Module 2: The Relational Data Model

Introduction to Information Systems

In This Module

- What is the Relational Data Model and what are its main components?
- What are integrity constraints and how are they enforced by a DBMS?
- How can we map an ER diagram to a relational schema?

Learning Outcomes

After successfully completing this module you should be able to reason with the logical foundation of the relational data model.

- Define the main components of the relational model: Relations, Domains, Attributes and Tuples.
- Explain and provide examples for each of the integrity constraints.
- Given an ER diagram, map it to a set of relations using the Relational Model.

Relational Model Concepts

Integrity Constraints

ER to Relational Mapping

Relational Model

Introduced by E.F. Codd in 1970

Many DBMS products based on this model

Based on a sound theoretical foundation with a simple and uniform data structure called relation

Four basic concepts:

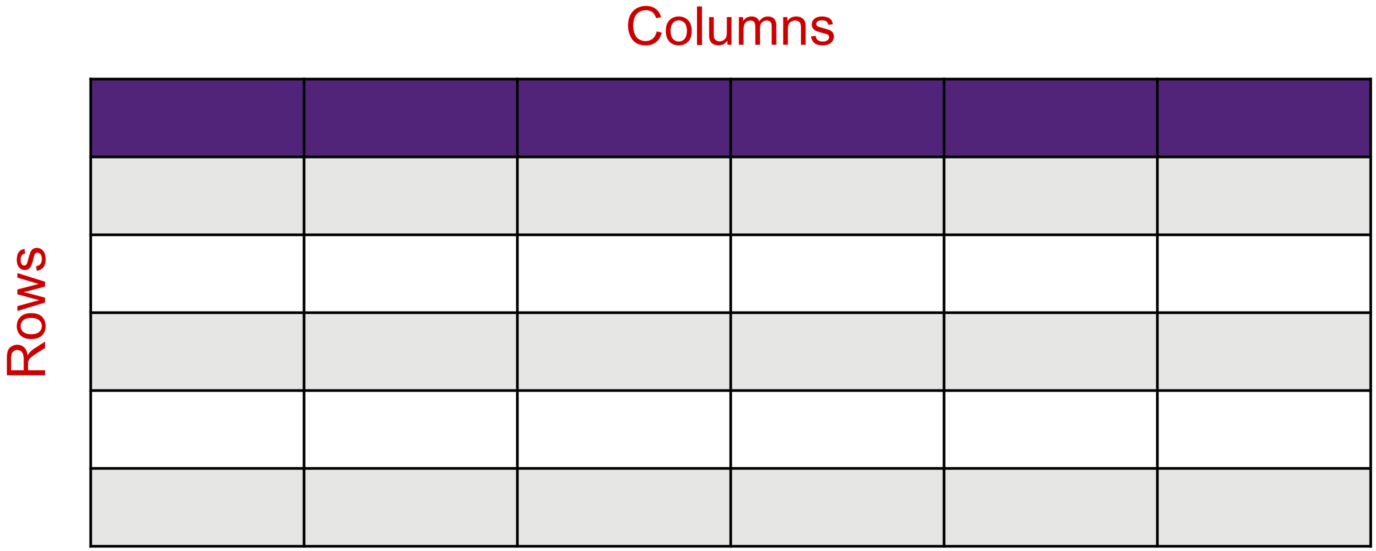
- Relations
- Attributes
- Domains
- Tuples

Relations

A Relation is the main construct for representing data in the Relational Model

Informally, a relation

- is a set of records
- is similar to a table with columns and rows



Columns

Rows

Relations, not Tables

The term table is used interchangeably with relation

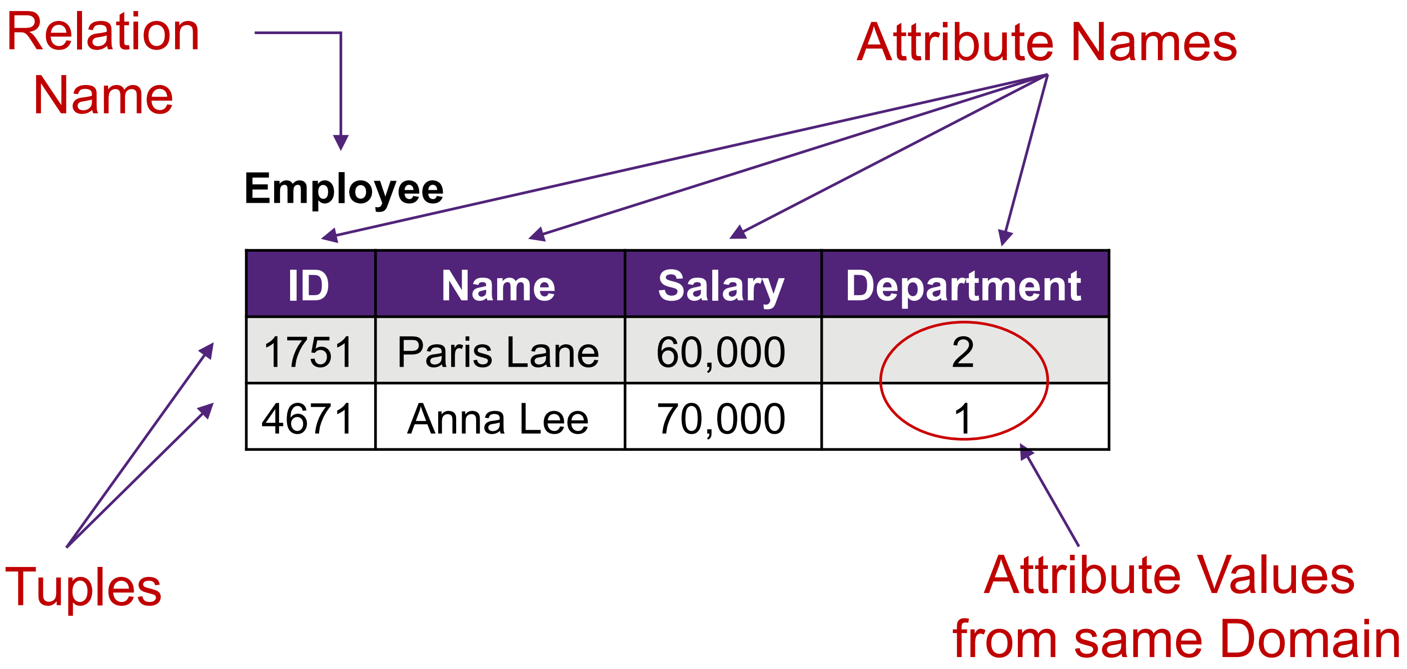
- Every relation is a table
- Not every table is a relation!

Relations have specific properties, based on the mathematical set theory

City: Brisbane		Product	Year: 1998			
Region	Suburb		Qtr 1	Qtr 2	Qtr 3	Qtr 4
South	Algester	Disks	32	243	23	246
South	Calamvale	Labels	4232	65	865	768
West	Taringa	Envelopes	3242	543	4554	454
North	McDowell	Toners	23	456	24	434
South	Sunnybank	Ribbons	324	65	56	657
West	Indooroopilly	Disks	234	6786	324	554

Not a Relation! ←

Relation Components



Relation Name

Attribute Names

Employee

ID	Name	Salary	Department
1751	Paris Lane	60,000	2
4671	Anna Lee	70,000	1

Tuples

Attribute Values from same Domain

Domain Types

A domain D is a set of atomic values

An atomic value is indivisible (as far as the relational data model is concerned)

Each domain has a data type or format

- Integers
- Numbers and currency
- Fixed or variable length character strings
- Date, timestamp
- Sub-range from a data type
  - e.g.,1 ≤ grade ≤ 7
- Enumerated data type
  - e.g. Gender in {‘Male’, ‘Female’, ‘Other’}

- Australian telephone numbers
  - Format: the digits “61” followed by 9 digits 0-9
- Car registration numbers
  - Format: 6 characters (either alpha or digits but no ‘Q’s allowed)

Attributes

Each attribute A is the name of a role played by some domain D in the relation named R

The number of attributes in a relation R is called the degree of R

Example: salary is an attribute name
(Each value of the attribute salary must belong to the domain of salary, which is integers)

Employee

ID	Name	Salary	Department
1751	Paris Lane	60,000	2
4671	Anna Lee	70,000	1

Domain/Attribute Restrictions

Same attribute name does not necessarily imply same domain

Department

ID	Name	Manager
1	Marketing	4671
2	Development	1751

Employee

ID	Name	Salary	Department
1751	Paris Lane	60,000	2
4671	Anna Lee	70,000	1

Domains for Department.ID and Employee.ID are different, even though the attribute names are the same

Domain/Attribute Restrictions

Different attribute name does not necessarily imply different domain

Employee

ID	Name	Salary	Department	ManagerID
1751	Paris Lane	60,000	2	NULL
4671	Anna Lee	70,000	1	NULL
2034	Jack Smith	40,000	1	4671
2670	Grace Mills	50,000	2	1751

Domains for ID and ManagerID are the same
but the attribute names are different

Tuples

Each tuple t is an ordered list of n values:

$$t = <v_1, v_2, \ldots, v_n>$$

where each value v_i $(1 \leq i \leq n)$ is an element of the corresponding domain of attribute A_i or a special value called “NULL”

Employee

ID	Name	Salary	Department	ManagerID
1751	Paris Lane	60,000	2	NULL
2670	Grace Mills	50,000	2	1751

t is called an n-tuple

- Example: (1751, Paris Lane, 60,000, 2, NULL) is a 5-tuple

Relation Schema and Instance

Relation Schema

- Denoted by R [A_1, A_2, A_3, …, A_n], includes a relation name R and list of attributes A_1, A_2, … A_n
- Integer n is termed “degree of the relation”
- A relation schema of degree 5
  - Employee [id, name, sex, salary, department]

Relation Instance

- A relation instance r of the relation schema R, denoted by r(R), is a set of n-tuples r = {t_1,t_2, …, t_m}.

Relation Schema and Instance Example

Relational Schema Example

Employee [id, name, salary, department]

Relation Instance example

Employee

ID	Name	Salary	Department
1751	Paris Lane	60,000	2
4671	Anna Lee	70,000	1
1023	Ben Cho	70,000	4
2034	Jack Smith	40,000	1
2670	Grace Mills	50,000	2

Question: Schema and Instance

The schema and instance of the database represent two distinct concepts. Associate each with the relevant characteristics in the table below.

Characteristics	Circle Schema OR Instance here	
Data in the database	Schema	Instance
Specified during database design	Schema	Instance
Data describing the data	Schema	Instance
Created through data update operations	Schema	Instance

Ordering of Tuples

Relations are *sets* of tuples

Mathematically, elements of a set have no implied order

Semantically, when reasoning with relations, e.g. when formulating queries, order is irrelevant

Physically, tuples reside on blocks of secondary storage, which have a partial ordering, hence tuples have an ordering

Department

ID	Name	Manager
1	Marketing	4671
2	Development	1751
4	Human Resources	1023

Department

ID	Name	Manager
1	Marketing	4671
4	Human Resources	1023
2	Development	1751

Same Relation

Ordering of Values within a Tuple

n-tuple is an *ordered* list of n values

Syntactically, all tuples in a relation have values in the same order

Semantically, the order chosen is irrelevant, as long as the correspondence between the attributes and the values can be maintained

Department

ID	Name	Manager
1	Marketing	4671
2	Development	1751
4	Human Resources	1023

Department

Name	ID	Manager
Marketing	1	4671
Development	2	1751
Human Resources	4	1023

Same Relation

Relational Model Concepts

Integrity Constraints

ER to Relational Mapping

Database Integrity Constraints

Integrity constraints are rules that enforce the 'integrity', or the correctness, of our database.

- They must hold on every instance of the schema.

Domain Constraint

Key Constraint

Entity Integrity Constraint

Referential Integrity Constraint

Semantic Integrity Constraint

Domain Constraint

A **domain** is a set of atomic values. Each attribute in a relation will belong to some domain.

A **domain constraint** violation occurs when an attribute's value does not appear in the corresponding domain.

Assuming the domain of Employee.id is a 4-digit integer then:

Employee

Domain constraint violation →

ID	Name	Salary	Department	
1751	Paris Lane	60,000	2	
LOL	**Anna Lee**	**70,000**	**4**	✗
4671	**Anna Lee**	**70,000**	**4**	✓

What is a **key**?

A **key** is a minimal set of attributes that uniquely identify tuples in a relation. The term *minimal* does not mean the smallest set of attributes but instead a set of attributes without any redundant attributes.

A schema may have more than one key

- Each is called a **candidate key**
- A **primary key** is candidate key chosen as main key for relation, which would be underlined.

Employee [id, name, salary, department]

Question: Key

Assuming that department IDs are unique, which of the following is a key for the Department relation?

A. (ID)
B. (ID, Name)
C. (ID, Manager)
D. All of the above

Department

ID	Name	Manager
1	Marketing	4671
2	Development	1751
4	Human Resources	1023

Question: Key

Assuming that department IDs are unique and that the combination of Name and Manager are also unique for each department, which of the following is a key(s) for the Department relation?

A. (ID)
B. (ID, Name)
C. (Name, Manager)
D. Both options A and C

Department

ID	Name	Manager
1	Marketing	4671
2	Development	1751
4	Human Resources	1023

Key Constraints

A **key constraint** violation occurs when a tuple is inserted or modified such that it has the same key value as another tuple.

Assuming ID is a key in the employee relation then:

Employee

ID	Name	Salary	Department	
1751	Paris Lane	60,000	2	
4671	Anna Lee	70,000	1	
4671	**Ben Cho**	**70,000**	**4**	✗
1023	**Ben Cho**	**70,000**	**4**	✓

Entity Integrity Constraints

An **entity integrity constraint** violation occurs when a tuple is inserted or modified such that part of its primary key contains the value NULL.

For primary keys that consists of multiple attributes, *no part* of the primary key can be null.

Assuming ID is a key in the employee relation then:

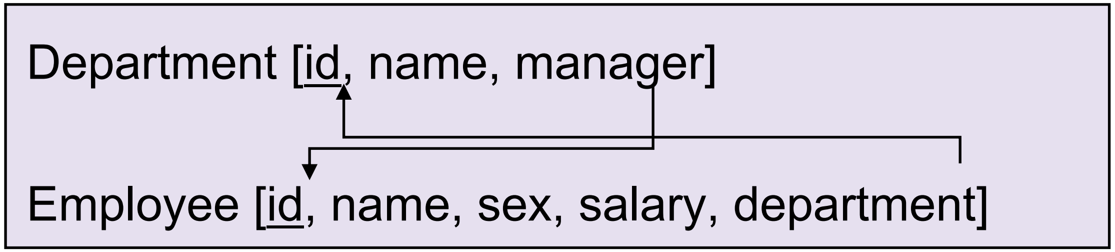
Employee

ID	Name	Salary	Department
1751	Paris Lane	60,000	2
4671	Anna Lee	70,000	1
1023	Ben Cho	70,000	4

NULL	**Jack Smith**	**40,000**	**1**	✗
2034	**Jack Smith**	**40,000**	**1**	✓

Foreign Keys

To preserve relationships, you may need to create a **foreign key (FK).** A foreign key is a primary key from one table placed into another table. This can be viewed graphically or textually.



Department [id, name, manager]

Employee [id, name, sex, salary, department]

Department.manager references Employee.id
Employee.department references Department.id

The key is called a foreign key in the table that received the key.

- e.g, the attribute manager is a FK.

Foreign Keys

- Let FK be a set of attributes in R1 and let PK be the primary attributes in R2
- FK in R1 is a **foreign key** referencing PK in R2 if
  - FK and PK have the same domain, and
  - For any tuple t_1 in R1, either t_1[FK] is null; or there exists a tuple t_2 in R2, such that $t_1[FK] = t_2[PK]$

Department

ID	Name	Manager
1	Marketing	4671
2	Development	1751
4	Human Resources	NULL

Department.manager references Employee.id

Employee

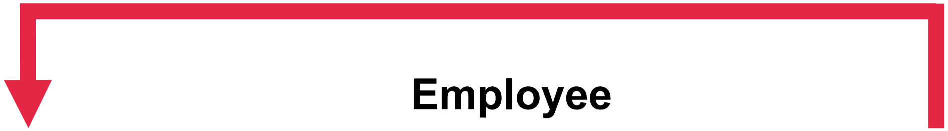
ID	Name	Salary	Department
1751	Paris Lane	60,000	2
4671	Anna Lee	70,000	1
1023	Ben Cho	70,000	4
2034	Jack Smith	40,000	NULL

Employee.department references Department.id

Self-Referencing Relations

It is also possible for a table to reference itself.

- In the given example, ManagerID references ID



Employee

ID	Name	Salary	Department	ManagerID
1751	Paris Lane	60,000	2	NULL
4671	Anna Lee	70,000	1	NULL
1023	Ben Cho	70,000	4	NULL
2034	Jack Smith	40,000	1	4671
2670	Grace Mills	50,000	2	1751

Employee.managerID references Employee.id

Relations with Composite keys

It is also possible to have FKs to relations that have a multi-attribute primary key.

Student [sid, name]

Course [cid, department, manager]

Enrollment [sid, cid, department, grade]

Enrollment.sid references Student.sid
Enrollment.{cid, department} references Course.{cid, department}

Referential Integrity Constraints

A **referential integrity constraint** can be utilised to guarantee that a department with department number 2 exists before the “Grace Mills” tuple is stored.

Employee

ID	Name	Salary	Department
1751	Paris Lane	60,000	2
4671	Anna Lee	70,000	1
1023	Ben Cho	70,000	4
2034	Jack Smith	40,000	1

2670	**Grace Mills**	**50,000**	**5**	✗
2670	**Grace Mills**	**50,000**	**2**	✓

Employee.department references Department.id

Department

Number	Name	Manager
1	Marketing	4671
2	Development	1751
4	Human Resources	1023

Semantic Integrity Constraint

Semantic integrity constraints are generally defined by the business or organization during client consultation.

Semantic constraints can be used to enforce organisation policies such as:

- “The salary of an employee should not exceed the employee’s supervisor’s salary”
- “The maximum number of hours that an employee can work on a project is 56”

Often implemented in a constraint specification language (SQL) using triggers and assertions.

Semantic Integrity Constraint

Assuming the salary of an employee should not exceed their supervisor's salary.

Employee

ID	Name	Salary	Department
1751	Paris Lane	60,000	2
4671	Anna Lee	70,000	1
1023	Ben Cho	70,000	4
2034	Jack Smith	40,000	1
2670	Grace Mills	50,000	2

2967	**Arron Dills**	**100,000**	**1**	✗
2967	**Arron Dills**	**40,000**	**1**	✓

Employee.department references Department.id

Department

Number	Name	Manager
1	Marketing	4671
2	Development	1751
4	Human Resources	1023

Constraints and Operations

Enforcement of integrity constraints ensures that the database remains consistent.

Changes to the database such as **insert**, **modification** and **deletion** must not violate integrity constraints (leave the database in an inconsistent state).

If a database update is submitted to the DBMS that would violate integrity, it must be rejected.

Constraints on Insertion & Modification

Insertion or modifications can violate five types of constraints.

Employee

ID	Name	Salary	Department
1751	Paris Lane	60,000	2
4671	Anna Lee	70,000	1
1023	Ben Cho	70,000	4
2034	Jack Smith	40,000	1
2670	Grace Mills	50,000	2

Department

ID	Name	Manager
1	Marketing	4671
2	Development	1751
4	Human Resources	1023

LOL	**John Doe**	**70,000**	**4**	✗	Domain constraints
4671	**John Doe**	**70,000**	**4**	✗	Key constraints
NULL	**John Doe**	**70,000**	**4**	✗	Entity Integrity constraints
1111	**John Doe**	**70,000**	**6**	✗	Referential Integrity constraints
1111	**John Doe**	**99,999**	**4**	✗	Semantic Integrity constraints

Constraints on Deletion

Deletion operations can lead to **referential** or **semantic** integrity constraints.

Referential integrity constraint violations can be either rejected, cascaded or the referencing attribute values can be modified to a default value.

Employee

ID	Name	Salary	Department
1751	Paris Lane	60,000	2
4671	Anna Lee	70,000	1
1023	Ben Cho	70,000	4
2034	Jack Smith	40,000	1
2670	Grace Mills	50,000	2

Department

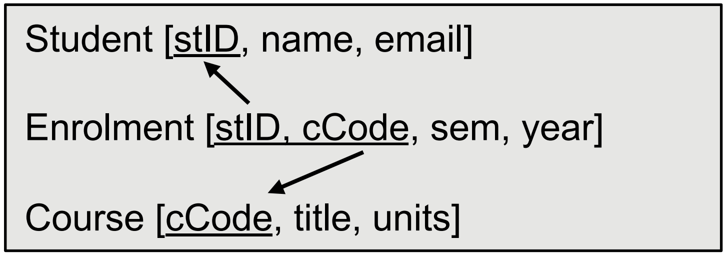
ID	Name	Manager
1	Marketing	4671
2	Development	1751
4	Human Resources	1023

Removing department with ID = 2 ✗ Referential Integrity constraints

Removing employee with ID =1023 assuming every department needs at least one employee ✗ Semantic Integrity constraints

In-class Exercise

Use this relational schema on the right to give examples of the following:



Student [stID, name, email]

Enrolment [stID, cCode, sem, year]

Course [cCode, title, units]

1. Key
2. Foreign Key
3. Domain Constraint

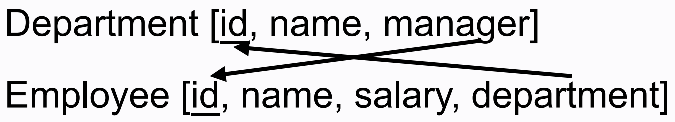
Question: Integrity Constraints

Imagine you are opening an online bank account. You are asked to enter a password, so you type in your usual password: "password123". However, a message "your password must contain at least one capital letter and a number" appears on your screen. What type of constraint in the bank's database is limiting you from using your usual password: "password123".

A. Domain constraint
B. Key constraint
C. Entity constraint
D. Referential integrity constraint
E. None of the above

Question: Integrity Constraints

Consider the following schema and instance of a database. Which constraint would be violated if (2670, James Smith, 40000, 1) was added to Employee?



Department [id, name, manager]

Employee [id, name, salary, department]

Department

ID	Name	Manager
1	Marketing	4671
2	Development	1751

Employee

ID	Name	Salary	Department
1751	Paris Lane	60,000	2
4671	Anna Lee	70,000	1
2670	Grace Mills	50,000	2
2034	Jack Smith	40,000	1

A. Domain constraint

B. Key constraint

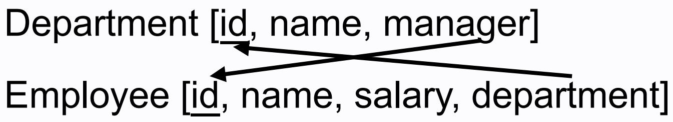
C. Entity constraint

D. Referential integrity constraint

E. None of the above

Question: Integrity Constraints

Consider the following schema and instance of a database. Which constraint would be violated if (2644, James, Smith, 1) was added to Employee?



Department [id, name, manager]

Employee [id, name, salary, department]

Department

ID	Name	Manager
1	Marketing	4671
2	Development	1751

Employee

ID	Name	Salary	Department
1751	Paris Lane	60,000	2
4671	Anna Lee	70,000	1
2670	Grace Mills	50,000	2
2034	Jack Smith	40,000	1

A. Domain constraint

B. Key constraint

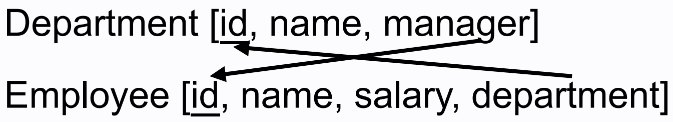
C. Entity constraint

D. Referential integrity constraint

E. None of the above

Question: Integrity Constraints

Consider the following schema and instance of a database. Which constraint would be violated if (2644, James Smith, 40000, 3) was added to Employee?



Department [id, name, manager]

Employee [id, name, salary, department]

Department

ID	Name	Manager
1	Marketing	4671
2	Development	1751

Employee

ID	Name	Salary	Department
1751	Paris Lane	60,000	2
4671	Anna Lee	70,000	1
2670	Grace Mills	50,000	2
2034	Jack Smith	40,000	1

A. Domain constraint

B. Key constraint

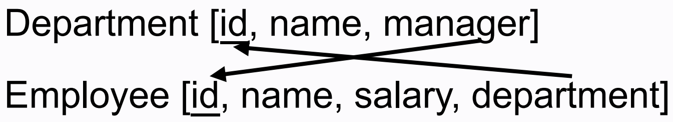
C. Entity constraint

D. Referential integrity constraint

E. None of the above

Question: Integrity Constraints

Consider the following schema and instance of a database. Which constraint would be violated if (2, Development, 1751) was deleted from Department?



Department [id, name, manager]

Employee [id, name, salary, department]

Department

ID	Name	Manager
1	Marketing	4671
2	Development	1751

Employee

ID	Name	Salary	Department
1751	Paris Lane	60,000	2
4671	Anna Lee	70,000	1
2670	Grace Mills	50,000	2
2034	Jack Smith	40,000	1

A. Domain constraint

B. Key constraint

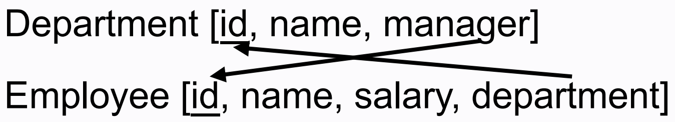
C. Entity constraint

D. Referential integrity constraint

E. None of the above

Question: Integrity Constraints

Consider the following schema and instance of a database. Which constraint would be violated if (2, Development, 1751) was updated to (2, Development, 2034) in Department?



Department [id, name, manager]

Employee [id, name, salary, department]

Department

ID	Name	Manager
1	Marketing	4671
2	Development	1751

Employee

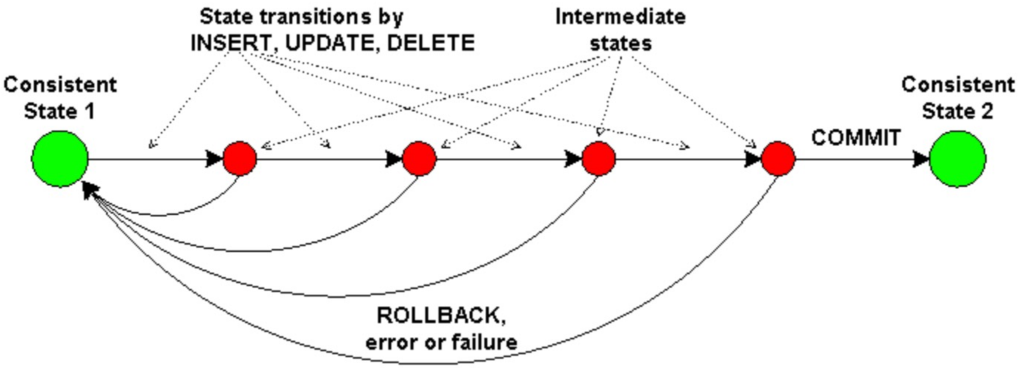
ID	Name	Salary	Department
1751	Paris Lane	60,000	2
4671	Anna Lee	70,000	1
2670	Grace Mills	50,000	2
2034	Jack Smith	40,000	1

A. Domain constraint
B. Key constraint
C. Entity constraint
D. Referential integrity constraint
E. None of the above

The Transaction Concept

A **transaction** is an executing program that includes some database operations, such as reading from the database, or applying insertions, deletions, or updates to the database.

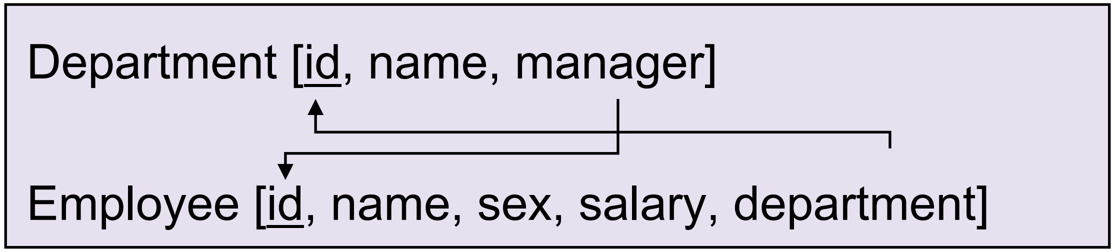
At the end of the transaction, it must leave the database in a valid or consistent state that satisfies all the constraints specified on the database schema.



Source: https://maxdb.sap.com/doc/7_7/81/74b30edc2142658e510080ef6917f1/ppt_img.gif

Transactions allow execution of a suite of queries where constraint violations in intermediate steps are allowed.

The Transaction Concept - Example



Department [id, name, manager]

Employee [id, name, sex, salary, department]

Constraint 1: Every department should have at least one employee.

Constraint 2: Every employee must work for a department.

Problem: Cannot create a new department since it has no employees.

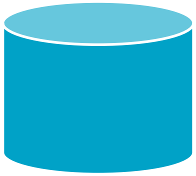
Solution: Use a transaction to insert information about a new department and its employee at the same time.

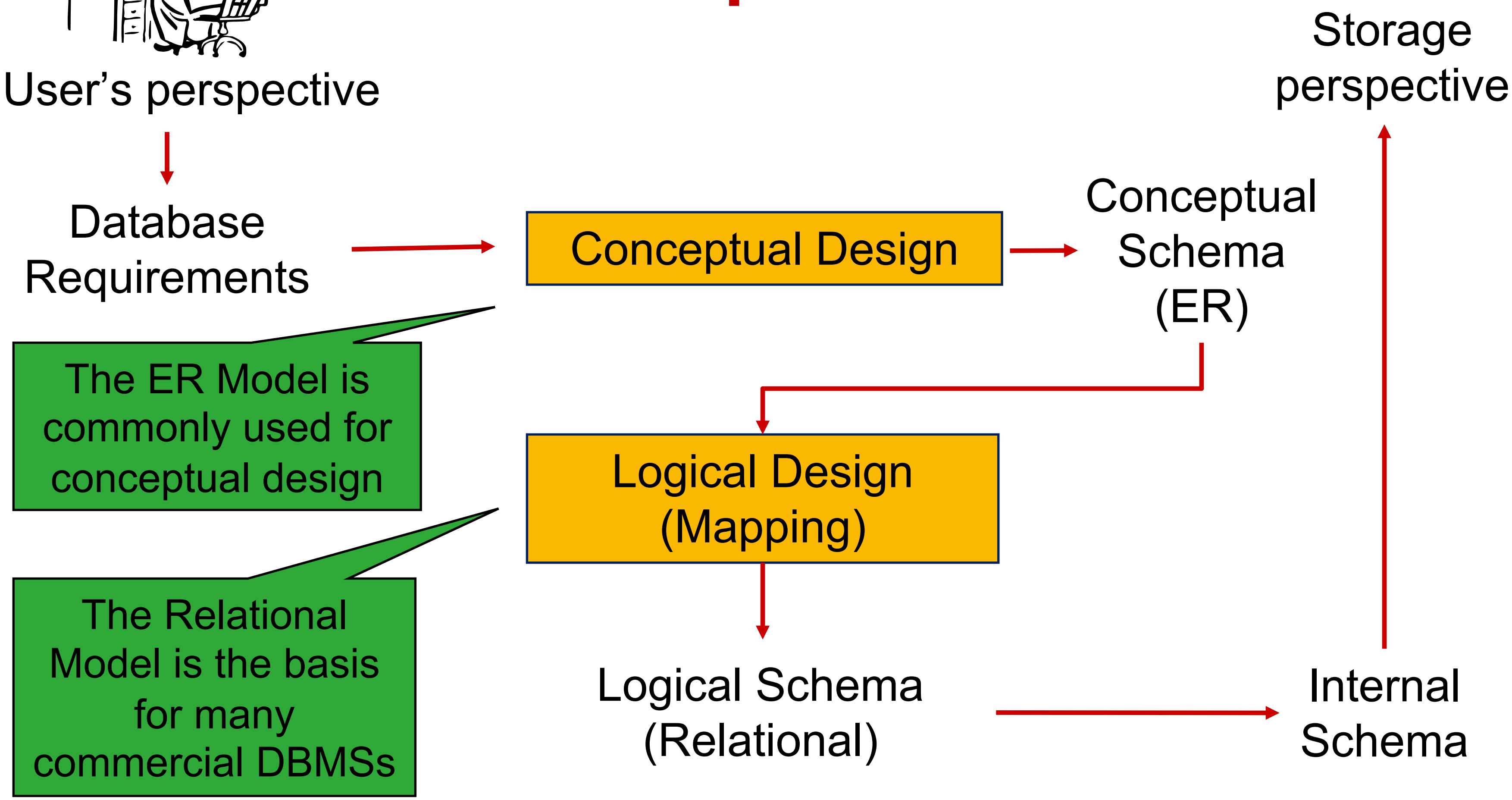
Relational Model Concepts

Integrity Constraints

ER to Relational Mapping

Conceptual Perspective





User's perspective

Database Requirements

Conceptual Design

Conceptual Schema (ER)

Logical Design (Mapping)

Logical Schema (Relational)

Internal Schema

Storage perspective

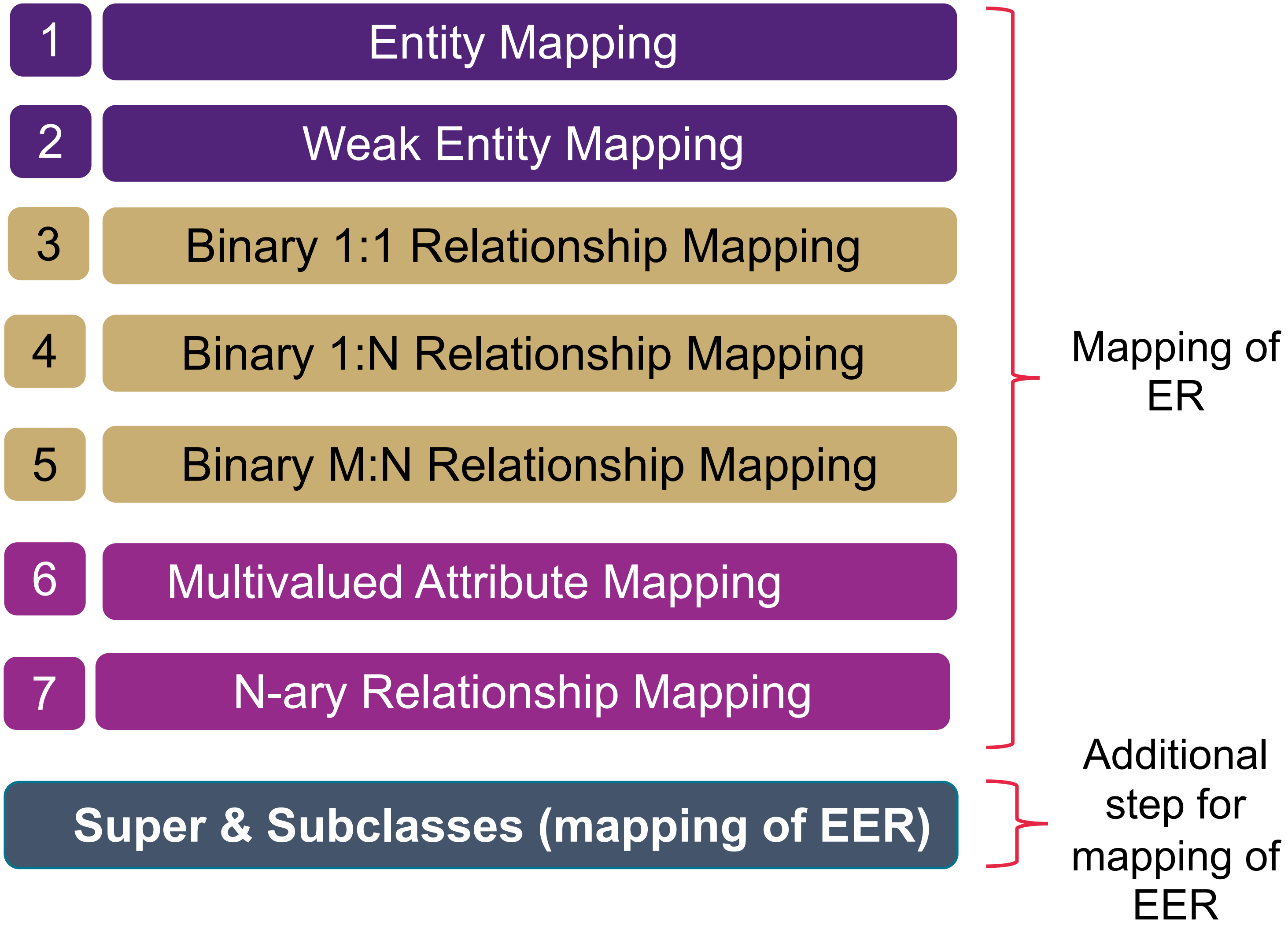
The ER Model is commonly used for conceptual design

The Relational Model is the basis for many commercial DBMSs

(7+1) – Steps for Mapping

Input:
An ER model

Output:
Relations with primary/foreign key constraints



1 Entity Mapping

2 Weak Entity Mapping

3 Binary 1:1 Relationship Mapping

4 Binary 1:N Relationship Mapping

5 Binary M:N Relationship Mapping

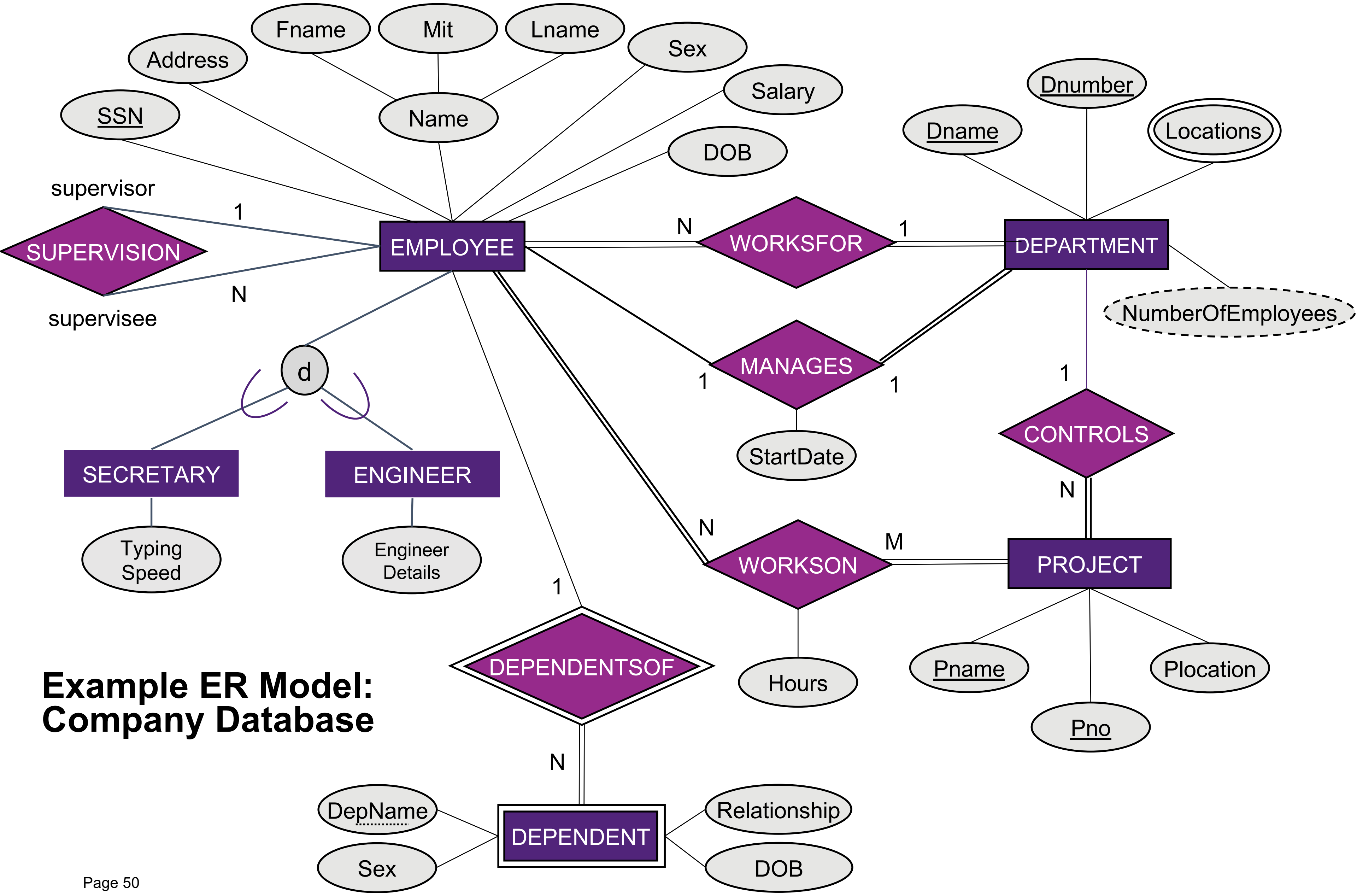
6 Multivalued Attribute Mapping

7 N-ary Relationship Mapping

Super & Subclasses (mapping of EER)

Mapping of ER

Additional step for mapping of EER

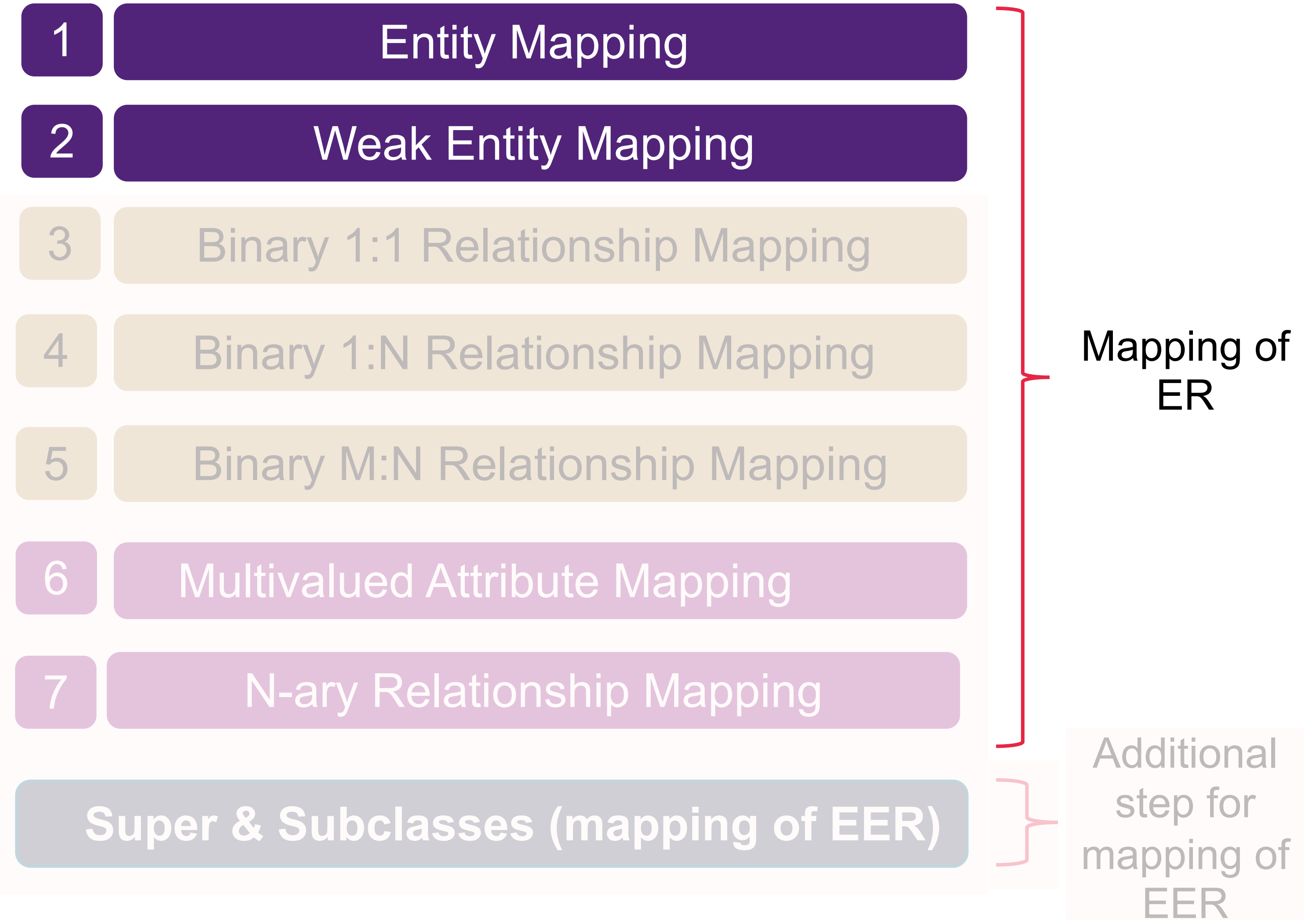


Example ER Model: Company Database

(7+1) – Steps for Mapping

Input:
An ER model

Output:
Relations with primary/foreign key constraints



1 Entity Mapping

2 Weak Entity Mapping

3 Binary 1:1 Relationship Mapping

4 Binary 1:N Relationship Mapping

5 Binary M:N Relationship Mapping

6 Multivalued Attribute Mapping

7 N-ary Relationship Mapping

Super & Subclasses (mapping of EER)

Mapping of ER

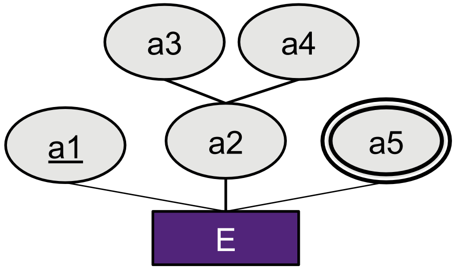
Additional step for mapping of EER

Step 1: Regular Entity

For each entity type

1. Create a relation
2. Choose a key as the primary key
3. Include all simple attributes (e.g., a3 and a4) which are not composite (e.g.,a2), derived or multivalued (e.g., a5)

ER Diagram

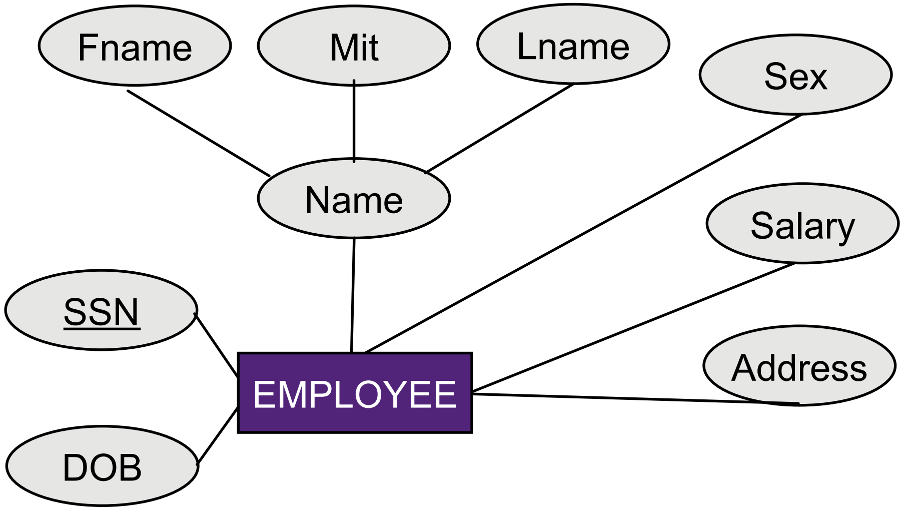


Mapping to Relation

E [a1, a3, a4]

Step 1: Company Example

ER Diagram

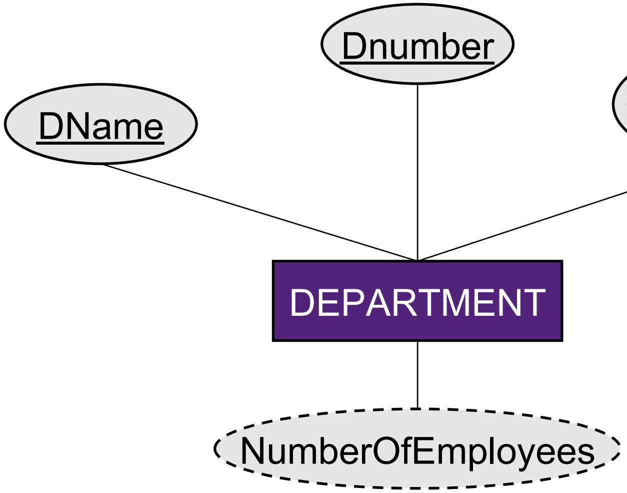


Mapping to Relation

Employee [ssn, fName, mIt, lName, dob, address, sex, salary]

Name will not be added to the relation.

Step 1: Company Example

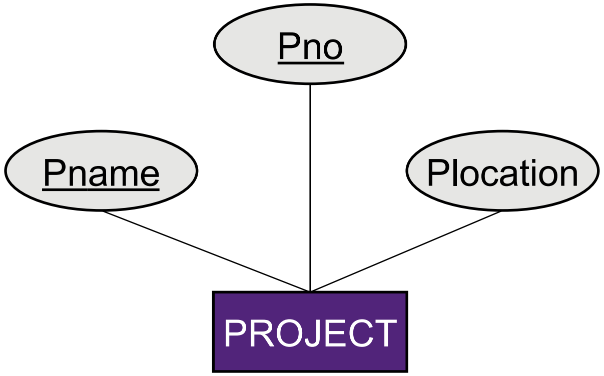


- Location will not be added for the time being
- Both Name and Number are keys. Number is taken as PK.
- NumberOfEmployees is not added to the relation as it is a derived attribute.

Mapping to Relation

Department [<u>dNumber</u>, dName]

Step 1: Company Example



- Both Name and Number are keys. Number is taken as PK.

Mapping to Relation

Project [pNo, pName, pLocation]

Schema (in progress)

Relations:

Employee [ssn, fName, mIt, lName, dob, address, sex, salary]

Department [dNumber, dName]

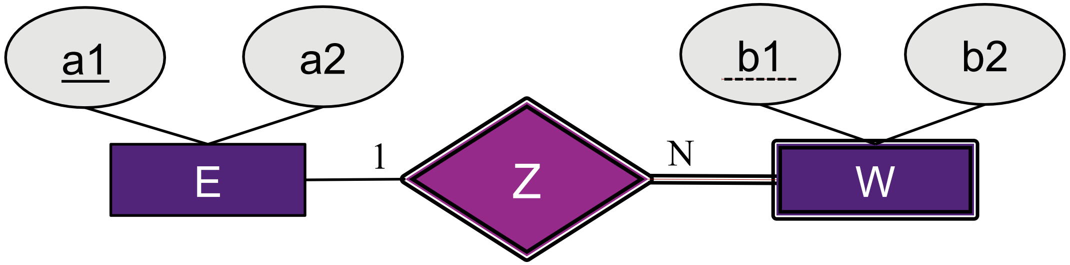
Project [pNo, pName, pLocation]

Step 2: Weak Entity

For each weak entity type

1. Create a relation
2. Set its primary key as the combination of the primary keys of owner entities and its own partial key.
3. Include a foreign key from the relation to the primary key of the relation of its owner entity types.
4. Include all simple attributes which are not composite, derived or multivalued.

ER Diagram



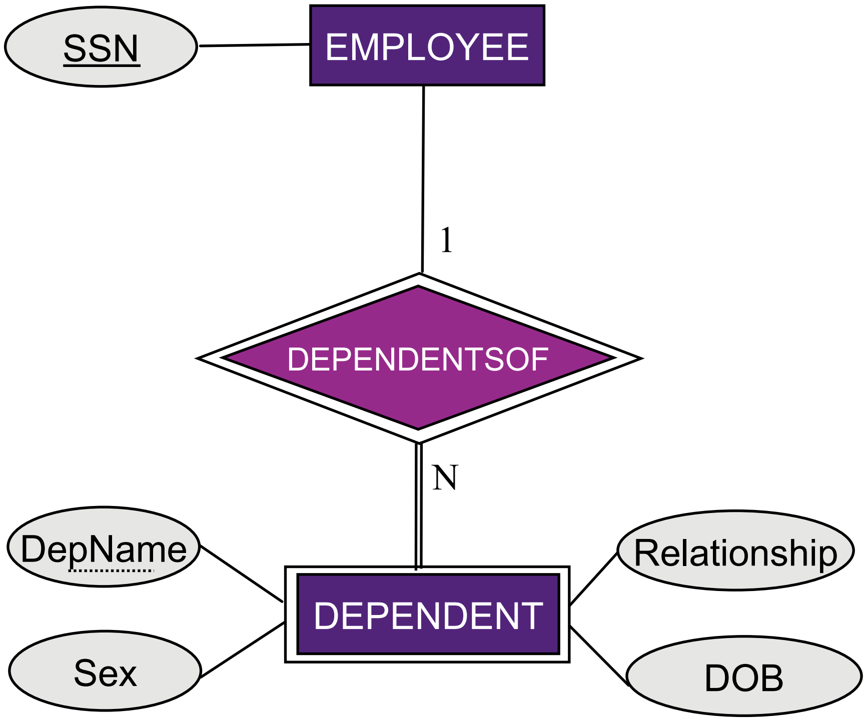
Mapping to Relation

E [a1, a2]

W [a1, b1, b2]

W.a1 references E.a1

Step 2: Company Example



- SSN, as the primary key of the EMPLOYEE relation, is added to the PK
- DepName is the partial key of DEPENDENT
- Dependent.ssn is a FK referring to Employee.ssn.

Mapping to Relation

Dependent [ssn, depName, sex, dob, relationship]
Dependent.snn references Employee.ssn

Schema (in progress)

Relations:

Employee [ssn, fName, mIt, lName, dob, address, sex, salary]

Department [dNumber, dName]

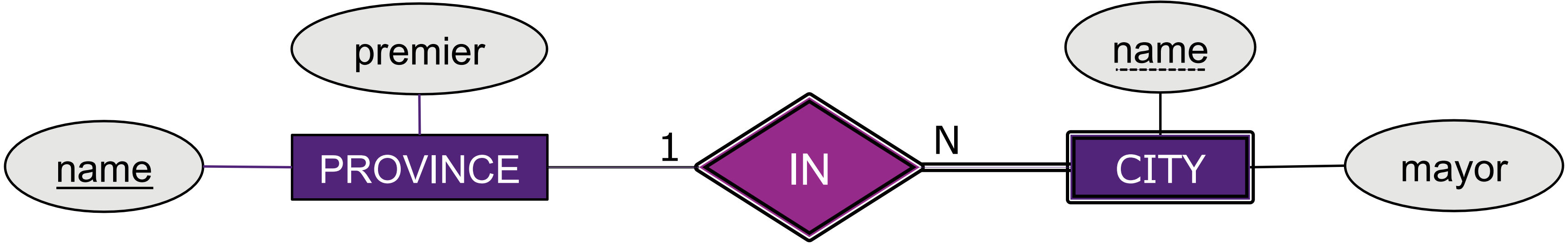
Project [pNo, pName, pLocation]

Dependent [ssn, depName, sex, dob, relationship]

Foreign Key:

Dependent.ssn references Employee.ssn

Question: Weak Entities



Convert this ER diagram to relations, resolving the dual use of "name" in some reasonable way. Which schema below is the most reasonable translation from ER to relations?

A. Cities [name, mayor], Provinces [name, premier]

B. Cities [cName, pName, mayor], Provinces [pName, premier]

- Cities.cName references Provinces.pName
- Cities.pName references Provinces.pName

C. Cities [cName, pName, mayor], Provinces [pName, premier]

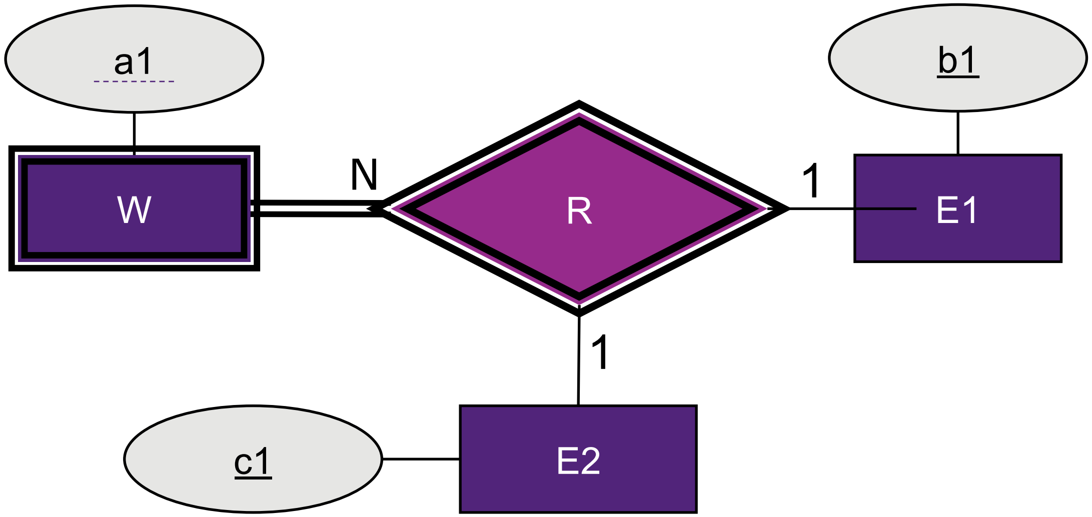
- Cities.pName references Provinces.pName

D. Cities [cName, pName, mayor], In [cName, pName], Provinces [name, premier]

- Cities.pName references Provinces.name

E. None of the above

Weak Entity with Multiple Owner Entities



Mapping to Relation

W [a1, b1, c1]

- W.b1 references E1.b1
- W.c1 references E2.c1

E1 [b1]

E2 [c1]

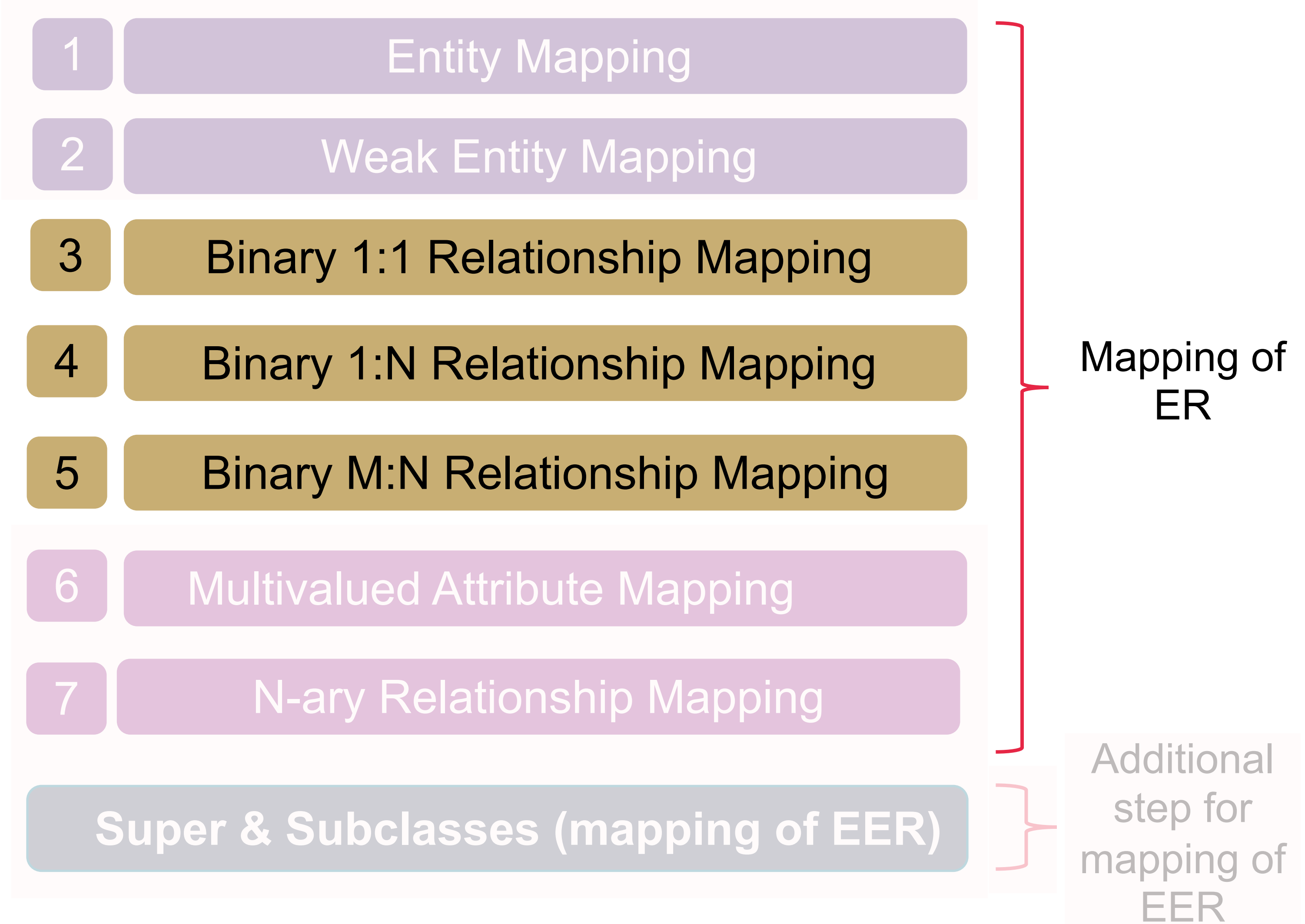
For each weak entity type with more than one owner entity type:

- Create a relation W that includes all simple attributes of that weak entity.
- Include as foreign key attributes in W the primary key attributes of owner entity types.
- The primary key of W is the combination of the primary key of owner entities and the partial key of the weak entity.

(7+1) – Steps for Mapping

Input:
An ER model

Output:
Relations with primary/foreign key constraints



1 Entity Mapping

2 Weak Entity Mapping

3 Binary 1:1 Relationship Mapping

4 Binary 1:N Relationship Mapping

5 Binary M:N Relationship Mapping

6 Multivalued Attribute Mapping

7 N-ary Relationship Mapping

Super & Subclasses (mapping of EER)

Mapping of ER

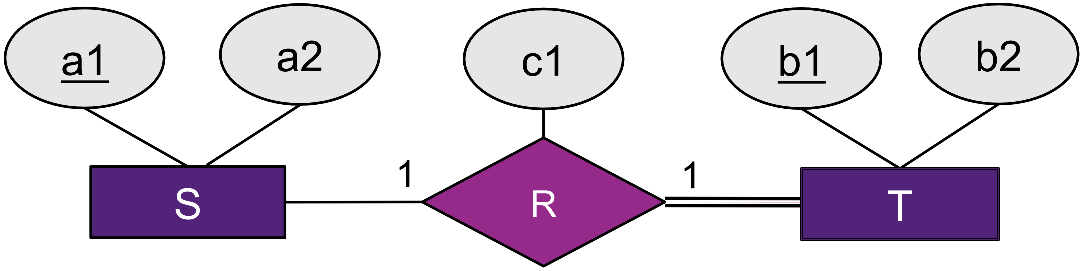
Additional step for mapping of EER

Step 3: Binary 1:1 Relationship

For each binary 1:1 relationship type

1. Choose one of the participating entity types (preference is given to entity types with total participation)
2. Include a foreign key from the chosen entity type back to the primary keys of the relation of the other entity type.
3. Include all the simple attributes of the relationship type to the relation of the chosen entity type.

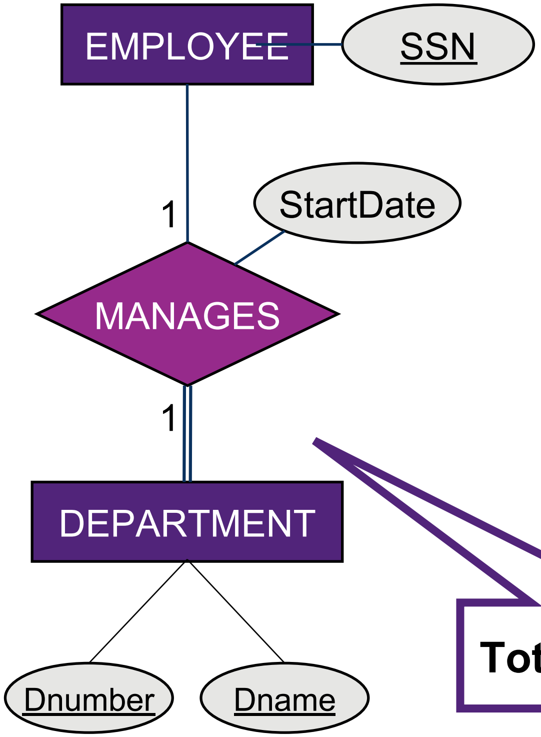
ER Diagram



Mapping to Relation

S [a1, a2]
T [b1, b2, a1, c1]
T.a1 references S.a1

Step 3: Company Example



- Given that department must have a manager, extending department is the better choice.
- Include the primary key of Employee as a foreign key in Department (renamed to mgrSSN)
- Include the simple attribute startDate of Manages (renamed to mgrStartDate)

Total Participation

Mapping to Relation

Department [dNumber, dName, mgrSSN, startDate]
Department.mgrSSN references Employee.ssn

Schema (in progress)

Relations:

Employee [ssn, fName, mIt, lName, dob, address, sex, salary]

Department [dNumber, dName, mgrSSN, startDate]

Project [pNo, pName, pLocation]

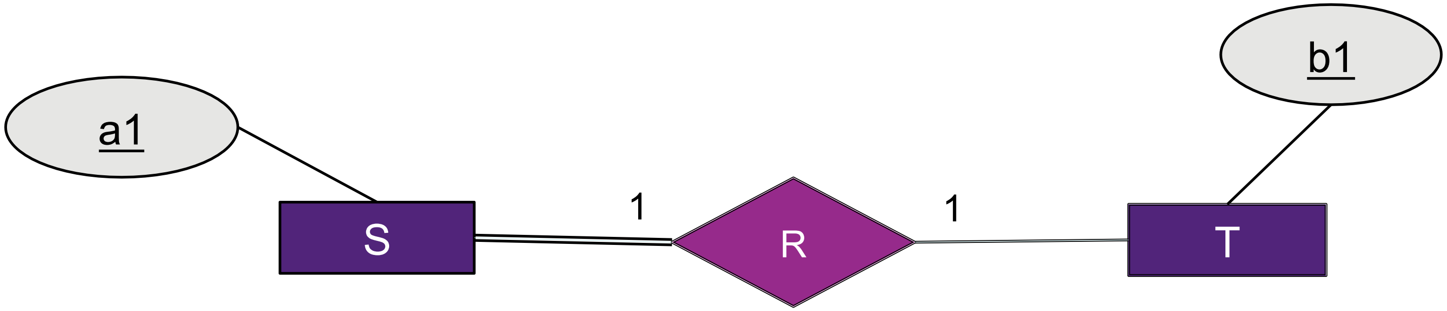
Dependent [ssn, depName, sex, dob, relationship]

Foreign Keys:

*Department.*mgrSSN *references Employee.Ssn*

Dependent.ssn references Employee.Ssn

Question: Binary Relationship



Which schema below is a reasonable translation from ER to relations?

A. S [a1, b1], T [b1] , S.b1 references T.b1

B. S [a1], T [b1]

C. ST [a1, b1]

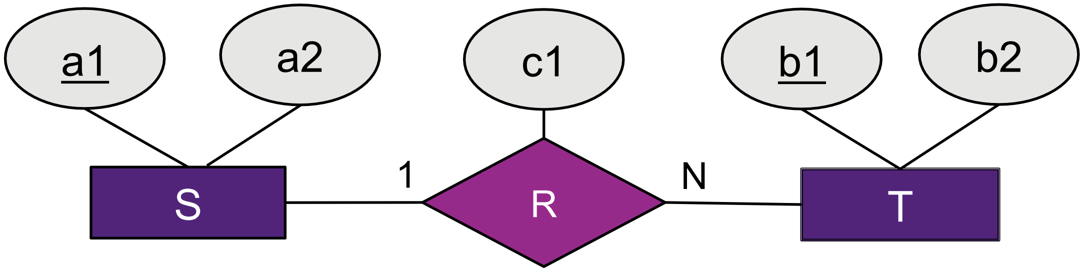
D. S [a1], T [b1, a1] , T.a1 references S.a1

Step 4: Binary 1:N Relationship

For each (non-weak) binary 1:N relationship type

1. Identify the participating entity type at the N-side of the relationship type
2. Include a foreign key to the entity type on the N-side. This foreign key should be the primary key of the entity type on the 1-side.
3. Include any simple attributes of the relationship type as attributes of the relation of the N side.

ER Diagram



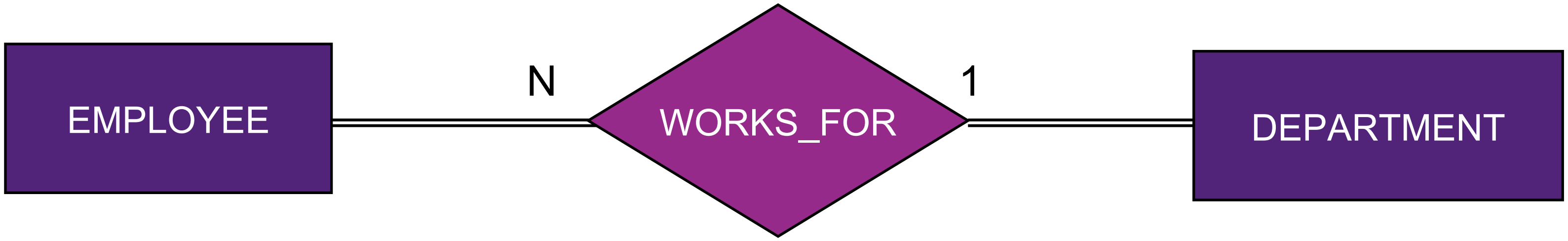
Mapping to Relation

S [a1, a2]
T [b1, b2, a1, c1]
T.a1 references S.a1

Step 4: Company Example

Binary 1:N relationships in the Company Database: WORKS_FOR, CONTROLS and SUPERVISION.

The primary key of DEPARTMENT is included as a foreign key in the EMPLOYEE relation (renamed dNumber)



Mapping to Relation

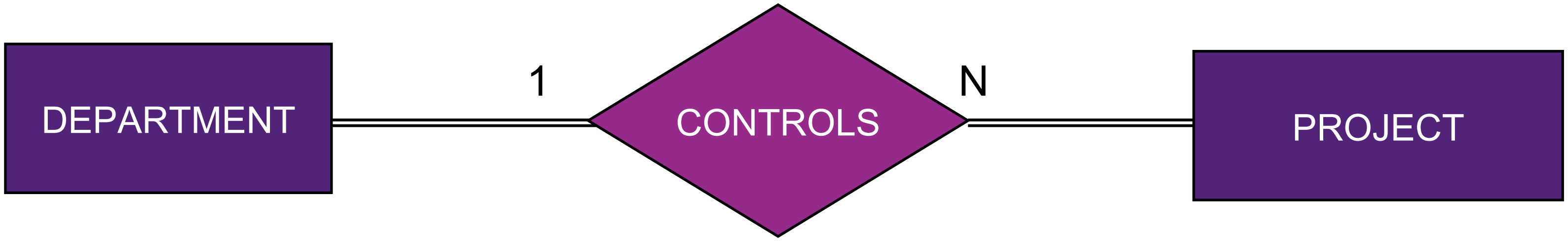
Employee [ssn, fName, mIt, lName, dob, address, sex, salary, dNumber]

Employee.dNumber references Department.dNumber

Step 4: Company Example

Binary 1:N relationships in the Company Database: WORKS_FOR, CONTROLS and SUPERVISION.

The primary key of the DEPARTMENT relation is included as a foreign key in the PROJECT relation.

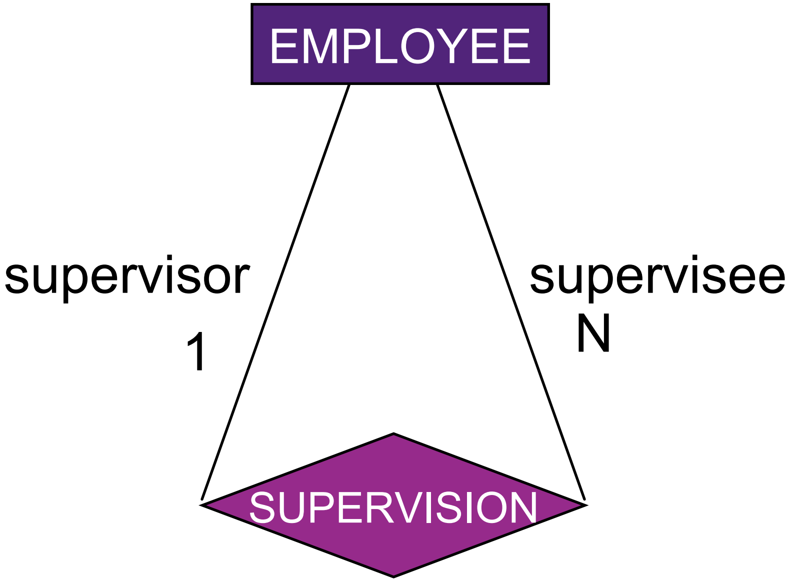


Mapping to Relation

Project [pNo, pName, pLocation, dNumber]

Project.dNumber references Department.dNumber

Step 4: Company Example



Binary 1:N relationships in the Company Database: WORKS_FOR, CONTROLS and SUPERVISION.

The primary key of the EMPLOYEE relation is included as a foreign key within the EMPLOYEE relation (called superSsn).

Mapping to Relation

Employee [<u>ssn</u>, fName, mIt, lName, dob, address, sex, salary, dNumber, superSSN]
Employee.superSSN references Employee.ssn

Schema (in Progress)

Relations:

Employee [ssn, fName, mIt, lName, dob, address, sex, salary, dNumber, superSSN]

Department [dNumber, dName, mgrSSN, startDate]

Project [pNo, pName, pLocation, dNumber]

Dependent [ssn, depName, sex, dob, relationship]

Foreign Keys:

Employee.dNumber references Department.dNumber

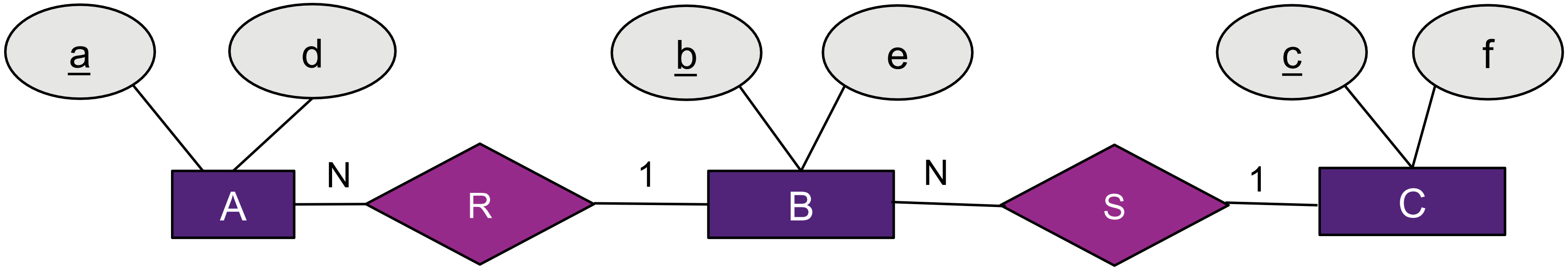
Employee.superSSN references Employee.ssn

Department.mgrSSN references Employee.ssn

Project.dNumber references Department.dNumber

Dependent.ssn references Employee.ssn

Question: Relationship Mapping



Translate the ER diagram to relational schema. Which of the following appears in your relational schema?

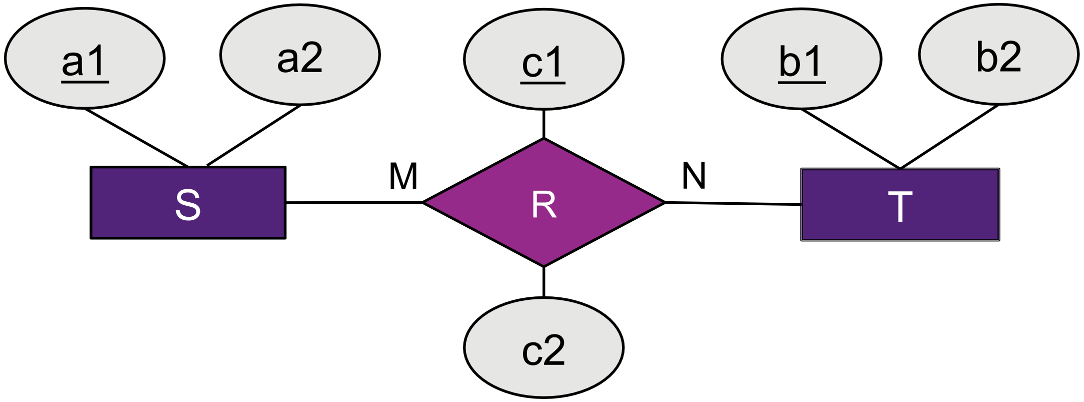
A. A [a,b,d], A.b references B.b
B. B [b,c,e], B.c references C.c
C. S [b,c]
D. All of the above
E. None of the above

Step 5: Binary M:N Relationship

For each binary M:N relationship type

1. Create a new relation
2. Include **foreign keys** from relation to the primary key of the participating entity types
3. The combination of foreign keys will form the **primary key** of R
4. R can have its own attributes that contribute to the primary key
5. Include any simple attributes of R as attributes of the new relation

ER Diagram



Mapping to Relation

S [a1, a2]

T [b1, b2]

R [a1, b1, c1, c2]

R.a1 references S.a1

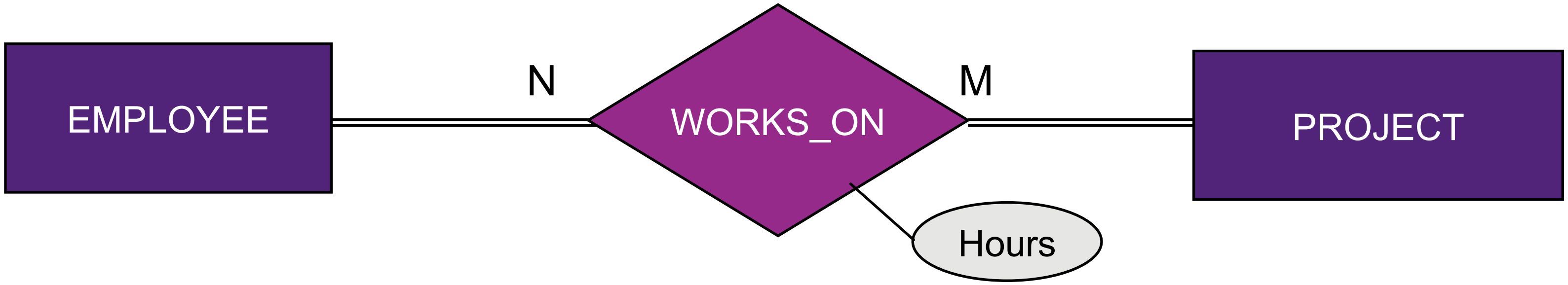
R.b1 references T.b1

Step 5: Company Example

Binary M:N relationships in the Company Database: WORKS_ON

The WORKS_ON includes the primary keys of PROJECT and EMPLOYEE as foreign keys.

Hours in WORKS_ON represents the attribute of the relationship type.



Mapping to Relation

WorksOn [<u>ssn, pNo</u>, hours]

WorksOn.snn references Employee.ssn

WorksOn.pNo references Project.pNo

Schema (in progress)

Relations:

Employee [ssn, fName, mIt, lName, dob, address, sex, salary, dNumber, superSSN]

Department [dNumber, dName, mgrSSN, startDate]

Project [pNo, pName, pLocation, dNumber]

Dependent [ssn, depName, sex, dob, relationship]

WorksOn [ssn, pNo, hours]

Foreign Keys:

Employee.dNumber references Department.dNumber

Employee.superSSN references Employee.ssn

Department.mgrSSN references Employee.ssn

Project.dNumber references Department.dNumber

Dependent.ssn references Employee.ssn

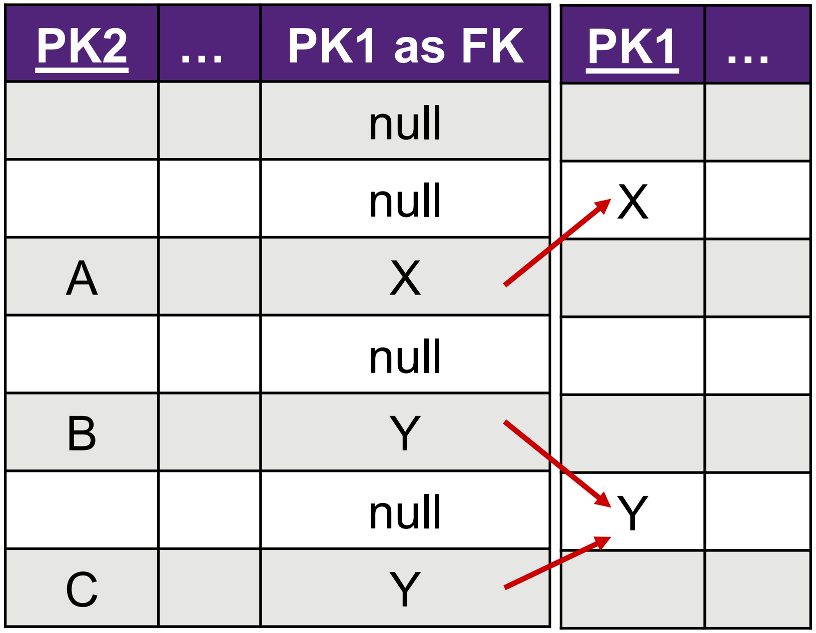
WorksOn.ssn references Employee.ssn

WorksOn.pNo references Project.pNo

Sparse Relationship Mapping

Note: 1:1 and 1:N relationships can be mapped in the same way as M:N

Advantageous when the relationship is sparse as it reduces the number of "NULLs" that appear as foreign key values.



PK2	…	PK1 as FK
		null
		null
A		X
		null
B		Y
		null
C		Y

PK1	…
X	
Y	

Standard Implementation

PK2	…
A	
B	
C	

PK1	…
X	
Y	

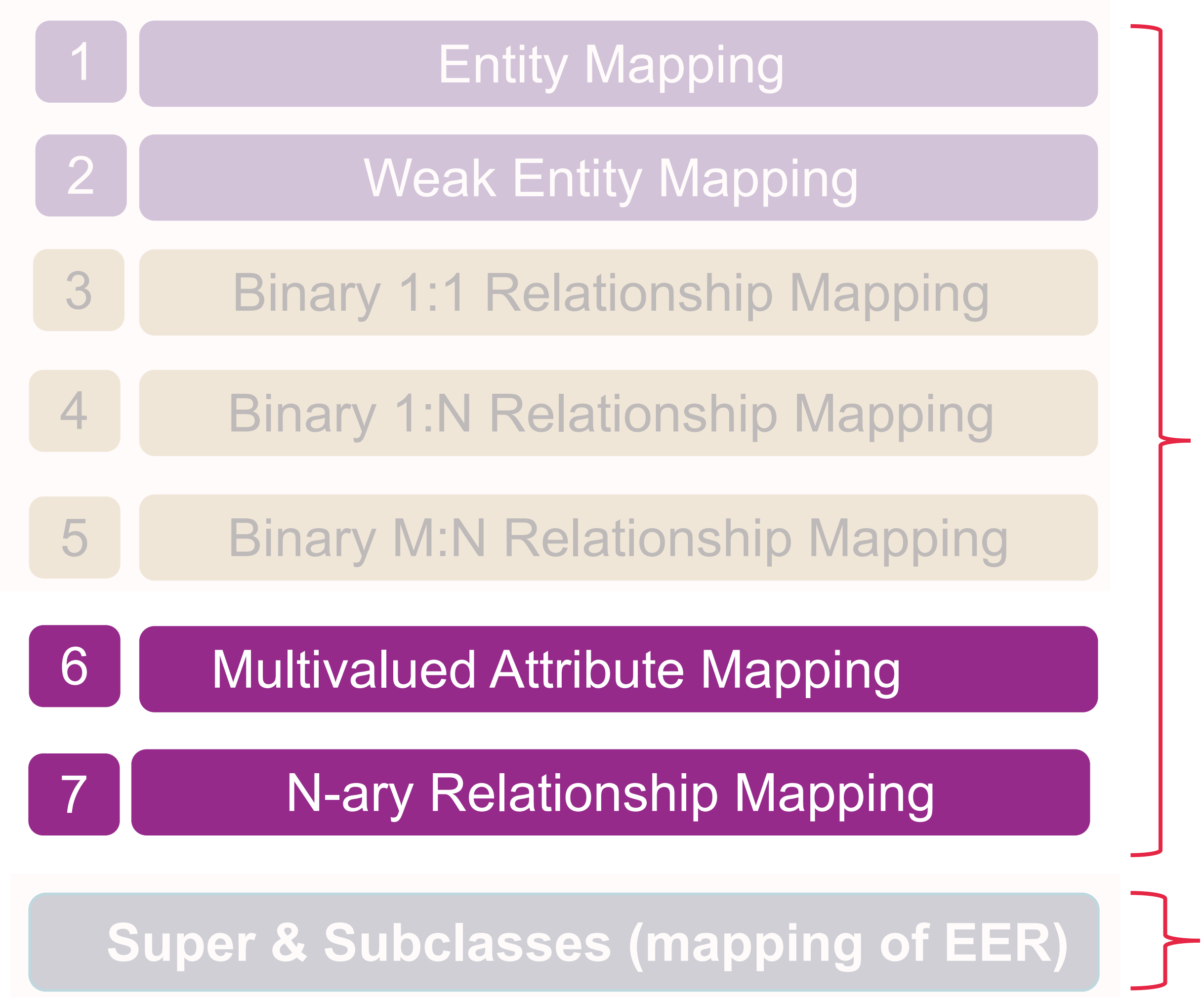
PK2	PK1
A	X
B	Y
C	Y

M:N Implementation

(7+1) – Steps for Mapping

Input:
An ER model

Output:
Relations with primary/foreign key constraints



1 Entity Mapping

2 Weak Entity Mapping

3 Binary 1:1 Relationship Mapping

4 Binary 1:N Relationship Mapping

5 Binary M:N Relationship Mapping

6 Multivalued Attribute Mapping

7 N-ary Relationship Mapping

Super & Subclasses (mapping of EER)

Mapping of ER

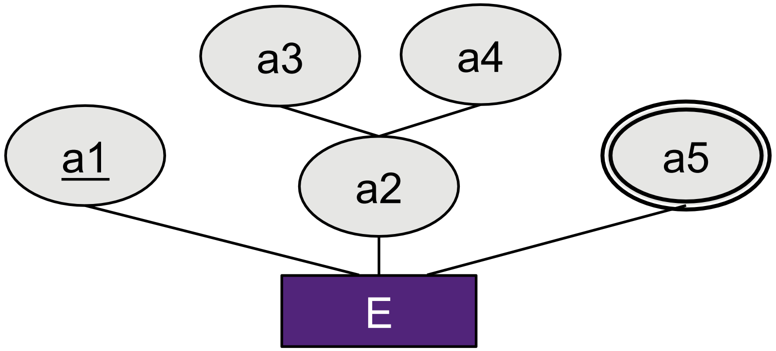
Additional step for mapping of EER

Step 6: Multivalued Attributes

For each multivalued attribute:

1. Create a new relation.
2. Include a foreign key from the relation to the primary key of the associated entity type.
3. The primary key is the combination of the multivalued attributes and the primary key of its associated entity type.
4. If the multivalued attribute is composite, include its simple components.

ER Diagram



Mapping to Relation

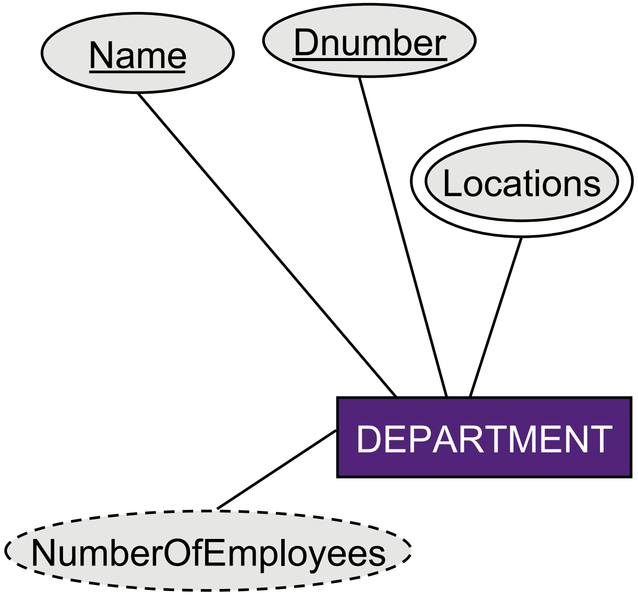
E [a1, a3, a4]

E2 [a1, a5]

E2.a1 references E.a1

Step 6: Company Example

There is only one multivalued attributes in the Company Database: **Locations**



DeptLocs includes the primary key of DEPARTMENT as a foreign key.

Location would also be included in the primary key.

Mapping to Relation

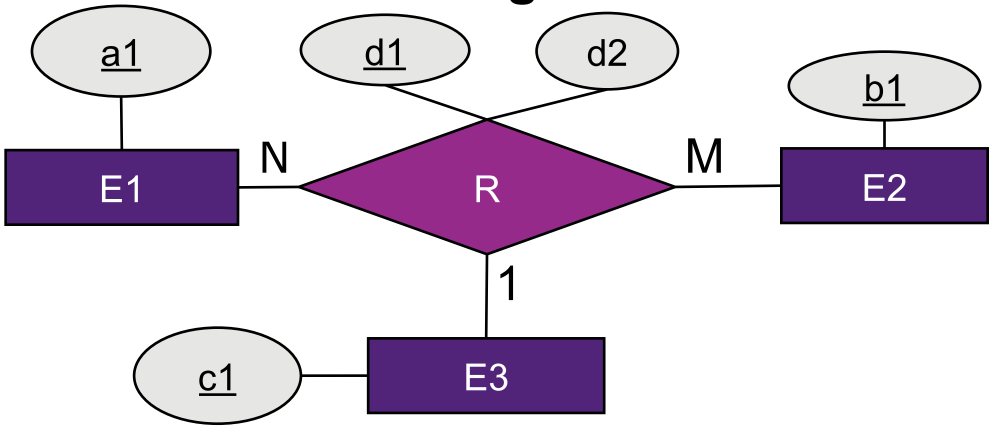
DeptLocs [dNumber, location]
DeptLocs.dNumber references Department.dNumber

Step 7: N-ary Relationships (Exc. 1:1:1 & N:1:1)

For each N-ary relationship type

1. Create a new relation
2. Include foreign keys from the relation to the primary key of the participating entity types
3. The combination of foreign keys from entity types with many cardinality will form the primary key
4. R can have its own attributes that contribute to the primary key
5. Include any simple attributes of R as attributes of the new relation

ER Diagram



Mapping to Relation

E1 [a1] E2 [b1] E3 [c1]

R [a1, b1, d1, c1 d2]

R.a1 references E1.a1

R.b1 references E2.b1

R.c1 references E3.c1

Step 7: Examples

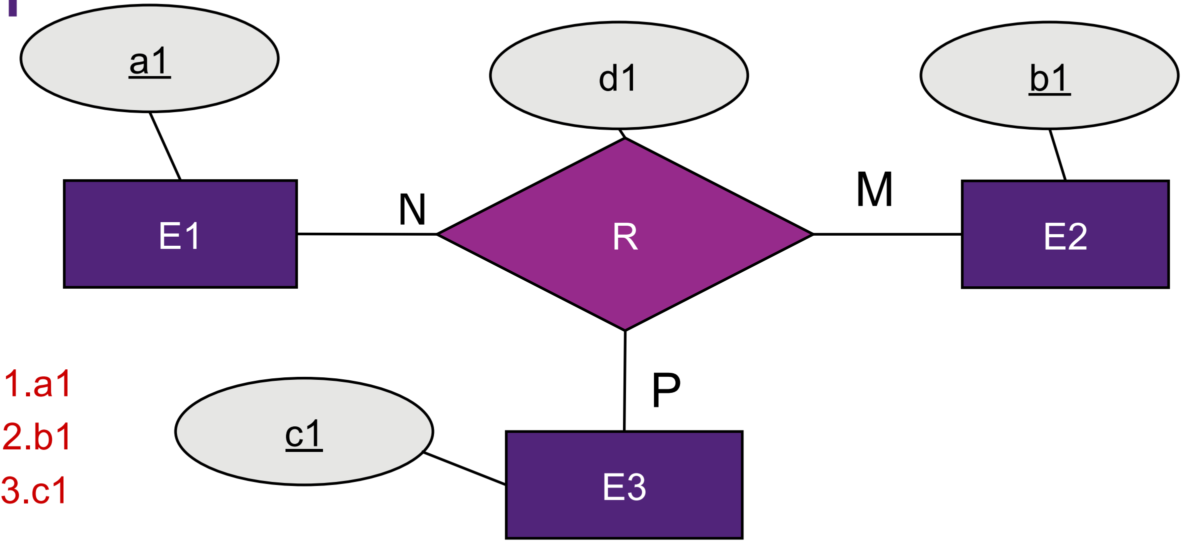
E1 [a1]

E2 [b1]

E3 [c1]

R [a1, b1, c1, d1]

- R.a1 references E1.a1
- R.b1 references E2.b1
- R.c1 references E3.c1



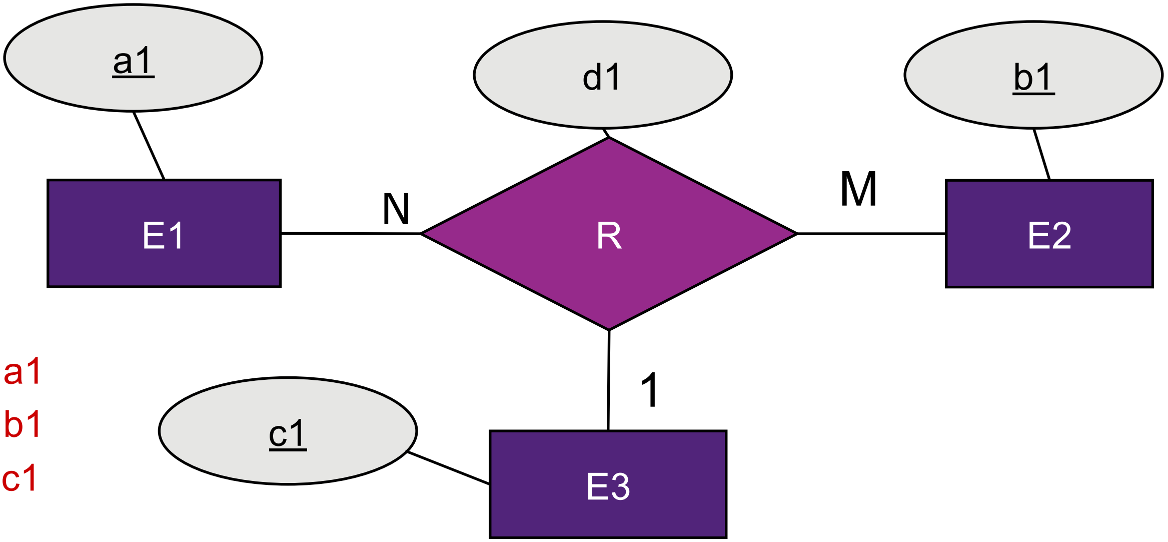
E1 [a1]

E2 [b1]

E3 [c1]

R [a1, b1, c1, d1]

- R.a1 references E1.a1
- R.b1 references E2.b1
- R.c1 references E3.c1



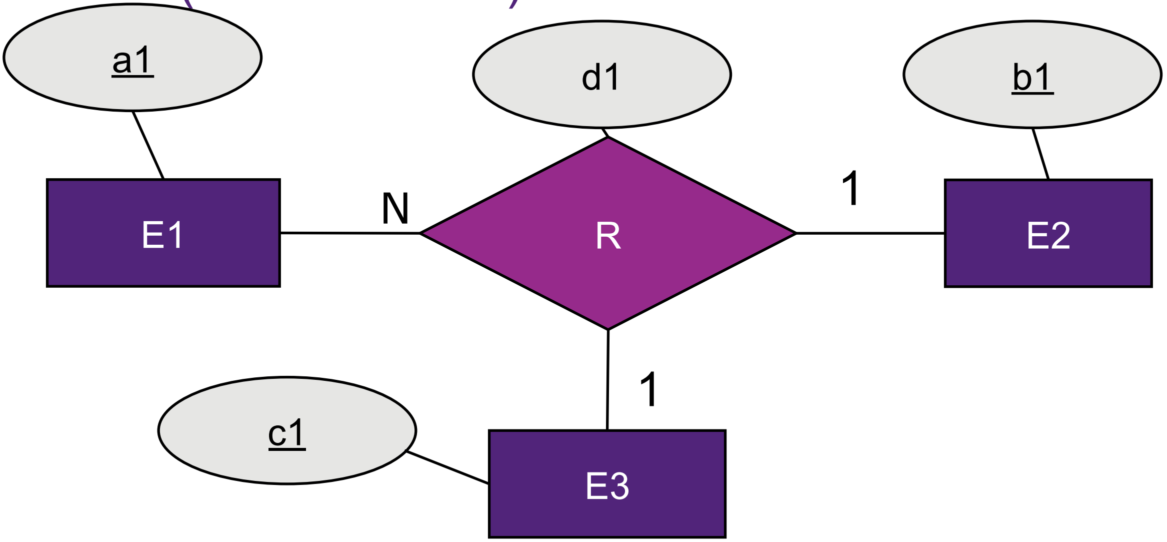
Step 7: Examples (1:1:1 & N:1:1)

E1 [a1, b1, c1, d1]

- E1.b1 references E2.b1
- E1.c1 references E3.c1

E2 [b1]

E3 [c1]

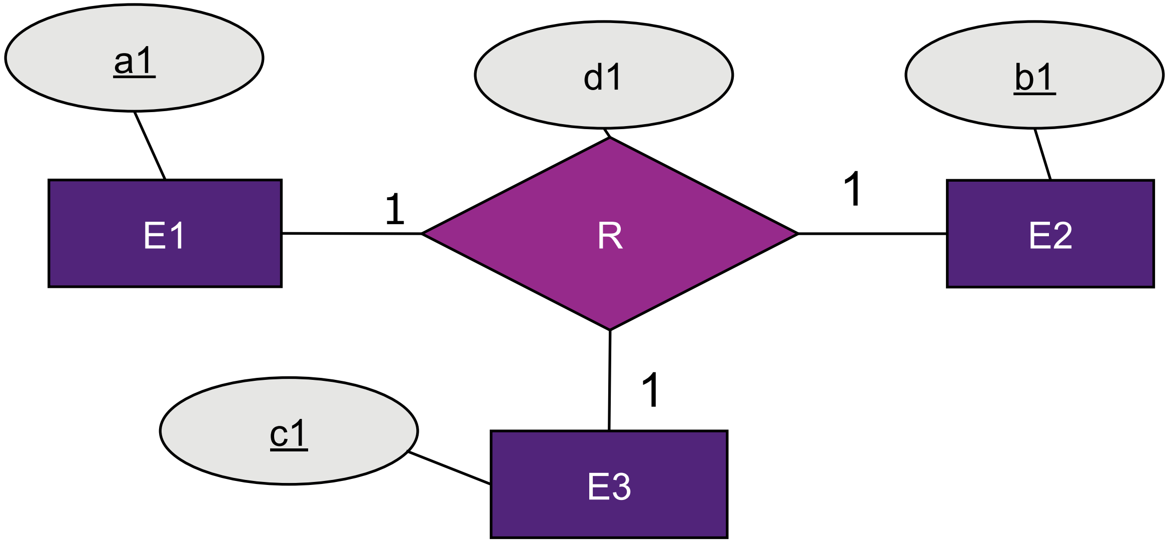


E1 [a1, ..., b1, c1, d1]

- E1.b1 references E2.b1
- E1.c1 references E3.c1

E2 [b1]

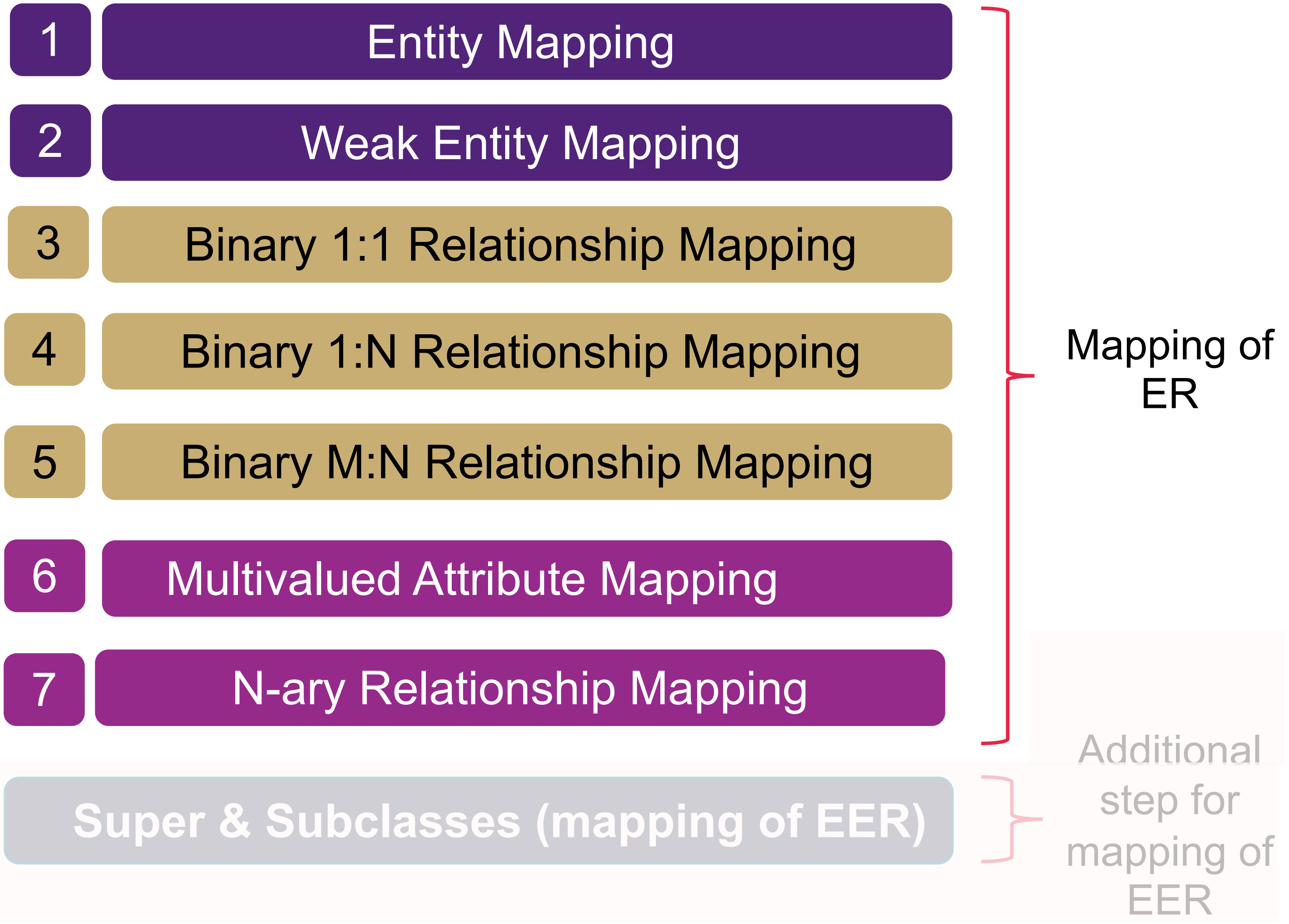
E3 [c1]



(7+1) – Review of 7-Steps for ER Mapping

Input:
An ER model

Output:
Relations with primary/foreign key constraints



1 Entity Mapping

2 Weak Entity Mapping

3 Binary 1:1 Relationship Mapping

4 Binary 1:N Relationship Mapping

5 Binary M:N Relationship Mapping

6 Multivalued Attribute Mapping

7 N-ary Relationship Mapping

Mapping of ER

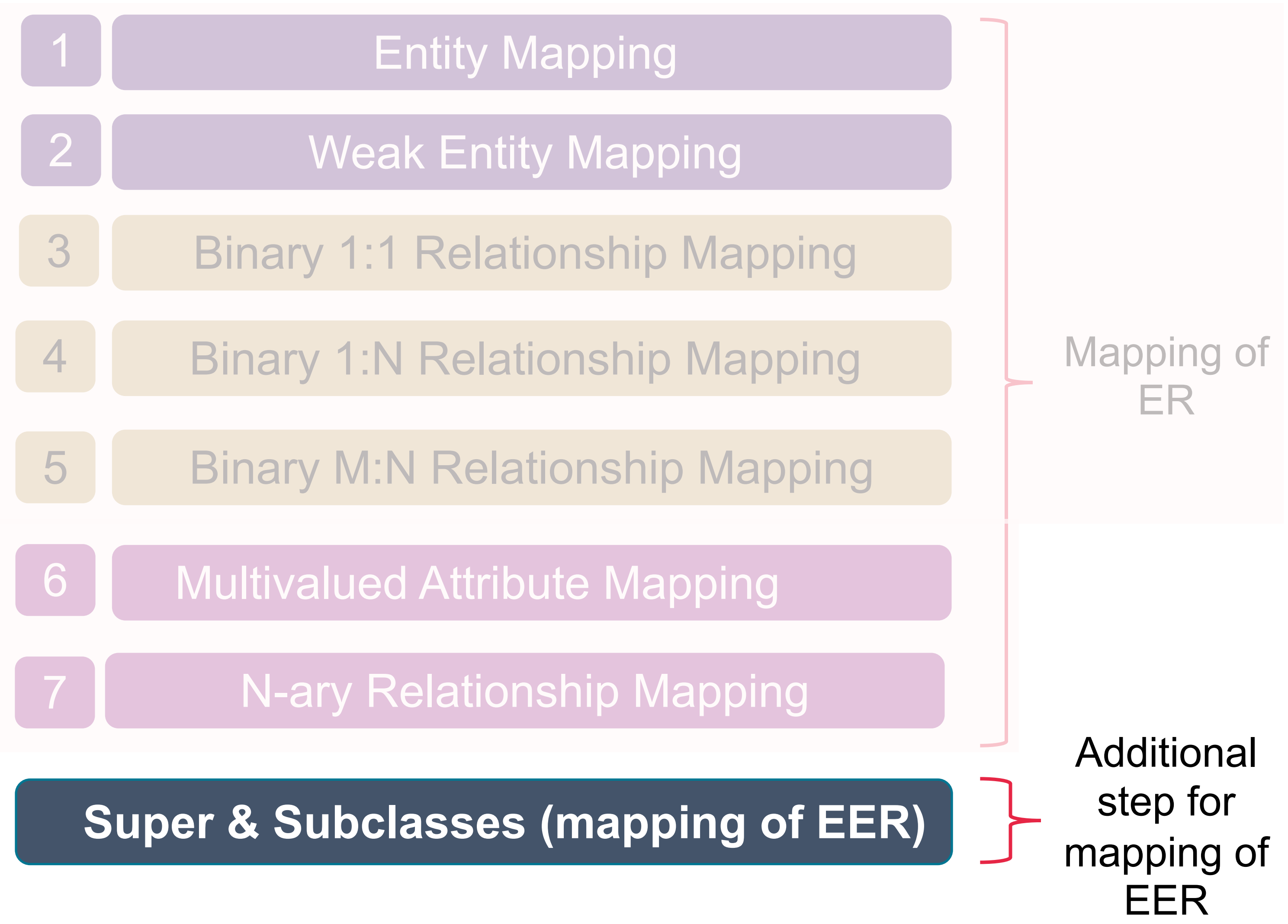
Super & Subclasses (mapping of EER)

Additional step for mapping of EER

(7+1) – Review of 7-Steps for ER Mapping

Input:
An ER model

Output:
Relations with primary/foreign key constraints



1 Entity Mapping

2 Weak Entity Mapping

3 Binary 1:1 Relationship Mapping

4 Binary 1:N Relationship Mapping

5 Binary M:N Relationship Mapping

6 Multivalued Attribute Mapping

7 N-ary Relationship Mapping

Mapping of ER

Super & Subclasses (mapping of EER)

Additional step for mapping of EER

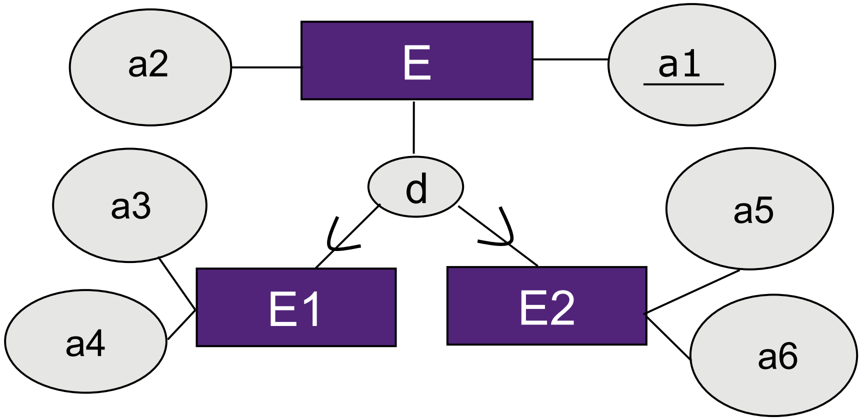
Super & Subclasses Mapping

The following method works for total/partial and disjoint/overlapping subclasses.

For each subclass entity type

1. Create a relation
2. The primary key of each of the subclasses is the primary key of the superclass.
3. Include a foreign key from the relation to the primary key of the relation of its superclass entity type.
4. Include all simple attributes which are not composite, derived or multivalued.

ER Diagram



Mapping to Relation

E [<u>a1</u>, a2]

E1 [<u>a1</u>, a3, a4]

E1.a1 references E.a1

E2 [<u>a1</u>, a5, a6]

E2.a1 references E.a1

Super & Subclasses Mapping

The following method works for total/partial and disjoint/overlapping subclasses.

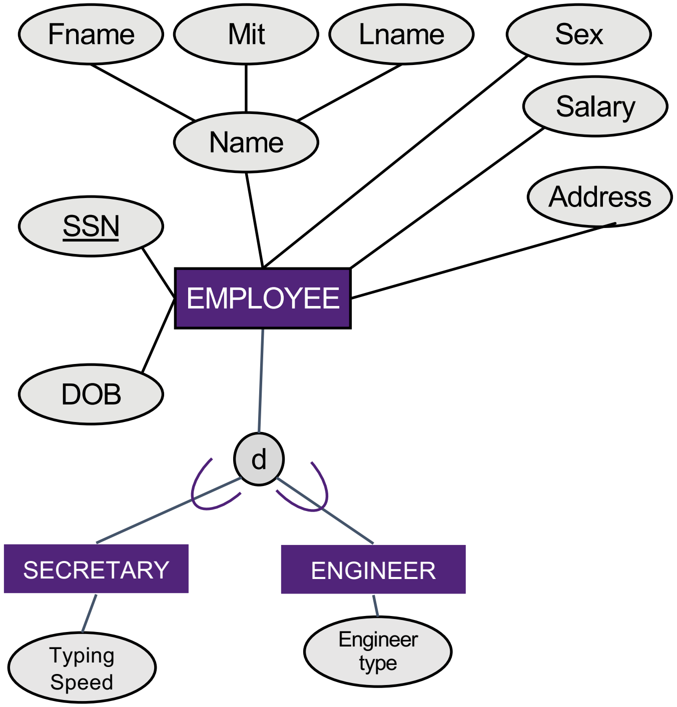
For each subclass entity type

1. Create a relation
2. The primary key of each of the subclasses is the primary key of the superclass.
3. Include a foreign key from the relation to the primary key of the relation of its superclass entity type.
4. Include all simple attributes which are not composite, derived or multivalued.

When to complete subclass mapping? This is an additional step for EER that does not occur at and specific stage, but is included as needed:

- Subclasses are generally mapped after either Step 1 or Step 2, depending on the specific ER diagram.
- This allows them to be modified or referenced during the relationship mapping steps.

Subclass Entity: Company Example



Mapping to Relation

Employee [ssn, fName, mIt, lName, dob, address, sex, salary]

Secretary [ssn, typingSpeed]
Secretary.ssn references Employee.ssn

Engineer [ssn, typing]
Engineer.ssn references Employee.ssn

Final Schema

Relations:

Employee [ssn, fName, mlt, lName, dob, address, sex,

salary, dNumber, superSSN]

Department [dNumber, dName, mgrSSN, startDate]

Project [pNo, pName, pLocation, dNumber]

Dependent [ssn, depName, sex, dob, relationship]

Secretary [ssn, typingSpeed]

Engineer [ssn, engineerType]

WorksOn [ssn, pNo, hours]

DeptLocs [dNumber, location]

Foreign Keys:

Employee.dNumber references Department.dNumber

Employee.superSSN references Employee.ssn

Department.mgrSSN references Employee.ssn

Dependent.ssn references Employee.ssn

Project.dNumber references Department.dNumber

Secretary.ssn references Employee.ssn

Engineer.ssn references Employee.ssn

WorksOn.Ssn references Employee.Ssn

WorksOn.pNo references Project.pNo

DeptLocs.dNumber references Department.dNumber

Another Example

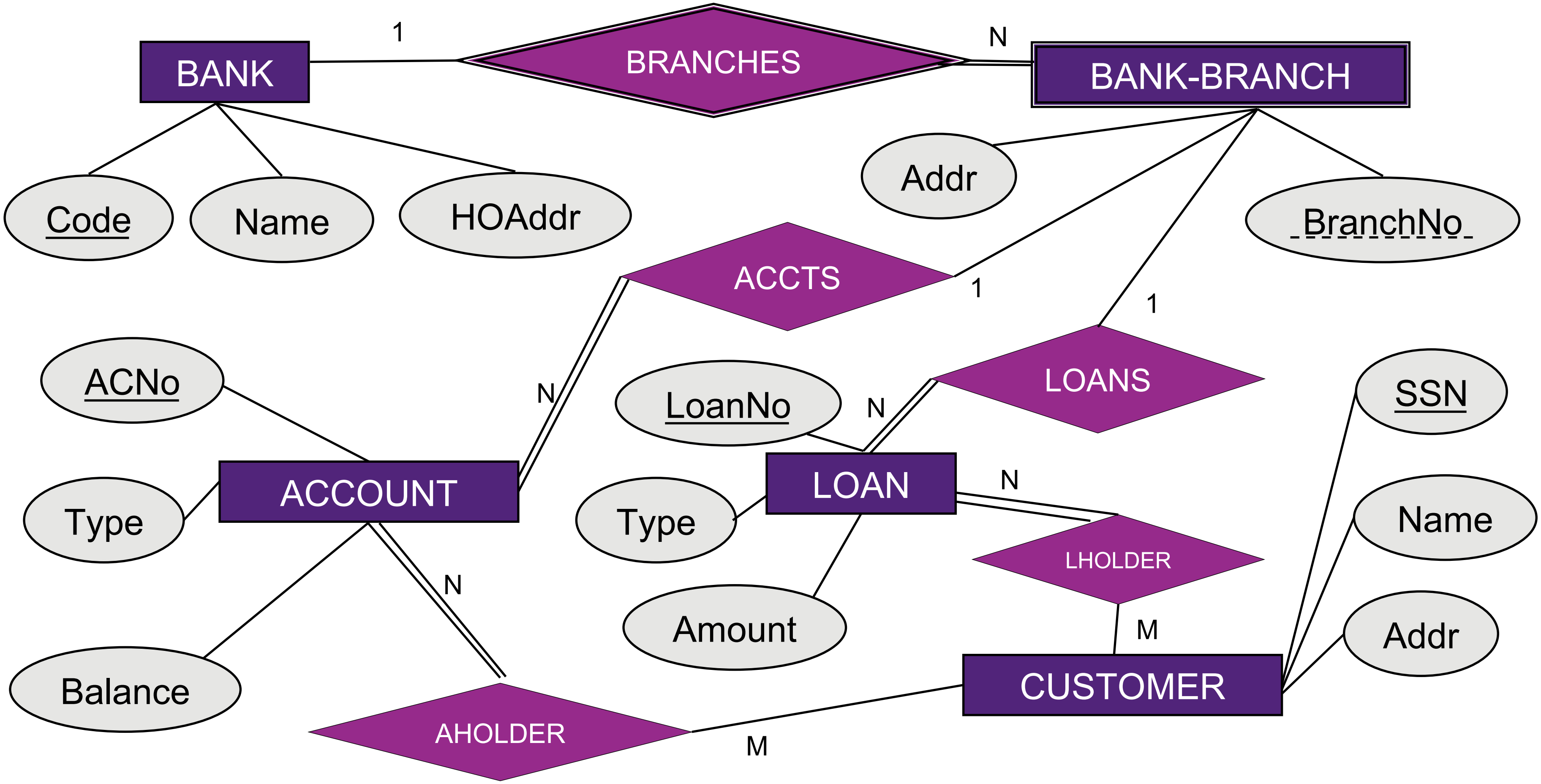
A bank, given by its unique code, name and head office address, can have several branches. Each branch within a given bank has a branch number and address

One branch can have several accounts, each identified by an AC number. Every account has a type, current balance, and one or more account holders

One branch can have several loans, each given by a unique loan number, type, amount and one or more loan holders

The name, address and SSN of all customers (account and loan holders) of the bank are recorded and maintained

ER Diagram


BANK
1
BRANCHES
N
BANK-BRANCH
Code
Name
HOAddr
Addr
BranchNo
ACCTS
1
1
LOANS
ACNo
N
LoanNo
N
SSN
Type
ACCOUNT
Type
LOAN
N
Name
LHOLDER
N
Amount
M
Addr
Balance
CUSTOMER
AHOLDER
M

Relational Schema (after step 1)

Relations:

Bank [code, name, hoAddr]

Account [acNo, type, balance]

Loan [loanNo, type, amount]

Customer [ssn, name, address]

Foreign Keys:

Relational Schema (after step 2)

Relations:

Bank [code, name, hoAddr]

Account [acNo, type, balance]

Loan [loanNo, type, amount]

Customer [ssn, name, address]

Branch [bankCode, branchNo, addr]

Foreign Keys:

Branch.bankCode references Bank.code

Relational Schema (after step 3)

Relations:

Bank [code, name, hoAddr]

Account [acNo, type, balance]

Loan [loanNo, type, amount]

Customer [ssn, name, address]

Branch [bankCode, branchNo, addr]

Foreign Keys:

Branch.bankCode → Bank.code

Relational Schema (after step 4)

Relations:

Bank [code, name, hoAddr]

Account [acNo, type, balance, **bankCode**, **branchNo**]

Loan [loanNo, type, amount, **bankCode, branchNo**]

Customer [ssn, name, address]

Branch [bankCode, branchNo, addr]

Foreign Keys:

Branch.bankCode → Bank.code

Account.{bankCode, branchNo} references Branch.{bankCode, branchNo}

Loan.{bankCode, branchNo} references Branch.{bankCode, branchNo}

Relational Schema (after step 5)

Relations:

Bank [code, name, hoAddr]

Account [acNo, type, balance, bankCode, branchNo]

Loan [loanNo, type, amount, bankCode, branchNo]

Customer [ssn, name, address]

Branch [bankCode, branchNo, addr]

AccountHolder [acNo, ssn]

LoanHolder [loanNo, ssn]

Foreign Keys:

Branch.bankCode references Brank.code

Account.{bankCode, branchNo} references Branch.{bankCode, branchNo}

Loan.{bankCode, branchNo} references Branch.{bankCode, branchNo}

AccountHolder.acNo references Account.acNo

AccountHolder.ssn references Customer.ssn

LoanHolder.loanNo references Loan.loanNo

LoanHolder.ssn references Customer.ssn

Review

Do you know …

- What is the Relational Data Model and what are its main components?
- What are integrity constraints and how are they enforced by a DBMS?
- How can we map an ER diagram to a relational schema?

Reading

- Chapters 5 and 9 in Elmasri & Navathe

Next Module

- Module 3: Relational Query Languages – SQL