

The background features a dark, textured surface with a network of light green circular icons connected by thin lines. The icons include a person, a folder, a cloud, a group of people, a house, a location pin, a padlock, and a globe. A hand is visible in the center, pointing towards the network.

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!

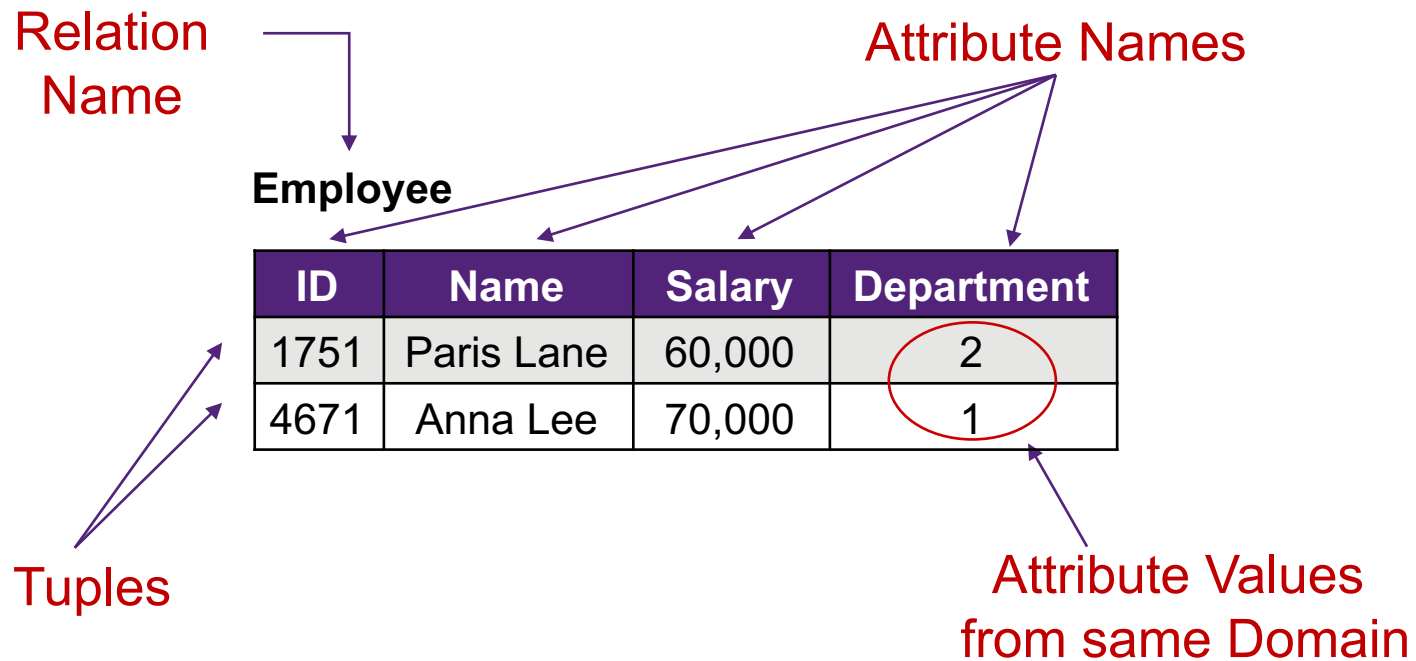
Relations have specific properties, based on the mathematical **set theory**

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



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., $1 \leq \text{grade} \leq 7$
- 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:

$$t = \langle v_1, v_2, \dots, v_n \rangle$$

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 $R [A_1, A_2, A_3, \dots, A_n]$, includes a relation name R and list of attributes A_1, A_2, \dots, 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 $r = \{t_1, t_2, \dots, t_m\}$.

Relation Schema and Instance Example

Relational Schema Example

Employee [id, name, sex, 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 specified on the database schema

- They must hold on every instance of that schema, as well as on transitions of the schema

Integrity constraints enforced by DBMS include:

- Domain constraints
- Key constraints
- Entity integrity constraints
- Referential integrity constraints
- Semantic integrity constraints



Domain Constraints

A **domain** is a set of atomic values

Each attribute in a relation will belong to some domain

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

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



Uniqueness and Superkey

Uniqueness Constraint: All tuples in a relation must be distinct (i.e. no two tuples can have same values for all attributes)

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

4671	Anna Lee	70,000	1
------	----------	--------	---



A **superkey** is a subset of attributes (SK) of a relation schema R, such that for any two tuples, t_i and t_j in a relation state r of R

$$t_i[\text{SK}] \neq t_j[\text{SK}]$$

Every relation has at least one superkey

- Trivially, the set of all its attributes

Key

K is a **key** in a relation schema R iff

- K is a superkey of R, and
- K does not contain any extraneous attributes
 - That is, any subset of K is no longer a superkey of R

A key is a minimal superkey

- Smallest set of attributes that uniquely identify a tuple

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

Superkey example = {**ID, Name**}

Key example = {**ID**}

A schema may have more than one key

- Each is called a **candidate key**
- One is selected as the **primary key**, which would be underlined.

Question: Superkey

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

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

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

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

Key Constraint Example

Keys must remain unique at all times

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 Constraint

No primary key can be null at any time

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
------	------------	--------	---



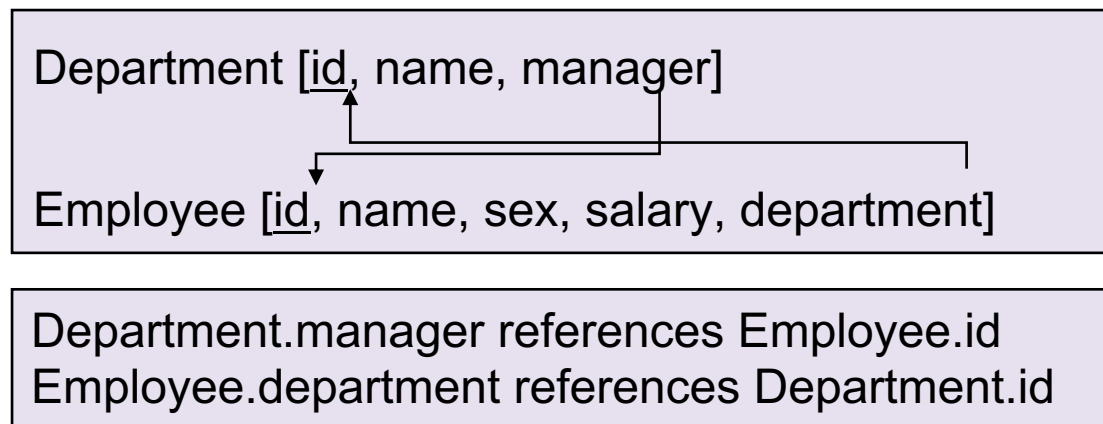
For primary keys that consists of multiple attributes, no part of the primary key can be null

Referential Integrity Constraint

Key and Entity Integrity constraints are specified on individual relations

Referential Integrity constraints are specified between two relations and are based on the notion of foreign keys

Foreign keys allow us to relate two different schemas. This can be viewed graphically or textually



Foreign Keys

Let FK be a set of attributes in R1

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 $t_1[\text{FK}]$ is null; or there exists a tuple t_2 in R2, such that $t_1[\text{FK}] = t_2[\text{PK}]$

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

Referential Integrity Constraint Example

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

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.department references Department.id

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}

Constraints and Operations

Enforcement of integrity constraints ensures that the database remains consistent

Changes to the database 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 & Insertion

Insert can violate four 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
-----	----------	--------	---

✗ What constraints is
violated in each one?

4671	John Doe	70,000	4
------	----------	--------	---

✗

NULL	John Doe	70,000	4
------	----------	--------	---

✗

1111	John Doe	70,000	6
------	----------	--------	---

✗

Constraints & Insertion

Insert can violate four 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

Constraints & Deletion

Referential integrity can be violated if the tuple being deleted is referenced by a foreign key from other tuples

The deletion can be rejected, cascaded or the referencing attribute values can be modified

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
will lead to a referential integrity violation.



Constraints & Modification

Non-key values

- domain check

Primary key

- similar to performing a delete and an insert

Foreign key

- DBMS must ensure new value refers to existing tuple in referenced relation

Employee

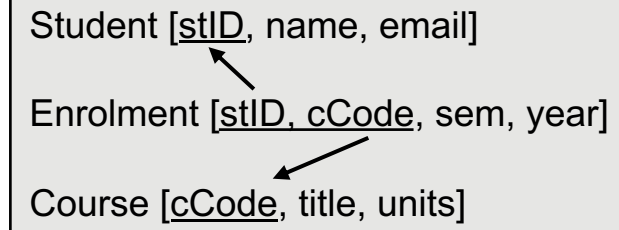
ID	Name	Salary	Department
1751	Paris Lane	60,000	2
4671	Anna Lee	70,000	1
1023	Ben Cho	70,000	4
2034	Jack Smith	40,000	1
2670	Grace Mills	50,000	2

Department

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

In-class Exercise

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



1. Superkey
2. Minimal Key
3. Foreign Key
4. 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, 40,000, 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, 40,000, 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]

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

Department

ID	Name	Manager
1	Marketing	4671
2	Development	1751

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

Semantic Integrity Constraint

Constraints that cannot be directly expressed in the schemas of the data model referred to as **semantic constraints** or business rules.

- **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 constraints can be violated during insertion, deletion or modification.

Semantic Integrity Constraint Example

Example: The salary of an employee should not exceed the employee's 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



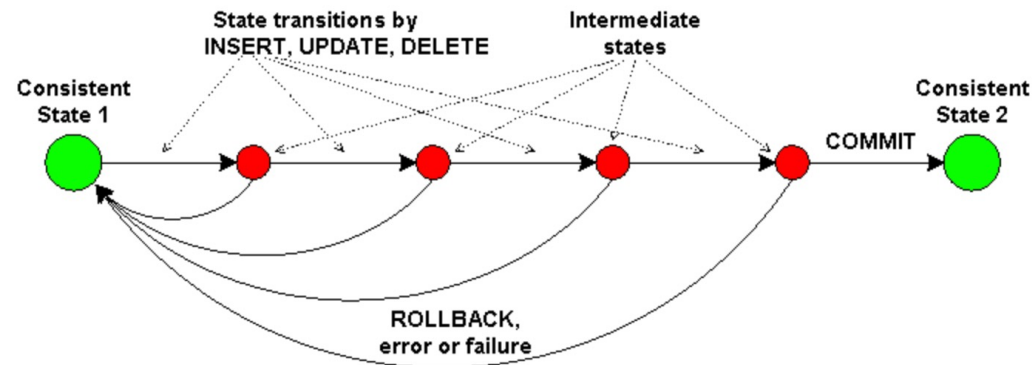
Employee.department references Department.id

Department

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

The Transaction Concept

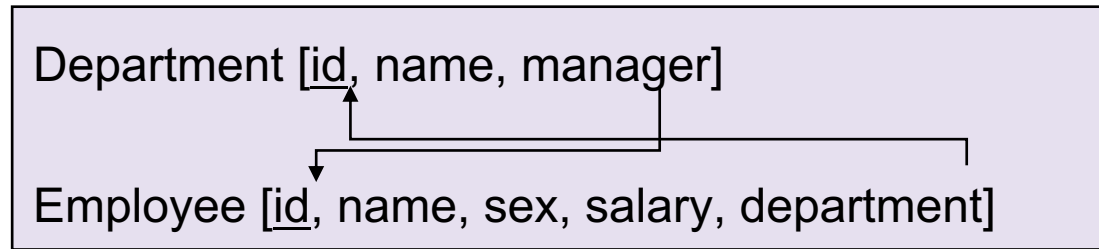
- A **transaction** is an executing program that includes some database operations, such as reading from the database, or applying insertions, deletions, or updates to the database.
- At the end of the transaction, it **must** leave the database in a valid or consistent state that satisfies all the constraints 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



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

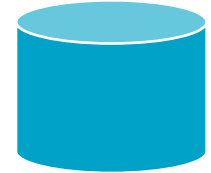
Integrity Constraints

ER to Relational Mapping



User's perspective

Conceptual perspective



Storage perspective

Database Requirements

Conceptual Design

Conceptual Schema (ER)

The ER Model is commonly used for conceptual design

Logical Design (Mapping)

The Relational Model is the basis for many commercial DBMSs

Logical Schema (Relational)

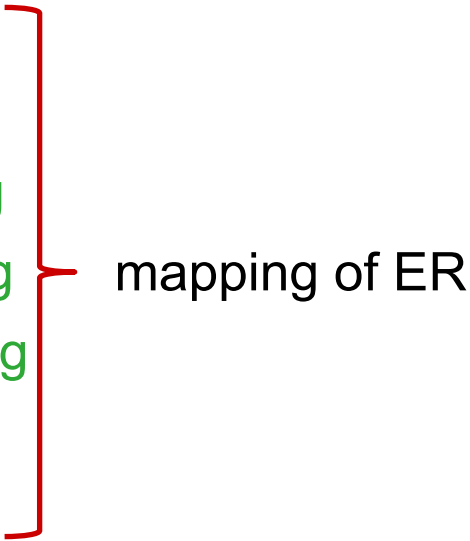
Internal Schema

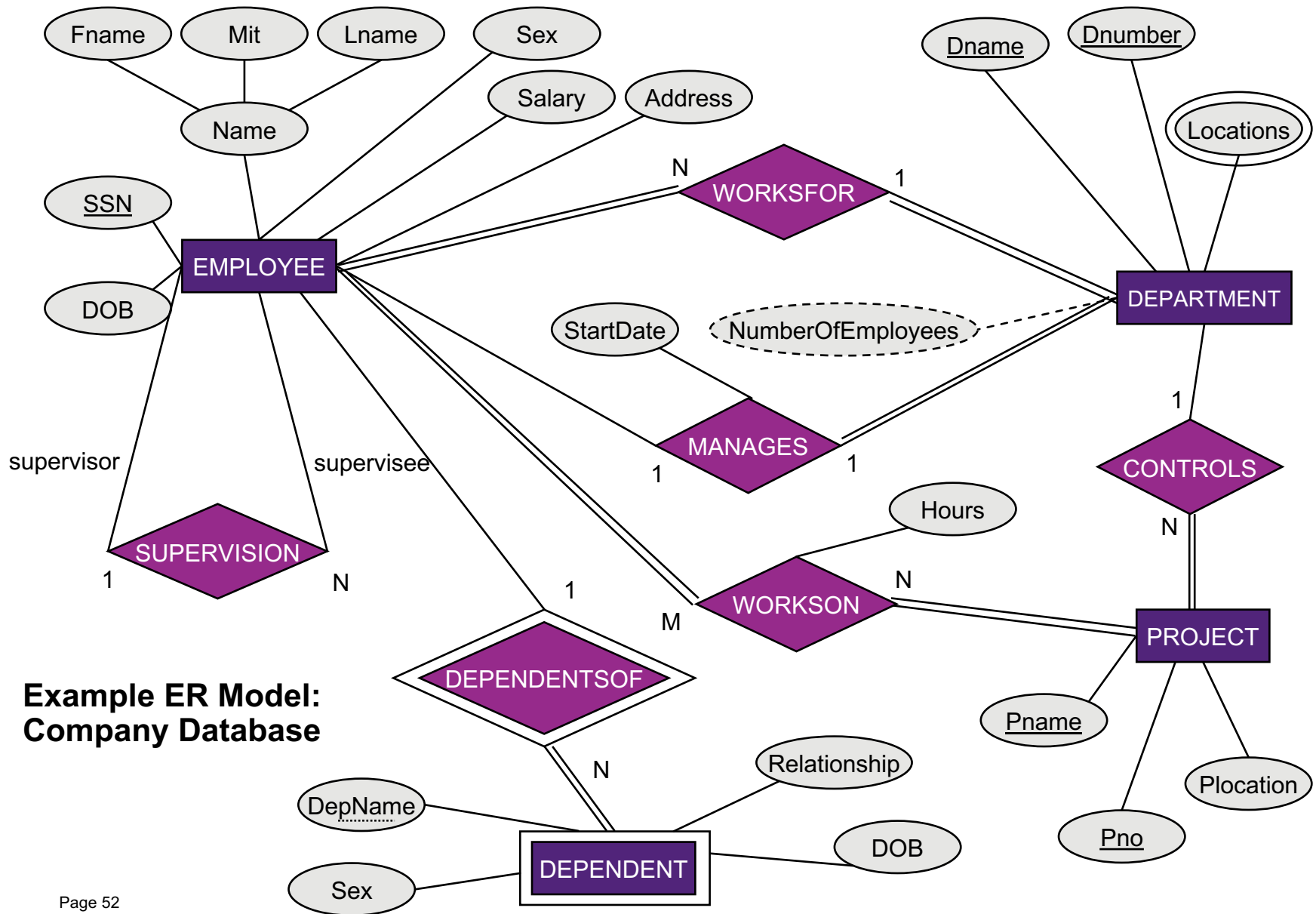
(7+1) – Steps for Mapping

Input: an ER model

Output: relations with primary/foreign key constraints

Steps:

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
 8. Super & Subclasses (mapping of EER)
- 
- mapping of ER



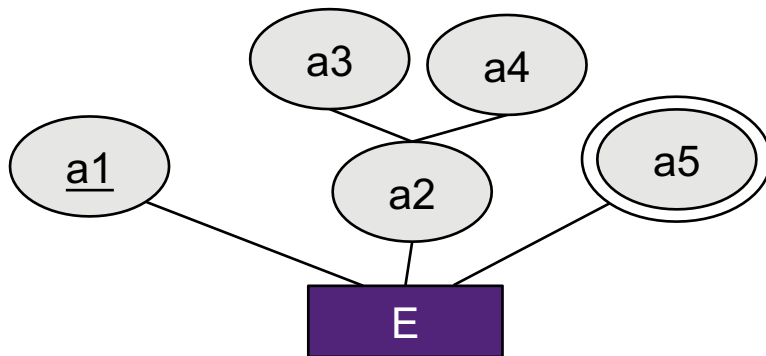
Step 1: Regular Entity Mapping

Regular: **non-weak** entity with **simple attributes**

For each entity type E, create a relation R that includes all attributes of E

- **Include only simple component attributes of a composite attribute**
- **Don't include derived attributes**
- **Choose one key attribute of E as primary key for R**

Note: Consider foreign key relationship, weak entities and multi-valued attributes in subsequent steps.

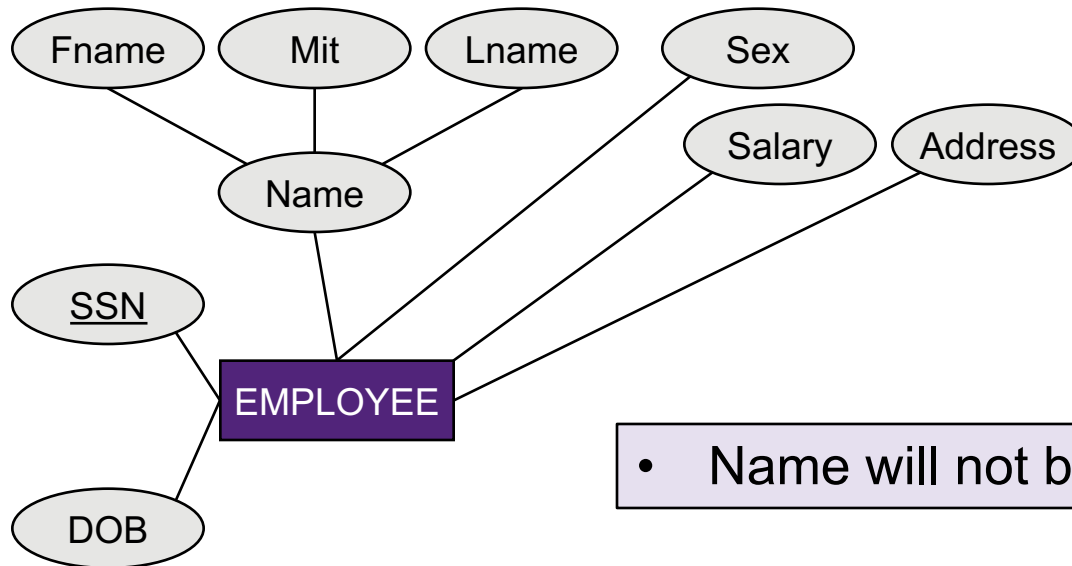


E [a1, a3, a4]

...what if a2 is the key?

Step 1: Example

Entities in the Company Database:
EMPLOYEE, DEPARTMENT, PROJECT

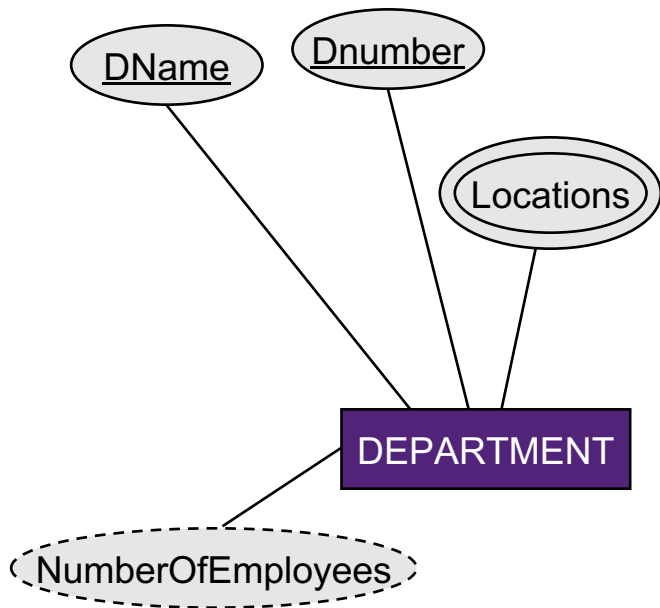


- Name will not be added to the relation.

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

Step 1: Example

Entities in the Company Database:
EMPLOYEE, DEPARTMENT, PROJECT

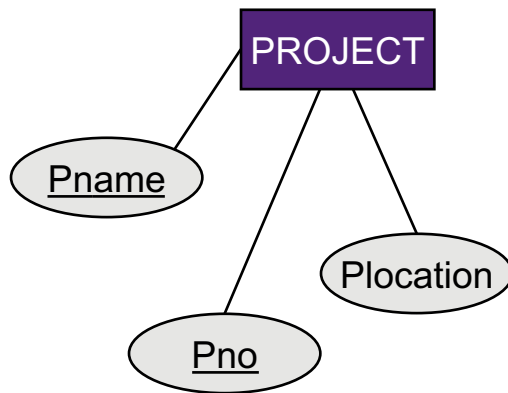


- Location will not be added for the time being
- Both Name and Number are keys. Number is taken as PK.
- NumberOfEmployees is not added to the relation as it is a derived attribute.

Department [dNumber, dName]

Step 1: Example

Entities in the Company Database:
EMPLOYEE, DEPARTMENT, PROJECT



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

Project [pNo, pName, pLocation]

Schema (in progress)

Relations:

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

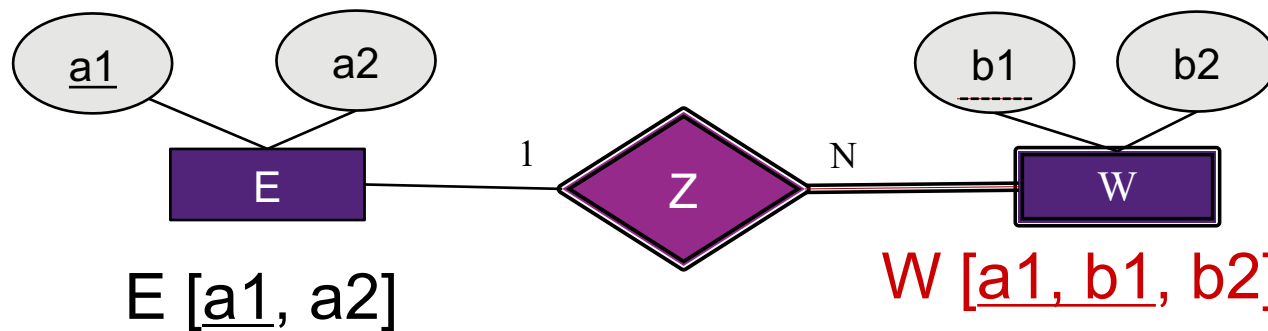
Department [dNumber, dName]

Project [pNo, pName, pLocation]

Step 2: Weak Entity

For each weak entity type W with owner entity type E

- Create a relation R that includes all simple attributes of W
- Include as foreign key attributes in R the primary key attributes of E
- The primary key of R is the combination of the primary key of E and the partial key of W (if any)

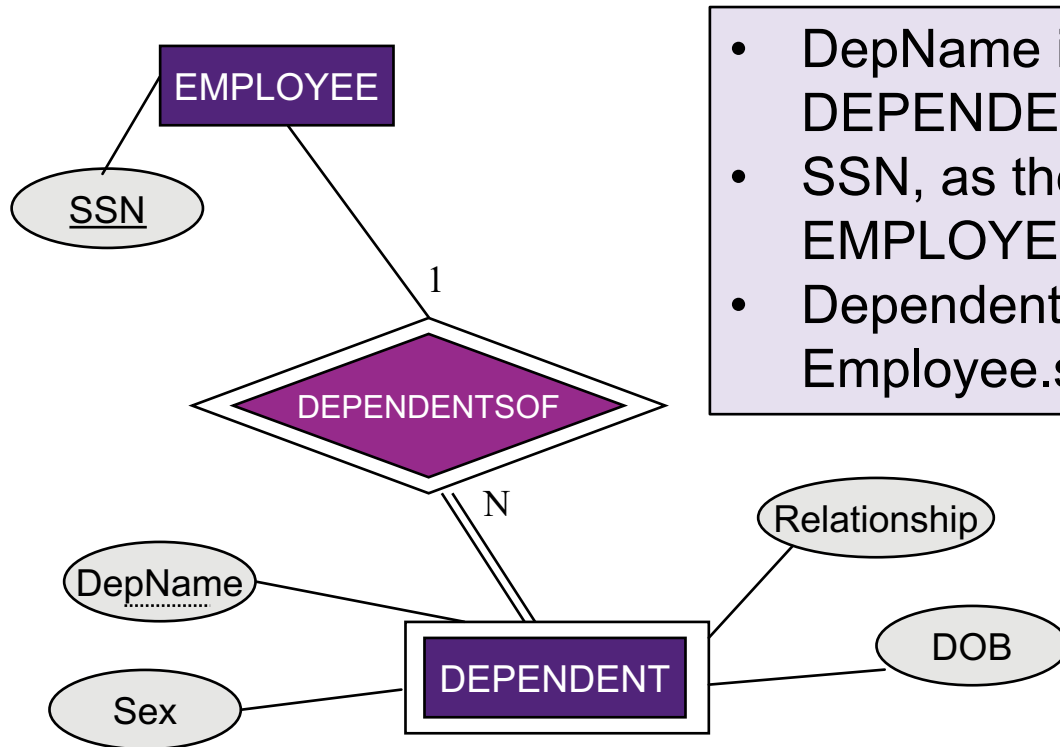


- $W.a1$ references $E.a1$

...if W has multiple owner entities, include the primary keys of all owner relations.

Step 2: Example

Weak Entities in the Company Database: DEPENDENT



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

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

- *Dependent.ssn references Employee.ssn*

Schema (in progress)

Relations:

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

Department [dNumber, dName]

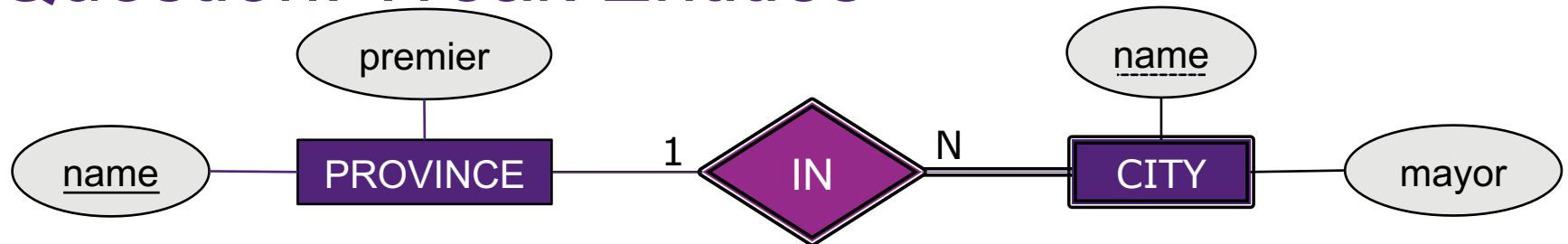
Project [pNo, pName, pLocation]

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

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

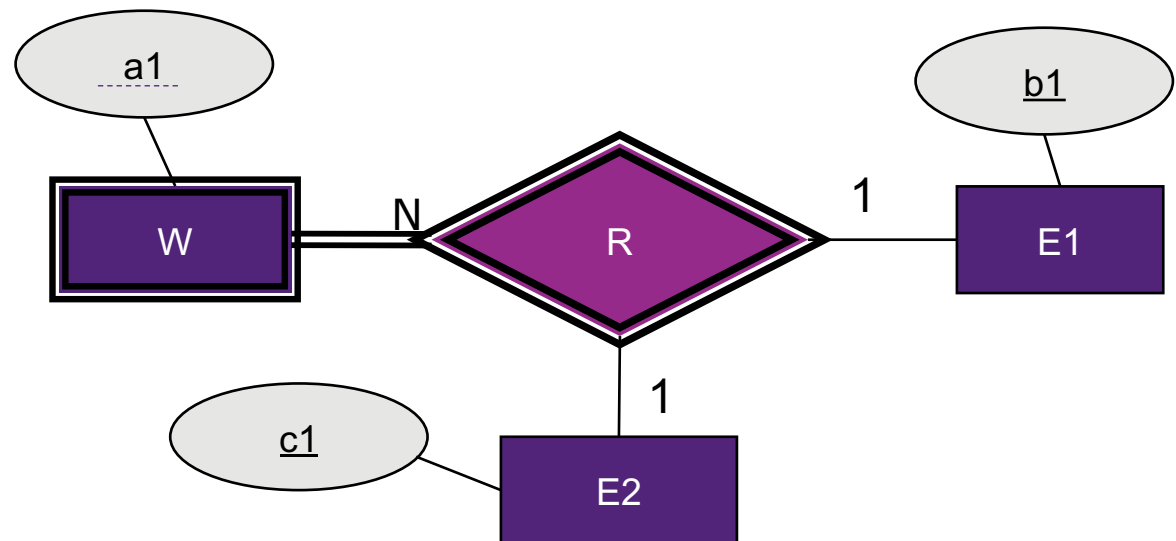
- Create a relation R that includes all simple attributes of W
- Include as foreign key attributes in R the primary key attributes of owner entity types.
- The primary key of R is the combination of the primary key of owner entities and the partial key of W

W [a1, b1, c1]

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

$E1$ [b1]

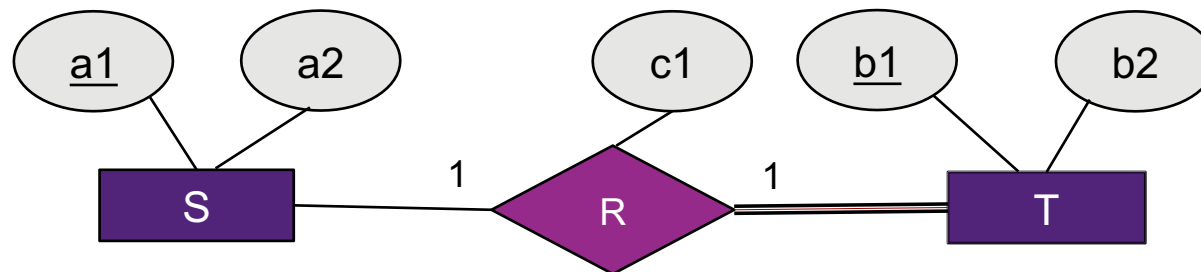
$E2$ [c1]



Step 3: Binary 1:1 Relationship

For each binary 1:1 relationship type R, with participating relations S & T

- Choose one relation (say T) and include as foreign key in T the primary key of S
 - It is better to choose the entity type with **total participation** in R
- Include all the simple attributes (or the simple components of composite attributes) of **R** as attributes of T



S [a1, a2]

T [b1, b2, **a1**, **c1**]

- T.a1 references S.a1

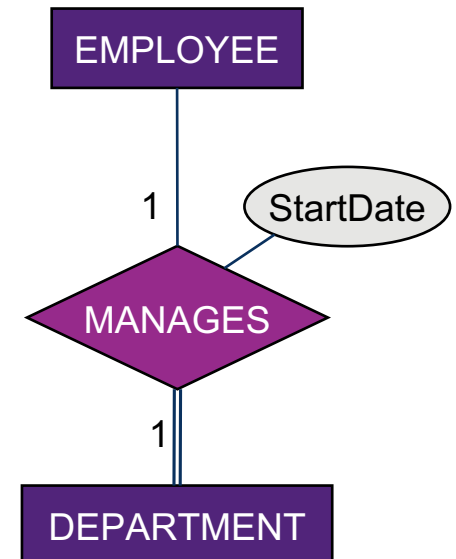
...why it's better to choose the side with total participation?

Step 3: Example

Binary 1:1 relationship type in the Company Database: **MANAGES**

- T = **DEPARTMENT**
- S = **EMPLOYEE**
- 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**)

Given that department must have a manager, extending department is the better choice.



Department [dNumber, dName, **mgrSSN**, **startDate**]

- *Department.mgrSSN references Employee.ssn*

Schema (in progress)

Relations:

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

Department [dNumber, dName, mgrSSN, startDate]

Project [pNo, pName, pLocation]

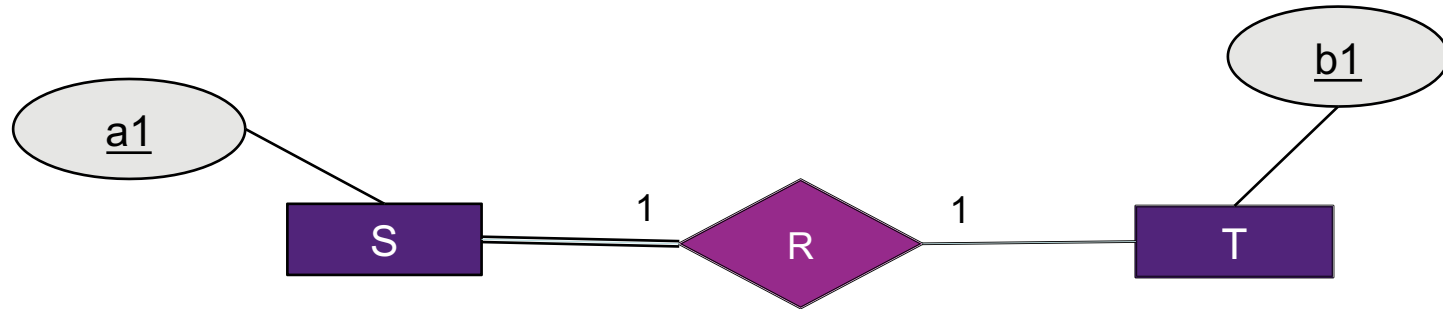
Dependent [ssn, depName, sex, dob, relationship]

Foreign Keys:

Department.mgrSSN references Employee.Ssn

Dependent.ssn references Employee.Ssn

Question: Binary Relationship



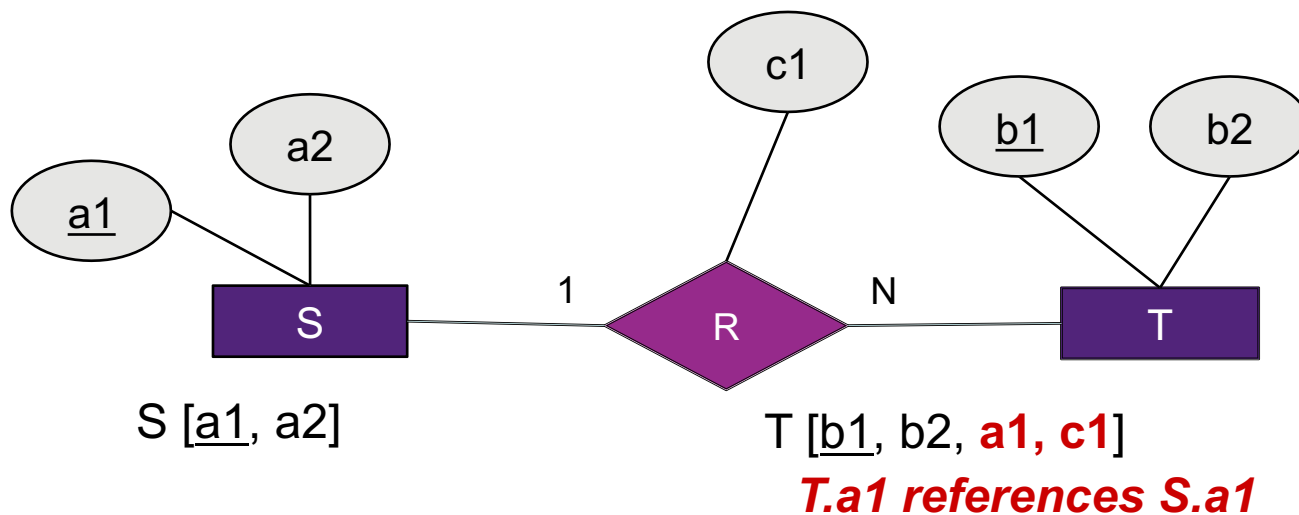
Which schema below is a reasonable translation from ER to relations?

- 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

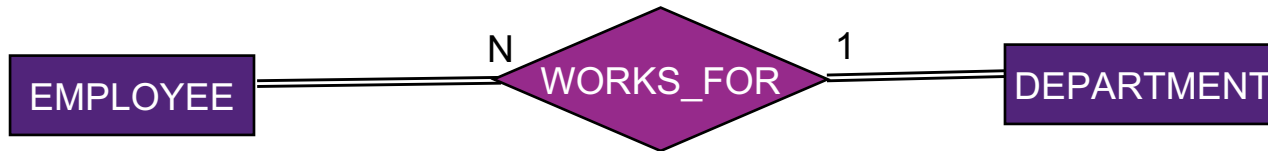
For each (non-weak) binary 1:N relationship R, identify relation T that represents the participating entity type at the **N-side** of the relationship type

- Include as **foreign key** of T the primary key of relation S that represents the other entity participating in R
- Include any **simple attributes** (or simple components of composite attributes) of the 1:N relationship as attributes of T



Step 4: Example

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



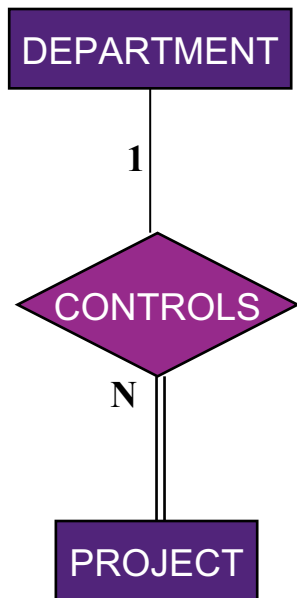
Employee [ssn, fName, mlt, lName, dob, address, sex, salary, dNumber]

- Employee.dNumber references Department.dNumber

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

Step 4: Example

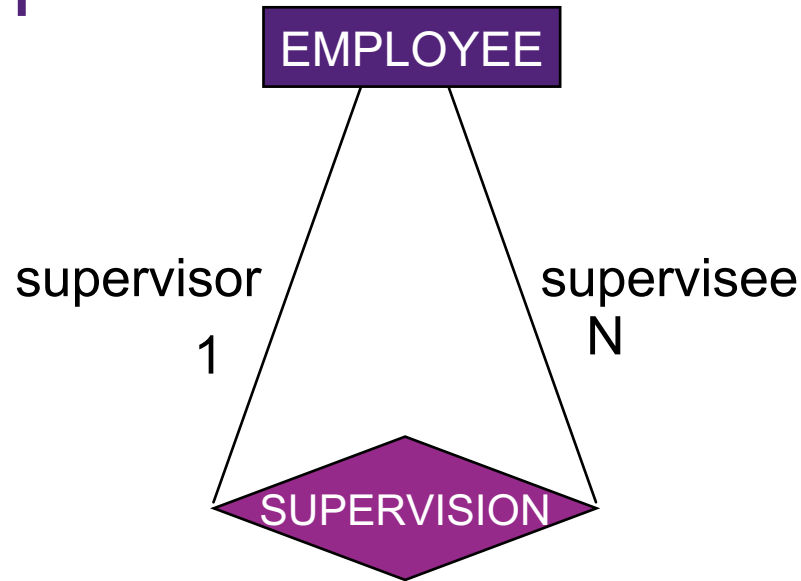
Binary 1:N relationships in the Company Database: WORKS_FOR, CONTROLS and SUPERVISES



Project [pNo, pName, pLocation, **dNumber**]

- *Project.dNumber references Department.dNumber*
- Where the primary key of the DEPARTMENT relation is included as a foreign key in the PROJECT relation

Step 4: Example



- Employee [ssn, fName, mlt, lName, dob, address, sex, salary, dNumber, **superSSN**]
 - **Employee.superSSN references Employee.ssn**
- Where the primary key of the EMPLOYEE relation is included as a foreign key within the EMPLOYEE relation (called superSsn)
 - *Note the recursive relationship!*

Schema (in Progress)

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]

Foreign Keys:

Employee.dNumber references Department.dNumber

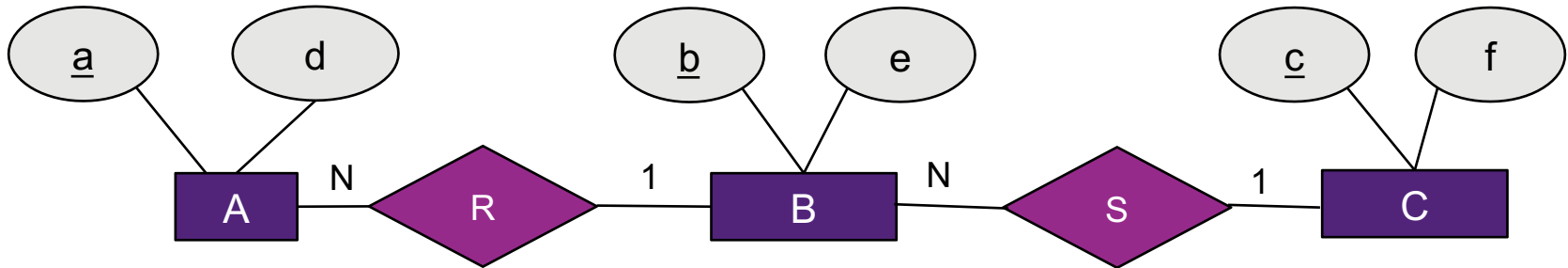
Employee.superSSN references Employee.ssn

Department.mgrSSN references Employee.ssn

Project.dNumber references Department.dNumber

Dependent.ssn references Employee.ssn

Question: Relationship Mapping



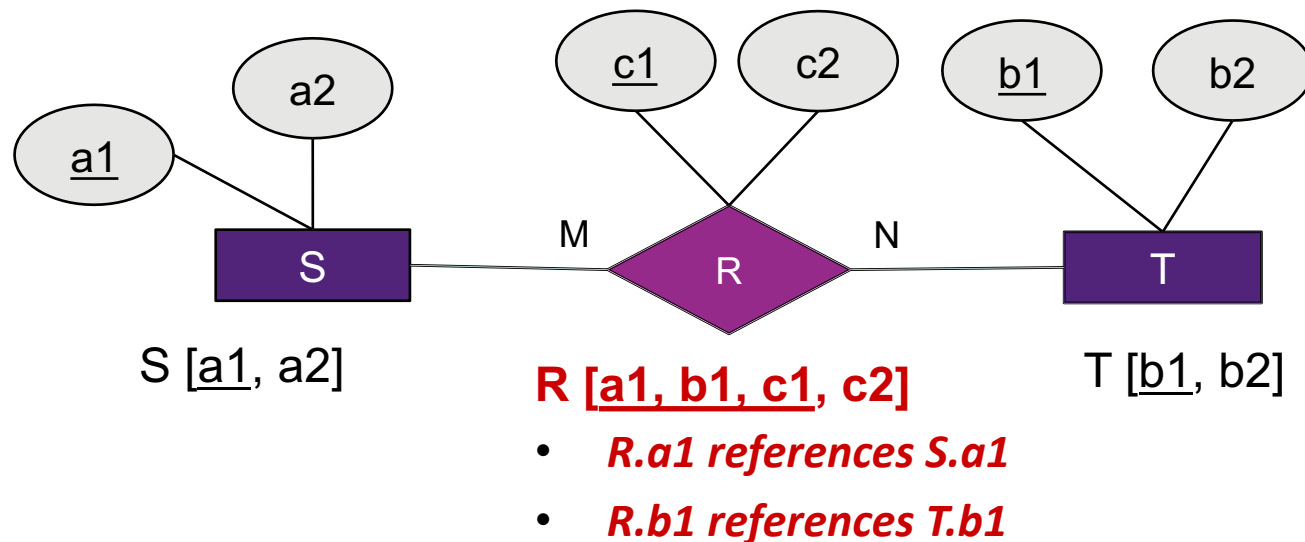
Translate the ER diagram to relational schema. Which of the following appears in your relational schema?

- 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

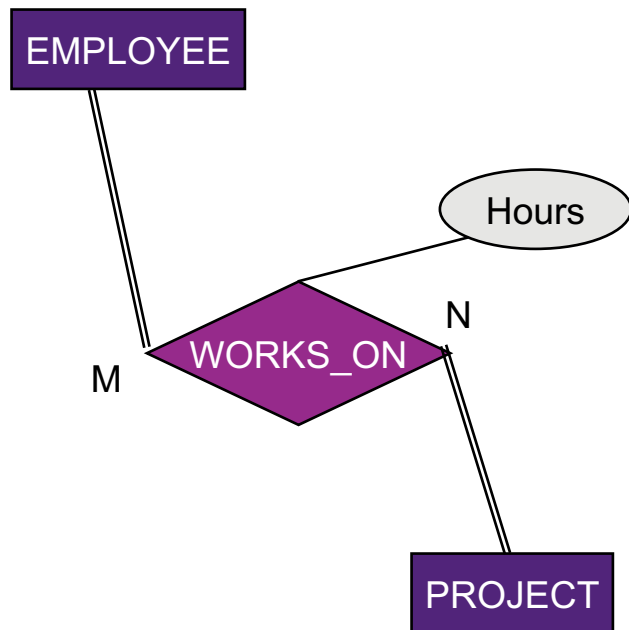
For each binary M:N relationship R, **create a new relation** R to represent it

- Include as **foreign keys** the primary keys of the relations that represent the participating entity types in R
- The combination of foreign keys will form the **primary key** of R
- R can have its own attributes that contribute to the primary key
- Include any simple attributes of R as attributes of the new relation.



Step 5: Example

Binary M:N relationships in the Company Database: **WORKS_ON**



WorksOn [ssn, pNo, hours]

- *WorksOn.ssn references Employee.ssn*
- *WorksOn.pNo references Project.pNo*
- Where **WORKS_ON** includes the primary keys of **PROJECT** and **EMPLOYEE** as foreign keys
- **Hours** in **WORKS_ON** represents the attribute of the relationship type

Schema (in progress)

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]

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 that 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

<u>PK2</u>	...	PK1 as FK	<u>PK1</u>	...
		null		
		null	X	
A		X		
		null		
B		Y		
		null	Y	
C		Y		

Standard Implementation

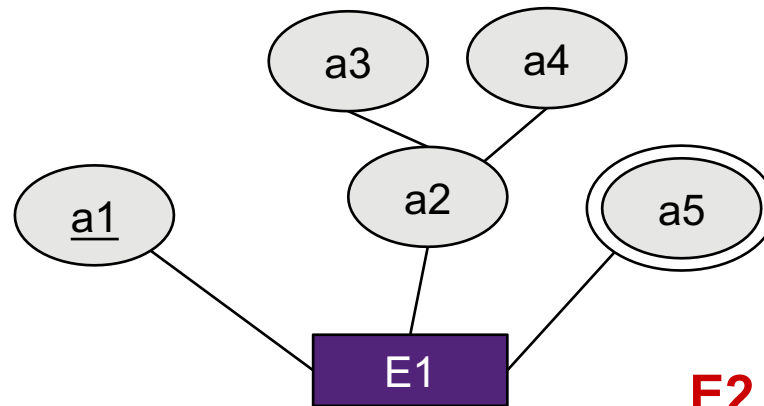
<u>PK2</u>	...	<u>PK1</u>	...	<u>PK2</u>	<u>PK1</u>
				A	X
		X		B	Y
A				C	Y
B					
		Y			
C					

M:N Implementation

Step 6: Multivalued Attributes

For each multivalued attribute A, create a new relation R that includes an attribute corresponding to A plus the primary key K (as a foreign key of 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 attributes A & K
- If the multivalued attribute is composite, include its simple components



E1 [a1, a3, a4]

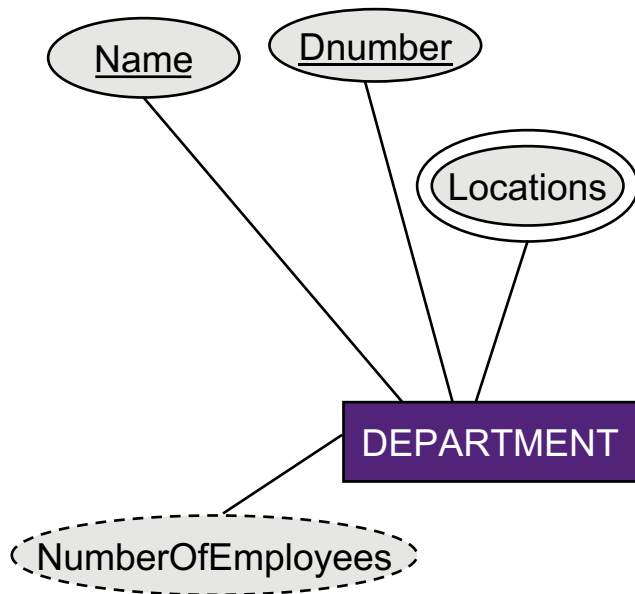
E2 [a1, a5]

- ***E2.a1 references E1.a1***

Step 6: Example

There is only one multivalued attributes in the Company Database:

Locations



DeptLocs [dNumber, location]

- *DeptLocs.dNumber references Department.dNumber*
- Where DeptLocs includes the primary key of DEPARTMENT as a foreign key.
- Location would also be included in the primary key

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]

WorksOn [ssn, pNo, hours]

DeptLocs [dNumber, location]

Foreign Keys:

Employee.dNumber references Department.dNumber

Employee.superSSN references Employee.ssn

Department.mgrSSN references Employee.ssn

Project.dNumber references Department.dNumber

WorksOn.Ssn references Employee.Ssn

WorksOn.pNo references Project.pNo

Dependent.ssn references Employee.ssn

DeptLocs.dNumber references Department.dNumber

Step 7: N-ary Relationships

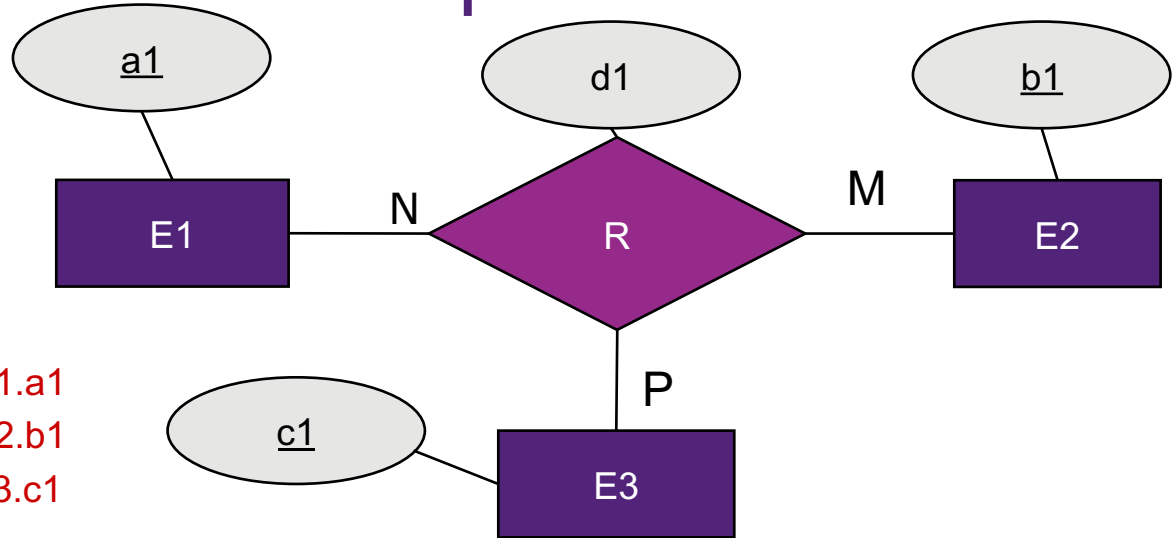
E1 [a1]

E2 [b1]

E3 [c1]

R [a1, b1, c1, d1]

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



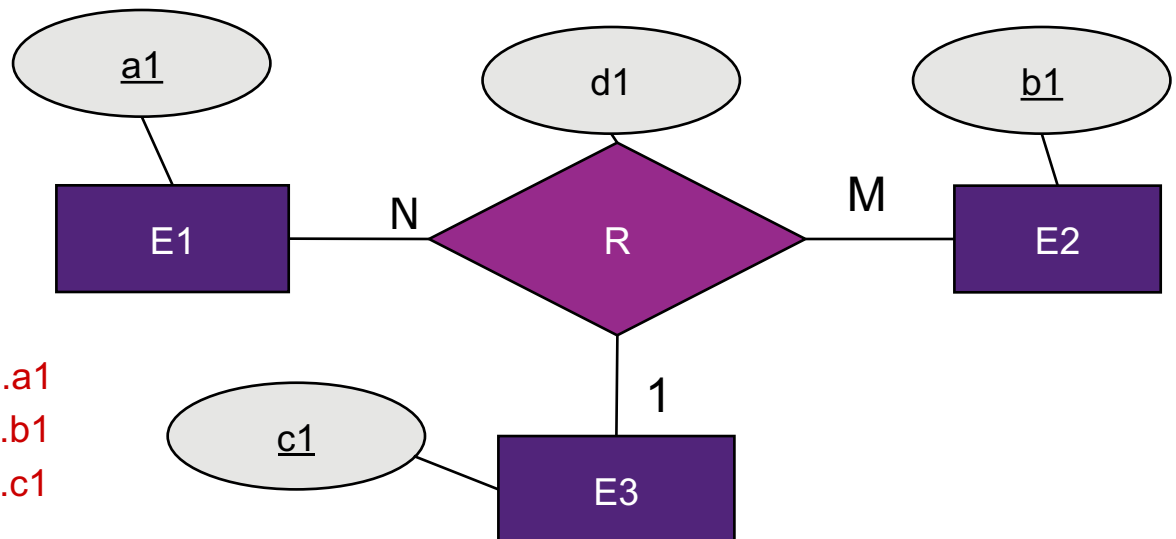
E1 [a1]

E2 [b1]

E3 [c1]

R [a1, b1, c1, d1]

- R.a1 references E1.a1
- R.v1 references E2.b1
- R.c1 references E3.c1



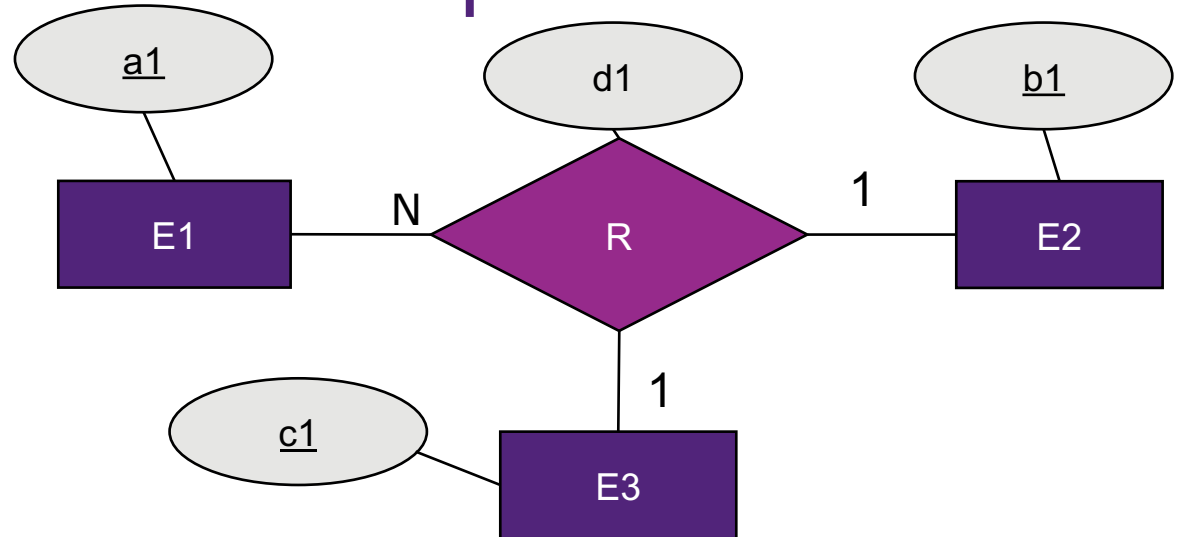
Step 7: N-ary Relationships

E1 [a1, b1, c1, d1]

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

E2 [b1]

E3 [c1]

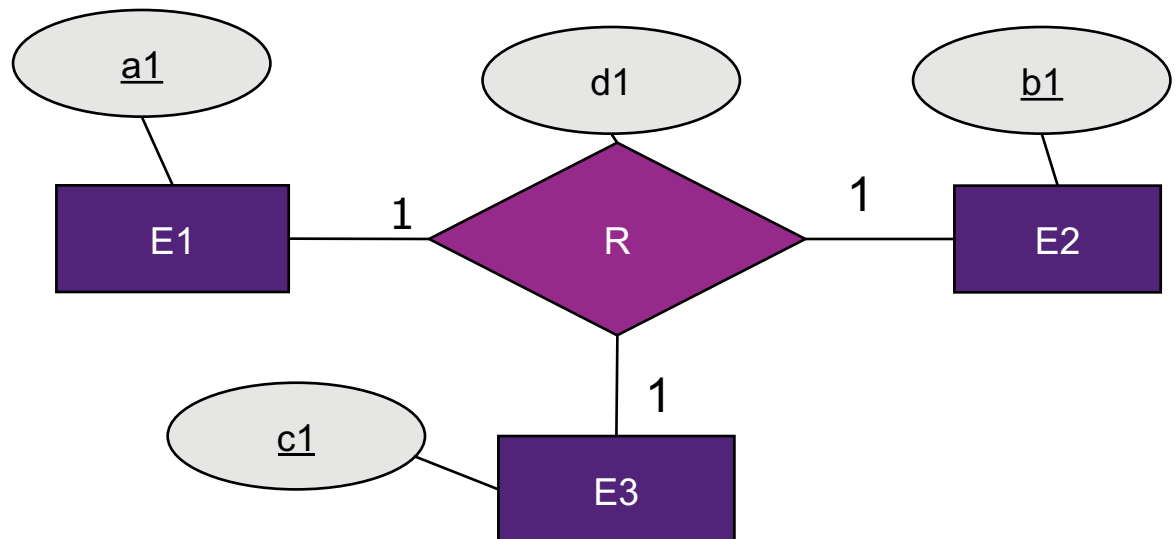


E1 [a1, ..., b1, c1, d1]

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

E2 [b1]

E3 [c1]

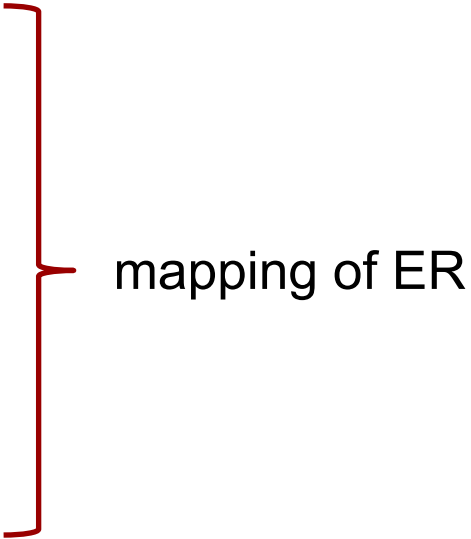


Review of 7-Steps for ER Mapping

Input: an ER model

Output: relations with primary/foreign key constraints

Steps:

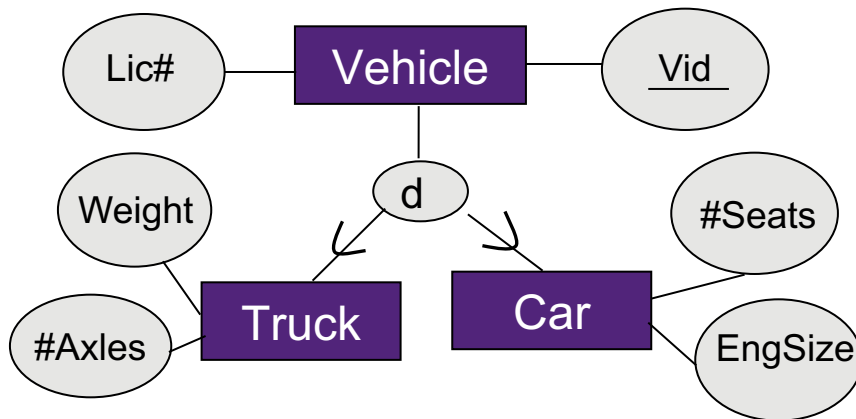
1. Entity Mapping
 2. Weak Entity Mapping
 3. Binary 1:1 Relationship Mapping
 4. Binary 1:N Relationship Mapping
 5. Binary M:N Relationship Mapping
 6. Multi-valued Attribute Mapping
 7. N-ary Relationship Mapping
 8. Super & Sub-classes (mapping of EER)
- 
- mapping of ER

Step 8: Super & Subclasses

Option 8A: Multiple relations - superclass and subclasses

We create a **relational table for the superclass** and create a **relational table for each subclass**.

The primary key of each of the subclasses is the primary key of the superclass.



Vehicle [vid, Lic#]

Truck [vid, weight, #Axles]

- Truck.vid → Vehicle.vid

Car [vid, #Seats, engSize]

- Car.vid → Vehicle.vid

Works for:

- **Disjoint/Overlapping**
- **Total/Partial**

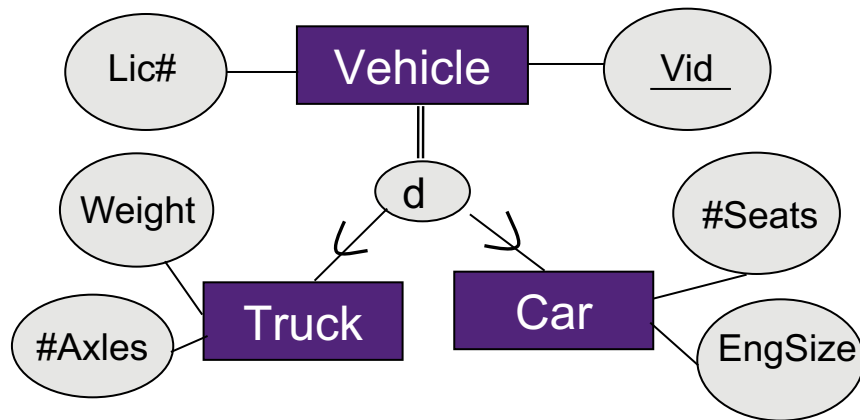


Step 8: Super & Subclasses

Option 8B: Multiple relations - subclass relations only

We create **a relational table for each subclass**. The attributes of the superclass are merged into each of the subclasses.

The primary key of the subclass table is the primary key of the superclass.



Truck [vid, Lic#, weight, #Axles]

Car [vid, Lic#, #Seats, engSize]



Works for **Disjoint Total** subclasses



Does not work for:

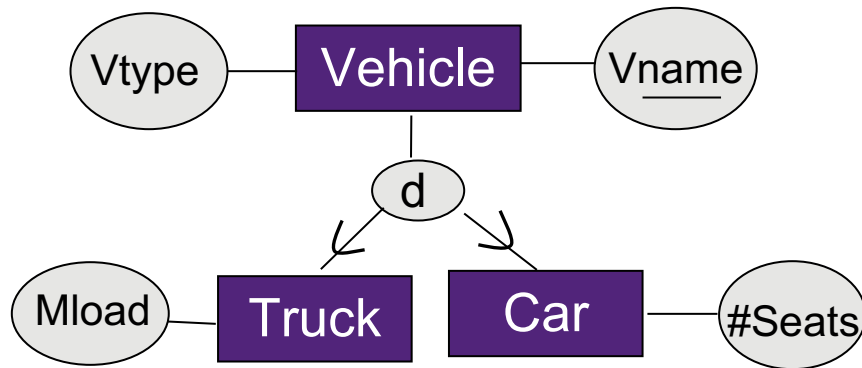
- **Overlapping:** redundancy
- **Partial:** may lose superclass entities not in any subclass

Step 8: Super & Subclasses

Option 8C: Single relation with one type attribute.

We create a **single relational** table for all subclasses and the superclass.

The attributes of the table is the union of all attributes plus the attribute T to indicate the subclass to which each tuple belongs. T is NULL in tuples that do not belong to any subclass (for partial constraints).



Vehicle [vName, vType, mLoad, #Seats, t]



Works for **Disjoint** subclasses that are **Total/Partial**



Does not work for **Overlapping**
Introduces a lot of NULLS

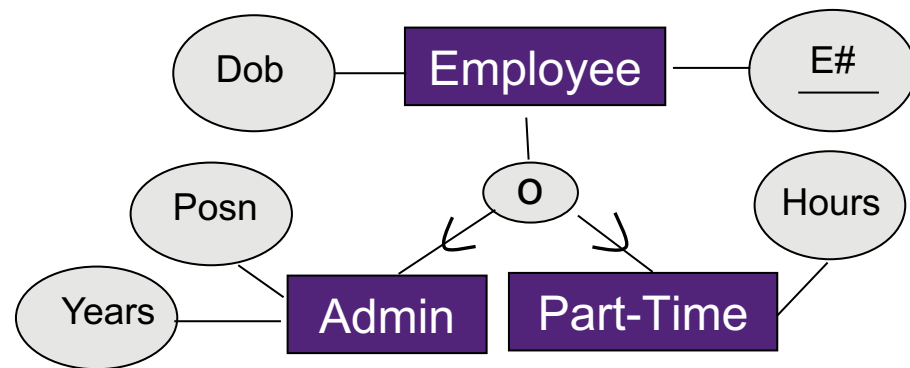
Step 8: Super & Subclasses

Option 8D: Single relation with multiple type attributes.

We create a **single relational** table for all subclasses and the superclass.

The attributes of the table is the union of all attributes plus **m** extra Boolean attributes for each subclass to indicate whether or not the tuple belongs to this subclass.

Employee [e#, dob, t1, posn, years, t2, hours]



Works for **Overlapping** subclasses that are **Total/Partial**



- Introduces a lot of NULLS
- Not recommended if many attributes in subclasses.

Another Example

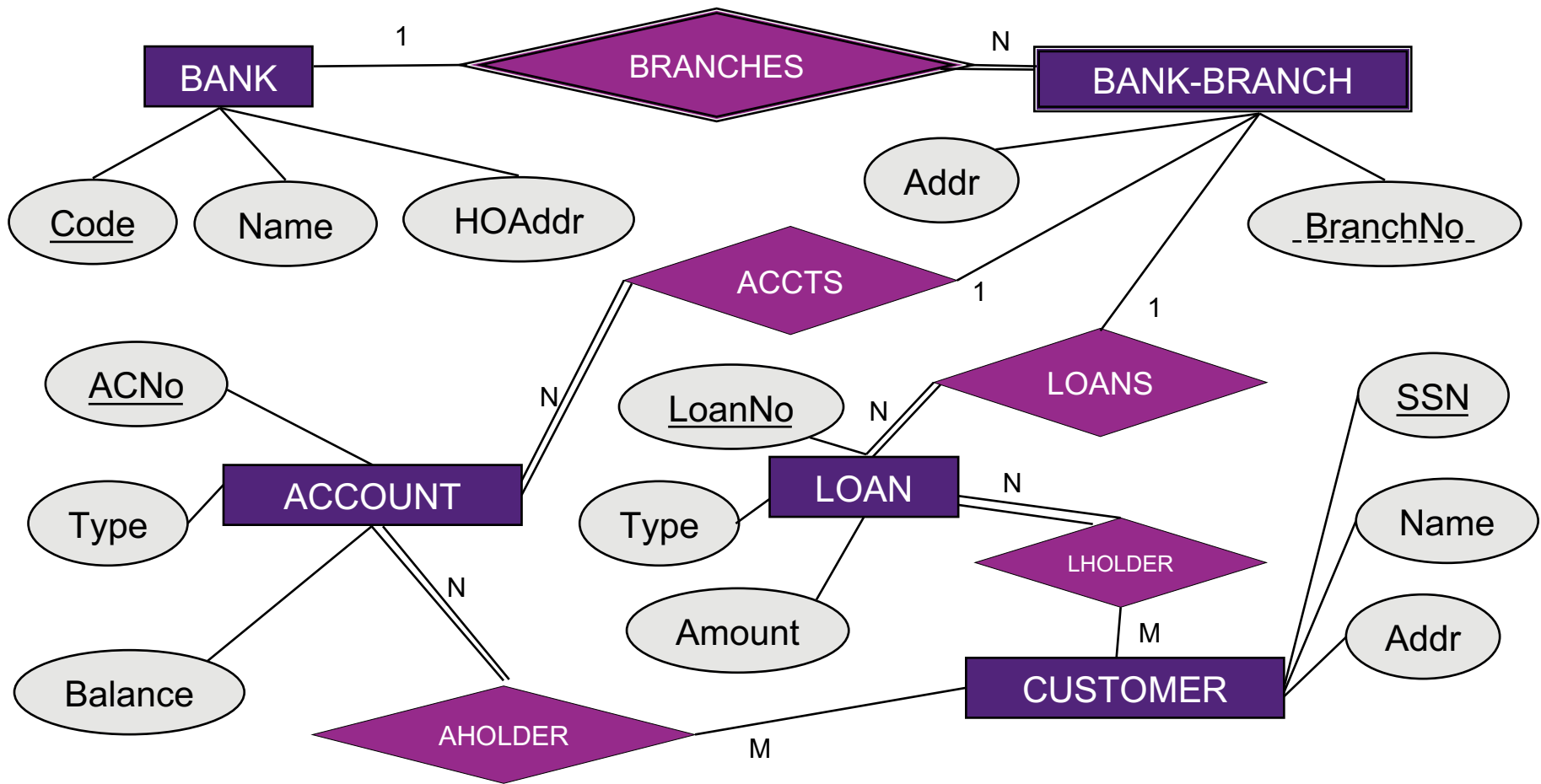
A bank, given by its unique code, name and head office address, can have several branches. Each branch within a given bank has a branch number and address

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



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 Bank.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