

INFS7901

Database Principles

The Relational Model

Rocky Chen
tong.chen@uq.edu.au

Schemas and Instances

Most data models have the concept of “schema” and “instance”

- A **Schema** is the meta-data, or data describing data
- Schema is specified during database design, and is not expected to change frequently

- An **Instance** is the data in the database at a particular time
- Instances are created during data updates and change frequently

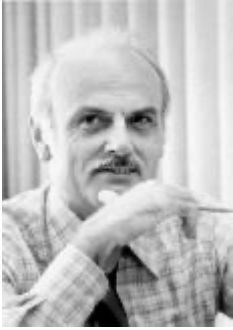
The Relational Model

Implementation of Constraints

Mapping ER Diagrams to Relational Models

Mapping Example

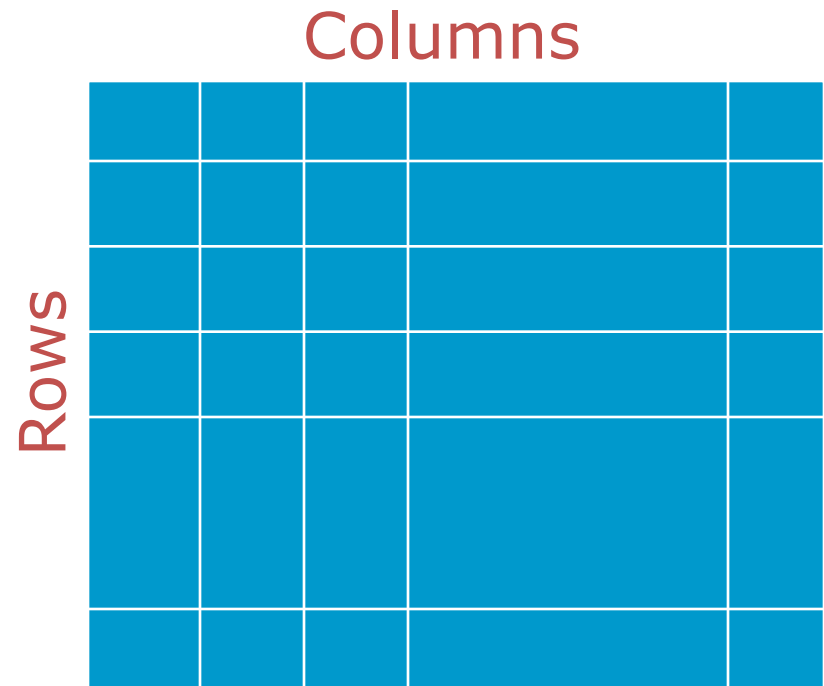
The Relational Model



- Introduced by Edgar Codd (IBM) in 1970
- Most widely used model today.
 - Vendors: IBM, Informix, Microsoft, Oracle, Sybase
- The Relational Model has four main concepts:
 - Relations
 - Domains
 - Attributes
 - Tuples

Relations, Informally

- 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

Domains

- A Domain D is a set of **atomic** values
- Each domain has a data type or format.
- Example: Auto registration numbers
 - 6 characters (either alpha or digits but no 'Q's allowed)
- Popular Domain Types
 - integers
 - real numbers
 - fixed or variable length character strings
 - date
 - time stamp
 - currency
 - sub-range from a data type, e.g. $1 \leq \text{Grade} \leq 7$
 - enumerated data type, e.g. {'Male', 'Female'}

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: StudentNo is an attribute name
(Each *value* of the attribute StudentNo must belong to the *domain* of StudentNos)

Name	StudentNo	Sex	Degree
J. Smith	606 567 333	M	BSc
A. Brown	321 638 999	F	BInfTch

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”
- t is called an n -tuple
- Example
 - (254, John, Smith, \$75K, Snr. Engineer, M)

Relation Schema and Instance

- Relation Schema

- A relation schema r is a set of attributes (a_1, a_2, \dots, a_n) where a_i is in D_i , the domain (set of allowed values) of the i -th attribute.
- Attribute values are atomic, i.e., integers, floats, strings
- A domain contains a special value **null** indicating that the value is not known.

- Relation Instance

- A relation instance r of the relation schema R is a set of n -tuples $r = \{t_1, t_2, \dots, t_m\}$. Each n -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 Domain (D_i) or is a special NULL value.

Relation Schema and Instance Example

- Relation Schema Example
 - Student [sid: integer, name: string, address: string, phone: string, major: string] or
 - Student [sid, name, address, phone, major]
- Relation Instance example

Student

sid	name	address	phone	major
99111120	G. Jones	1234 W. 12 th Ave., Van.	889-4444	CPSC
92001200	G. Smith	2020 E. 18 th St., Van	409-2222	MATH
94001020	A. Smith	2020 E. 18 th St., Van	222-2222	CPSC
94001150	S. Wang	null	null	null

Relation Schema and Instance Example

- Relational Schema Example

Movie (MovieID, Title, Year)
StarsIn (MovieID, StarID, Role)
MovieStar (StarID, Name, Gender)

- Relation Instance example

MovieID	Title	Year
1	Star Wars	1977
2	Gone with the Wind	1939
3	The Wizard of Oz	1939
4	Indiana Jones and the Raiders of the Lost Ark	1981

MovieID	StarID	Role
1	1	Han Solo
4	1	Indiana Jones
2	2	Scarlett O'Hara
3	3	Dorothy Gale

StarID	Name	Gender
1	Harrison Ford	Male
2	Vivian Leigh	Female
3	Judy Garland	Female

Clicker Question

Here is a table representing a relation instance named R. Which of the following is a true statement about R?

A	B	C
0	1	2
3	4	5
6	7	8
9	10	11

- A. R has four tuples.
- B. (6,7,8) is a tuple of R.
- C. The schema of R is R(A,B,C).
- D. All of the above

Clicker Question

Here is a table representing a relation instance named R. Which of the following is a true statement about R?

A	B	C
0	1	2
3	4	5
6	7	8
9	10	11

A. R has four tuples.

B. (6,7,8) is a tuple of R.

C. The schema of R is R(A,B,C).

D. All of the above

D All are true

History and the Three Schema Architecture

The Relational Model

Implementation of Constraints

Mapping ER Diagrams to Relational Models

Mapping Example

Implementation of Constraints

- **Integrity Constraints (ICs)** are conditions that must be true for any instance of the database; e.g.,
domain constraints
 - ICs are specified when schema is defined
 - ICs are checked when relations are modified
- A **legal** instance of a relation is one that satisfies all specified ICs
 - DBMS should not allow illegal instances
 - Avoids data entry errors, too!

Where do ICs Come From?

- ICs are based upon the real-world semantics being described (in the database relations).
- We *can* check a database instance to verify an IC, but we *cannot* tell the ICs by looking at the instance.
 - For example, even if all student names differ, we cannot assume that name is a key.
 - An IC is a statement about *all possible* instances.
- All constraints must be identified during the conceptual design.
- Some constraints can be explicitly specified in the conceptual model
 - For example, keys are shown on ER diagrams.
- Others are written in a more general language.

Integrity Constraint Types

- Domain constraints
- Key constraints
- Entity constraints
- Referential integrity constraints
- User-defined constraints

Domain Constraints

- Each attribute in a relation must belong to some domain.

StarID	Name	Gender
1	Harrison Ford	Male
2	Vivian Leigh	Female
3	Judy Garland	5

- This instance violates the domain constraint of Gender, which is {Male, Female, other}

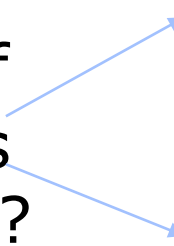
Integrity Constraint Types

- Domain constraints
- **Key constraints**
- Entity constraints
- Referential integrity constraints
- User-defined constraints

Key Constraints

- All tuples in a relation must be distinct, that is no two tuples can have same values for all attributes --
→ uniqueness constraint

Violation of
Uniqueness
Constraint ?



StarID	Name	Gender
1	Harrison Ford	Male
3	Judy Garland	Female
2	Vivian Leigh	Female
1	Harrison Ford	Male

Notion of a Superkey

- A Superkey (SK) is a subset of attributes of a relation schema R , such that for any two tuples, t_i and t_j in a relation instance r of R
 - $t_i[\text{SK}] \neq t_j[\text{SK}]$
- Every relation has **at least one** superkey - the set of all its attributes
- Superkey can have **redundant attributes**, that is, by removing some attributes, the uniqueness constraint is still maintained

Clicker Question

Assuming that StarIDs are unique, which of the following is a superkey for the MovieStar relation?

- A. (Name, Gender)
- B. (StarID, Gender)
- C. (StarID, Name)
- D. Both B and C

StarID	Name	Gender
1	Harrison Ford	Male
3	Judy Garland	Female
2	Vivian Leigh	Female

Clicker Question

Assuming that StarIDs are unique, which of the following is a superkey for the MovieStar relation?

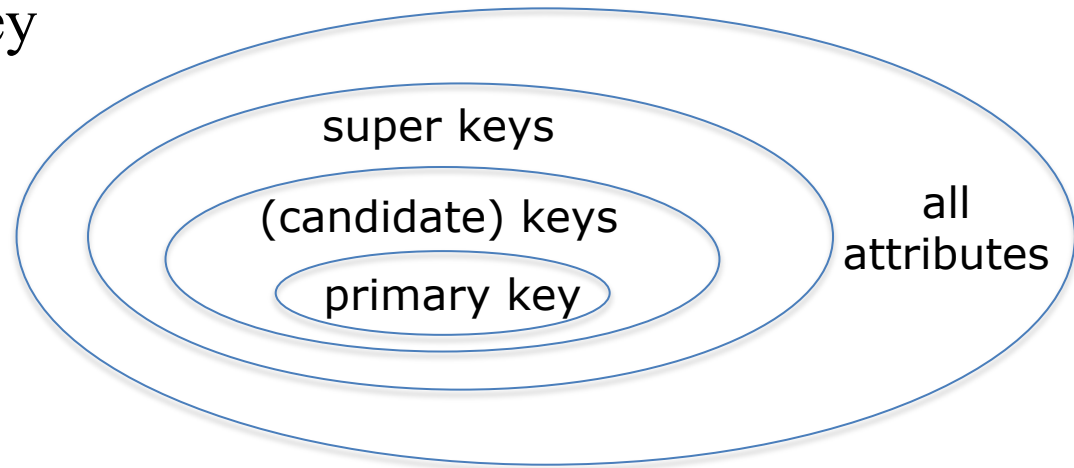
- A. (Name, Gender)
- B. (StarID, Gender)
- C. (StarID, Name)
- D. Both B and C

Correct

StarID	Name	Gender
1	Harrison Ford	Male
3	Judy Garland	Female
2	Vivian Leigh	Female

Notion of a Key

- Key is a minimal Superkey
 - Minimal: Removing any attribute means the proposed key is no longer a Superkey



- Each individual key may have multiple attributes.

StarID would be key

StarID	Name	Gender
1	Harrison Ford	Male
3	Judy Garland	Female
2	Vivian Leigh	Female

Characteristics of Keys

- **Uniqueness:** Value of key attributes uniquely identify a tuple in a relation
- **Key constraints** hold on every relation instance
 - Name cannot always be used as key
- Each individual key **may have multiple attributes.**
- A schema may have more than one key
 - Each is called a “**candidate**” key
 - One is selected as the “**primary**” key

Integrity Constraint Types

- Domain constraints
- Key constraints
- Entity constraints
- Referential integrity constraints
- User-defined constraints

Entity Integrity Constraint

- No primary key can be null
 - How would you distinguish between **Emma Watson** and **Emily Watson**

StarID	Name	Gender
?	E Watson	Female
?	E Watson	Female
2	Vi Leigh	Female
1	H Ford	Male

Integrity Constraint Types

- Domain constraints
- Key constraints
- Entity constraints
- Referential integrity constraints
- User-defined constraints

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

- Foreign keys allow us to relate two different schemas (e.g., R1, R2)
- A set of attributes FK in relation schema R1 is a foreign key if
 - the attributes of FK have the same domain as the the primary key attributes PK of another schema R2
 - $t_1[FK] = t_2[PK]$ or $t_1[FK]$ is null
- Referential integrity: All foreign keys reference existing entities.
 - i.e. there are no dangling references
 - all foreign key constraints are enforced

Referential Integrity Example

- CustomerID in Order references Customer
- Only customers listed in the Customer relation/table should be allowed in Order.

Order

OrderID	CusID	Product
O1	1	Coke
O4	1	7 Up
O2	2	Sprite
O3	3	Pepsi

Customer

CusID	Name	Gender
1	Harrison Ford	Male
2	Vivian Leigh	Female
3	Judy Garland	Female



CusID is the foreign key in Order, and it is referencing the Customer table.

Self Referencing Relations

- Goal: have managerID be foreign key reference for same table Emps.

id	sin	name	managerID
1	1000	Jane	Null
2	1001	Jack	1

- Again, why foreign keys can be NULL?
 - For **referential integrity** to hold in a relational database, any field in a table that is declared a foreign key should contain either a **NULL** value, or only values from a parent table's **primary key**.

$$t_1[\text{FK}] = t_2[\text{PK}] \quad \text{or} \quad t_1[\text{FK}] \text{ is NULL}$$

Integrity Constraint Types

- Domain constraints
- Key constraints
- Entity constraints
- Referential integrity constraints
- User-defined constraints

User-Defined Constraints

- General user defined constraints that cannot be enforced by the other constraints
- Implemented by using Checks, Assertions and Triggers. These are not covered in detail in this course
- Example: For better movie quality, no actors can play in more than 10 movies in one year

Movie (MovieID, Title, Year)
StarsIn (MovieID, StarID, Role)
MovieStar (StarID, Name, Gender)

In-class Exercise

Use this relational schema (**foreign keys in bold font**)

- STUDENT (StID, Name, Email)
- COURSE (CCode, Title, Units)
- ENROLMENT (**StID**, **CCode**, Sem, Year)

To give examples of the following:

- Super Key
- Minimal Key
- Foreign Key
- Domain Constraint

In-class Exercise - Solution

Use this relational schema (foreign keys in bold font)

- STUDENT (StID, Name, Email)
- COURSE (CCode, Title, Units)
- ENROLMENT (**StID**, **CCode**, Sem, Year)

To give examples of the following:

- Super Key StID, Name, Email in relation Student
- Minimal Key StID in relation Student
- Foreign Key StID in relation Enrolment
- Domain Constraint Ccode 4 letters followed by 4 num

History and the Three Schema Architecture

The Relational Model

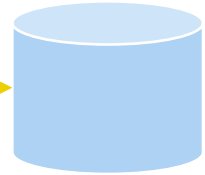
Implementation of Constraints

Mapping ER Diagrams to Relational Models

Mapping Example



Conceptual
perspective



CONCEPTUAL
DESIGN

The Entity
Relationship (ER)
Model is one of
the most widely
used methods for
conceptual design

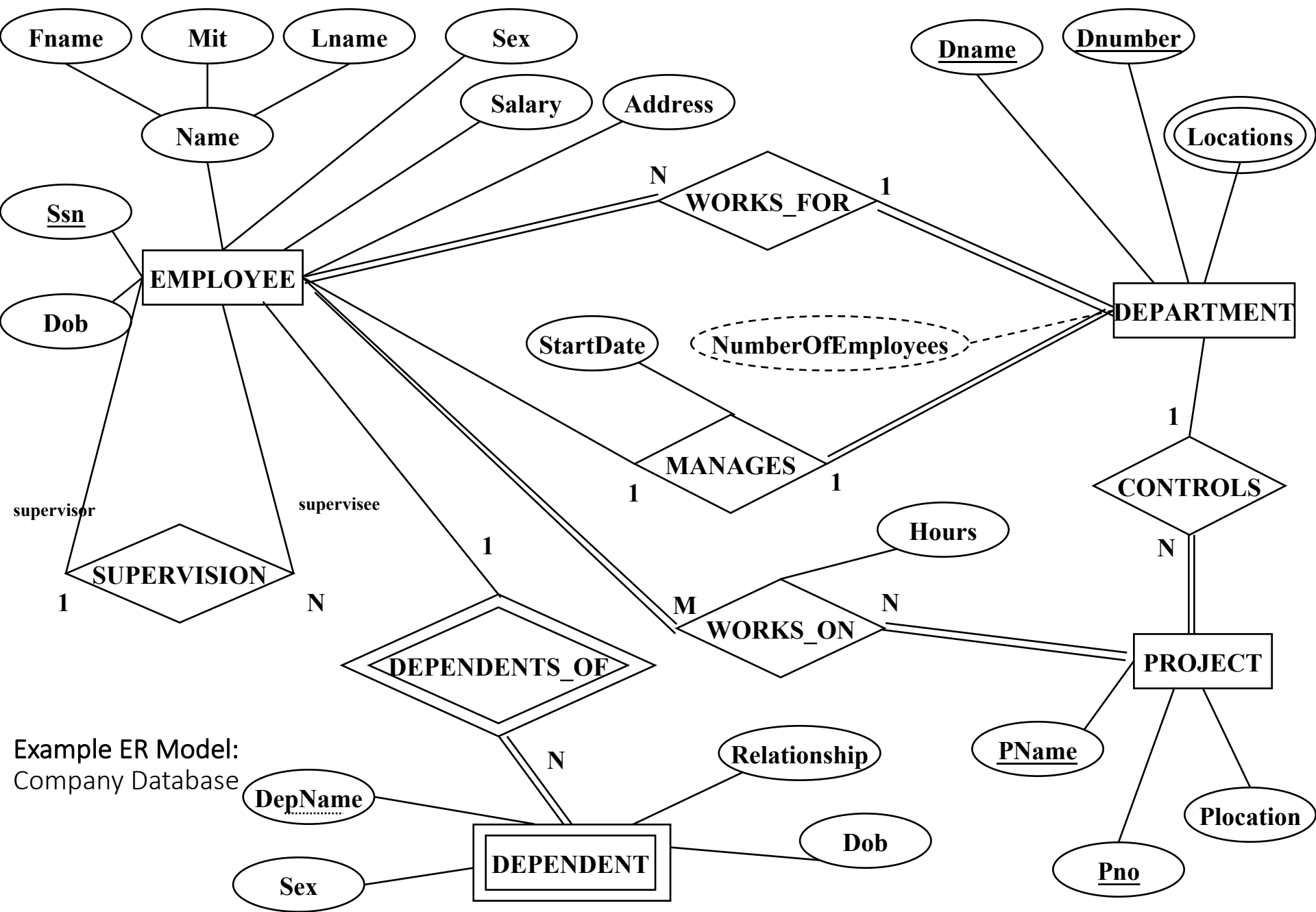
LOGICAL
DESIGN
(MAPPING)

The Relational
Model is the
basis for
several
commercial
DBMSs

**Focus of the rest
of this Module**

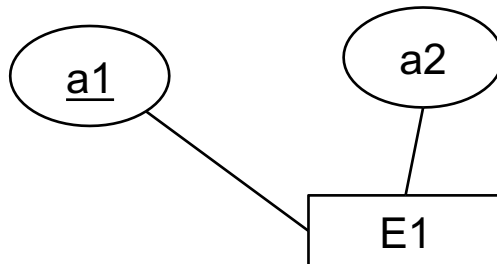
Mapping ER Diagrams to Relational Models

- Method for mapping a conceptual schema developed using the ER model to a relational database schema comprises **8 steps**
- 1. Entity Mapping (create relations)
- 2. Weak Entity Mapping (create relations)
- 3. Binary 1:1 Relationship Mapping (define foreign keys)
- 4. Binary 1: N Relationship Mapping (define foreign keys)
- 5. Binary M:N Relationship Mapping (create relations and define foreign keys)
- 6. Multi-valued Attribute Mapping (create relations and define foreign keys)
- 7. N-ary Relationship Mapping (create relations and define foreign keys)
- 8. Super & Sub-classes (mapping of EER)



Step 1: Entity Mapping

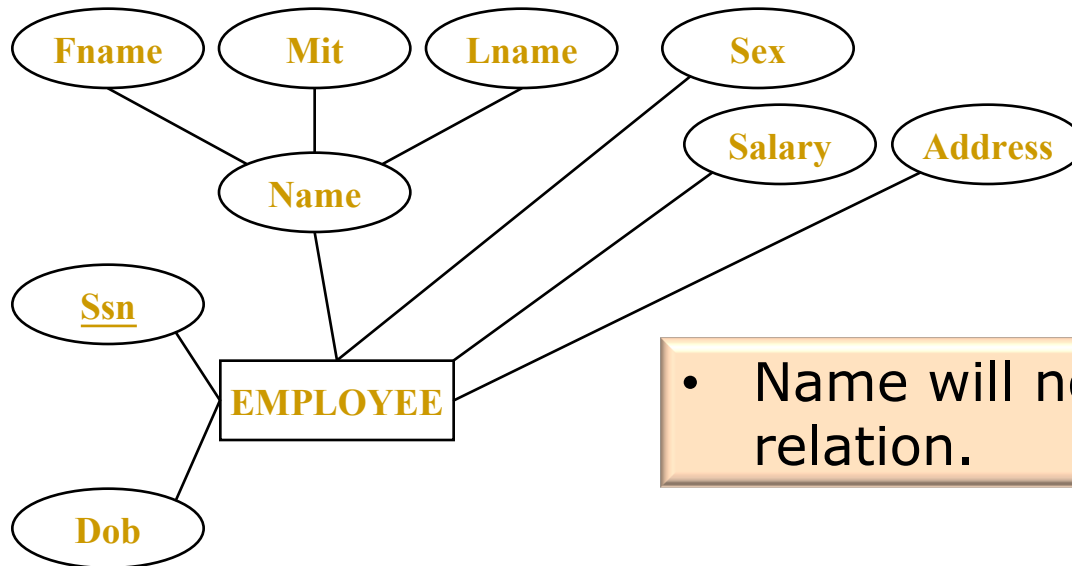
- For each regular (non-weak) entity E, **create a relation R** that includes all simple attributes of E
 - Include only simple component attributes of a composite attribute
 - Choose one key attribute of E as **primary key** for R. If key of E is composite, the set of simple attributes together should form the key
 - Add foreign key attributes in subsequent steps



E1 [a1, a2]

Step 1: Example

Entities in the Company Database: **EMPLOYEE**, **DEPARTMENT**, **PROJECT**

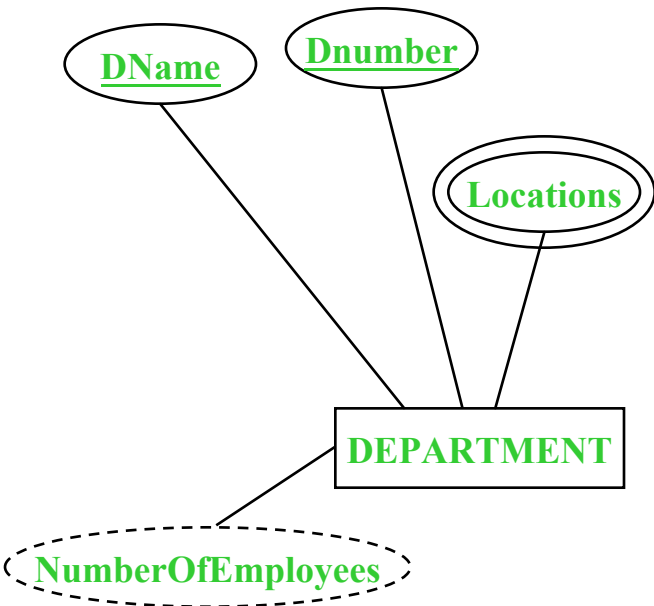


- Name will not be added to the relation.

EMPLOYEE [Ssn, Fname, Mit, Lname, Dob, Address, Sex, Salary]

Step 1: Example

Entities in the Company Database: EMPLOYEE, DEPARTMENT, PROJECT

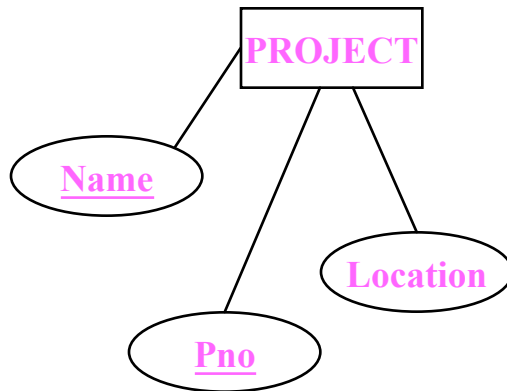


- Location will not be added for the time being
- Both DName and DNumber are keys. DNumber is taken as PK.
- NumberOfEmployees is not added to the relation as it is a derived attribute.

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]

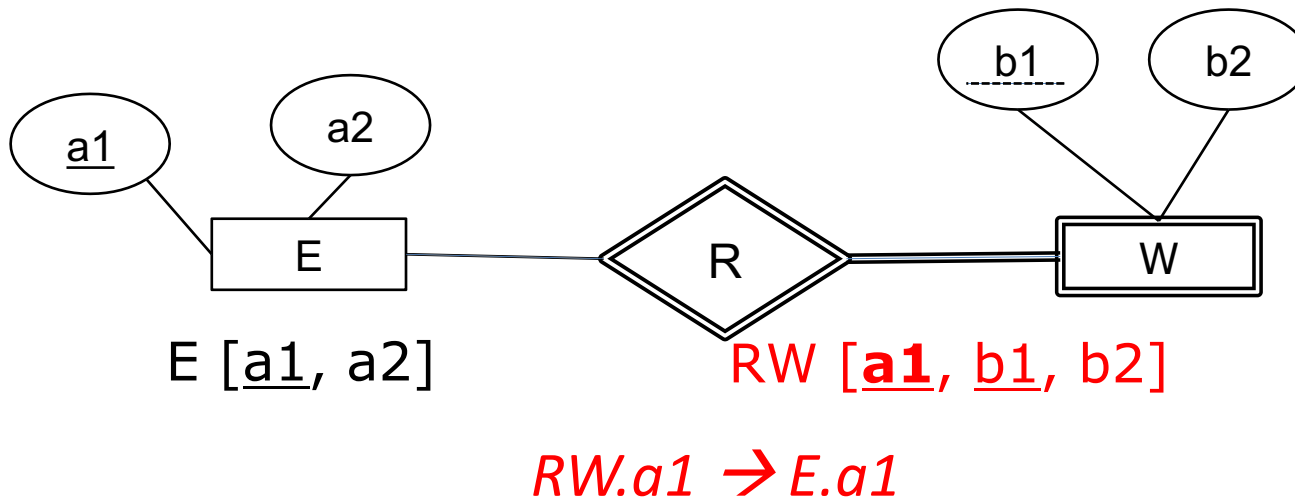
Schema (in progress)

Relations:

- EMPLOYEE [SSN, Fname, Mit, Lname, Dob, Address, Sex, Salary]
- DEPARTMENT [Dnumber, DName]
- PROJECT [Pno, PName, Plocation]

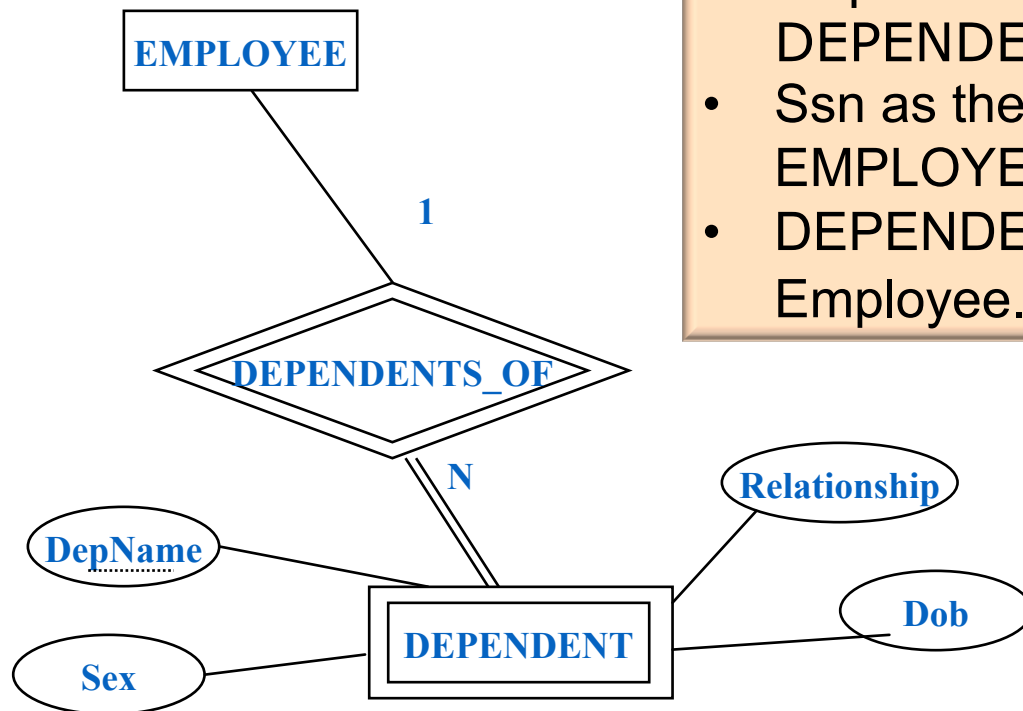
Step 2: Weak Entity Mapping

- For each weak entity W with owner entity E **create a relation (table)** RW that includes all simple attributes of W
 - Include the owner entity's primary key attributes as the **foreign key attributes** of RW w.r.t the relation. (This maps the identifying relationship(s) of W)
 - The **primary key** of RW is the combination of the primary key(s) of the owner(s) and the primary key of the weak entity W (if any)



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 an FK referring to Employee.SSN

DEPENDENT [SSN, DepName, Sex, Dob, Relationship]

- *DEPENDENT.SSN* → *EMPLOYEE.SSN*

Schema (in progress)

Relations:

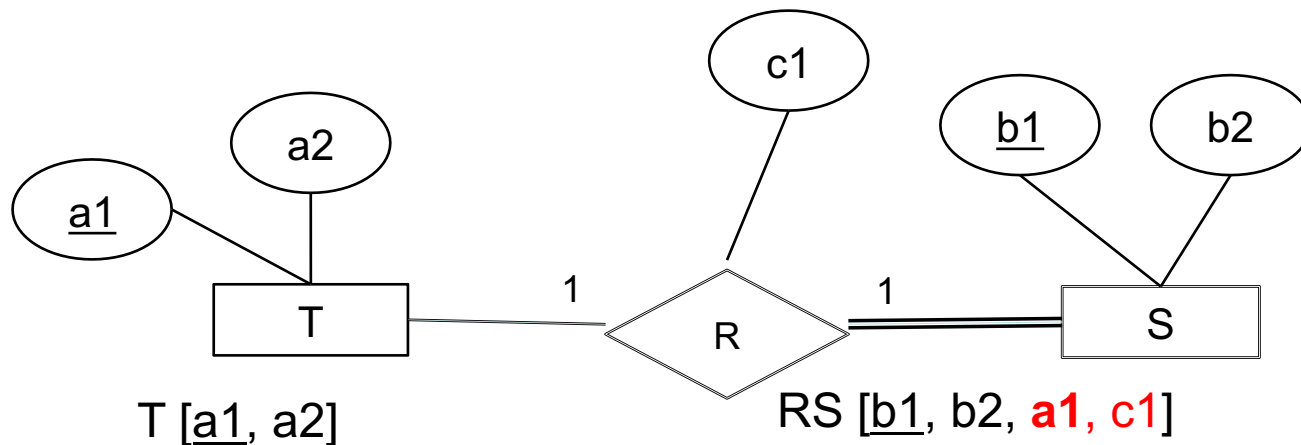
- EMPLOYEE [SSN, Fname, Mit, Lname, Dob, Address, Sex, Salary]
- DEPARTMENT [Dnumber, DName]
- PROJECT [Pno, PName, Plocation]
- DEPENDENT [SSN, DepName, Sex, Dob, Relationship]

Foreign Key:

- DEPENDENT.SSN → EMPLOYEE.SSN

Step 3: Binary 1:1 Relationship

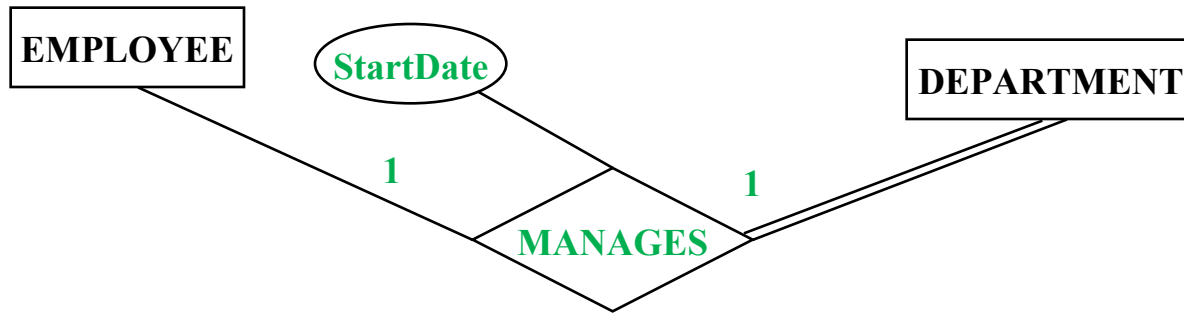
- For each binary 1:1 relationship R, identify relations S & T that correspond to the entities participating in R
 - Choose one relation (say S) and include as the primary key of T as the foreign key in S
 - It is better to choose S – the entity with total participation
 - Include all the simple attributes (or simple components of composite attributes) of the relationship R as attributes of S



RS.a1 → T.a1

Step 3: Example

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



DEPARTMENT [Dnumber, DName, **SSN**, **StartDate**]

- *DEPARTMENT.SSN → EMPLOYEE.SSN*

- Include the primary key of the EMPLOYEE relation as a foreign key in the DEPARTMENT relation (SSN)
- Include the simple attribute StartDate of the MANAGES relation

Schema (in progress)

Relations:

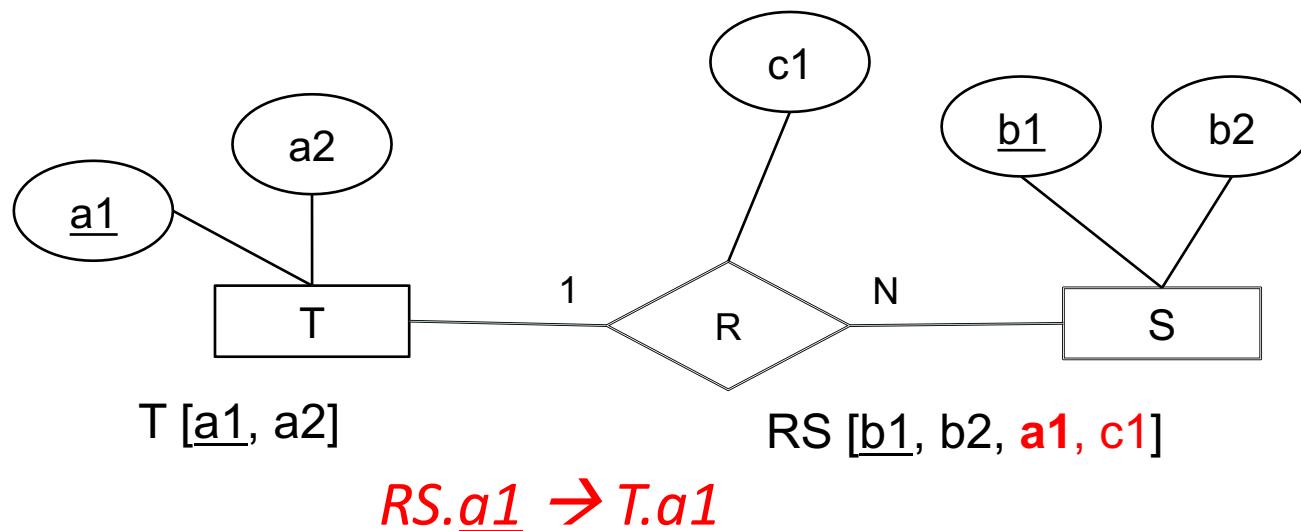
- EMPLOYEE [SSN, Fname, Mit, Lname, Dob, Address, Sex, Salary]
- DEPARTMENT [Dnumber, Dname, **SSN**, StartDate]
- PROJECT [Pno, PName, Plocation]
- DEPENDENT [SSN, DepName, Sex, Dob, Relationship]

Foreign Keys:

- DEPENDENT.SSN → EMPLOYEE.SSN
- DEPARTMENT.SSN → EMPLOYEE.SSN

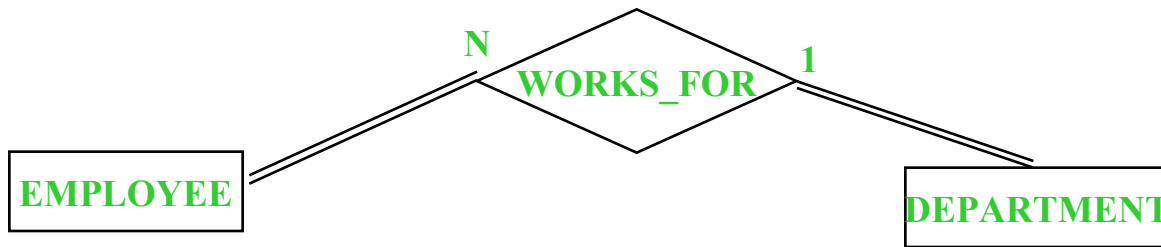
Step 4: Binary 1:N Relationship

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



Step 4: Example

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



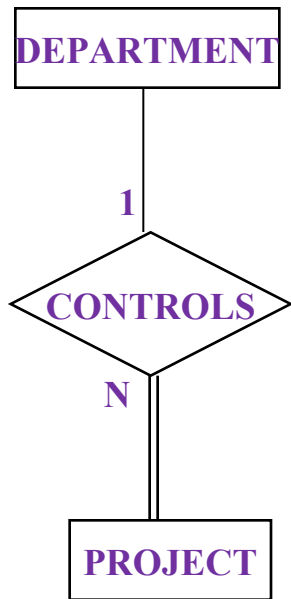
- Where primary key of the DEPARTMENT relation is included as a foreign key in the EMPLOYEE relation

EMPLOYEE [SSN, Fname, Minit, Lname, Dob, Address, Sex, Salary, **Dnumber**]

- *EMPLOYEE.Dnumber* → *DEPARTMENT.Dnumber*

Step 4: Example

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



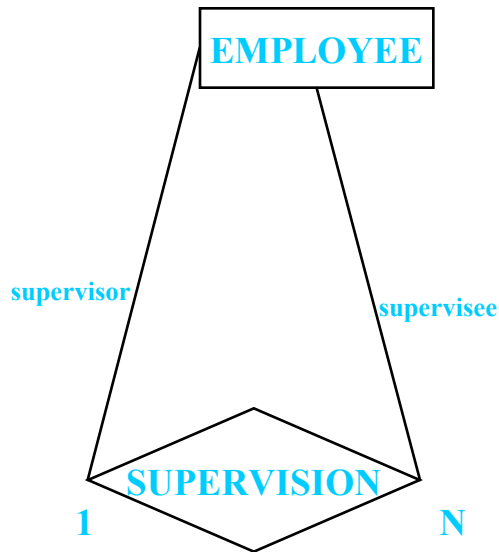
- Primary key of the DEPARTMENT relation is included as a foreign key in the PROJECT relation

PROJECT [Pno, PName, Plocation, **Dnumber**]

- *PROJECT.Dnumber* → *DEPARTMENT.Dnumber*

Step 4: Example

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



- primary key of the **EMPLOYEE** relation is included as a foreign key within the **EMPLOYEE** relation (called SuperSSN)
- *Note the recursive relationship!*

EMPLOYEE [Ssn, Fname, Mit, Lname, Dob, Address, Sex, Salary, **Dno**, **SuperSsn**]

- *EMPLOYEE.SuperSSN → EMPLOYEE.SSN*

Schema (in Progress)

Relations:

- EMPLOYEE [SSN, Fname, Mit, Lname, Dob, Address, Sex, Salary, **Dnumber**, **SuperSSN**]
- DEPARTMENT [Dnumber, Dname, **SSN**, StartDate]
- PROJECT [Pno, PName, Plocation, **DNumber**]
- DEPENDENT [SSN, DepName, Sex, Dob, Relationship]

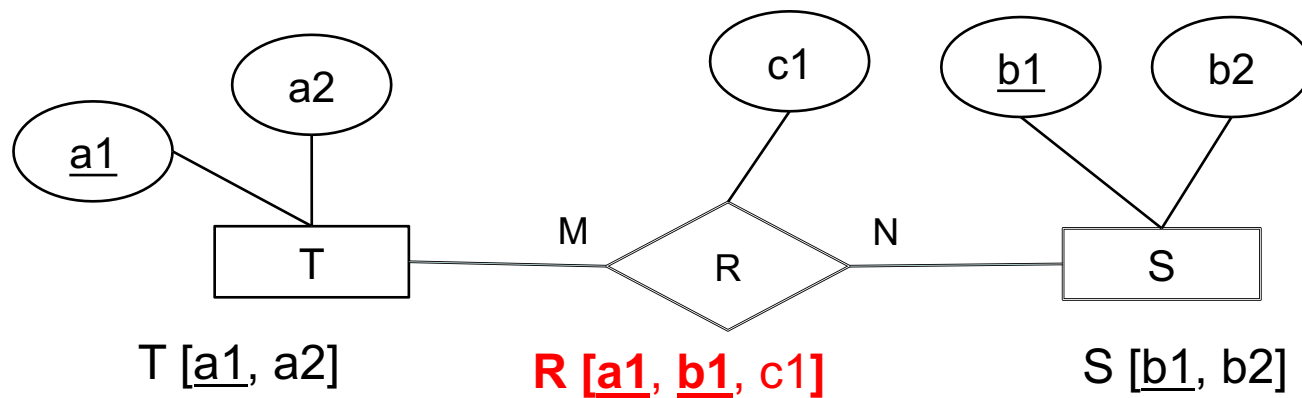
Foreign Keys:

- DEPARTMENT.SSN → EMPLOYEE.SSN
- DEPENDENT.SSN → EMPLOYEE.SSN
- PROJECT.Dnumber → DEPARTMENT.Dnumber
- EMPLOYEE. Dnumber → DEPARTMENT.Dnumber
- EMPLOYEE.SuperSSN → EMPLOYEE.SSN

Step 5: Binary M:N Relationship

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

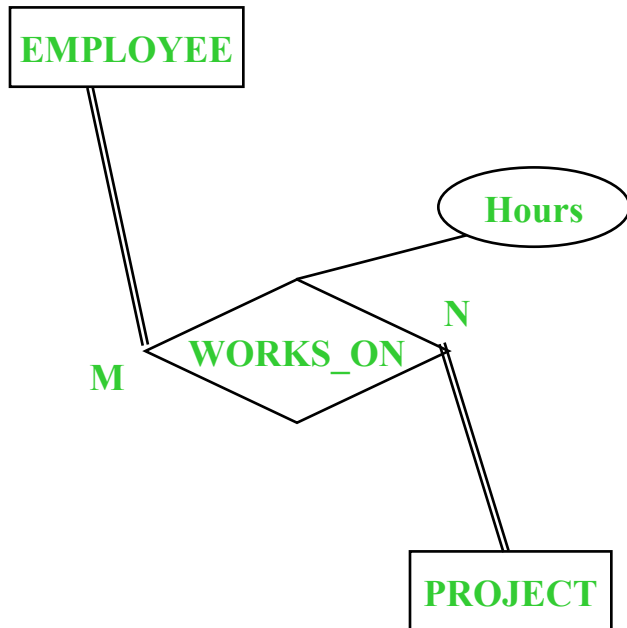
- Include as foreign keys the primary keys of the relations that represent the participating entity in R
- The combination of foreign keys will form the primary key of R
(Note: cannot represent the M:N using a single foreign key in one relation because of the M:N cardinality ratio)
- Include any simple attributes (or simple components of composite attributes) of the M:N relationship as attributes of RS.



R.a1 → T.a1 and R.b1 → S.b1

Step 5: Example

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



- The primary key of **WORKS_ON** is the combination of the **foreign key attributes**

WORKS_ON [**SSN**, **Pno**, Hours]

- *WORKS_ON.SSN* → *EMPLOYEE.SSN*
- *WORKS_ON.Pno* → *PROJECT.Pno*

Schema (in progress)

Relations:

- EMPLOYEE [SSN, Fname, Mit, Lname, Dob, Address, Sex, Salary, **Dnumber**, SuperSSN]
- DEPARTMENT [Dnumber, Dname, SSN, StartDate]
- PROJECT [Pno, PName, Plocation, **Dnumber**]
- DEPENDENT [SSN, DepName, Sex, Dob, Relationship]
- WORKS_ON [**SSN**, **Pno**, Hours]

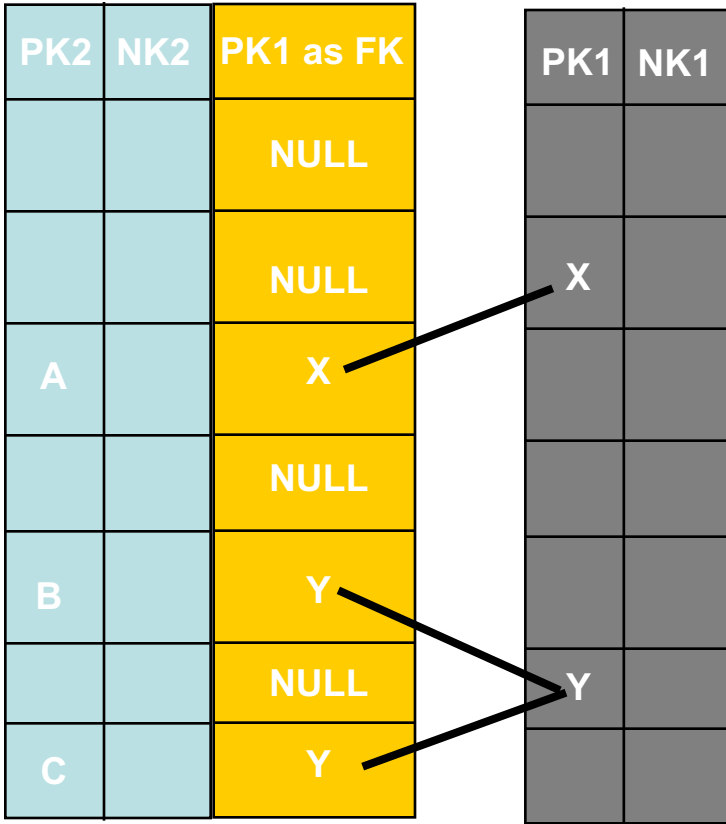
Foreign Keys:

- EMPLOYEE.Dnumber → DEPARTMENT.Dnumber
- EMPLOYEE.SuperSSN → EMPLOYEE.SSN
- DEPARTMENT.SSN → EMPLOYEE.SSN
- PROJECT.Dnumber → DEPARTMENT.Dnumber
- DEPENDENT.SSN → EMPLOYEE.SSN
- WORKS_ON.SSN → EMPLOYEE.SSN
- WORKS_ON.Pno → PROJECT.Pno

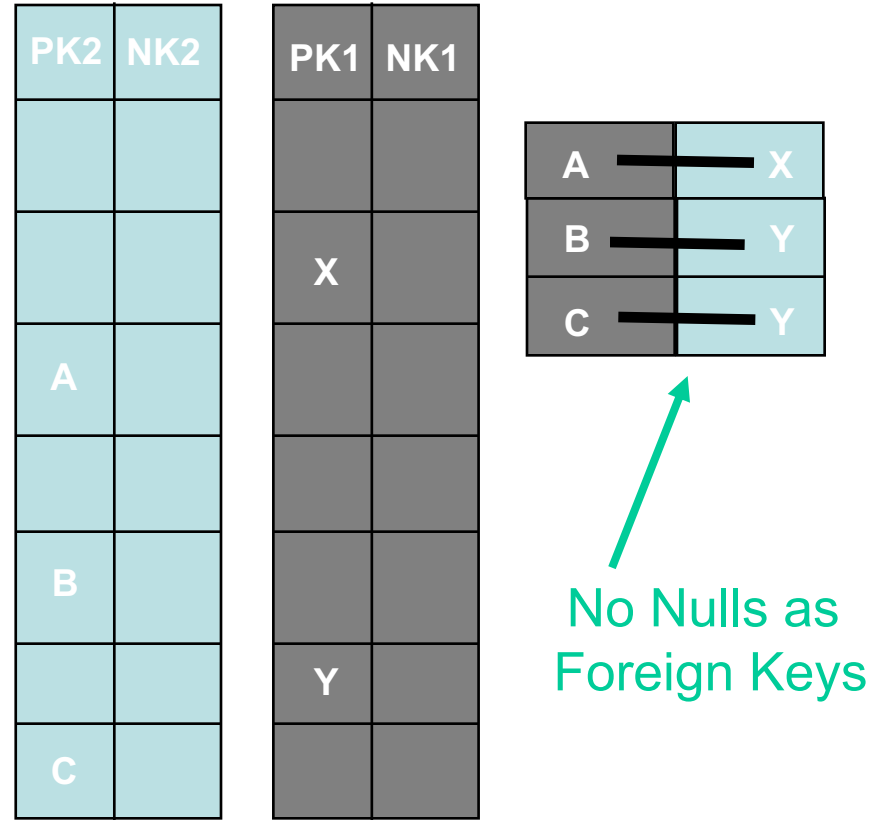
More on M:N Mapping

- Note that **1:1** and **1:N** relationships can be mapped in the same way as **M:N**
- Advantageous when few relationship instances exist (**Sparse 1:1 Relationship**) as it reduces the number of “nulls” that appear as foreign key values

Sparse 1:1 Relationship



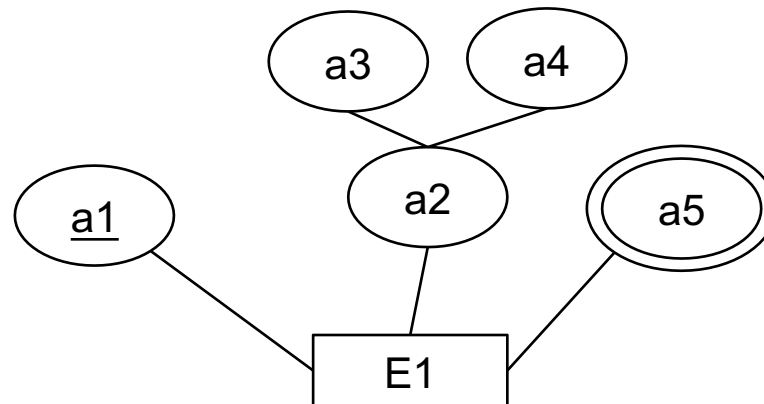
Standard Implementation



M:N Implementation

Step 6: Multivalued Attributes

- For each multi-valued 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 attribute is **composite**, include its simple components

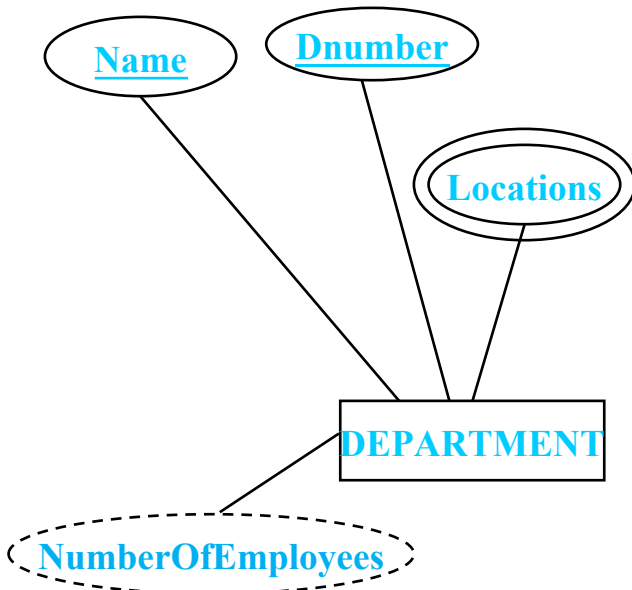


E1 [a1, a3, a4]

E2 [a1, a5]
E2.a1 → E.a1

Step 6: Example

Multi-valued attributes in the Company Database: **Locations**



- Attribute Location will represent the multi-valued attributes Locations of DEPARTMENT

DEPT_LOCS [**Dnumber**, Location]

- *DEPT_LOCS. Dnumber → DEPARTMENT.Dnumber*

Final Schema

Relations:

- EMPLOYEE [SSN, Fname, Mit, Lname, Dob, Address, Sex, Salary, **Dnumber**, **SuperSSN**]
- DEPARTMENT [Dnumber, Dname, **SSN**, StartDate]
- PROJECT [Pno, PName, Plocation, **Dnumber**]
- DEPENDENT [SSN, DepName, Sex, Dob, Relationship]
- WORKS_ON [SSN, Pno, Hours]
- DEPT_LOCS [Dnumber, Location]

Foreign Keys:

- EMPLOYEE.Dnumber → DEPARTMENT.Dnumber
- EMPLOYEE.SuperSSN → EMPLOYEE.SSN
- DEPARTMENT.SSN → EMPLOYEE.SSN
- PROJECT.Dnumber → DEPARTMENT.Dnumber
- WORKS_ON.SSN → EMPLOYEE.SSN
- WORKS_ON.Pno → PROJECT.Pno
- DEPENDENT.SSN → EMPLOYEE.SSN
- DEPT_LOCS.Dnumber → DEPARTMENT.Dnumber

Step 7: N-ary Relationships

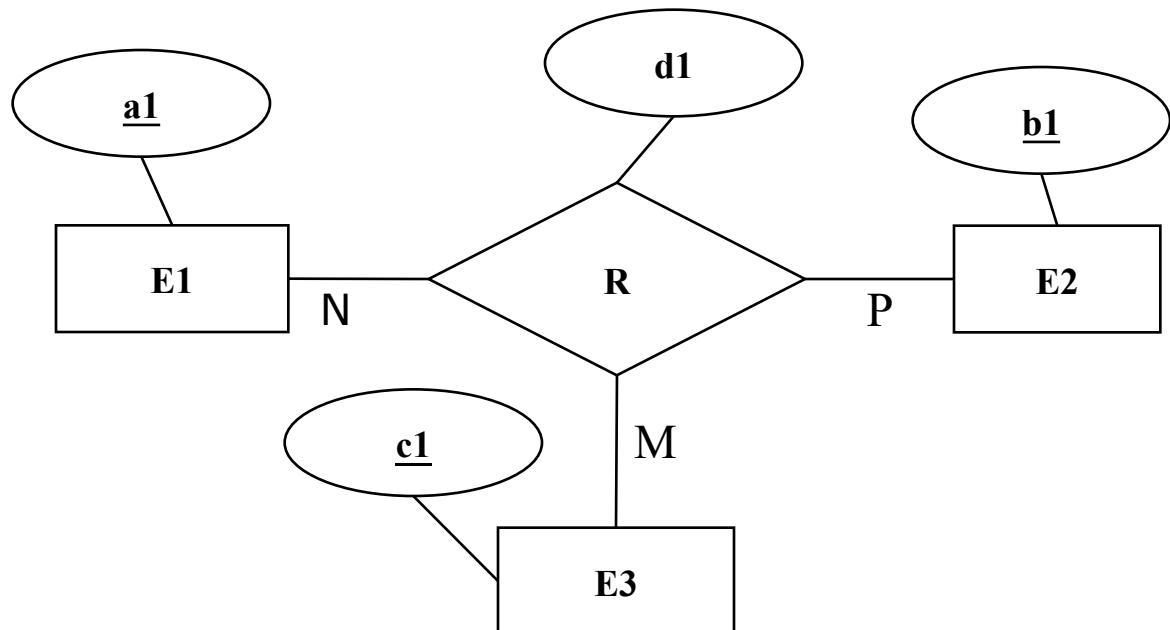
- For each “n-ary” relationship R, **create a new relation** R to represent it.
 - Include as **foreign key attributes** of R the primary keys of the relations that represent the participating entity types in R
 - Include any **simple attributes** of the n-ary relationship
 - The combination of foreign keys referencing the relations representing the participating entities is used to form **primary key** of R

E1 [a1, ...]

E2 [b1, ...]

E3 [c1, ...]

R [a1, b1, c1, d1]



Step 8: Super & Sub-classes

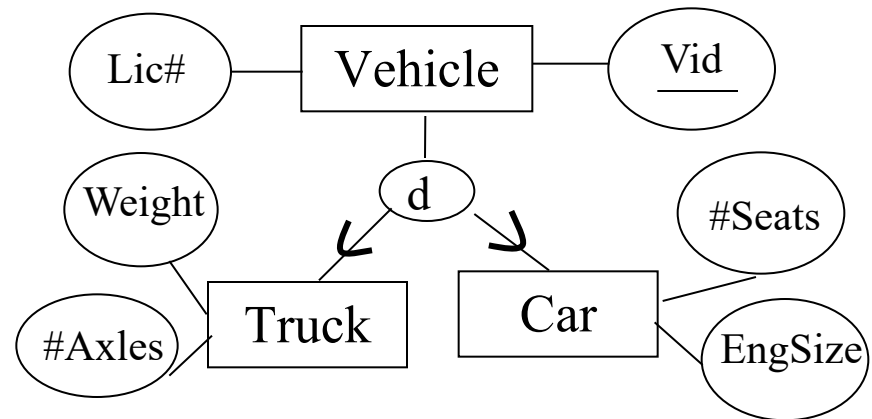
- Option 8A
 - We create a **relational table for the superclass** and create a **relational table for each subclass**.
 - The primary key of each of the subclass is the primary key of the superclass.

Vehicle(Vid, Lic#)

Truck(Vid, Weight, #Axles)

Car(Vid, #Seats, Engsize)

Works for all constraints:
Disjoint/Overlapping
Total/Partial



Step 8 (cont)

- Option 8B

- 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.

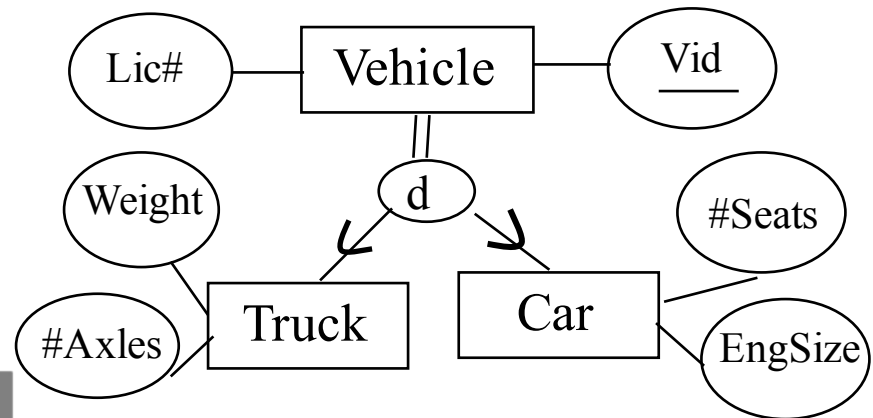
Truck(Vid, Lic#, Weight, #Axles)

Car(Vid, Lic#, #Seats, Engsize)

Disjoint Total only

Overlapping: redundancy

Partial: may lose superclass entities not in any subclass



Step 8 (cont)

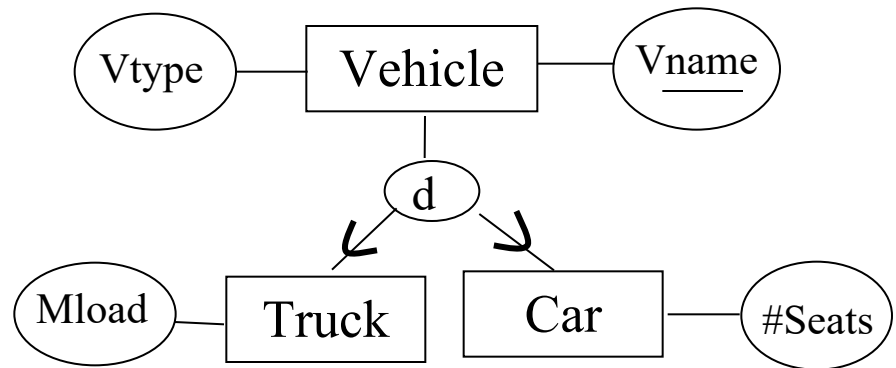
- Option 8C
 - We create **a single relational table** for all subclasses and the superclass.
 - The attributes of the table is the union of all attributes **plus the attribute T** to indicate the subclass to which each tuple belongs. T is NULL in tuples that do not belong to any subclass (for partial constraints)

Vehicle(Vname, Vtype, Mload, #Seats, T)

Disjoint only

In the case attribute **Vtype** is used to discriminate subclasses, the attribute T can be omitted.

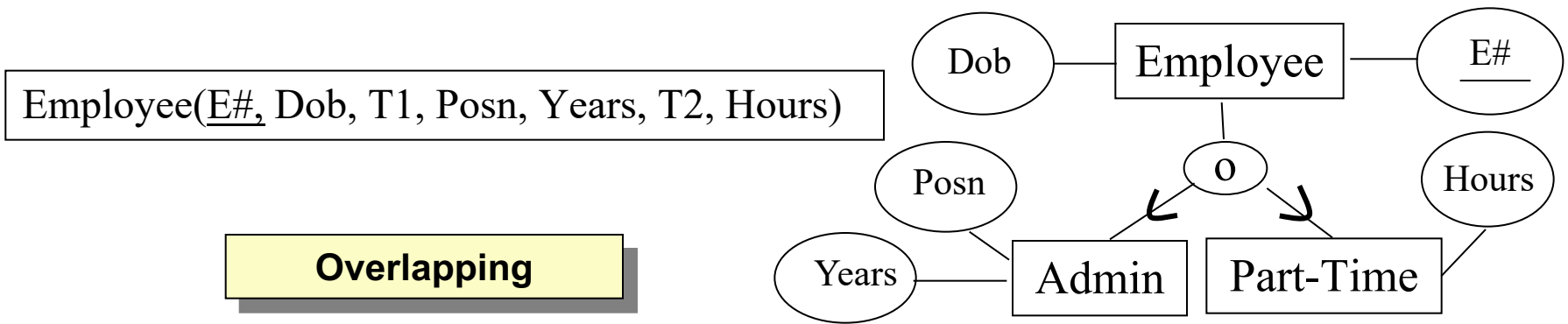
Many NULLS!



Step 8 (cont)

- Option 8D

- 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.



History and the Three Schema Architecture

The Relational Model

Implementation of Constraints

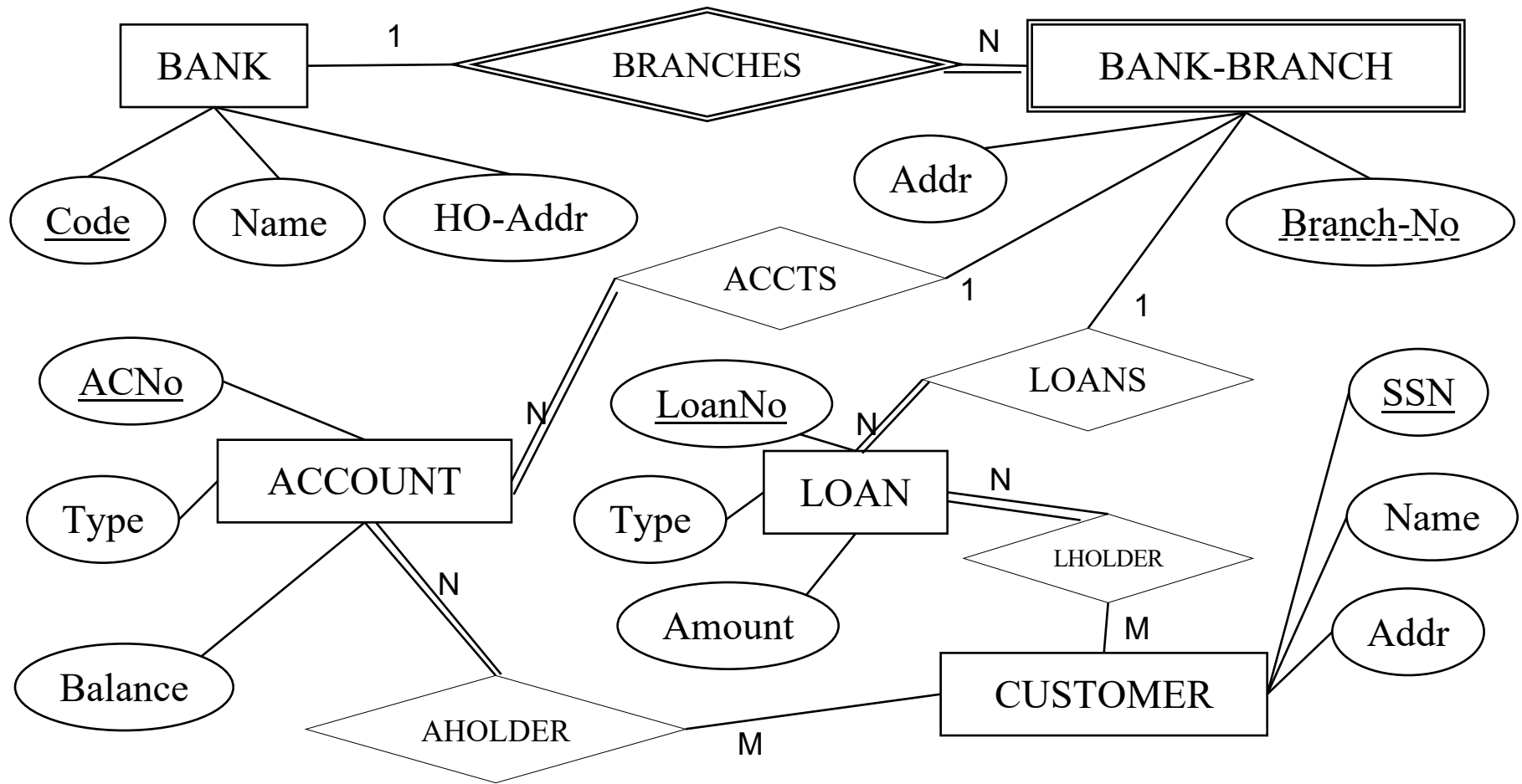
Mapping ER Diagrams to Relational Models

Mapping Example

Specifications

- A bank, given by its 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 id of all customers (account and loan) 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 → 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} → BRANCH.{BankCode, BranchNo}
- LOAN.{BankCode, BranchNo} → 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]
- ACCOUNT-HOLDER [ACNo, SSN]
- LOAN-HOLDER [LoanNo, SSN]

Foreign Keys:

- BRANCH.BankCode → BANK.Code
- ACCOUNT.{BankCode, BranchNo} → BRANCH.{BankCode, BranchNo}
- LOAN.{BankCode, BranchNo} → BRANCH.{BankCode, BranchNo}
- ACCOUNT-HOLDER.ACNo → ACCOUNT.ACNo
- ACCOUNT-HOLDER.SSN → CUSTOMER.SSN
- LOAN-HOLDER.LoanNo → LOAN.LoanNo
- LOAN-HOLDER.SSN → CUSTOMER.SSN