Chapter 4:
Relational Data Model and ER/EER-to-Relational Mapping
Contents

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

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

  - \( \text{STUDENT}(\text{Name}, \text{SSN}, \text{HomePhone}, \text{Address}, \text{OfficePhone}, \text{Age}, \text{GPA}) \)
  - Degree = ??
  - \( \text{dom}(\text{GPA}) = ?? \)
Basic Concepts

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

A relation r (or relation state, relation instance) of the relation schema R(A1, A2, . . ., An), also denoted by r(R), is a set of n-tuples r = {t1, t2, . . ., tm}.

- Each n-tuple t is an ordered list of n values t = <v1, v2, . . ., vn>, where each value vi, i=1..n, is an element of dom(Ai) or is a special null value. The i\textsuperscript{th} value in tuple t, which corresponds to the attribute Ai, is referred to as t[Ai]
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

<table>
<thead>
<tr>
<th>Relation name</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>STUDENT</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>SSN</th>
<th>HomePhone</th>
<th>Address</th>
<th>OfficePhone</th>
<th>Age</th>
<th>GPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Benjamin Bayer</td>
<td>305-61-2435</td>
<td>373-1616</td>
<td>2918 Bluebonnet Lane</td>
<td>null</td>
<td>19</td>
<td>3.21</td>
</tr>
<tr>
<td>Katherine Ashly</td>
<td>381-62-1245</td>
<td>375-4409</td>
<td>125 Kirby Road</td>
<td>null</td>
<td>18</td>
<td>2.89</td>
</tr>
<tr>
<td>Dick Davidson</td>
<td>422-11-2320</td>
<td>null</td>
<td>3452 Elgin Road</td>
<td>749-1253</td>
<td>25</td>
<td>3.53</td>
</tr>
<tr>
<td>Charles Cooper</td>
<td>489-22-1100</td>
<td>376-9821</td>
<td>265 Lark Lane</td>
<td>749-6492</td>
<td>28</td>
<td>3.93</td>
</tr>
<tr>
<td>Barbara Benson</td>
<td>533-69-1238</td>
<td>839-8461</td>
<td>7384 Fontana Lane</td>
<td>null</td>
<td>19</td>
<td>3.25</td>
</tr>
</tbody>
</table>

Tuples
## 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
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), \( t_1[SK] \neq t_2[SK] \)

- **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:

\[
\text{CAR(State, Reg\#), SerialNo, Make, Model, Year) has two keys}
\]

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

![Database Table Image]

- **DEPARTMENT**
  - DNAME: Research, Administration, Headquarters
  - DNUMBER: 5, 4, 1
  - MGRSSN: 333445555, 987654321, 888665555

- **EMPLOYEE**
  - FNAME: John, Franklin, Alicia, Jennifer, Ramesh, Joyce, Ahmad, James
  - LNAME: Smith, Wong, Zelaya, Wallace, Narayan, English, Jabbar, Borg
  - SSN: 123456789, 333445555, 999887777, 987654321, 666884444, 453453453, 987987987, 888665555
  - ADDRESS: 731 Fondren, Houston, TX, 638 Voss, Houston, TX, 3321 Castle, Spring, TX, 291 Berry, Bellaire, TX, 975 Fire Oak, Humble, TX, 5631 Rice, Houston, TX, 980 Dallas, Houston, TX, 450 Stone, Houston, TX
  - SALARY: 30000, 40000, 25000, 43000, 38000, 25000, 25000, 55000
  - SUPERSSN: 333445555, 888665555, 987654321, 333445555, 333445555, 333445555, 987654321, null
  - DNO: 5, 5, 4, 4, 5, 5, 4, 1
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

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
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 !!
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
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
ER-to-Relational Mapping

**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
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
**ER-to-Relational Mapping**

- **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:

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

**Relationships Types**

- EMPLOYEE
  - Fname
  - Lname
  - Name
  - Address
  - Sex
  - Salary
  - Bdate
  - Supervisor
  - Supervisee
  - Dependants_of

- DEPARTMENT
  - Name
  - Locations
  - NumberOfEmployees

- MANAGES
  - 1
  - Works_for

- SUPERVISION
  - 1
  - Works_on

- DEPENDENT
  - Name
  - Sex
  - BirthDate
  - Relationship
Result of mapping the COMPANY ER schema into a relational schema
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.
ER-to-Relational Mapping

- 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
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:
  
  \[ \text{ENROL(SNUMBER, MDL-NUMBER, DATE)} \]
The ERD for the COMPANY database

[Diagram of an entity-relationship diagram (ERD) showing relationships between entities such as Employee, Department, Project, Supervision, and Dependents.]
Result of mapping the COMPANY ER schema into a relational schema
ER-to-Relational Mapping

- 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

![Diagram of ER-to-Relational Mapping]

Jan - 2014  CuuDuongThanCong.com  https://fb.com/tailieudientucntt
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)
ER-to-Relational Mapping

- **Example 2: 1:N involuted relationship**
  - If the relationship is mandatory or almost mandatory:
    \[ \text{EMPLOYEE}(\text{ID}, \text{ENAME}, ..., \text{SUPERVISOR}_\text{ID}) \]
  - If the relationship is optional:
    \[ \text{EMPLOYEE}(\text{ID}, \text{ENAME}, ...) \]
    \[ \text{SUPERVISE}(\text{ID}, \text{START}_\text{DATE}, ..., \text{SUPERVISOR}_\text{ID}) \]

- **Example 3: N:M involuted relationship**
  \[ \text{PART}(\text{PNUMBER}, \text{DESCRIPTION}, ...) \]
  \[ \text{COMPRISER}(\text{MAJOR}-\text{PNUMBER}, \text{MINOR}-\text{PNUMBER}, \text{QUANTITY}) \]
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
ER-to-Relational Mapping

- **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
ER-to-Relational Mapping

**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

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

**ER Model**

- Entity type
- 1:1 or 1:N relationship type
- M:N relationship type
- n-ary relationship type
- Simple attribute
- Composite attribute
- Multivalued attribute
- Value set
- Key attribute

**Relational Model**

- “Entity” relation
- Foreign key (or “relationship” relation)
- “Relationship” relation & 2 foreign keys
- “Relationship” relation & n foreign keys
- Attribute
- Set of simple component attributes
- Relation and foreign key
- Domain
- Primary (or secondary) key
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
EER-to-Relational Mapping

- **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
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. **This option works only for specialization whose subclass are disjoint.**

### 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
Example: Option 8D

(d) PART

<table>
<thead>
<tr>
<th>PartNo</th>
<th>Description</th>
<th>MFlag</th>
<th>DrawingNo</th>
<th>ManufactureDate</th>
<th>BatchNo</th>
<th>PFlag</th>
<th>SupplierName</th>
<th>ListPrice</th>
</tr>
</thead>
</table>

CuuDuongThanCong.com

https://fb.com/tailieudientucntt
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
Example: Mapping of Shared Subclasses

\[\text{PERSON} \quad \text{EMPLOYEE} \quad \text{ALUMNUS} \quad \text{ALUMNUS\_DEGREES} \quad \text{STUDENT}\]

- \text{PERSON}:
  - \text{Ssn} \quad \text{Name} \quad \text{Birth\_date} \quad \text{Sex} \quad \text{Address}

- \text{EMPLOYEE}:
  - \text{Ssn} \quad \text{Salary} \quad \text{Employee\_type} \quad \text{Position} \quad \text{Rank} \quad \text{Percent\_time} \quad \text{Ra\_flag} \quad \text{Ta\_flag} \quad \text{Project} \quad \text{Course}

- \text{ALUMNUS} \quad \text{ALUMNUS\_DEGREES}:
  - \text{Ssn}
  - \text{Ssn} \quad \text{Year} \quad \text{Degree} \quad \text{Major}

- \text{STUDENT}:
  - \text{Ssn} \quad \text{Major\_dept} \quad \text{Grad\_flag} \quad \text{Undergrad\_flag} \quad \text{Degree\_program} \quad \text{Class} \quad \text{Student\_assist\_flag}
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
Contents

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

1) Define the following terms as they apply to the relational model of data: **domain**, **attribute**, **n-tuple**, **relation schema**, **relation state**, **degree of a relation**, **relational database schema**, and **relational database state**.

2) Discuss the **entity integrity** and **referential integrity constraints**. Why is each considered important?