B.Sc. (General) Degree Program
Faculty of Applied Sciences
University of Sri Jayewardenepura

CSC209 2.0 Database Management Systems

Presented By:
Surani Tissera(PhD)
Department of Computer Science

The Relational Data Model and Relational Database Constraints

You will learn

The Relational Database Model

- Relational Model Concepts
- Relational Model Constraints and Relational Database Schemas
- Update Operations and Dealing with Constraint Violations

Relational Model Concepts

- The relational Model of Data is based on the concept of a Relation
- The strength of the relational approach to data management comes from the formal foundation provided by the theory of relations

Relational Model Concepts

- A Relation is a mathematical concept based on the ideas of sets.
- The model was first proposed by Dr. E.F. Codd of IBM Research in 1970 in the following paper:

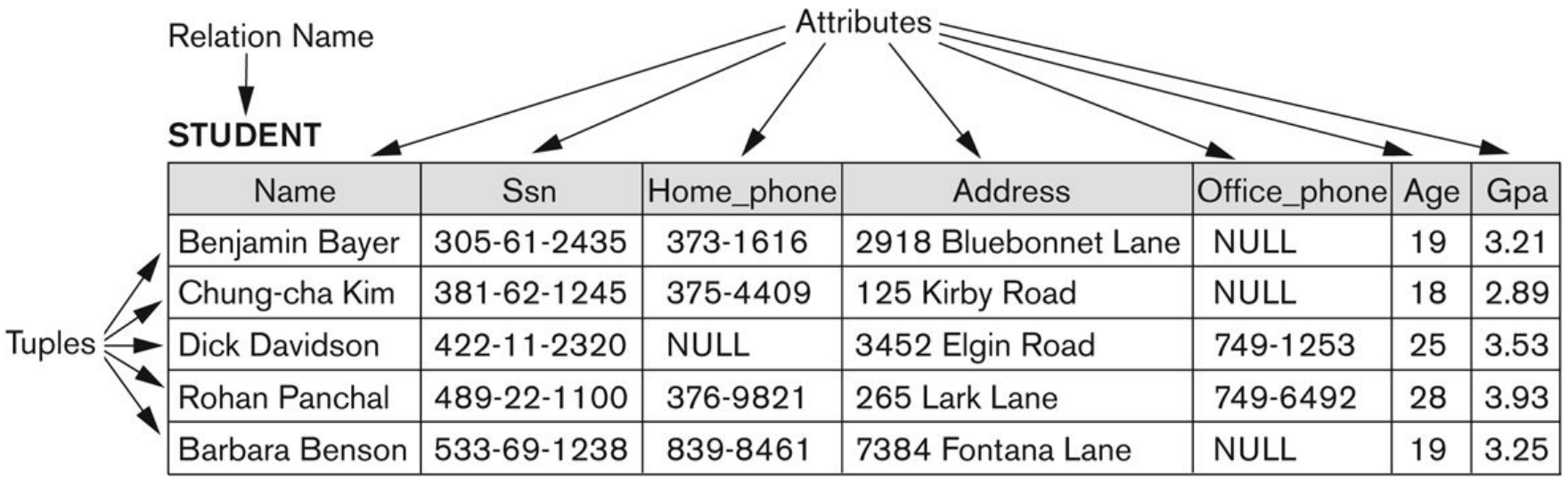
  "A Relational Model for Large Shared Data Banks," Communications of the ACM, June 1970
- The above paper caused a major revolution in the field of database management and earned Dr. Codd the coveted ACM Turing Award
-

Informal Definition

- Informally, a relation looks like a table of values.
- A relation typically contains a set of rows.
- The data elements in each row represent certain facts that correspond to a real-world entity or relationship
  - In the formal model, rows are called **tuples**
- Each column has a column header that gives an indication of the meaning of the data items in that column
  - In the formal model, the column header is called an **attribute** name (or just attribute)

Example of a Relation

-



Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	749-1253	25	3.53
Rohan Panchal	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	NULL	19	3.25

Figure 5.1
The attributes and tuples of a relation STUDENT.

Informal Definition

- Key of a Relation:
- Each row has a value of a data item (or set of items) that **uniquely identifies** that row in the table
  - Called the **key**
- In the STUDENT table, SSN is the key
- Sometimes **row-ids or sequential numbers are assigned as keys to identify the rows in a table**
  - Called artificial key or surrogate key

Formal Definitions - Schema

The Schema (or description) of a Relation:

- Denoted by R(A1, A2, ...An)
- R is the name of the relation
- The attributes of the relation are A1, A2, ..., An

Example:

CUSTOMER (Cust-id, Cust-name, Address, PhoneNo)

- CUSTOMER is the relation name
- Defined over the four attributes:
  Cust-id, Cust-name, Address, PhoneNo

Formal Definitions - Schema

- Each attribute has a domain or a set of valid values.

  For example, the domain of **Cust-id is 6 digit numbers.**

Formal Definitions - Tuple

- A tuple is an ordered set of values (enclosed in angled brackets ‘< … >’)
- Each value is derived from an appropriate domain.
- A row in the CUSTOMER relation is a 4-tuple and would consist of four values, for example:

  <632895, "John Smith", "101 Main St. Atlanta, GA 30332", "(404) 894-2000">
  - This is called a 4-tuple as it has 4 values
  - A tuple (row) in the CUSTOMER relation.

- A relation is a set of such tuples (rows)

Formal Definitions - Domain

- A domain has a logical definition:
  - Example: “USA_phone_numbers” are the set of 10 digit phone numbers valid in the U.S.
- A domain also has a data-type or a format defined for it.
  - The USA_phone_numbers may have a format: (ddd)ddd-dddd where each d is a decimal digit.
  - Dates have various formats such as year, month, date formatted as yyyy-mm-dd, or as dd mm,yyyy etc.

Formal Definitions - Domain

- The attribute name designates the role played by a domain in a relation:
  - Used to interpret the meaning of the data elements corresponding to that attribute
  - Example: The domain Date may be used to define two attributes named “Invoice-date” and “Payment-date” with different meanings

Formal Definitions - State

- The relation state is a subset of the Cartesian product of the domains of its attributes
  - each domain contains the set of all possible values the attribute can take.
- Example: attribute Cust-name is defined over the domain of character strings of maximum length 25
  - dom(Cust-name) is varchar(25)
- The role these strings play in the CUSTOMER relation is that of the name of a customer.

Formal Definitions - Summary

- Formally,
  - Given R(A1, A2, ..., An)
  - r(R) ⊂ dom (A1) X dom (A2) X ....X dom(An)
- R(A1, A2, …, An) is the schema of the relation
- R is the name of the relation
- A1, A2, …, An are the attributes of the relation
- r(R): a specific state (or "value" or “population”) of relation R – this is a set of tuples (rows)
  - r(R) = {t1, t2, …, tn} where each ti is an n-tuple
  - ti = <v1, v2, …, vn> where each vj element-of dom(Aj)

Formal Definitions - Example

- Let R(A1, A2) be a relation schema:
  - Let dom(A1) = {0,1}
  - Let dom(A2) = {a,b,c}
- Then: dom(A1) X dom(A2) is all possible combinations:
  - {<0,a> , <0,b> , <0,c>, <1,a>, <1,b>, <1,c> }
- The relation state r(R) $\subset$ dom(A1) X dom(A2)
- For example: r(R) could be {<0,a> , <0,b> , <1,c> }
  - this is one possible state (or "population" or "extension") r of the relation R, defined over A1 and A2.
  - It has three 2-tuples: <0,a> , <0,b> , <1,c>

Definitions - Summary

Informal Terms		Formal Terms
Table		Relation
Column Header		Attribute
All possible Column Values		Domain
Row		Tuple
Table Definition		Schema of a Relation
Populated Table		State of the Relation

Example - A Relation Student

-

Relation Name

STUDENT

Attributes

Tuples

Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	749-1253	25	3.53
Rohan Panchal	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	NULL	19	3.25

Figure 5.1
The attributes and tuples of a relation STUDENT.

A Characteristic of Relation

- Ordering of tuples in a relation r(R):
  - The tuples are **not considered to be ordered,** even though they appear to be in the tabular form.
- Ordering of attributes in a relation schema R (and of values within each tuple):
  - We will consider the attributes in R(A1, A2, ..., An) and the values in t=<v1, v2, ..., vn> to be ordered .
    - (However, a more general alternative definition of relation does not require this ordering).

Same state as previous Figure (but with different order of tuples)

Figure 5.2
The relation STUDENT from Figure 5.1 with a different order of tuples.

STUDENT

Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	749-1253	25	3.53
Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	NULL	19	3.25
Rohan Panchal	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
Chung-cha Kim	381-62-1245	375-4409	125 Kirby Road	NULL	18	2.89
Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	NULL	19	3.21

A Characteristic of Relation

- Values in a tuple:
  - All values are considered atomic (indivisible).
  - Each value in a tuple must be from the domain of the attribute for that column
    - If tuple t = <v1, v2, …, vn> is a tuple (row) in the relation state r of R(A1, A2, …, An)
    - Then each vi must be a value from dom(Ai)
- A special null value is used to represent values that are unknown or inapplicable to certain tuples.

A Characteristic of Relation

- In general, we can have several meanings for NULL values,
    - value unknown,
    - value exists but is not available,
    - attribute does not apply to this tuple (also known as value undefined).
- Example of the last type of NULL is add an attribute Visa_status to the STUDENT relation. It applies only to tuples representing foreign students.

A Characteristic of Relation

- Notation:
  - We refer to component values of a tuple t by:
    - t[Ai] or t.Ai
    - This is the value vi of attribute Ai for tuple t
  - Similarly, t[Au, Av, ..., Aw] refers to the subtuple of t containing the values of attributes Au, Av, ..., Aw, respectively in t

Relational Integrity Constraints

- Constraints are conditions that must hold on all valid relation states.
- There are three main types of constraints in the relational model:
  - Key constraints
  - Entity integrity constraints
  - Referential integrity constraints
- Another implicit constraint is the domain constraint
  - Every value in a tuple must be from the domain of its attribute (or it could be null, if allowed for that attribute)

Key Constraints

- Superkey of R:
  - Is a set of attributes SK of R with the following condition:
    - No two tuples in any valid relation state 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]
    - This condition must hold in any valid state r(R)

Key Constraints

- Key of R:
  - A "minimal" superkey
  - That is, a key is a superkey K such that removal of any attribute from K results in a set of attributes that is not a superkey (does not possess the superkey uniqueness property)

Key Constraints

Example: Consider the CAR relation schema:

- CAR(State, Reg#, SerialNo, Make, Model, Year)
- CAR has two keys:
  - Key1 = {State, Reg#}
  - Key2 = {SerialNo}
- Both are also superkeys of CAR
- {SerialNo, Make} is a superkey but not a key.

Key Constraints

- In general, a relation schema may have more than one key.
- In this case, each of the keys is called a **candidate key.**
  - For example, the CAR relation has two candidate keys: License_number and Engine_serial_number.
- It is common to designate one of the candidate keys as the **primary key** of the relation.
- This is the candidate key whose values are used to identify tuples in the relation.
- We use the convention that the attributes that form the primary key of a relation schema are underlined.

Key Constraints

- If a relation has several candidate keys, one is chosen arbitrarily to be the primary key.
  - The primary key attributes are underlined.
- Example: Consider the CAR relation schema:
  - CAR(State, Reg#, SerialNo, Make, Model, Year)
  - We chose SerialNo as the primary key
- The primary key value is used to uniquely identify each tuple in a relation
  - Provides the tuple identity

Key Constraints

Also used to reference the tuple from another tuple

- General rule: Choose as primary key the smallest of the candidate keys (in terms of size)
- Not always applicable – choice is sometimes subjective

CAR table with two candidate keys – LicenseNumber chosen as Primary Key

-

CAR

License_number	Engine_serial_number	Make	Model	Year
Texas ABC-739	A69352	Ford	Mustang	02
Florida TVP-347	B43696	Oldsmobile	Cutlass	05
New York MPO-22	X83554	Oldsmobile	Delta	01
California 432-TFY	C43742	Mercedes	190-D	99
California RSK-629	Y82935	Toyota	Camry	04
Texas RSK-629	U028365	Jaguar	XJS	04

Figure 5.4
The CAR relation, with two candidate keys: License_number and Engine_serial_number.

Relational Database Schema

- Relational Database Schema:
  - A set S of relation schemas that belong to the same database.
  - S is the name of the whole database schema
  - S = {R1, R2, ..., Rn}
  - R1, R2, …, Rn are the names of the individual relation schemas within the database S
- Following slide shows a COMPANY database schema with 6 relation schemas

Company Database Schema

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

DEPT_LOCATIONS

Dnumber	Dlocation

PROJECT

Pname	Pnumber	Plocation	Dnum

WORKS_ON

Essn	Pno	Hours

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship

Figure 5.5
Schema diagram for the COMPANY relational database schema.

Entity Integrity

- **Entity Integrity:**
  - The *primary key attributes* PK of each relation schema R in S cannot have null values in any tuple of r(R).
    - This is because primary key values are used to *identify* the individual tuples.
    - t[PK] ≠ null for any tuple t in r(R)
    - If PK has several attributes, null is not allowed in any of these attributes
  - Note: Other attributes of R may be constrained to disallow null values, even though they are not members of the primary key.

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**.

Referential Integrity

- Tuples in the **referencing relation** R1 have attributes FK (called **foreign key** attributes) that reference the primary key attributes PK of the **referenced relation** R2.
  - A tuple t1 in R1 is said to **reference** a tuple t2 in R2 if t1[FK] = t2[PK].
- A referential integrity constraint can be displayed in a relational database schema as a directed arc from R1.FK to R2.

Referential Integrity (Foreign Key) constraint

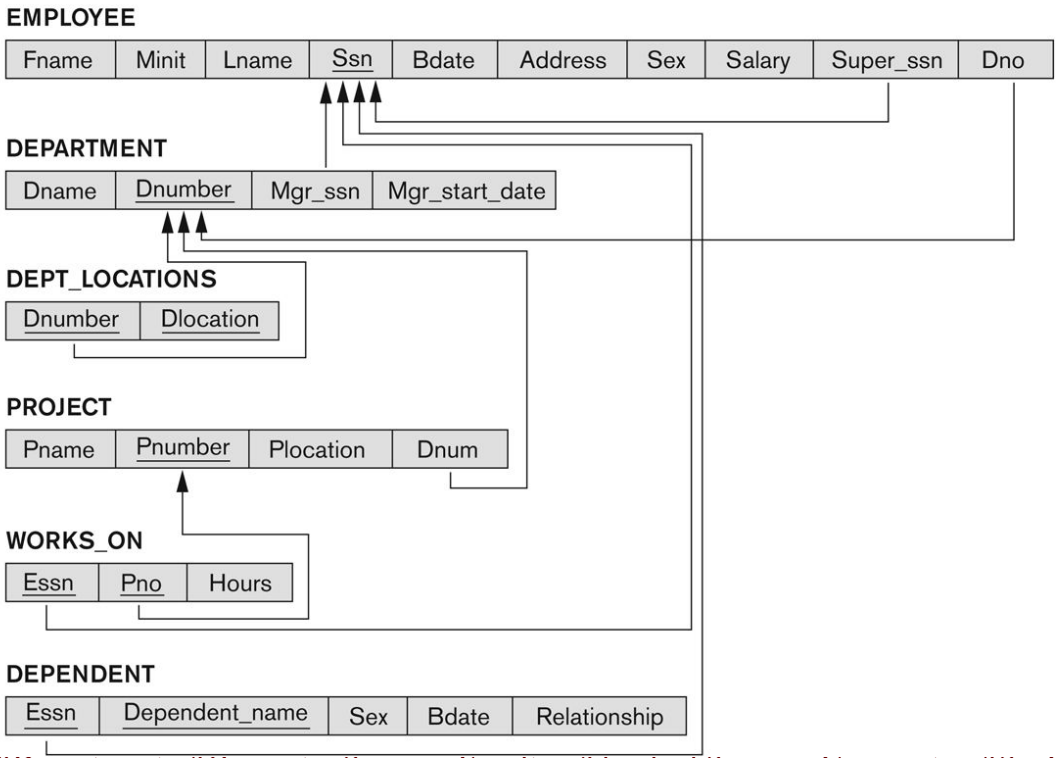
- Statement of the constraint
  - The value in the foreign key column (or columns) FK of the the **referencing relation** R1 can be **either**:
    - (1) a value of an existing primary key value of a corresponding primary key PK in the **referenced relation** R2, or
    - (2) a **null**.
- In case (2), the FK in R1 should **not** be a part of own primary key.

Referential Integrity (Foreign Key) constraint

- Statement of the constraint
  - The value in the foreign key column (or columns) FK of the the **referencing relation** R1 can be **either**:
    - (1) a value of an existing primary key value of a corresponding primary key PK in the **referenced relation** R2, or
    - (2) a **null**.
- In case (2), the FK in R1 should **not** be a part of own primary key.

Referential Integrity Constraints for COMPANY database

Figure 5.7
Referential integrity constraints displayed on the COMPANY relational database schema.



Other Types of Constraints

- Semantic Integrity Constraints:
  - based on application semantics and cannot be expressed by the model per se
  - Example: “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 express for some of these

Populated Database State

- Each *relation* will have many tuples in its current relation state
- The *relational database state* is a union of all the individual relation states
- Whenever the database is changed, a new state arises
- Basic operations for changing the database:
  - INSERT a new tuple in a relation
  - DELETE an existing tuple from a relation
  - MODIFY an attribute of an existing tuple
- Next slide shows an example state for the COMPANY database

Populated Database State for Company

Figure 5.6

One possible database state for the COMPANY relational database schema.

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

Update Operations on Relations

- INSERT a tuple.
- DELETE a tuple.
- MODIFY a tuple.
- Integrity constraints should not be violated by the update operations.
- Several update operations may have to be grouped together.
- Updates may **propagate** to cause other updates automatically. This may be necessary to maintain integrity constraints.

Update Operations on Relations

- In case of integrity violation, several actions can be taken:
  - Cancel the operation that causes the violation (RESTRICT or 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

Possible violation for each operation

- INSERT may violate any of the constraints:
  - Domain constraint:
    - if one of the attribute values provided for the new tuple is not of the specified attribute domain
  - Key constraint:
    - if the value of a key attribute in the new tuple already exists in another tuple in the relation
  - Referential integrity:
    - if a foreign key value in the new tuple references a primary key value that does not exist in the referenced relation
  - Entity integrity:
    - if the primary key value is null in the new tuple

Example - INSERT

- Operation: Insert <‘Cecilia’, ‘F’, ‘Kolonsky’, NULL, ‘1960-04-05’, ‘6357 Windy Lane, Katy, TX’, F, 28000, NULL, 4> into EMPLOYEE. Result: This insertion violates the entity integrity constraint (NULL for the primary key Ssn), so it is rejected.
- Operation: Insert <‘Alicia’, ‘J’, ‘Zelaya’, ‘999887777’, ‘1960-04-05’, ‘6357 Windy Lane, Katy, TX’, F, 28000, ‘987654321’, 4> into EMPLOYEE. Result: This insertion violates the key constraint because another tuple with the same Ssn value already exists in the EMPLOYEE relation, and so it is rejected.
- Operation: Insert <‘Cecilia’, ‘F’, ‘Kolonsky’, ‘677678989’, ‘1960-04-05’, ‘6357 Windswept, Katy, TX’, F, 28000, ‘987654321’, 7> into EMPLOYEE. Result: This insertion violates the referential integrity constraint specified on Dno in EMPLOYEE because no corresponding referenced tuple exists in DEPARTMENT with Dnumber = 7.

Possible violation for each operation

- DELETE may violate only referential integrity:
  - If the primary key value of the tuple being deleted is referenced from other tuples in the database
    - Can be remedied by several actions: RESTRICT, CASCADE, SET NULL
      - RESTRICT option: reject the deletion
      - CASCADE option: propagate the new primary key value into the foreign keys of the referencing tuples
      - SET NULL option: set the foreign keys of the referencing tuples to NULL
  - One of the above options must be specified during database design for each foreign key constraint

Example - DELETE

- Operation:

Delete the WORKS_ON tuple with Essn = ‘999887777’ and Pno = 10.
Result: This deletion is acceptable and deletes exactly one tuple.

- Operation:

Delete the EMPLOYEE tuple with Ssn = ‘999887777’.
Result: This deletion is not acceptable, because there are tuples in WORKS_ON that refer to this tuple. Hence, if the tuple in EMPLOYEE is deleted, referential integrity violations will result.

- Operation:

Delete the EMPLOYEE tuple with Ssn = ‘333445555’.
Result: This deletion will result in even worse referential integrity violations, because the tuple involved is referenced by tuples from the EMPLOYEE, DEPARTMENT, WORKS_ON, and DEPENDENT relations.

Possible violation for each operation

- UPDATE may violate domain constraint and NOT NULL constraint on an attribute being modified
- Any of the other constraints may also be violated, depending on the attribute being updated:
  - Updating the primary key (PK):
    - Similar to a DELETE followed by an INSERT
    - Need to specify similar options to DELETE
  - Updating a foreign key (FK):
    - May violate referential integrity
  - Updating an ordinary attribute (neither PK nor FK):
    - Can only violate domain constraints

Example - UPDATE

- Operation:

Update the salary of the EMPLOYEE tuple with Ssn = ‘999887777’ to 28000.
Result: Acceptable.

- Operation:

Update the Dno of the EMPLOYEE tuple with Ssn = ‘999887777’ to 1.
Result: Acceptable.

- Operation:

Update the Dno of the EMPLOYEE tuple with Ssn = ‘999887777’ to 7.
Result: Unacceptable, because it violates referential integrity.

- Operation:

Update the Ssn of the EMPLOYEE tuple with Ssn = ‘999887777’ to ‘987654321’.
Result: Unacceptable, because it violates primary key constraint by repeating a value that already exists as a primary key in another tuple; it violates referential integrity constraints because there are other relations that refer to the existing value of Ssn

Summary

- Presented Relational Model Concepts
  - Definitions
  - Characteristics of relations
- Discussed Relational Model Constraints and Relational Database Schemas
  - Domain constraints'
  - Key constraints
  - Entity integrity
  - Referential integrity
- Described the Relational Update Operations and Dealing with Constraint Violations

Class Exercise

Consider the following relations for a database that keeps track of student enrollment in courses and the books adopted for each course:

- STUDENT(SSN, Name, Major, Bdate)
- COURSE(Course#, Cname, Dept)
- ENROLL(SSN, Course#, Quarter, Grade)
- BOOK_ADOPTION(Course#, Quarter, Book_ISBN)
- TEXT(Book_ISBN, Book_Title, Publisher, Author)

Draw a relational schema diagram specifying the foreign keys for this schema.

The Relational Data Model and Relational Database Constraints

Chapter Outline

- **ER-to-Relational Mapping Algorithm**
  - Step 1: Mapping of Regular Entity Types
  - Step 2: Mapping of Weak Entity Types
  - Step 3: Mapping of Binary 1:1 Relation 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.

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

FIGURE 7.1

The ER conceptual schema diagram for the COMPANY database.

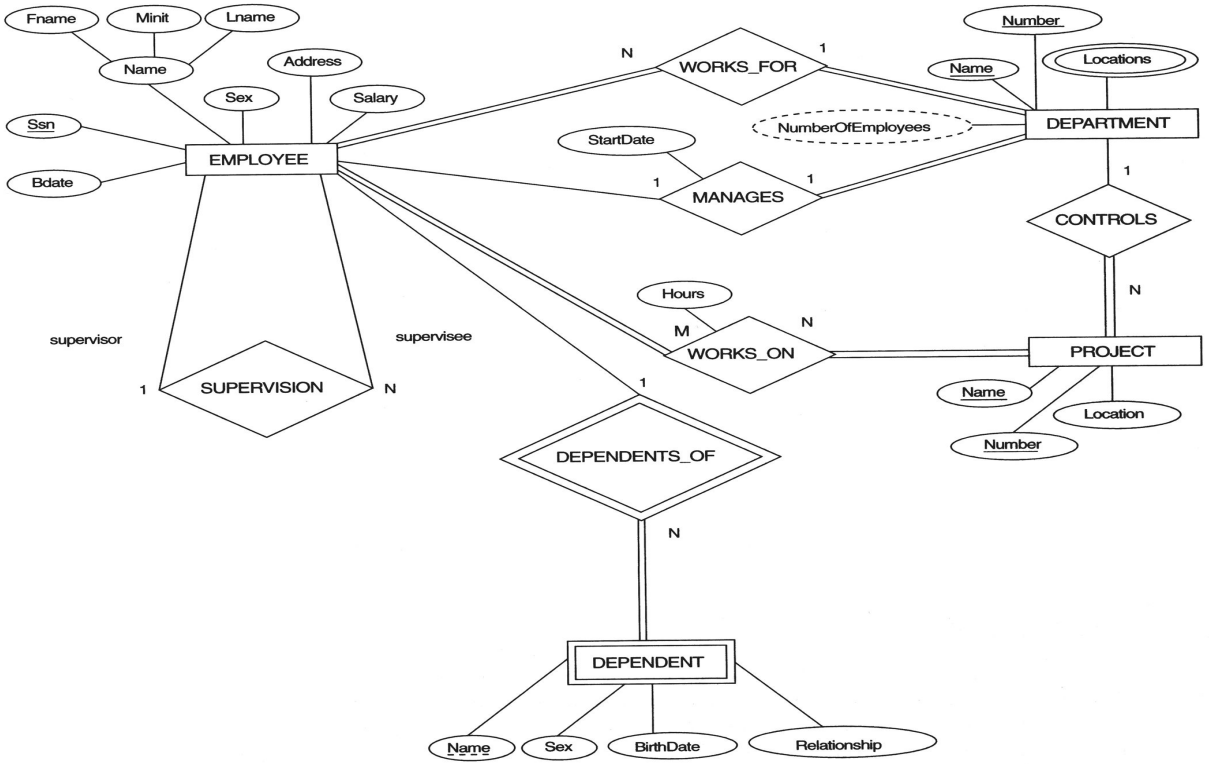


FIGURE 7.2

Result of mapping the COMPANY ER schema into a relational schema.

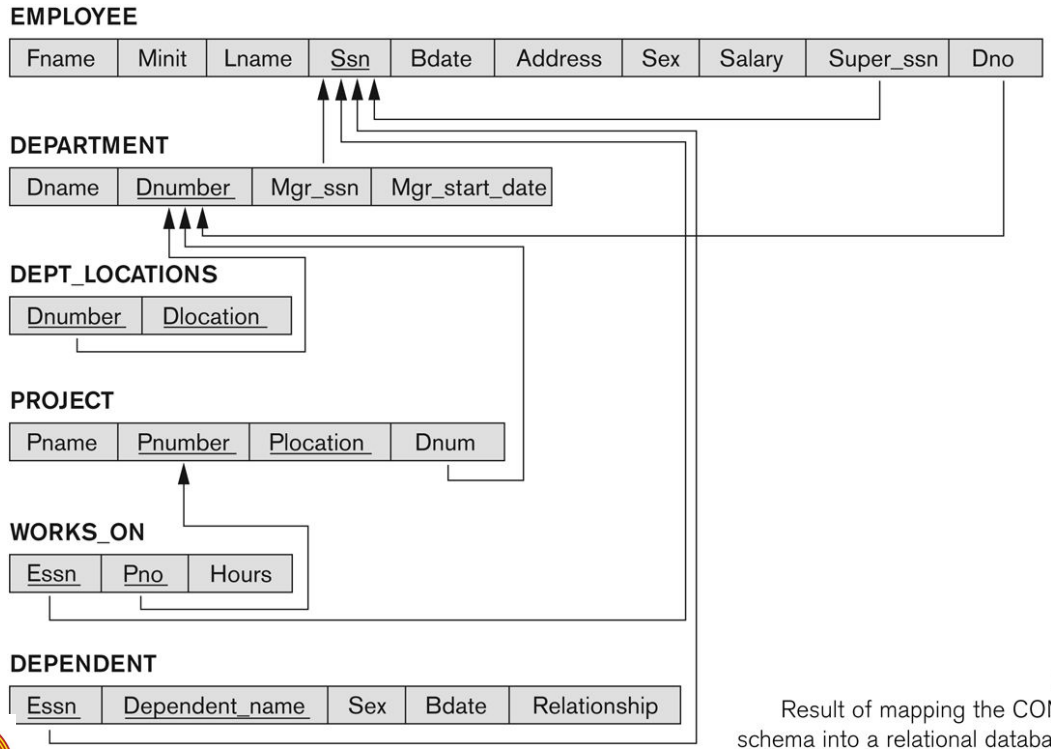


Figure 7.2
Result of mapping the COMPANY ER schema into a relational database schema.

ER-to-Relational Mapping Algorithm

- Step 1: Mapping of Regular Entity Types.
  - For each regular (strong) entity type E in the ER schema, create a relation R that includes all the simple attributes of E.
  - Choose one of the key attributes of E as the primary key for R.
  - If the chosen key of E is composite, the set of simple attributes that form it will together form the primary key of R.

Step 1: Mapping of Regular Entity Types.

- 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.

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary

DEPARTMENT

Dname	Dnumber

PROJECT

Pname	Pnumber	Plocation

ER-to-Relational Mapping Algorithm (contd.)

- **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 & include all simple attributes (or simple components of composite attributes) of W as attributes of R.
  - Also, 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.

Step 2: Mapping of Weak Entity Types

- **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.

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship

ER-to-Relational Mapping Algorithm (contd.)

- **Step 3: Mapping of Binary 1:1 Relation Types**
  - For each binary 1:1 relationship type R in the ER schema, identify the relations S and T that correspond to the entity types participating in R.

Step 3: Mapping of Binary 1:1 Relation Types

- There are three possible approaches:

1. **Foreign Key approach:** Choose one of the relations-say S-and include a foreign key in S the primary key of T. It is better to choose an entity type with total participation in R in the role of S. Include all the simple attributes (or simple components of composite attributes) of the 1:1 relationship type R as attributes of S.
   - Example: 1:1 relation MANAGES is mapped by choosing the participating entity type DEPARTMENT to serve in the role of S, because its participation in the MANAGES relationship type is total.

DEPARTMENT

Dname	Dnumber	Mgr_ssn

1. Foreign Key approach:

- Example: 1:1 relation MANAGES is mapped by choosing the participating entity type DEPARTMENT to serve in the role of S, because its participation in the MANAGES relationship type is total.
- We include the primary key of the EMPLOYEE relation as foreign key in the DEPARTMENT relation and rename it to Mgr_ssn. We also include the simple attribute Start_date of the MANAGES relationship type in the DEPARTMENT relation and rename it Mgr_start_date

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

Step 3: Mapping of Binary 1:1 Relation Types

- There are three possible approaches:

2. **Merged relation option:** An alternate mapping of a 1:1 relationship type is possible by merging the two entity types and the relationship into a single relation. This may be appropriate when both participations are total.

3. **Cross-reference or relationship relation option:** The third alternative is to set up a third relation R for the purpose of cross-referencing the primary keys of the two relations S and T representing the entity types.

ER-to-Relational Mapping Algorithm (contd.)

- Step 4: Mapping of Binary 1:N Relationship Types.
  - For each regular binary 1:N relationship type R, identify the relation S that represent the participating entity type at the N-side of the relationship type.
  - Include as foreign key in S the primary key of the relation T that represents the other entity type participating in R.
  - Include any simple attributes of the 1:N relation type as attributes of S.

Step 4: Mapping of Binary 1:N Relationship Types.

- Example: 1:N relationship types WORKS_FOR, CONTROLS, and SUPERVISION in the figure.
  - For WORKS_FOR we include the primary key DNUMBER of the DEPARTMENT relation as foreign key in the EMPLOYEE relation and call it DNO.

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Dno

  - For SUPERVISION we include the primary key of the EMPLOYEE relation as foreign key in the EMPLOYEE relation itself—because the relationship is recursive—and call it Super_ssn.

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

Step 4: Mapping of Binary 1:N Relationship Types.

- Example: 1:N relationship types WORKS_FOR, CONTROLS, and SUPERVISION in the figure.
  - The CONTROLS relationship is mapped to the foreign key attribute Dnum of PROJECT, which references the primary key Dnumber of the DEPARTMENT relation.

PROJECT

Pname	Pnumber	Plocation	Dnum

ER-to-Relational Mapping Algorithm (contd.)

- **Step 5: Mapping of Binary M:N Relationship Types.**
  - For each regular binary M:N relationship type R, *create a new relation* S to represent R.
  - Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types; *their combination will form the primary key* of S.
  - Also include any simple attributes of the M:N relationship type (or simple components of composite attributes) as attributes of S.

Step 5: Mapping of Binary M:N Relationship Types.

- Example: The M:N relationship type WORKS_ON from the ER diagram is mapped by creating a relation WORKS_ON in the relational database schema.
  - The primary keys of the PROJECT and EMPLOYEE relations are included as foreign keys in WORKS_ON and renamed PNO and ESSN, respectively.
  - Attribute HOURS in WORKS_ON represents the HOURS attribute of the relation type. The primary key of the WORKS_ON relation is the combination of the foreign key attributes {ESSN, PNO}.

ER-to-Relational Mapping Algorithm (contd.)

- **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 of 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.

ER-to-Relational Mapping Algorithm (contd.)

- **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}.

ER-to-Relational Mapping Algorithm (contd.)

- **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.

ER-to-Relational Mapping Algorithm (contd.)

- **Example:** The relationship type SUPPLY in the ER on the next slide.
  - 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}

FIGURE 4.11

Ternary relationship types. (a) The SUPPLY relationship.

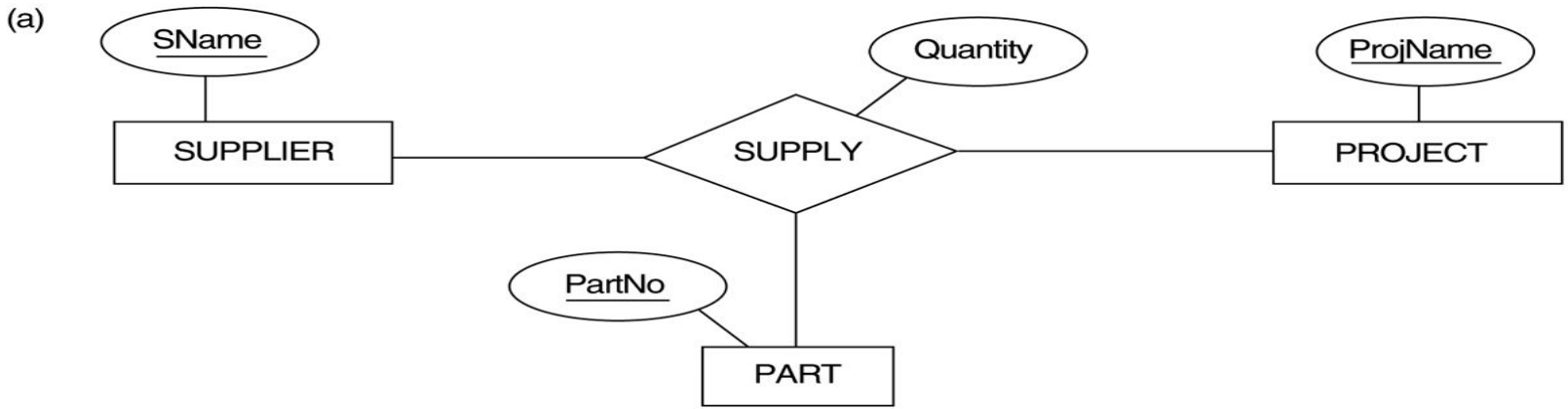
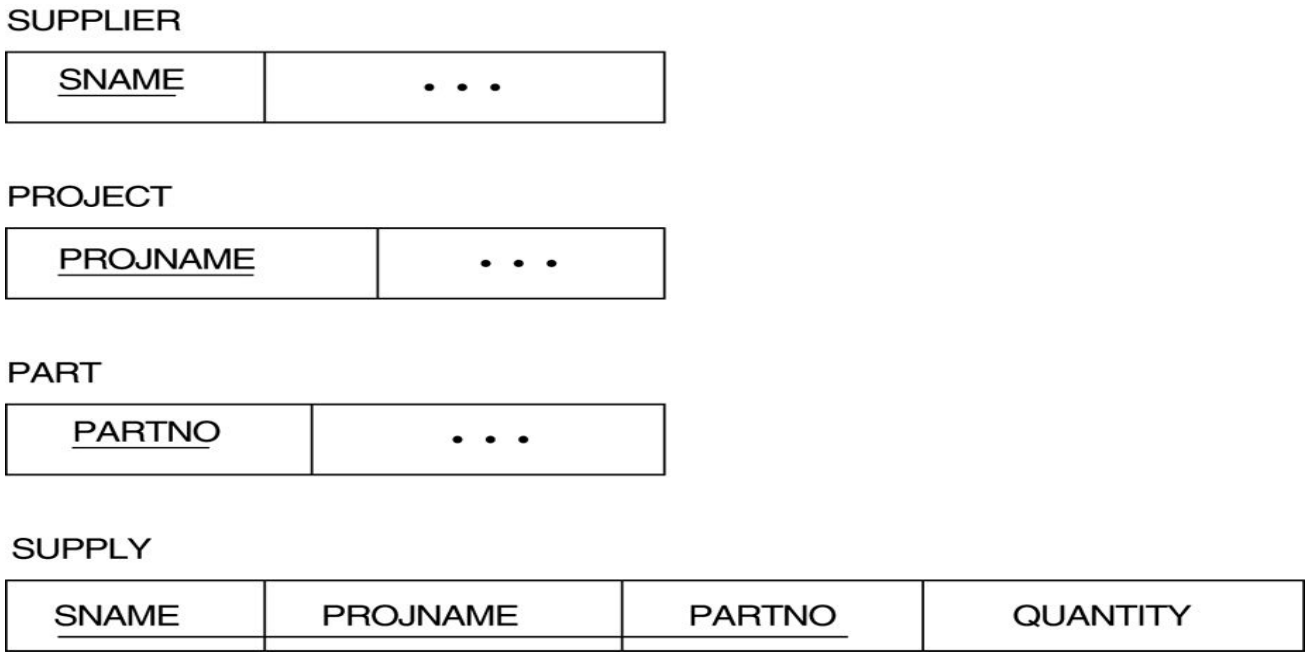


FIGURE 7.3

Mapping the *n*-ary relationship type SUPPLY from Figure 4.11a.



Summary of Mapping constructs and constraints

Table 7.1 Correspondence between ER and Relational Models

ER Model	Relational Model
Entity type	"Entity" relation
1:1 or 1:N relationship type	Foreign key (or "relationship" relation)
M:N relationship type	"Relationship" relation and two foreign keys
n-ary relationship type	"Relationship" relation and n foreign keys
Simple attribute	Attribute
Composite attribute	Set of simple component attributes
Multivalued attribute	Relation and foreign key
Value set	Domain
Key attribute	Primary (or secondary) key

Mapping EER Model Constructs to Relations

- **Step8: Options for Mapping Specialization or Generalization.**
  - Convert each specialization with m subclasses {S1, S2,….,Sm} and generalized superclass C, where the attributes of C are {k,a1,…an} 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

Mapping EER Model Constructs to Relations

- **Option 8A: Multiple relations-Superclass and subclasses**
  - Create a relation L for C with attributes Attrs(L) = {k,a1,…an} and PK(L) = k.
  - Create a relation Li for each subclass Si, 1 < i < m, with the attributesAttrs(Li) = {k} U {attributes of Si} and PK(Li)=k.
  - This option works for any specialization (total or partial, disjoint of over-lapping).

FIGURE 4.4

EER diagram notation for an attribute-defined specialization on JobType.

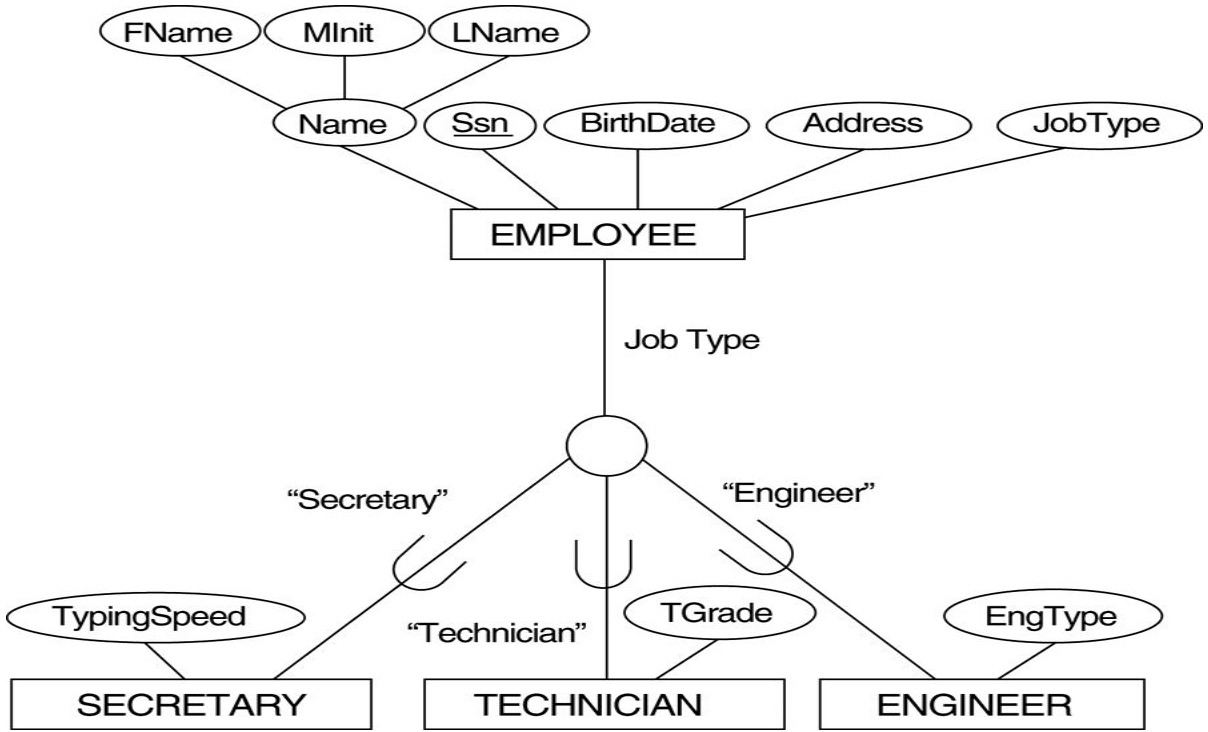
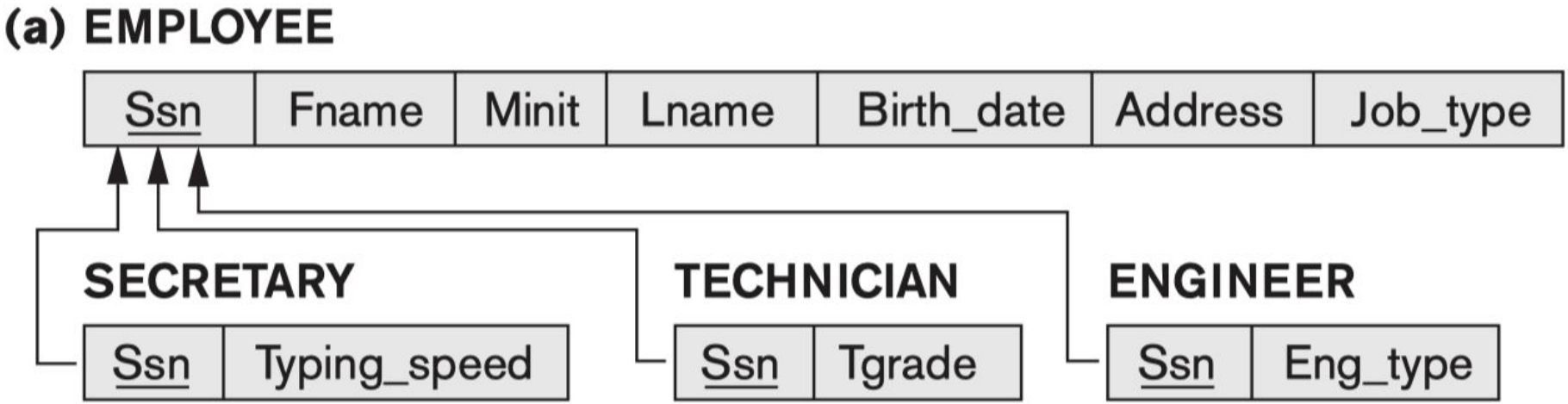


FIGURE 7.4

Options for mapping specialization or generalization.

(a) Mapping the EER schema in Figure 4.4 using option 8A.



Mapping EER Model Constructs to Relations

- **Option 8B: Multiple relations-Subclass relations only**
  - Create a relation Li for each subclass Si, 1 < i < m, with the attributes Attr(Li) = {attributes of Si} U {k,a1…,an} and PK(Li) = 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).

FIGURE 4.3

Generalization. (b) Generalizing CAR and TRUCK into the superclass VEHICLE.

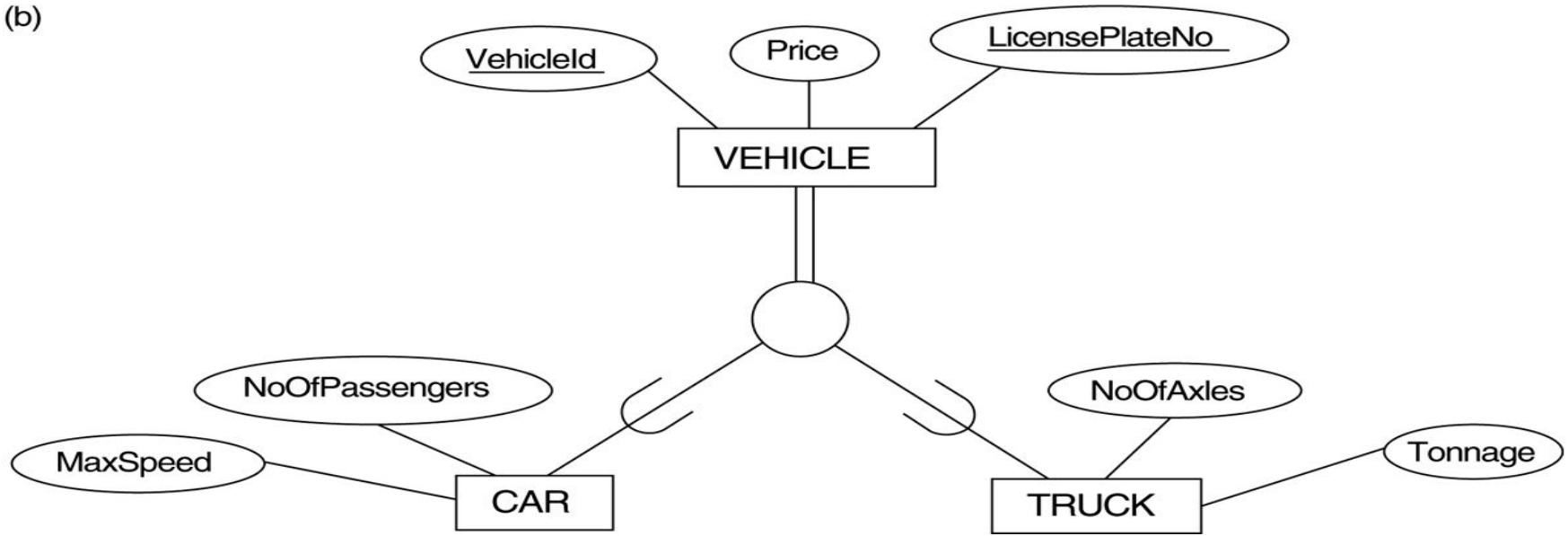


FIGURE 7.4

Options for mapping specialization or generalization.

(b) Mapping the EER schema in Figure 4.3b using option 8B.

(b) CAR

Vehicle_id	License_plate_no	Price	Max_speed	No_of_passengers

TRUCK

Vehicle_id	License_plate_no	Price	No_of_axles	Tonnage

Mapping EER Model Constructs to Relations (contd.)

- **Option 8C: Single relation with one type attribute**
  - Create a single relation L with attributes Attrs(L) = $\{k,a_1,\ldots a_n\}$ U {attributes of S_1} U…U {attributes of S_m} U {t} and PK(L) = k.
  - The attribute t is called a type (or **discriminating**) attribute that indicates the subclass to which each tuple belongs

FIGURE 4.4

EER diagram notation for an attribute-defined specialization on JobType.

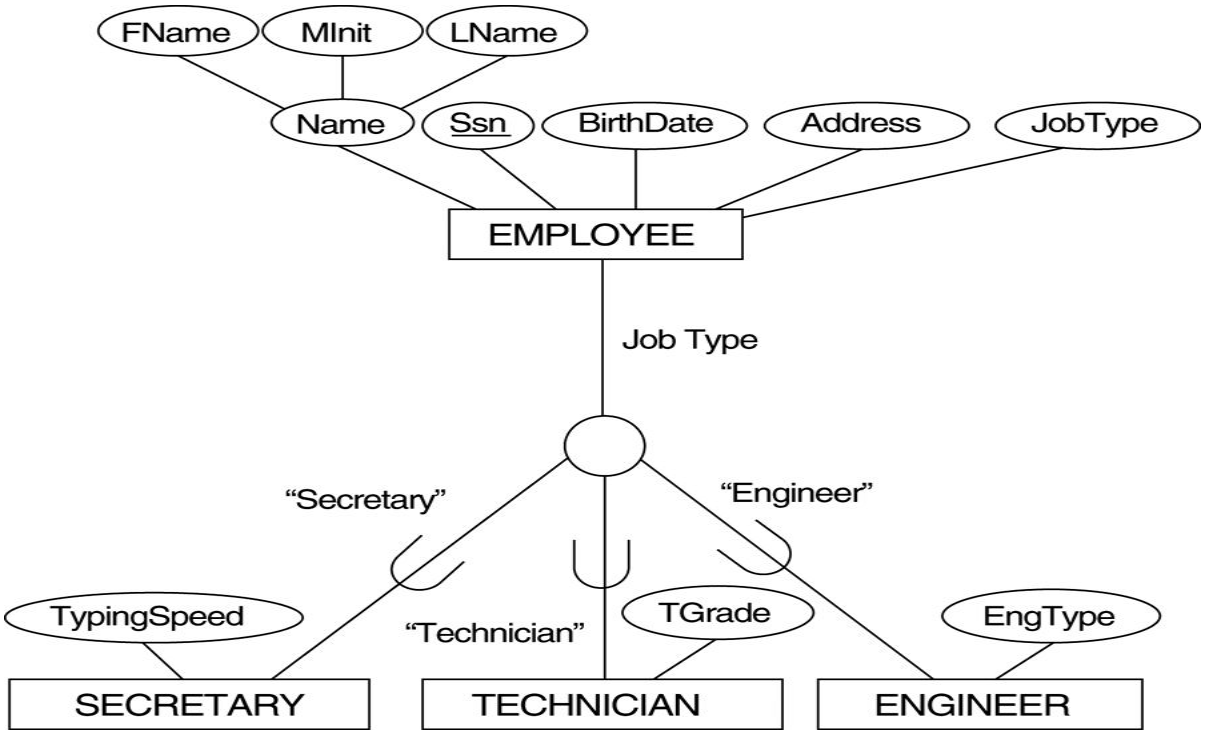


FIGURE 7.4

Options for mapping specialization or generalization.

(c) Mapping the EER schema in Figure 4.4 using option 8C.

(c) EMPLOYEE

Ssn	Fname	Minit	Lname	Birth_date	Address	Job_type	Typing_speed	Tgrade	Eng_type

Mapping EER Model Constructs to Relations (contd.)

- **Option 8D: Single relation with multiple type attributes**
  - Create a single relation schema L with attributes Attrs(L) = $\{k,a_1,\ldots a_n\}$ U {attributes of S_1} U…U {attributes of S_m} U $\{t_1, t_2,\ldots,t_m\}$ and PK(L) = k. Each t_i, 1 < I < m, is a Boolean type attribute indicating whether a tuple belongs to the subclass S_i.

FIGURE 4.5

EER diagram notation for an overlapping (non-disjoint) specialization.

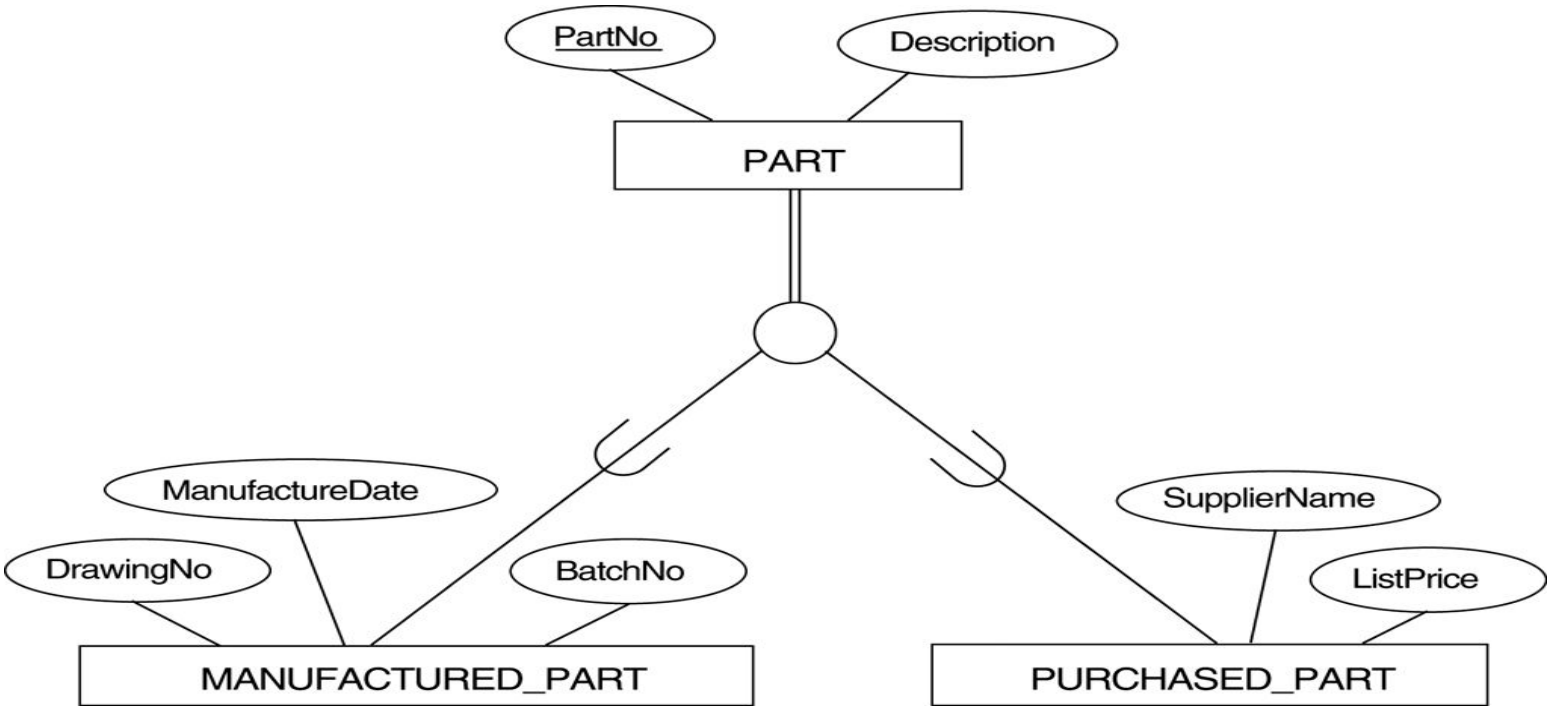


FIGURE 7.4

Options for mapping specialization or generalization. (d) Mapping Figure 4.5 using option 8D with Boolean type fields Mflag and Pflag.

(d) PART

Part_no	Description	Mflag	Drawing_no	Manufacture_date	Batch_no	Pflag	Supplier_name	List_price

Mapping EER Model Constructs to Relations

- 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.

FIGURE 4.7

A specialization lattice with multiple inheritance for a UNIVERSITY database.

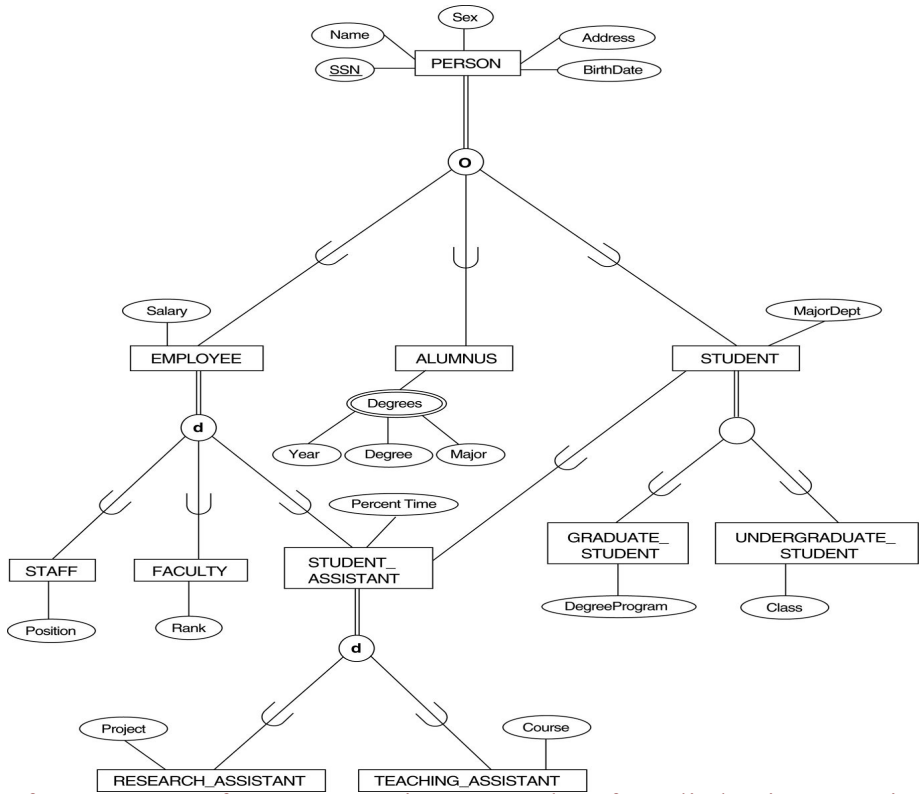


FIGURE 7.5

Mapping the EER specialization lattice in Figure 4.6 using multiple options.

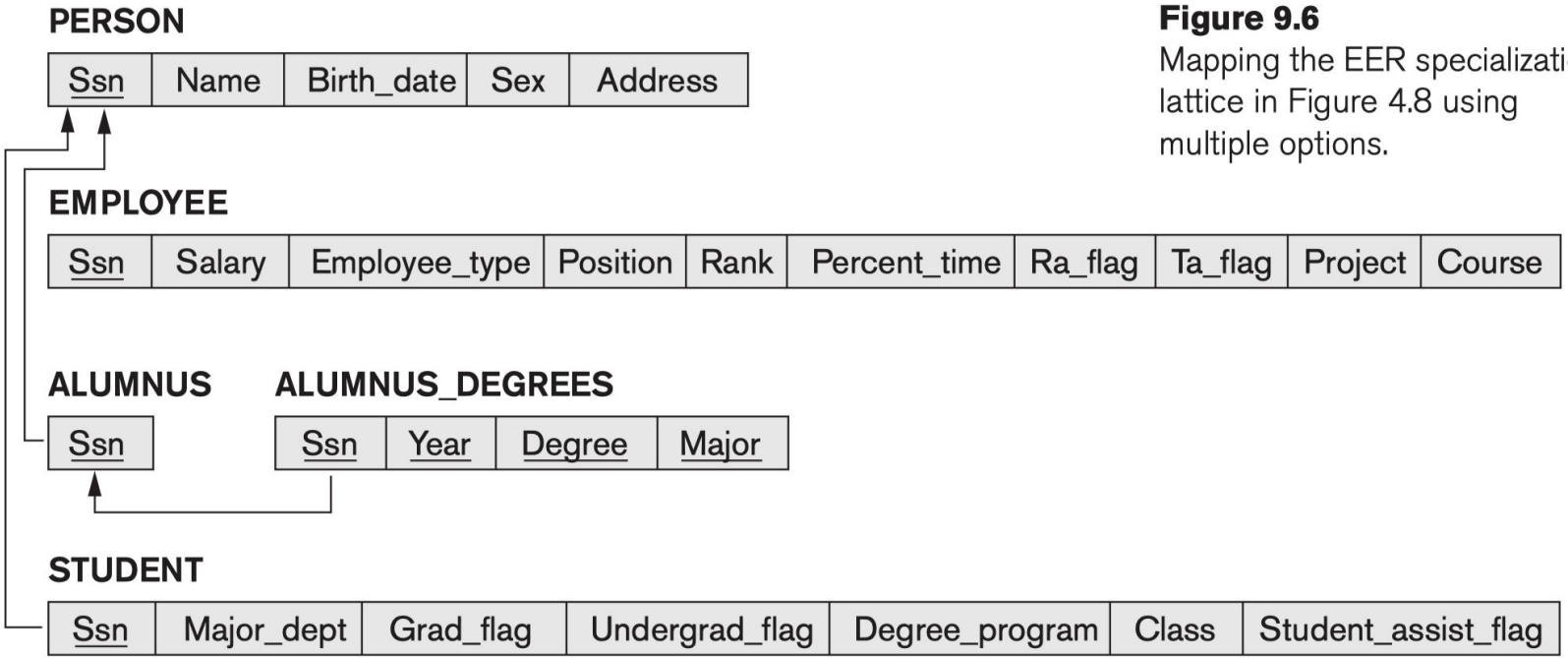


Figure 9.6
Mapping the EER specializati
lattice in Figure 4.8 using
multiple options.

Option 8C is used in the EMPLOYEE relation (Employee_type attribute) and option 8D is used in the STUDENT relation (Student_assist_flag attribute).

Mapping EER Model Constructs to Relations (contd.)

- **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.

FIGURE 4.8

Two categories (union types): OWNER and REGISTERED_VEHICLE.

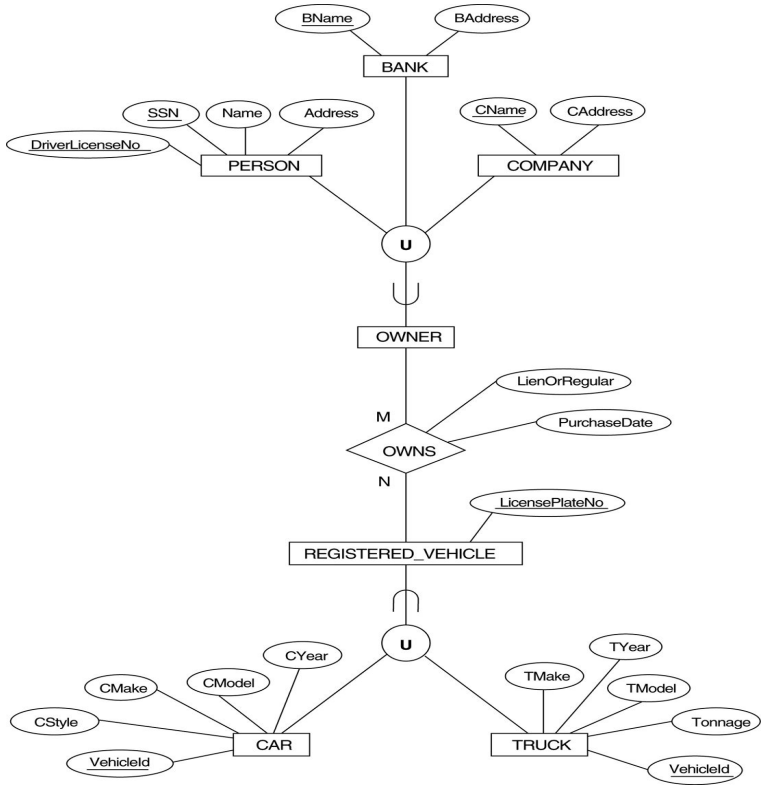
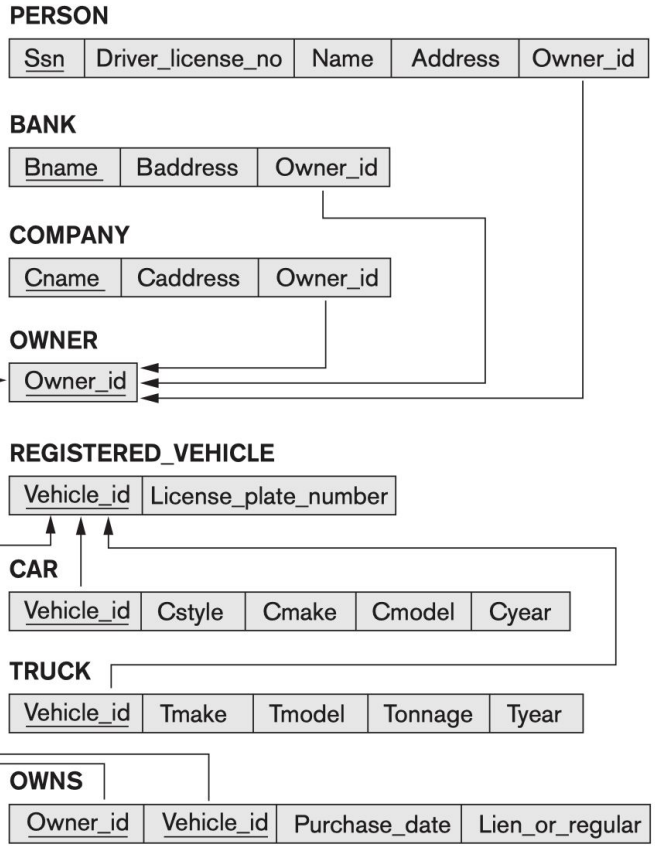


FIGURE 7.6

Mapping the EER categories (union types) in Figure 4.7 to relations.



Mapping Exercise

Exercise 7.4.

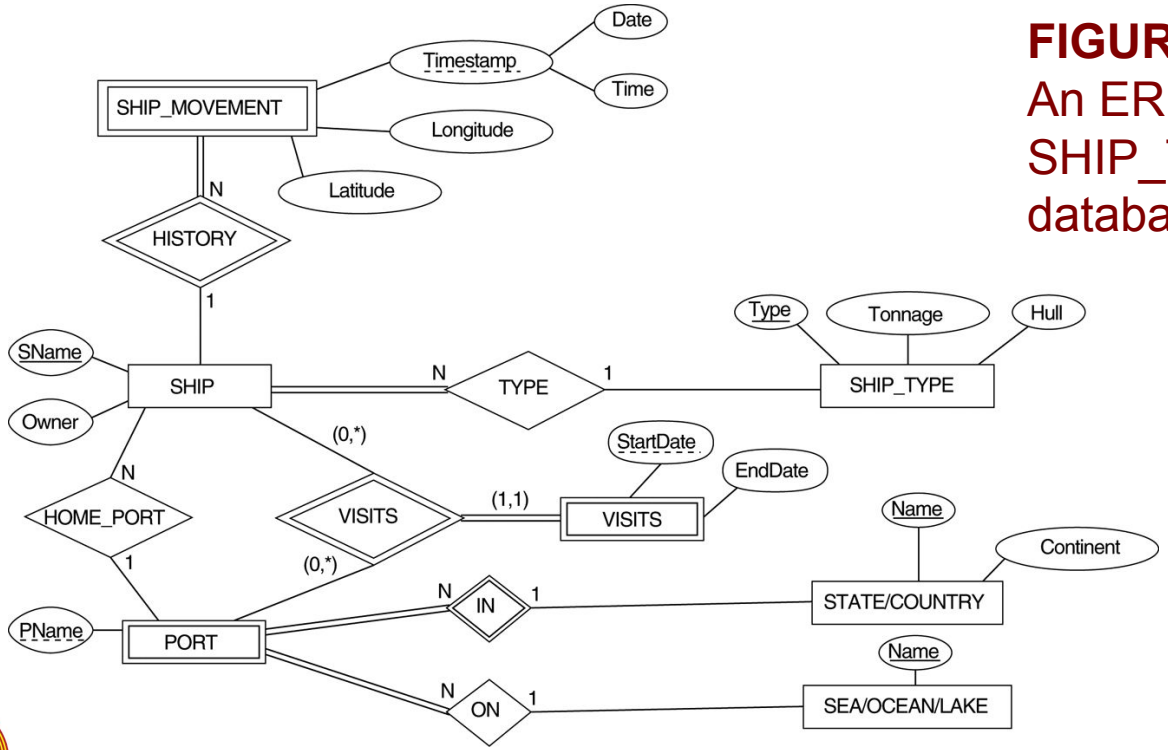


FIGURE 7.7
An ER schema for a SHIP_TRACKING database.

Chapter Summary

- **ER-to-Relational Mapping Algorithm**
  - Step 1: Mapping of Regular Entity Types
  - Step 2: Mapping of Weak Entity Types
  - Step 3: Mapping of Binary 1:1 Relation 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.

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