better alternative:
BORROWER(BNUMBER, NAME, ADDRESS, ...)  
BOOK(ISBN, TITLE, ...)  
ON_LOAN(ISBN, BNUMBER)
The ERD for the COMPANY database

1:N (both optional)
Result of mapping the COMPANY ER schema into a relational schema

[1]: Step 4, chapter 7
ER-to-Relational Mapping

- **N:M binary relationships:**
  - An N:M relationship is always represented by a new relation which consists of the prime attributes of both participating entity types together with any attributes of the relationship.
  - The combination of the prime attributes will form the primary key of the new relation.

**Example:** ENROL is an M:N relationship between STUDENT and MODULE. To represent the relationship, we have a new relation:

```
ENROL(SNUMBER, MDL-NUMBER, DATE)
```
The ERD for the COMPANY database

Relational Data Model & ER/EER-to-Relational Mapping
Result of mapping the COMPANY ER schema into a relational schema
ER-to-Relational Mapping

- **ER-**
  - Step 1: Mapping of Regular Entity Types
  - Step 2: Mapping of Weak Entity Types
  - Step 3: Mapping of Binary 1:1 Relationship Types
  - Step 4: Mapping of Binary 1:N Relationship Types
  - Step 5: Mapping of Binary M:N Relationship Types
  - Step 6: Mapping of Multivalued attributes
  - Step 7: Mapping of N-ary Relationship Types
Transformation of recursive/involuted relationships

- Relationship among different instances of the same entity
- The name(s) of the prime attribute(s) needs to be changed to reflect the role each entity plays in the relationship
Example 1: 1:1 involuted relationship, in which the memberships for both entities are optional

PERSON(ID, NAME, ADDRESS, ...)
MARRY(HUSBAND-ID, WIFE_ID, DATE_OF_MARRIAGE)
Example 2: 1:M involuted relationship
- If the relationship is mandatory or almost mandatory:
  \[
  \text{EMPLOYEE}(\text{ID, ENAME, } \ldots, \text{SUPERVISOR\_ID})
  \]
- If the relationship is optional:
  \[
  \text{EMPLOYEE}(\text{ID, ENAME, } \ldots)
  \]
  \[
  \text{SUPERVISE}(\text{ID, START\_DATE, } \ldots, \text{SUPERVISOR\_ID})
  \]

Example 3: N:M involuted relationship
\[
\text{PART}(\text{PNUMBER, DESCRIPTION, } \ldots)
\]
\[
\text{COMPRISSE}(\text{MAJOR\_PNUMBER, MINOR\_PNUMBER, QUANTITY})
\]
Step 6: Mapping of Multivalued attributes

- For each multivalued attribute $A$, create a new relation $R$. This relation $R$ will include an attribute corresponding to $A$, plus the primary key attribute $K$ as a foreign key in $R$ of the relation that represents the entity type or relationship type that has $A$ as an attribute.
- The primary key of $R$ is the combination of $A$ and $K$. If the multivalued attribute is composite, we include its simple components.

Example: The relation `DEPT_LOCATIONS` is created. The attribute `DLOCATION` represents the multivalued attribute `LOCATIONS` of `DEPARTMENT`, while `DNUMBER` as foreign key represents the primary key of the `DEPARTMENT` relation. The primary key of $R$ is the combination of {`DNUMBER`, `DLOCATION`}.
The ERD for the COMPANY database

Multivalued Attribute
Result of mapping the COMPANY ER schema into a relational schema
Step 7: Mapping of N-ary Relationship Types

- For each n-ary relationship type R, where n>2, create a new relationship S to represent R.
- Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types.
- Also include any simple attributes of the n-ary relationship type (or simple components of composite attributes) as attributes of S.

**Example:** The relationship type SUPPY in the ER below. This can be mapped to the relation SUPPLY shown in the relational schema, whose primary key is the combination of the three foreign keys {SNAME, PARTNO, PROJNAME}. 

ER-to-Relational Mapping
Ternary relationship types: The SUPPLY relationship

SUPPLIER

| SNAME | . . . |

PROJECT

| PROJNAME | . . . |

PART

| PARTNO | . . . |

Note: if the cardinality constraint on any of the entity types E participating in the relationship is 1, the PK should not include the FK attributes that reference the relation E’ corresponding to E
## ER-to-Relational Mapping
Correspondence between ER and Relational Models

<table>
<thead>
<tr>
<th>ER Model</th>
<th>Relational Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity type</td>
<td>“Entity” relation</td>
</tr>
<tr>
<td>1:1 or 1:N relationship type</td>
<td>Foreign key (or “relationship” relation)</td>
</tr>
<tr>
<td>M:N relationship type</td>
<td>“Relationship” relation &amp; 2 foreign keys</td>
</tr>
<tr>
<td>n-ary relationship type</td>
<td>“Relationship” relation &amp; n foreign keys</td>
</tr>
<tr>
<td>Simple attribute</td>
<td>Attribute</td>
</tr>
<tr>
<td>Composite attribute</td>
<td>Set of simple component attributes</td>
</tr>
<tr>
<td>Multivalued attribute</td>
<td>Relation and foreign key</td>
</tr>
<tr>
<td>Value set</td>
<td>Domain</td>
</tr>
<tr>
<td>Key attribute</td>
<td>Primary (or secondary) key</td>
</tr>
</tbody>
</table>
ER- & EER-to-Relational Mapping

ER-
- Step 1: Mapping of Regular Entity Types
- Step 2: Mapping of Weak Entity Types
- Step 3: Mapping of Binary 1:1 Relationship Types
- Step 4: Mapping of Binary 1:N Relationship Types
- Step 5: Mapping of Binary M:N Relationship Types
- Step 6: Mapping of Multivalued attributes
- Step 7: Mapping of N-ary Relationship Types

EER-
- Step 8: Options for Mapping Specialization or Generalization.
- Step 9: Mapping of Union Types (Categories)
EER-to-Relational Mapping

- **Step 8: Options for Mapping Specialization or Generalization.**

Convert each specialization with m subclasses \{S_1, S_2, \ldots, S_m\} and generalized superclass C, where the attributes of C are \{k, a_1, \ldots, a_n\} and k is the (primary) key, into relational schemas using one of the four following options:

- Option 8A: Multiple relations-Superclass and subclasses
- Option 8B: Multiple relations-Subclass relations only
- Option 8C: Single relation with one type attribute
- Option 8D: Single relation with multiple type attributes
Option 8A: Multiple relations-Superclass and subclasses

Create a relation L for C with attributes $\text{Attrs}(L) = \{k,a_1,\ldots,a_n\}$ and $\text{PK}(L) = k$. Create a relation $L_i$ for each subclass $S_i$, $1 < i < m$, with the attributes $\text{Attrs}(L_i) = \{k\} \cup \{\text{attributes of } S_i\}$ and $\text{PK}(L_i) = k$. This option works for any specialization (total or partial, disjoint or over-lapping).

Option 8B: Multiple relations-Subclass relations only

Create a relation $L_i$ for each subclass $S_i$, $1 < i < m$, with the attributes $\text{Attr}(L_i) = \{\text{attributes of } S_i\} \cup \{k,a_1,\ldots,a_n\}$ and $\text{PK}(L_i) = k$. This option only works for a specialization whose subclasses are total (every entity in the superclass must belong to (at least) one of the subclasses).
Example: Option 8A

(a) EMPLOYEE

<table>
<thead>
<tr>
<th>SSN</th>
<th>FName</th>
<th>MInit</th>
<th>LName</th>
<th>BirthDate</th>
<th>Address</th>
<th>JobType</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>SSN</th>
<th>TypingSpeed</th>
<th></th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>SSN</th>
<th>TGrade</th>
<th></th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>SSN</th>
<th>EngType</th>
<th></th>
</tr>
</thead>
</table>
Example: Option 8B

```
(b) CAR
<table>
<thead>
<tr>
<th>VehicleId</th>
<th>LicensePlateNo</th>
<th>Price</th>
<th>MaxSpeed</th>
<th>NoOfPassengers</th>
</tr>
</thead>
</table>

(b) TRUCK
<table>
<thead>
<tr>
<th>VehicleId</th>
<th>LicensePlateNo</th>
<th>Price</th>
<th>NoOfAxles</th>
<th>Tonnage</th>
</tr>
</thead>
</table>
```
Option 8C: Single relation with one type attribute

Create a single relation L with attributes $\text{Attrs}(L) = \{k,a_1,\ldots,a_n\} \cup \{\text{attributes of } S_1\} \cup \ldots \cup \{\text{attributes of } S_m\} \cup \{t\}$ and $\text{PK}(L) = k$. The attribute $t$ is called a type (or discriminating) attribute that indicates the subclass to which each tuple belongs.

Option 8D: Single relation with multiple type attributes

Create a single relation schema L with attributes $\text{Attrs}(L) = \{k,a_1,\ldots,a_n\} \cup \{\text{attributes of } S_1\} \cup \ldots \cup \{\text{attributes of } S_m\} \cup \{t_1, t_2, \ldots, t_m\}$ and $\text{PK}(L) = k$. Each $t_i$, $1 < i < m$, is a Boolean type attribute indicating whether a tuple belongs to the subclass $S_i$. 
Example: Option 8C

(c) EMPLOYEE

<table>
<thead>
<tr>
<th>SSN</th>
<th>FName</th>
<th>MInit</th>
<th>LName</th>
<th>BirthDate</th>
<th>Address</th>
<th>JobType</th>
<th>TypingSpeed</th>
<th>TGrade</th>
<th>EngType</th>
</tr>
</thead>
</table>

Relational Data Model & ER/EER-to-Relational Mapping
Example: Option 8D

```
<table>
<thead>
<tr>
<th>PART</th>
</tr>
</thead>
<tbody>
<tr>
<td>PartNo</td>
</tr>
<tr>
<td>Description</td>
</tr>
<tr>
<td>MANUFACTURED_PART</td>
</tr>
<tr>
<td>SupplierName</td>
</tr>
<tr>
<td>PURCHASED_PART</td>
</tr>
<tr>
<td>SupplierName</td>
</tr>
<tr>
<td>ListPrice</td>
</tr>
<tr>
<td>ManufactureDate</td>
</tr>
<tr>
<td>DrawingNo</td>
</tr>
<tr>
<td>BatchNo</td>
</tr>
<tr>
<td>MFlag</td>
</tr>
<tr>
<td>Description</td>
</tr>
<tr>
<td>PartNo</td>
</tr>
<tr>
<td>ListPrice</td>
</tr>
</tbody>
</table>
```

Relational Data Model & ER/EER-to-Relational Mapping
EER-to-Relational Mapping

- Mapping of Shared Subclasses (Multiple Inheritance)
  - A shared subclass, such as STUDENT_ASSISTANT, is a subclass of several classes, indicating multiple inheritance. These classes must all have the same key attribute; otherwise, the shared subclass would be modeled as a category.
  - We can apply any of the options discussed in Step 8 to a shared subclass, subject to the restriction discussed in Step 8 of the mapping algorithm. Below both 8C and 8D are used for the shared class STUDENT_ASSISTANT.
Example: Mapping of Shared Subclasses

Relational Data Model & ER/EER-to-Relational Mapping
Example: Mapping of Shared Subclasses

<table>
<thead>
<tr>
<th>PERSON</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SSN</td>
<td>Name</td>
<td>BirthDate</td>
<td>Sex</td>
<td>Address</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EMPLOYEE</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SSN</td>
<td>Salary</td>
<td>EmployeeType</td>
<td>Position</td>
<td>Rank</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ALUMNUS</th>
<th>ALUMNUS_DEGREES</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SSN</td>
<td></td>
<td>SSN</td>
<td>Year</td>
<td>Degree</td>
<td>Major</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>STUDENT</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SSN</td>
<td>MajorDept</td>
<td>GradFlag</td>
<td>UndergradFlag</td>
<td>DegreeProgram</td>
<td>Class</td>
</tr>
</tbody>
</table>
EER-to-Relational Mapping

Step 9: Mapping of Union Types (Categories).

- For mapping a category whose defining superclass have different keys, it is customary to specify a new key attribute, called a surrogate key, when creating a relation to correspond to the category.

- In the example below we can create a relation OWNER to correspond to the OWNER category and include any attributes of the category in this relation. The primary key of the OWNER relation is the surrogate key, which we called OwnerId.
Example: Mapping of Union Types

PERSON
- SSN
- DriverLicenseNo
- Name
- Address
- OwnerId

BANK
- BName
- BAddress
- OwnerId

COMPANY
- CName
- CAddress
- OwnerId

OWNER
- OwnerId

REGISTRED_VEHICLE
- VehcicleId
- LicensePlateNumber

CAR
- VehcicleId
- CStyle
- CMake
- CModel
- CYear

TRUCK
- VehcicleId
- TMake
- TModel
- Tonnage
- TYear

OWNED
- OwnerId
- VehcicleId
- PurchaseDate
- LienOrRegular

Relational Data Model & ER/EER-to-Relational Mapping
<table>
<thead>
<tr>
<th></th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Relational Data Model</td>
</tr>
<tr>
<td>2</td>
<td>Main Phases of Database Design</td>
</tr>
<tr>
<td>3</td>
<td>ER-/EER-to-Relational Mapping</td>
</tr>
</tbody>
</table>
Exercise