INFS1200/7900
Introduction to Information Systems

The Relational Model

Hassan Khosravi

Notes

- **Feedback:** Please use the link under Learning Resources → Module 1→ Feedback to provide feedback on your Module 1 learning experience
- **Lectures**: Starting "Module 2: The Relational Model"
- **Tutorials**: Tutorial 2 on the Relational Model. Tutorial 1 solution released.
- **Practicals**: Working through part 1.1 and 1.2 of the Practical work
- **Mini-project:** Simulates a basic scenario where you have been recruited as a MySQL Database Architect.
- **RiPPLE**: Round 1 ends 10 August at 11:59 pm. User rating mark will be reviewed at the end of the semester.
- **Assignment 1**: This assignment will be released by the end of this week.

UQ Active Learn

- UQ Active Learn contains muliple applications includding **UQpoll** and **UQwordcloud**

URL: apps.elearning.uq.edu.au
ID: 60140

Disclosure: This application allows me to track authors of the responses.

Learning Outcomes

Description	Tag
Explain the three-Schema architecture, separating internal level, conceptual level and external level.	DBMS
Explain the concept of physical and logical data independence.	
Define the main components of the relational model: Relations, Domains, Attributes and Tuples.	Relational-model
Explain and provide examples for domain constraints.	
Explain and provide examples for key constraints.	
Explain and provide examples for notion of superkeys.	
Explain and provide examples for notion of keys.	
Explain and provide examples for referential integrity constraints.	
Explain and provide examples for user-specified constraints.	
Map entities to relations.	Map-ER-Schema
Map relationships to relations.	
Map weak entities to relations.	
Map Super & Sub-classes to relations.	
Given an ER diagram, map it to a set of relations using the Relational Model.	
Given a relational schema, use the Relational Model to determine the underlying ER diagram.	Map-Schema-ER

Data Abstraction and Data Model

- **Data Abstraction**: an abstract view of data that excludes many details that are either too complex or not of interest to the users

- **Data Model**: a collection of concepts that can be used to describe the structure of a database to achieve data abstraction.

Schemas and Instances

Most data models have the concept of "schema" and "instance"

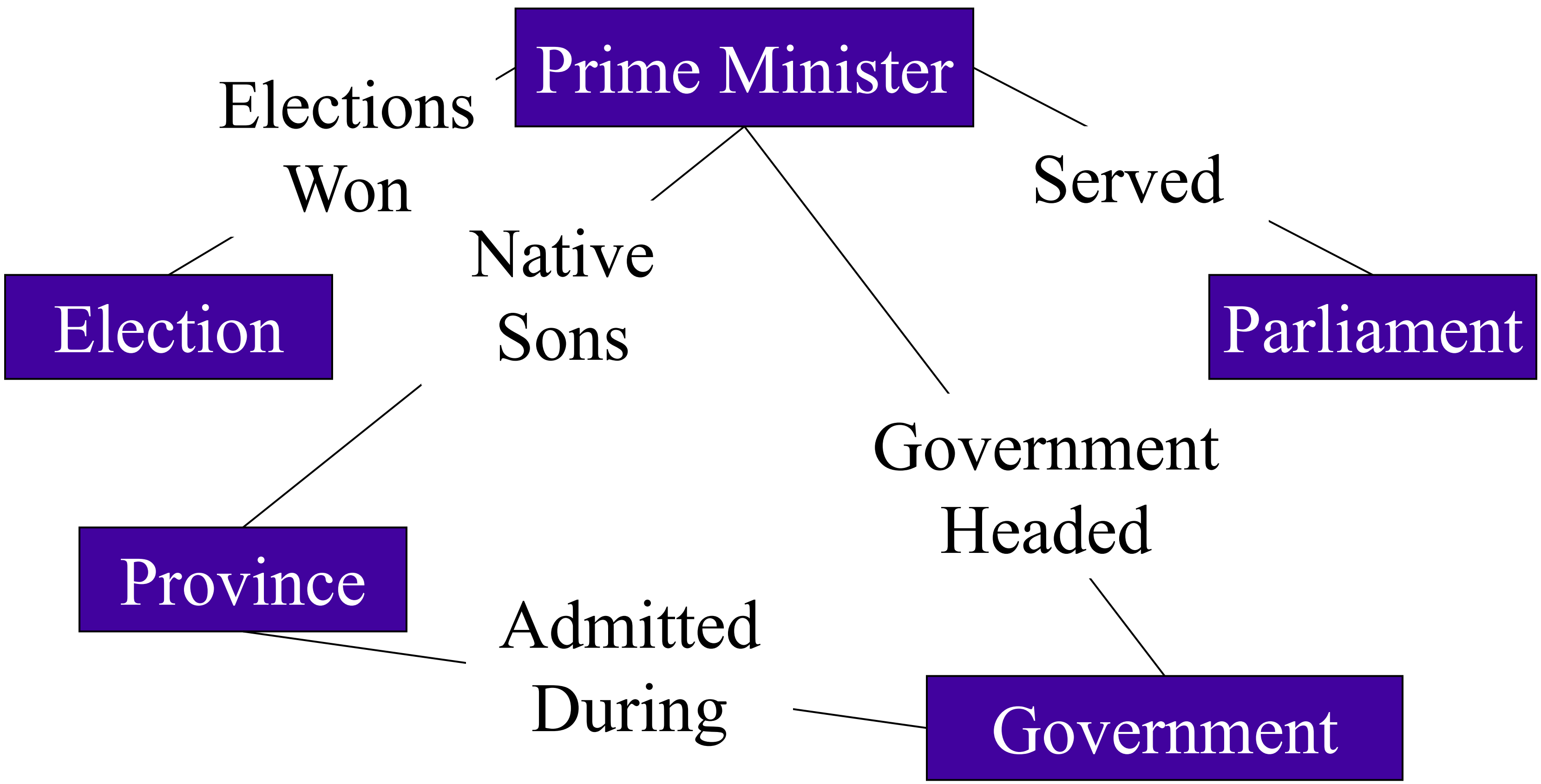
- A **Schema** is the meta-data, or data describing data
- Schema is specified during database design, and is not expected to change frequently

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

Before the Relational Model

- Prior to the relational model, there were two main contenders
  - Network model and databases
  - Hierarchical model and databases
- Network databases had a complex data model
- Hierarchical databases integrated the application in the data model

Example Hierarchical Model



Looks similar to ER diagrams
but has fewer concepts. But let's see how you query it…

Example IMS (Hierarchical) query: Print the names of all the provinces admitted during a Liberal Government

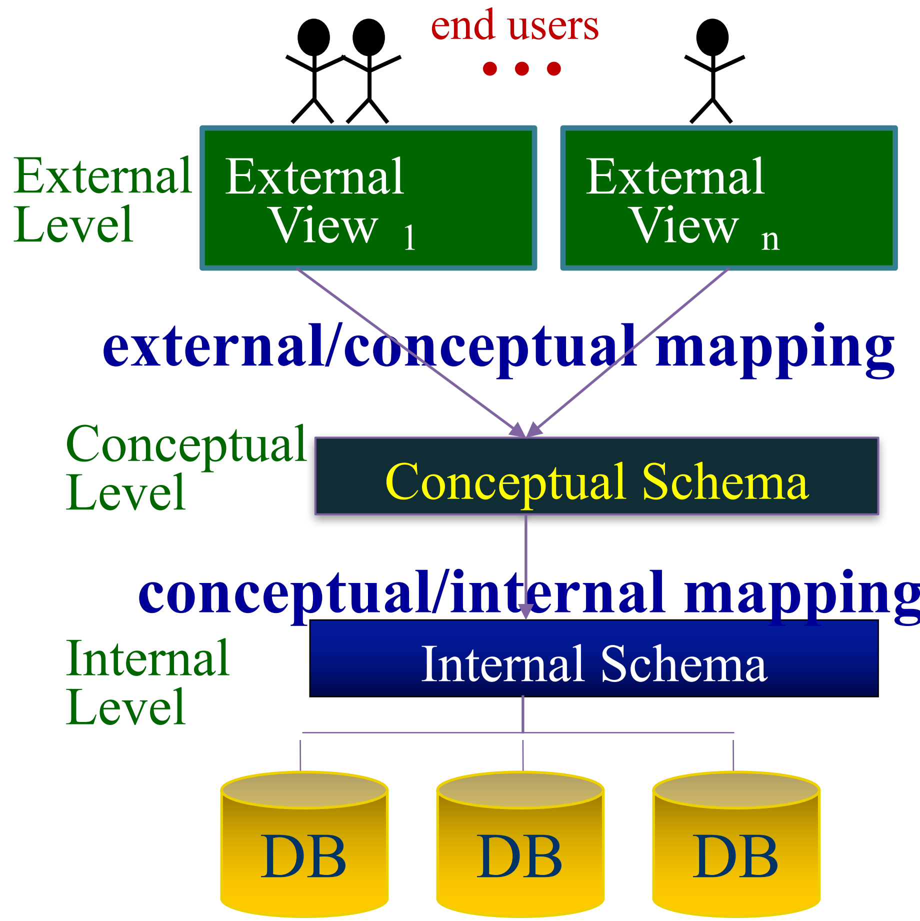
```
DLITPLI:PROCEDURE (QUERY_PCB) OPTIONS (MAIN);

 DECLARE QUERY_PCB POINTER;
 /*Communication Buffer*/
 DECLARE 1 PCB BASED(QUERY_PCB),
  2 DATA_BASE_NAME CHAR(8),
  2 SEGMENT_LEVEL CHAR(2),
  2 STATUS_CODE CHAR(2),
  2 PROCESSING_OPTIONS CHAR(4),
  2 RESERVED_FOR_DLI FIXED BIRARY(31,0),
  2 SEGMENT_NAME_FEEDBACK CHAR(8)
  2 LENGTH_OF_KEY_FEEDBACK_AREA FIXED BINARY(31,0),
  2 NUMBER_OF_SENSITIVE_SEGMENTS FIXED BINARY(31,0),
  2 KEY_FEEDBACK_AREA CHAR(28);
 /* I/O Buffers*/
 DECLARE PRES_IO_AREA CHAR(65),
  1 PRESIDENT DEFINED PRES_IO_AREA,
  2 PRES_NUMBER CHAR(4),
  2 PRES_NAME CHAR(20),
  2 BIRTHDATE CHAR(8)
  2 DEATH_DATE CHAR(8),
  2 PARTY CHAR(10),
  2 SPOUSE CHAR(15);
 DECLARE SADMIT_IO_AREA CHAR(20),
  1 province_ADMITTED DEFINED SADMIT_IO_AREA,
  2 province_NAME CHAR(20);
 /* Segment Search Arguments */
 DECLARE 1 PRESIDENT_SSA STATIC UNALIGNED,
  2 SEGMENT_NAME CHAR(8) INIT('PRES '),
  2 LEFT_PARENTHESIS CHAR (1) INIT('('),
  2 FIELD_NAME CHAR(8) INIT ('PARTY '),
  2 CONDITIONAL_OPERATOR CHAR (2) INIT('='),
  2 SEARCH_VALUE CHAR(10) INIT ('Liberal '),
2 RIGHT_PARENTHESIS CHAR(1) INIT(')');
 DECLARE 1 province_ADMITTED_SSA STATIC UNALIGNED,
  2 SEGMENT_NAME CHAR(8) INIT('SADMIT ');
 /* Some necessary variables */
 DECLARE GU CHAR(4) INIT('GU '),
  GN CHAR(4) INIT('GN '),
  GNP CHAR(4) INIT('GNP '),
  FOUR FIXED BINARY (31) INIT (4),
  SUCCESSFUL CHAR(2) INIT(' '),
  RECORD_NOT_FOUND CHAR(2) INIT('GE');
 /*This procedure handles IMS error conditions */
 ERROR;PROCEDURE(ERROR_CODE);
  *
  *
  *
 END ERROR;
 /*Main Procedure */
 CALL PLITDLI(FOUR,GU,QUERY_PCB,PRES_IO_AREA,PRESIDENT_SSA);
 DO WHILE(PCB.STATUS_CODE=SUCCESSFUL);
  CALL PLITDLI(FOUR,GNP,QUERY_PCB,SADMIT_IO_AREA,province_ADMITTED_SSA);
  DO WHILE(PCB.STATUS_CODE=SUCCESSFUL);
   PUT EDIT(province_NAME)(A);
  CALL PLITDLI(FOUR,GNP,QUERY_PCB,SADMIT_IO_AREA,province_ADMITTED_SSA);
   END;
  IF PCB.STATUS_CODE NOT = RECORD_NOT_FOUND
   THEN DO;
    CALL ERROR(PCB.STATUS_CODE);
    RETURN;
    END;
   CALL PLITDLI(FOUR,GN,QUERY_PCB,PRES_IO_AREA,PRESDIENT_SSA);
  END;
  IF PCB.STATUS_CODE NOT = RECORD_NOT_FOUND
   THEN DO;
    CALL ERROR(PCB.STATUS_CODE);
    RETURN;
    END;
END DLITPLI;
```

Three-Schema Architecture

- **External Level:** provides access to particular parts of the database to users

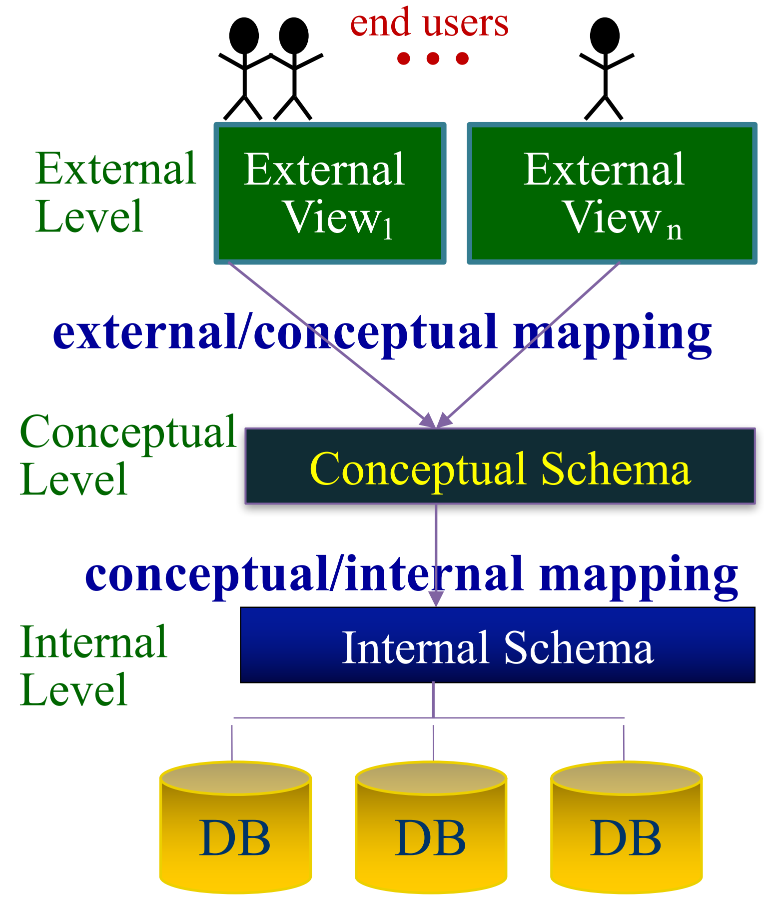
- **Conceptual Level:** describes the structure of the whole database for a community of users.

- **Internal Level:** describes the physical storage structure of the database.



Data Independence via Three-Schema Architecture

- **Data Independence**: Ability to change the schema at one level of a database system without having to change the schema at the next higher level.
  1. **Logical Data Independence**: Ability to change the conceptual schema without changing applications
  2. **Physical Data Independence**: Ability to modify physical schema w/o changing logical schema



History and the Three Schema Architecture

The Relational Model

Implementation of Constraints

Mapping ER Diagrams to Relational Models

Mapping Example

From Schema to ER Diagrams

The Relational Model

- Introduced by Edgar Codd (IBM) in 1970

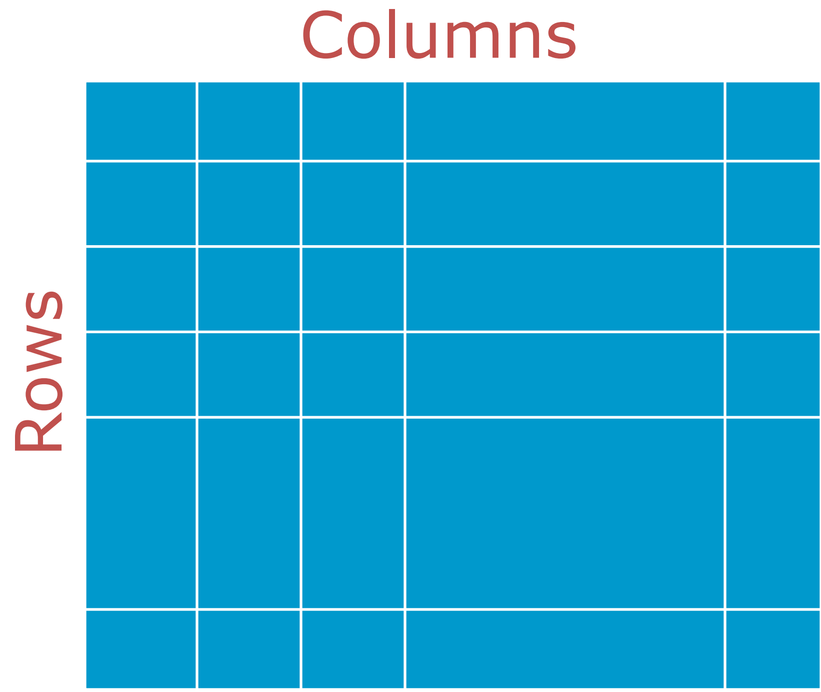
- Most widely used model today.
  - Vendors: IBM, Informix, Microsoft, Oracle, Sybase,

- The Relational Model has four main concepts:
  - Relations
  - Domains
  - Attributes
  - Tuples

- Recent competitors (triggered by the needs of Web):
  - XML data model and NoSQL

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



Relations, not Tables

- The term Table is used interchangeably with Relation
  - Every relation is a table
  - Every table is not necessarily 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	Calam Vale	Labels	4232	65	865	768
West	Taringa	Envelops	3242	543	4554	454
North	McDowell	Toners	23	456	24	434
South	Sunny Bank	Ribbons	324	65	56	657
West	Indooroopilly	Disks	234	6786	324	554

Not a Relation !

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. $1 \leq$ Grade ≤ 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
Joe Smith	606 567 333	M	BSc
A. Brown	321 638 999	F	BInfTch

Tuples

- Each Tuple **t** is an ordered list of n values:

  $$\mathbf{t = <v_1,v_2, \ldots , v_n>}$$

  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"
- t is called an n-tuple
- Example

  (254, John, Smith, $45K, 3453-2543,M)

Example of a Relation Instance

relation name

attribute, column name

tuple, row, record

domain value

Student

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

- degree/arity = 5;
- Order of rows isn't important
- Order of attributes isn't important (except in some query languages)

Relation Schema and Instance

- Relation Schema
  - A relation schema r is a set of attributes (a_1, a_2,…, 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 $r= \{t_1, t_2, \ldots, t_m\}$. Each n-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 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. 12th Ave., Van.	889-4444	CPSC
92001200	G. Smith	2020 E. 18th St., Van	409-2222	MATH
94001020	A. Smith	2020 E. 18th St., Van	222-2222	CPSC
94001150	S. Wang	null	null	null

Relation Schema and Instance Example

- Relational Schema Example

Movie (<u>MovieID</u>, Title, Year)
StarsIn (<u>MovieID, StarID</u>, Role)
MovieStar (<u>StarID</u>, 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. C is an attribute of R.

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

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

E. 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. C is an attribute of R.

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

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

E. All of the above

E All are true

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

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

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

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

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

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

Same Relation

History and the Three Schema Architecture

The Relational Model

Implementation of Constraints

Mapping ER Diagrams to Relational Models

Mapping Example

From Schema to ER Diagrams

Relational Database Design

- Relational Database
  - A collection of relations with distinct relation names
- Relational Database Schema
  - A collection of schemas for the relations in the database
- Relational Database Design
  - Process of capturing the semantics of an application, and translating it into a relational database schema

Implementation of Constraints

- **Integrity Constraints** 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
  - Key and foreign key ICs 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 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 instance r of R

  $$t_i\,[SK] \neq t_j[SK]$$

- 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
E. All A, 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**
E. All A, B and C

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
- Formally, K is a key in a relation schema R iff
  - K is a Superkey of R, and
  - removing any attribute from K leaves a set of attributes K', where K' is not a superkey of R, that is, K' does NOT maintain the uniqueness constraint

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

- StarID would be key

Characteristics of Keys

- 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

- 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

- StarID in StarsIn references MovieStar
- Only movie stars listed in the MovieStar relation should be allowed star in Movies.

StarsIn

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

MovieStar

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

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

- Could foreign key 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.

t1[FK] = t2[PK] or t1[FK] 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 <u>Checks, Assertions</u> and <u>Triggers</u>. These are not covered in detail in this course.
- Example: No actor can play in more than 10 movies in one year.

Movie (<u>MovieID</u>, Title, Year)
StarsIn (<u>MovieID, StarID</u>, Role)
MovieStar (<u>StarID</u>, Name, Gender)

In-class Exercise

Use this relational schema

STUDENT (StID, Name, Email)

COURSE (CCode, Title, Units)

ENROLMENT (StID, CCode, Sem, Year)

to give examples of the following:

1. Super Key
2. Minimal Key
3. Foreign Key
4. Domain Constraint

In-class Exercise - Solution

Use this relational schema

STUDENT (<u>StID</u>, Name, Email)

COURSE (<u>CCode</u>, Title, Units)

ENROLMENT (<u>StID, CCode</u>, Sem, Year)

to give examples of the following:

1. Super Key - <u>StID</u>, Name, Email in relation Student
2. Minimal Key <u>StID in relation Student</u>
3. Foreign Key <u>StID in relation Enrolment</u>
4. Domain Constraint <u>Ccode 4 letters followed by 4 num</u>

History and the Three Schema Architecture

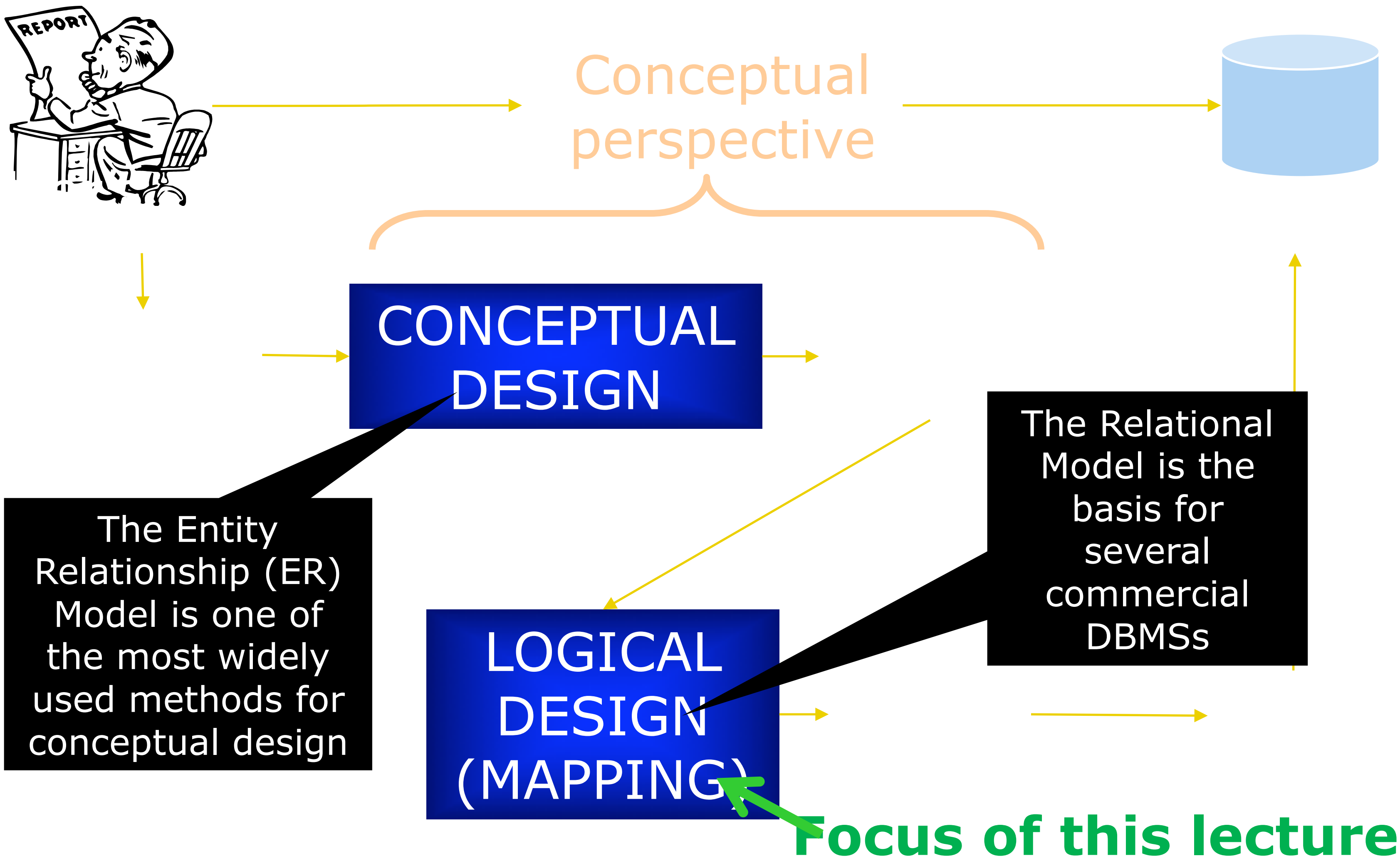
The Relational Model

Implementation of Constraints

Mapping ER Diagrams to Relational Models

Mapping Example

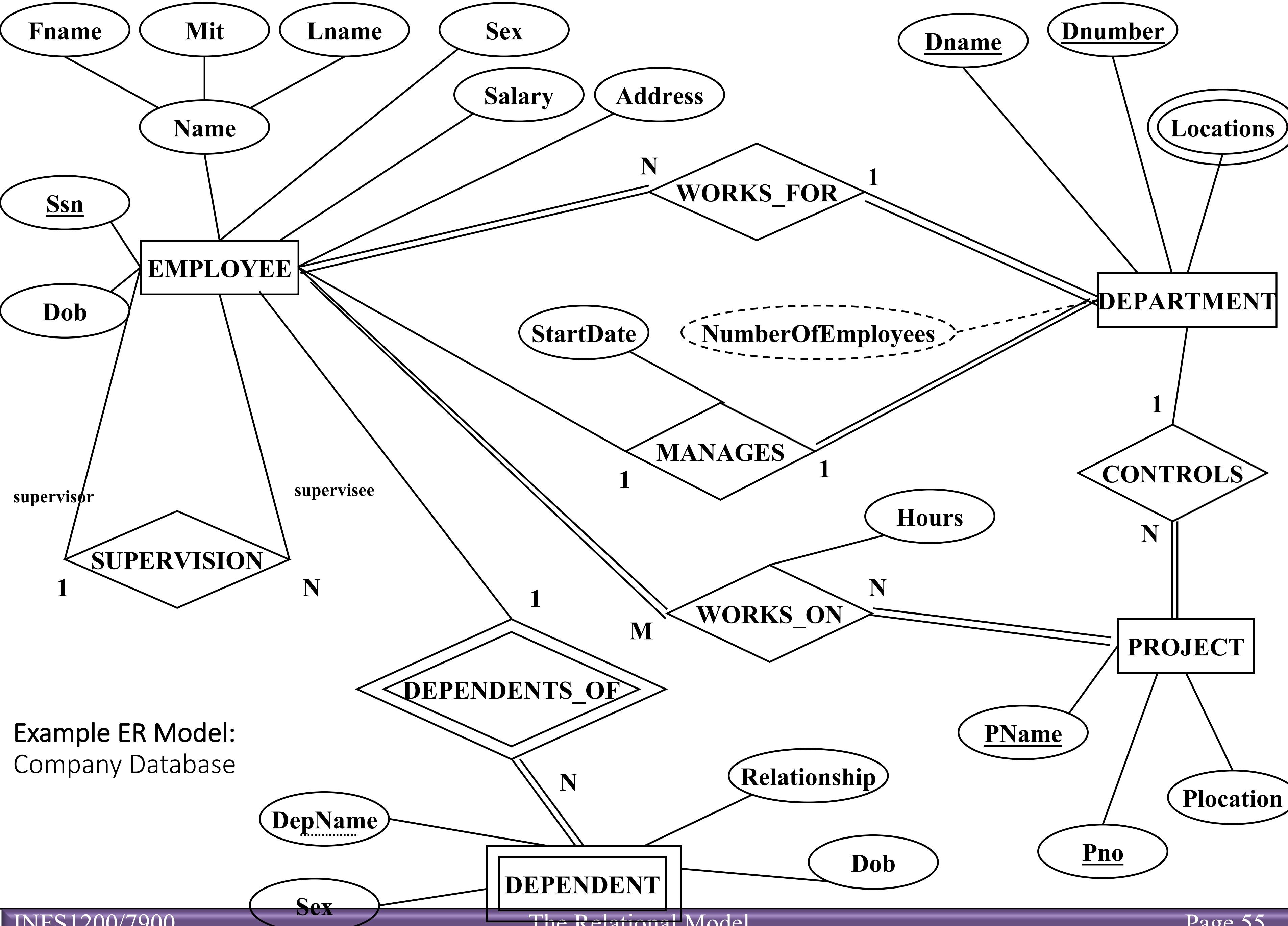
From Schema to ER Diagrams


REPORT
Conceptual perspective
CONCEPTUAL DESIGN
The Entity Relationship (ER) Model is one of the most widely used methods for conceptual design
The Relational Model is the basis for several commercial DBMSs
LOGICAL DESIGN (MAPPING)
Focus of this lecture

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)

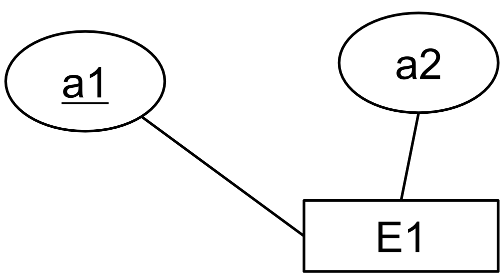


Example ER Model:
Company Database

Step 1: Entity Mapping

For each regular (non-weak) entity E, create a relation R that includes all simple attributes of E

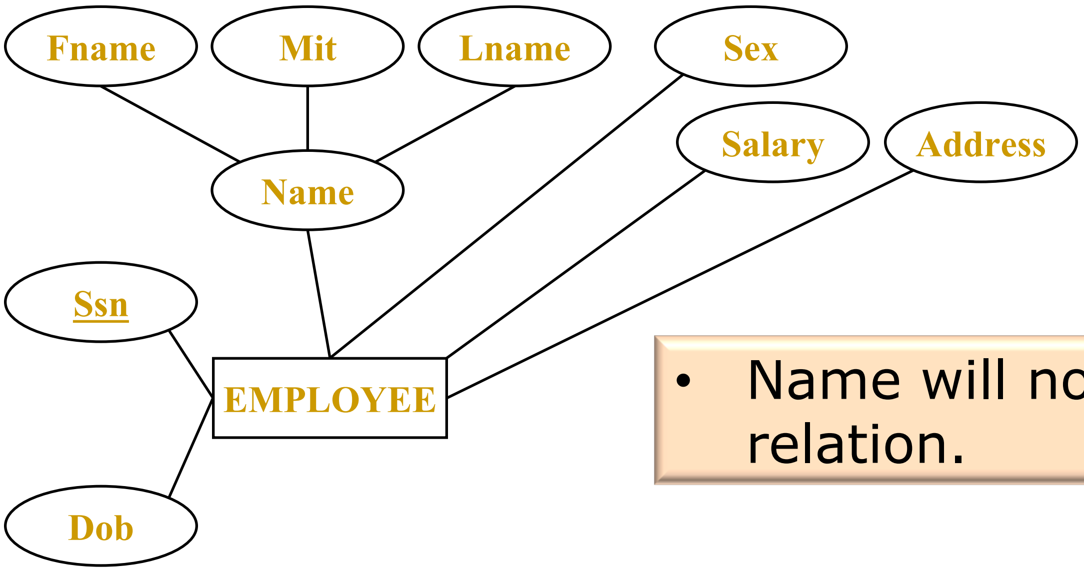
- Include only simple component attributes of a composite attribute
- Choose one key attribute of E as primary key for R. If key of E is composite, the set of simple attributes together should form the key
- Add foreign key attributes in subsequent steps



E1 [a1, a2]

Step 1: Example

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

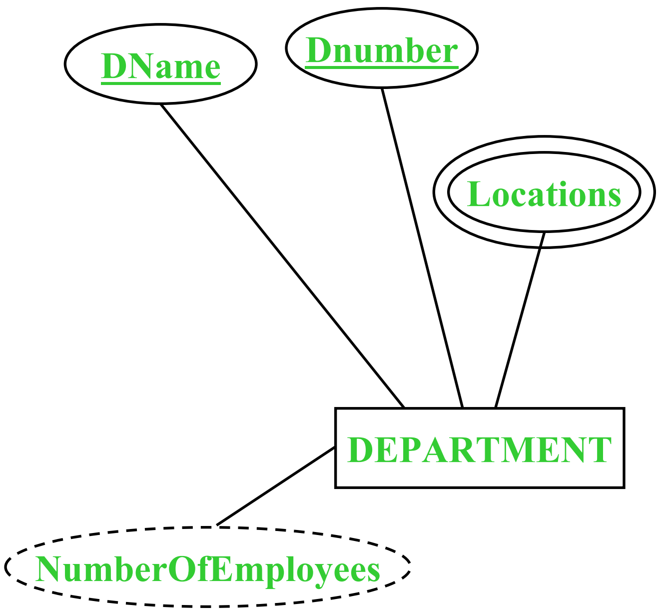


- Name will not be added to the relation.

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

Step 1: Example

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

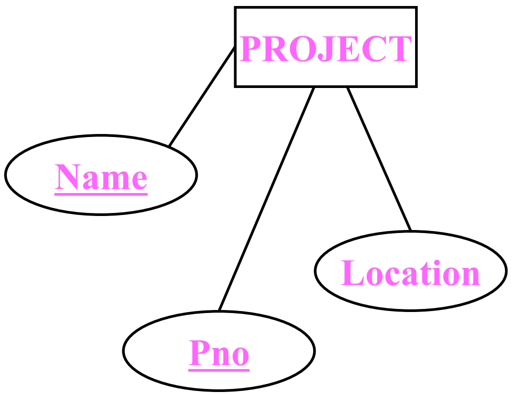


- Location will not be added for the time being
- Both Name and Number are keys. Number is taken as PK.
- NumberOfEmployees is not added to the relation as it is a derived attribute.

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 [<u>Pno</u>, PName, Plocation]

Schema (in progress)

Relations:

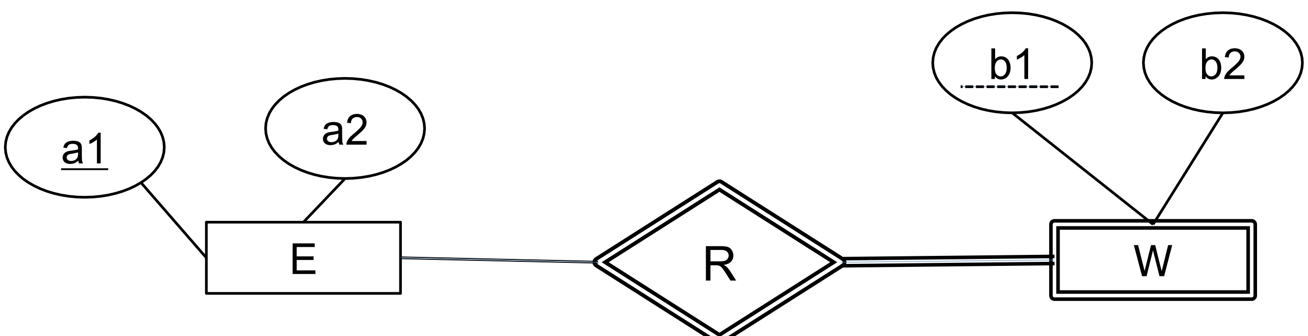
- EMPLOYEE [Ssn, Fname, Mit, Lname, Dob, Address, Sex, Salary]
- DEPARTMENT [Dnumber, DName]
- PROJECT [Pno, PName, Plocation]

Step 2: Weak Entity Mapping

For each weak entity W with owner entity E create a relation R that includes all simple attributes of W

– Include as foreign key attributes in R the primary key attributes of the relation(s) that correspond to the owner entity. (This maps the identifying relationship(s) of W)

– The primary key of R is the combination of the primary key(s) of the owner(s) and the primary key of the weak entity W (if any)



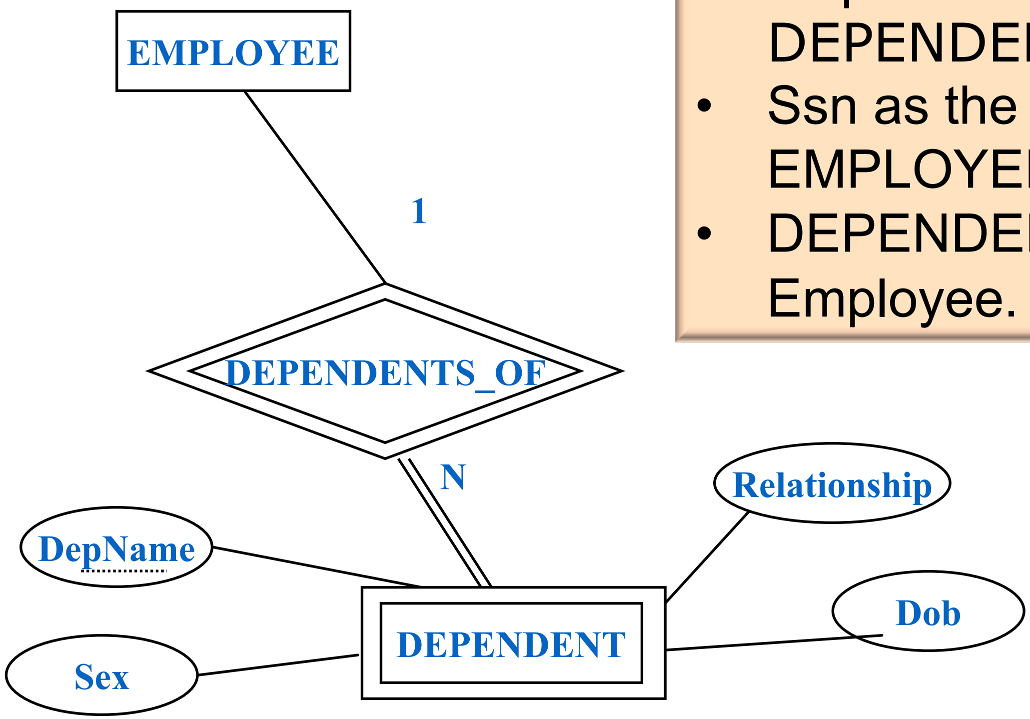
E [a1, a2]

RW [a1, b1, b2]

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]
where Primary Key includes {Ssn, DepName}

Schema (in progress)

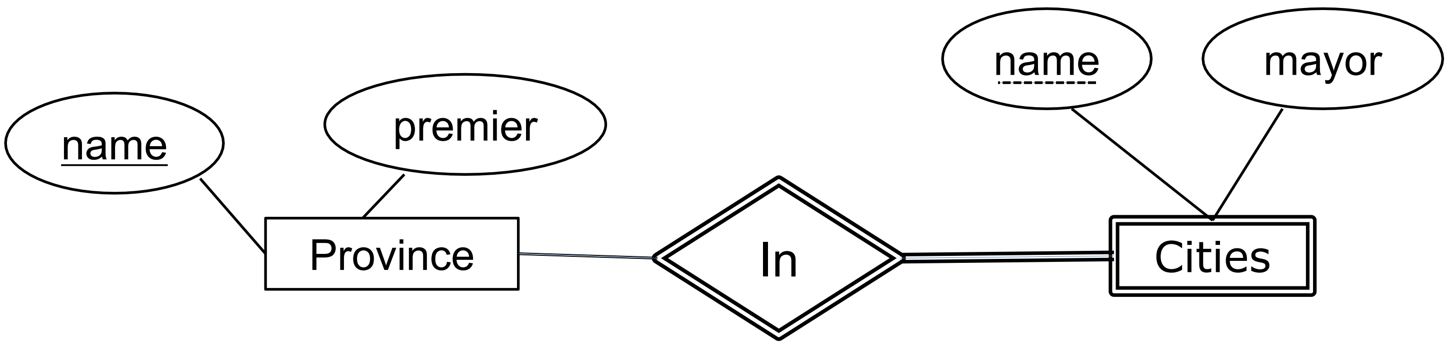
Relations:

- 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*

Multiple-Choice Question



- Convert this E/R 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? (foreign key attributes are shown in bold)

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

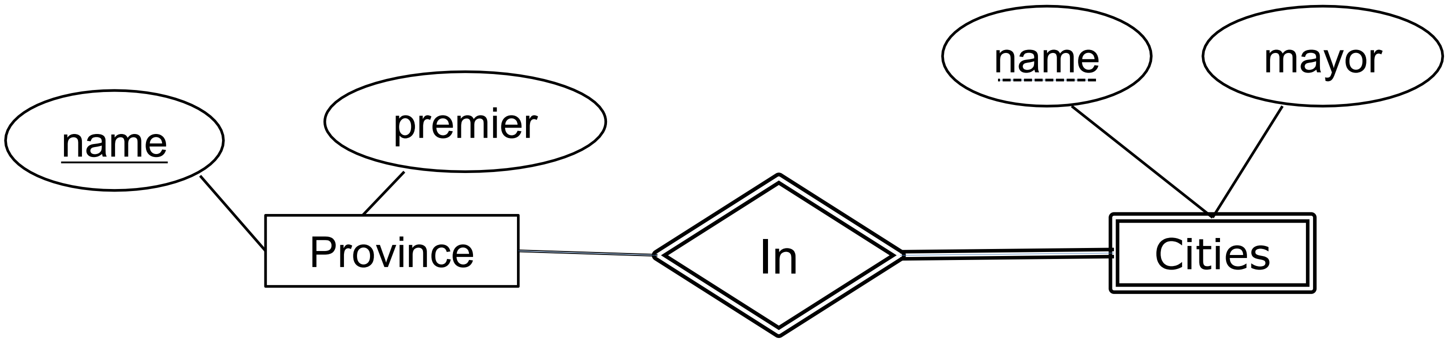
B. Cities [**cname**, **pname**, mayor], Provinces [pname, premier]

C. Cities [cname, **pname**, mayor], Provinces [pname, premier]

D. Cities [cname, **pname**, mayor], In [cname, pname], Provinces [name, premier]

E. None of the above

Multiple-Choice Question



- Convert this E/R 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? (foreign key attributes are shown in bold)

A. Cities [name, mayor], Provinces [name, premier] Cities W.E, so needs pname

B. Cities [**cname**, **pname**, mayor], Provinces [pname, premier] Cname not FK

C. Cities [cname, **pname**, mayor], Provinces [pname, premier] C. Right answer

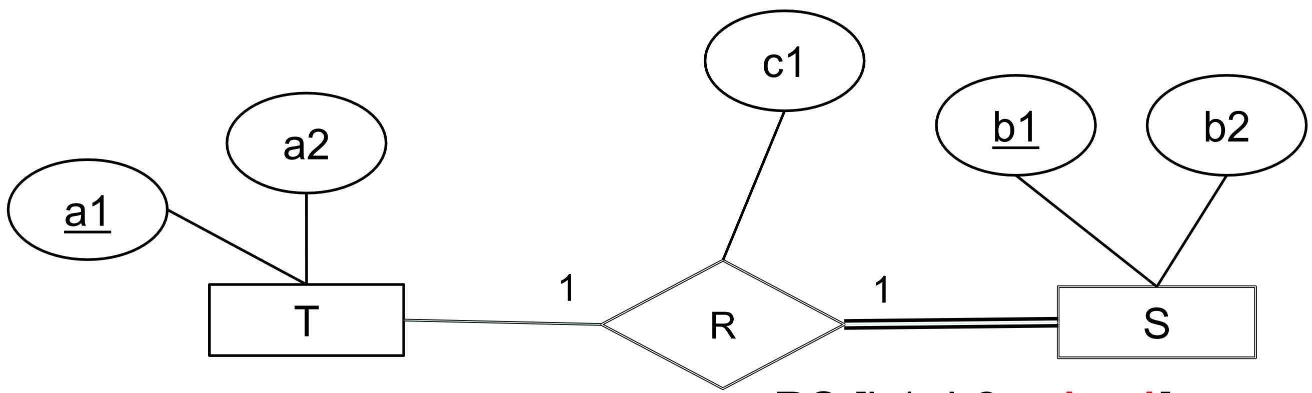
D. Cities [cname, **pname**, mayor], In [cname, pname], Provinces [name, premier] no IN

E. None of the above

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 foreign key in S the primary key of T
- It is better to choose as S, the entity with total participation in RT
- Include all the simple attributes (or simple components of composite attributes) of the 1:1 relationship R as attributes of S



T [a1, a2]

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

Step 3: Example

Binary 1:1 relationship in the Company Database: MANAGES

Schema (in progress)

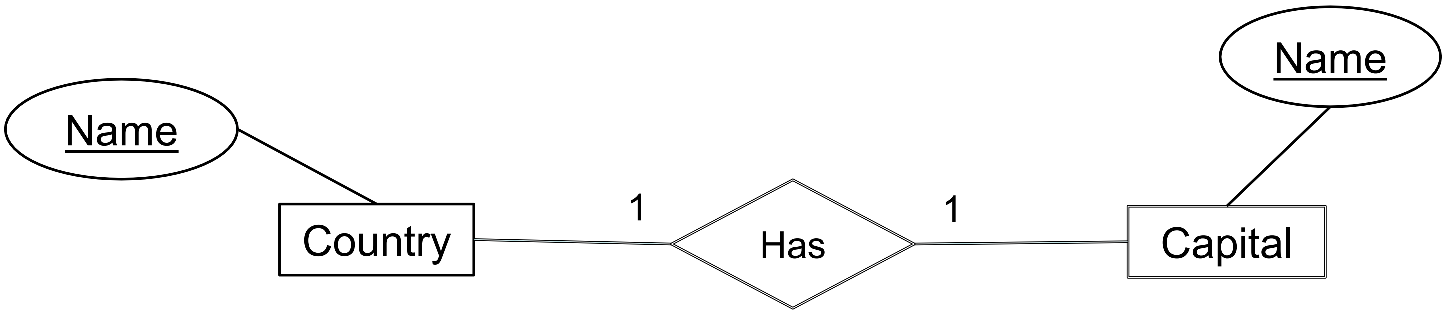
Relations:

- EMPLOYEE [<u>Ssn</u>, Fname, Mit, Lname, Dob, Address, Sex, Salary]
- DEPARTMENT [<u>Dnumber</u>, Dname, MGRSSN, MgrStart]
- PROJECT [<u>Pno</u>, PName, Plocation]
- DEPENDENT [<u>ESSN,DepName</u>, Sex, DOB, Relationship]

Foreign Keys:

- *DEPARTMENT.MGRSSN → EMPLOYEE.Ssn*
- *DEPENDENT.ESSN → EMPLOYEE.Ssn*

Multiple-Choice Question



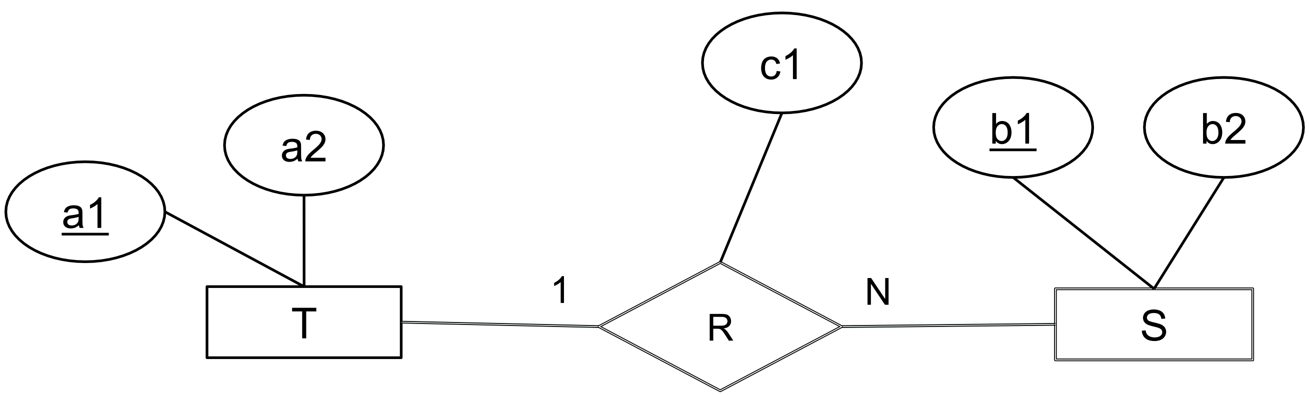
Which schema below is a reasonable translation from ER to relations? ?
(foreign key attributes are shown in bold)

A. Country [coName, **caName**], Capital [caName]
B. Country [name], Capital [name]
C. Country [coName, caName]
D. Both A and C
E. All of A, B, and C

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



T [a1, a2]

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

Step 4: Example

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

Step 4: Example

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

Step 4: Example

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

Schema (in Progress)

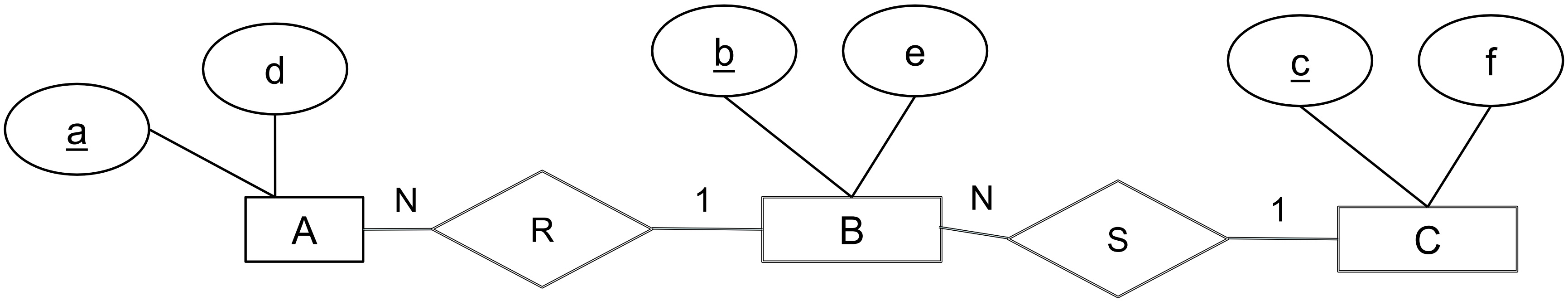
Relations:

- EMPLOYEE [<u>Ssn</u>, Fname, Mit, Lname, Dob, Address, Sex, Salary, Dno, SuperSSN]
- DEPARTMENT [<u>Dnumber</u>, Dname, MGRSSN,MgrStart]
- PROJECT [<u>Pno</u>, PName, Plocation, DNum]
- DEPENDENT [<u>ESSN,DepName</u>, Sex, DOB, Relationship]

Foreign Keys:

- *EMPLOYEE.Dno → DEPARTMENT.Dnumber*
- *EMPLOYEE.SuperSSN → EMPLOYEE.Ssn*
- *DEPARTMENT.MGRSSN → EMPLOYEE.Ssn*
- *PROJECT.DNum → DEPARTMENT.Dnumber*
- *DEPENDENT.ESSN → EMPLOYEE.Ssn*

Multiple-Choice Question



Translate the ER diagram to relational.

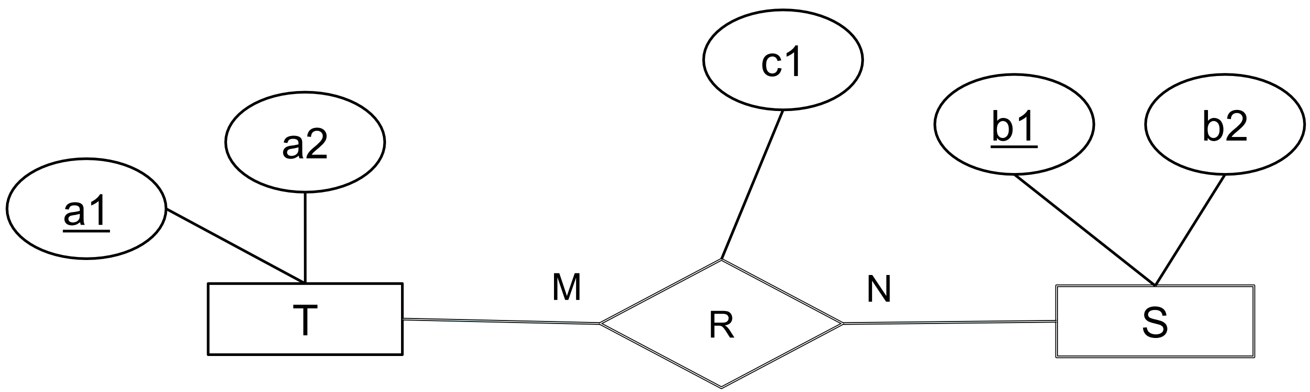
Which of the following appears in your relational schema?(foreign key attributes are shown in bold):

A. AR [<u>a</u>,<u>**b**</u>,d]
B. BS [<u>b</u>,**c**,e]
C. S [<u>b</u>,<u>**c**</u>]
D. All of these
E. None of these

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.



T [a1, a2]   **R [a1, b1, c1]**   S [b1, b2]

Step 5: Example

Binary M:N relationships in the Company Database: WORKS_ON

Schema (in progress)

Relations:

- EMPLOYEE [<u>Ssn</u>, Fname, Mit, Lname, Dob, Address, Sex, Salary, Dno, SuperSSN]
- DEPARTMENT [<u>Dnumber</u>, Dname, MGRSSN,MgrStart]
- PROJECT [<u>Pno</u>, PName, Plocation, DNum]
- DEPENDENT [<u>ESSN,DepName</u>, Sex, DOB, Relationship]
- WORKS_ON [<u>ESSN, PNo</u>, Hours]

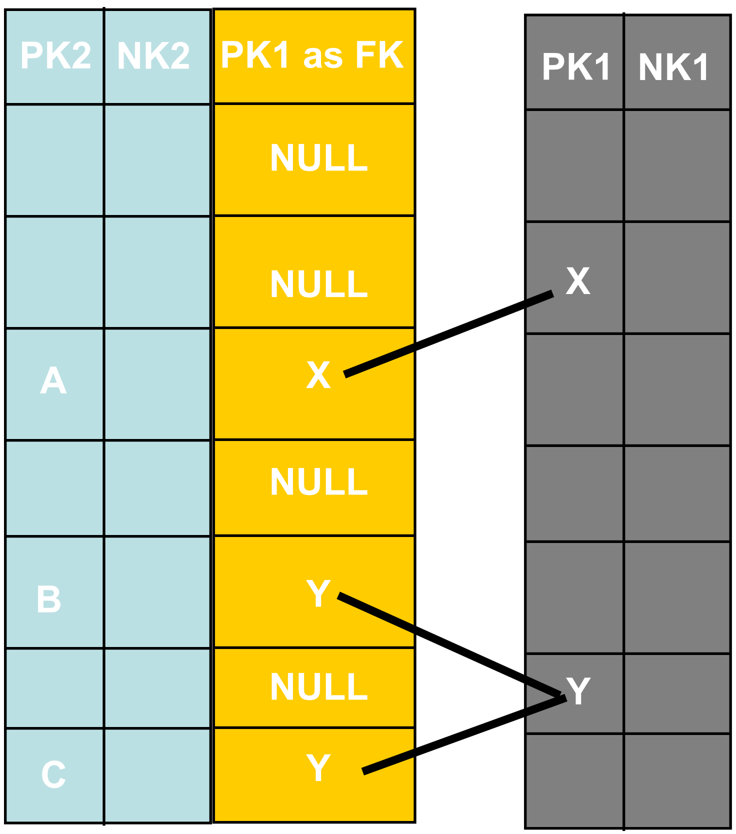
Foreign Keys:

- *EMPLOYEE.Dno →DEPARTMENT.Dnumber*
- *EMPLOYEE.SuperSSN → EMPLOYEE.Ssn*
- *DEPARTMENT.MGRSSN → EMPLOYEE.Ssn*
- *PROJECT.Dnum → DEPARTMENT.Dnumber*
- *WORKS_ON.ESSN → EMPLOYEE.Ssn*
- *WORKS_ON.PNo → PROJECT.Pno*
- *DEPENDENT.ESSN → EMPLOYEE.Ssn*

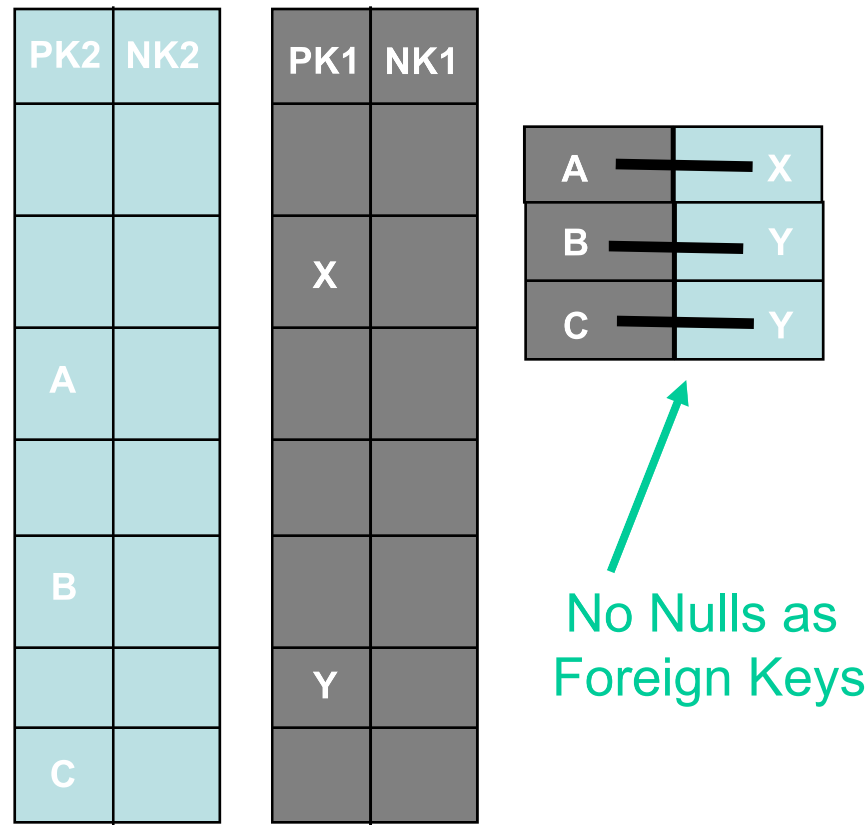
More on M:N Mapping

- 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

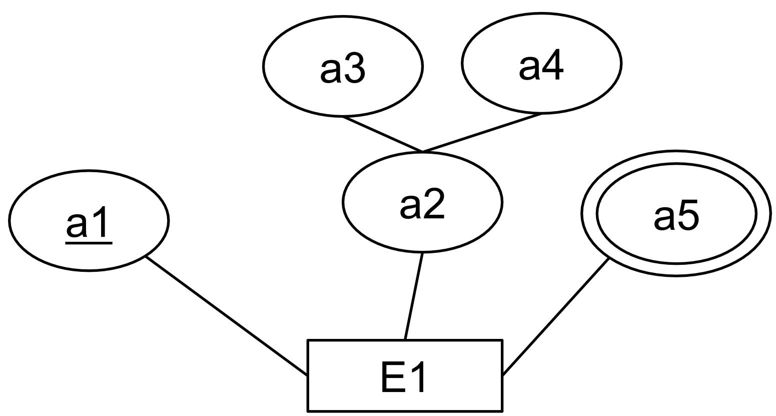


M:N Implementation

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 multi-valued attribute is composite, include its simple components



E2 [a1, a5]

E1 [a1, a3, a4]

Step 6: Example

Multi-valued attributes in the Company Database: Locations

Final Schema

Relations:

- EMPLOYEE [Ssn, Fname, Mit, Lname, Dob, Address, Sex, Salary, Dno, SuperSSN]
- DEPARTMENT [Dnumber, Dname, MGRSSN, MgrStart]
- PROJECT [Pno, PName, Plocation, DNum]
- DEPENDENT [ESSN,DepName, Sex, DOB, Relationship]
- WORKS_ON [ESSN, PNo, Hours]
- DEPT_LOCS[DNumber, DLocation]

Foreign Keys:

EMPLOYEE.Dno →DEPARTMENT.Dnumber

EMPLOYEE.SuperSSN → EMPLOYEE.Ssn

DEPARTMENT.MGRSSN → EMPLOYEE.Ssn

PROJECT.Dnum → DEPARTMENT.Dnumber

WORKS_ON.ESSN → EMPLOYEE.Ssn

WORKS_ON.PNo → PROJECT.Pno

DEPENDENT.ESSN → 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.

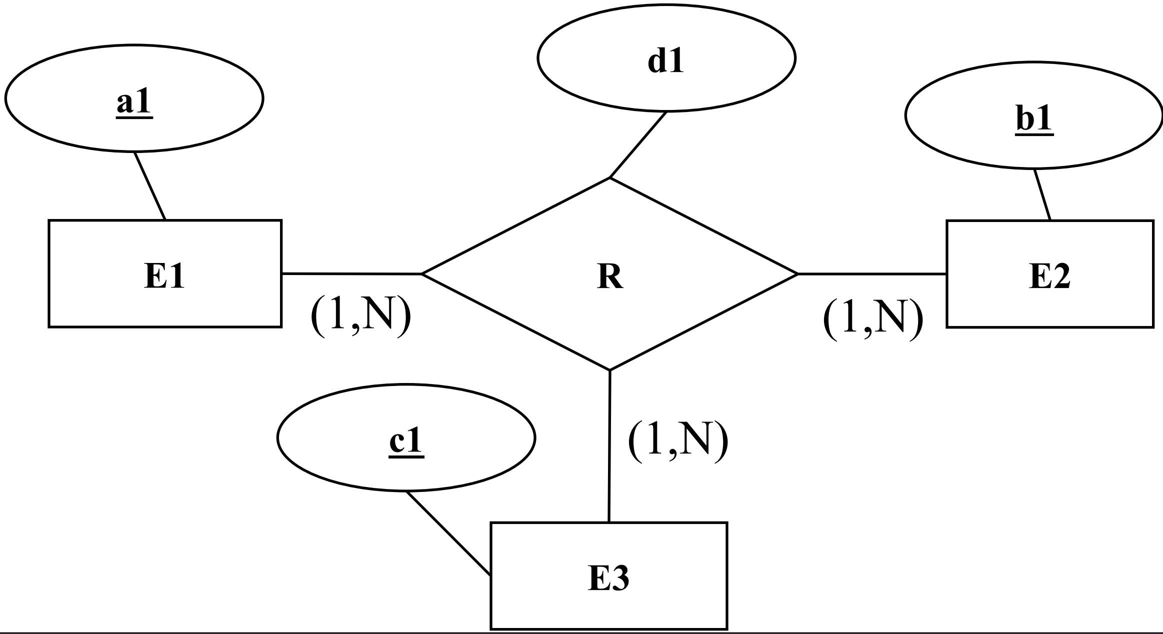
- Include as foreign key attributes of R the primary keys of the relations that represent the participating entity types in R
- Include any simple attributes of the n-ary relationship
- The combination of foreign keys referencing the relations representing the participating entities is used to form primary key of R

E1 [a1, ...]

E2 [b1, ...]

E3 [c1, ...]

R [**a1, b1, c1**, d1]



Step 8: Super & Sub-classes

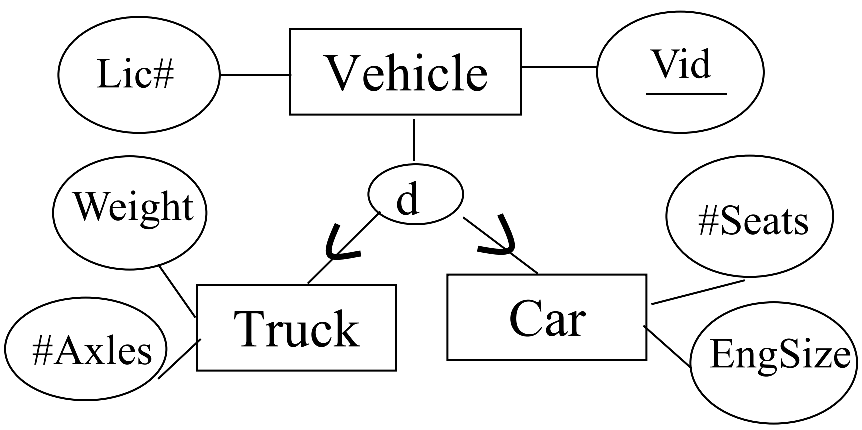
- Option 8A
  - We create a relational table for the superclass and create a relational table for each subclass.
  - The primary key of each of the subclass is the primary key of the superclass.

Vehicle(Vid, Lic#)

Truck(Vid, Weight, #Axles)

Car(Vid, #Seats, Engsize)

Works for all constraints:
Disjoint/Overlapping
Total/Partial



Step 8 (cont)

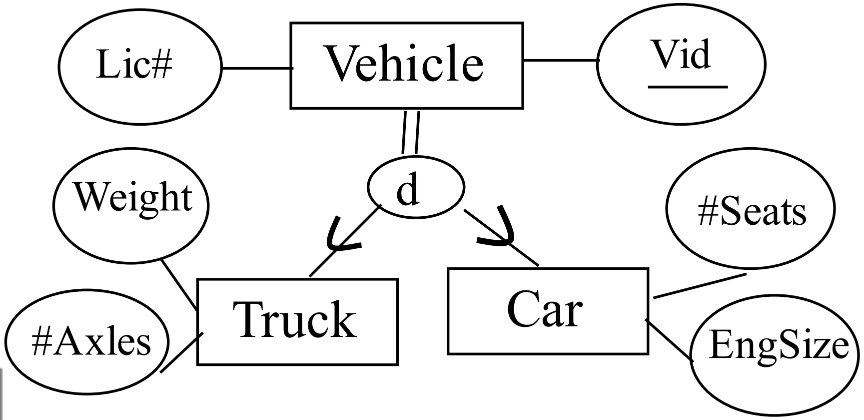
- Option 8B
  - We create a relational table for each subclass. The attributes of the superclass are merged into each of the subclasses.
  - The primary key of the subclass table is the primary key of the superclass.

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)

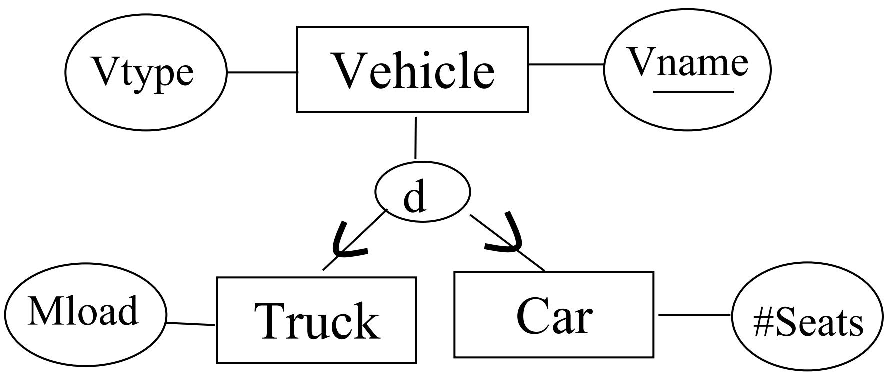
- Option 8C
  - We create a single relational table for all subclasses and the superclass.
  - The attributes of the table is the union of all attributes plus the attribute T to indicate the subclass to which each tuple belongs. T is NULL in tuples that do not belong to any subclass (for partial constraints)

Vehicle(<u>Vname</u>, Vtype, Mload, #Seats, T)

Disjoint only

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

Many NULLS

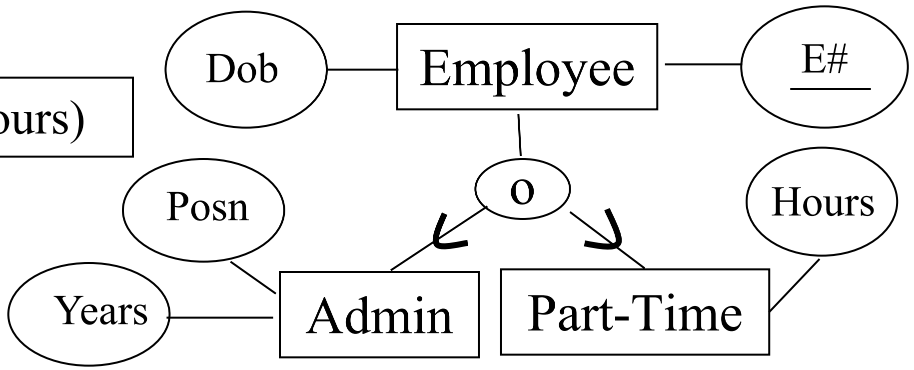


Step 8 (cont)

- 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.

Employee(<u>E#</u>, Dob, T1, Posn, Years, T2, Hours)

Overlapping



A Brief History

The Relational Model

Implementation of Constraints

Mapping ER Diagrams to Relational Models

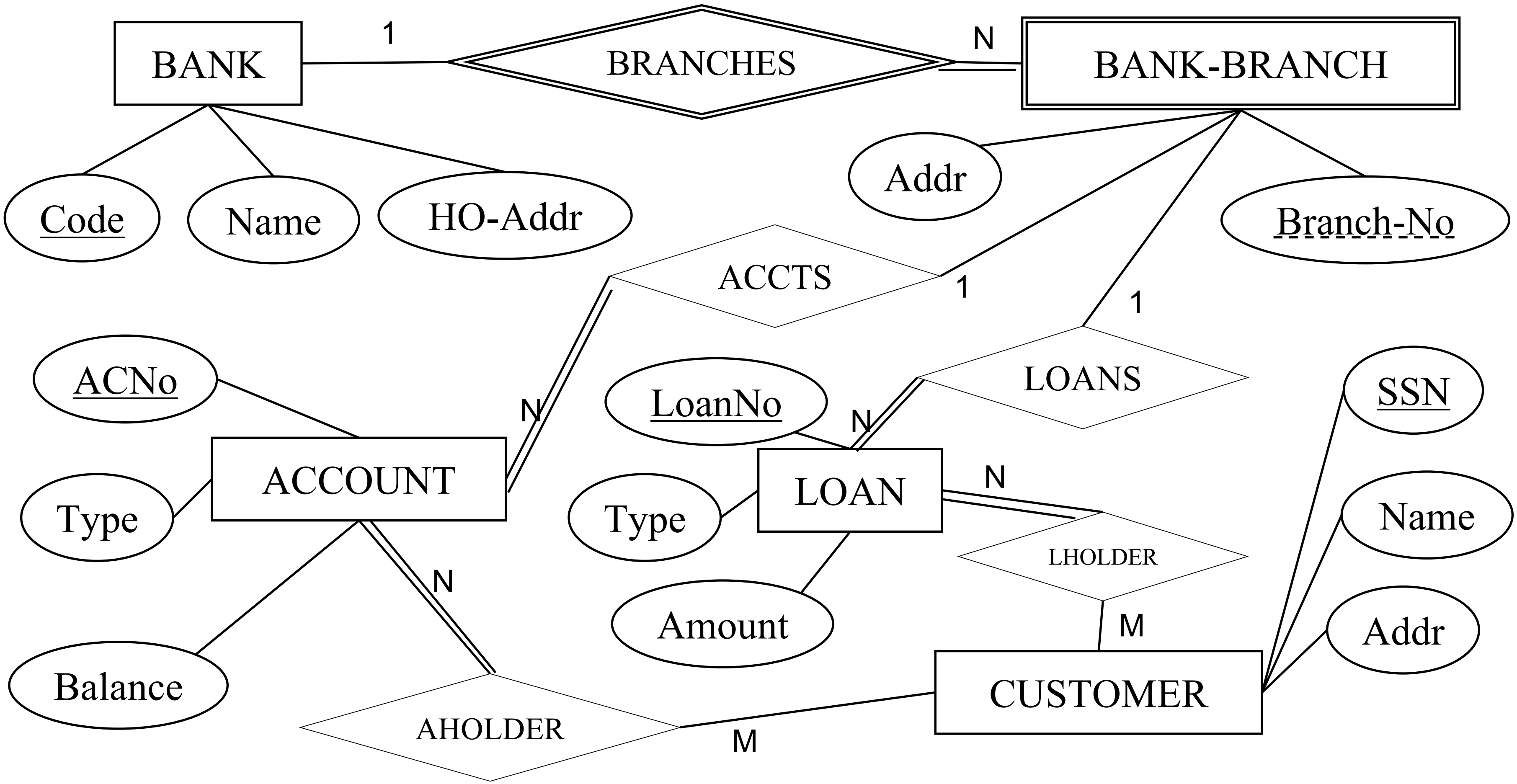
Mapping Example

From Schema to ER Diagrams

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


BANK
BRANCHES
BANK-BRANCH
1
N
Code
Name
HO-Addr
Addr
Branch-No
ACCTS
1
LOANS
1
ACNo
LoanNo
SSN
N
N
ACCOUNT
LOAN
N
Type
Type
Name
LHOLDER
N
Amount
M
Addr
Balance
CUSTOMER
AHOLDER
M

Relational Schema (after step 1)

Relational Schema (after step 2)

Relational Schema (after step 3)

Relational Schema (after step 4)

Relational Schema (after step 5)

A Brief History

The Relational Model

Implementation of Constraints

Mapping ER Diagrams to Relational Models

Mapping Example

From Schema to ER Diagrams

From Schema to ER Diagrams

- Reverse Engineering a table schema is the process of starting with a schema and working back to its ER Diagram.
- This may become useful if you inherit a project which has not been well documented.

A[a, h]
B [a, b]
C [a, c, d]
D (e)
E(a, e)
F(e, f)
G(e, f)
H(h, i)

B.b references A.b
C.a references A.a
A.h references H.h
E.a references A.a
E.e references D.e
F.e reference D.e
G.e referencs D.e

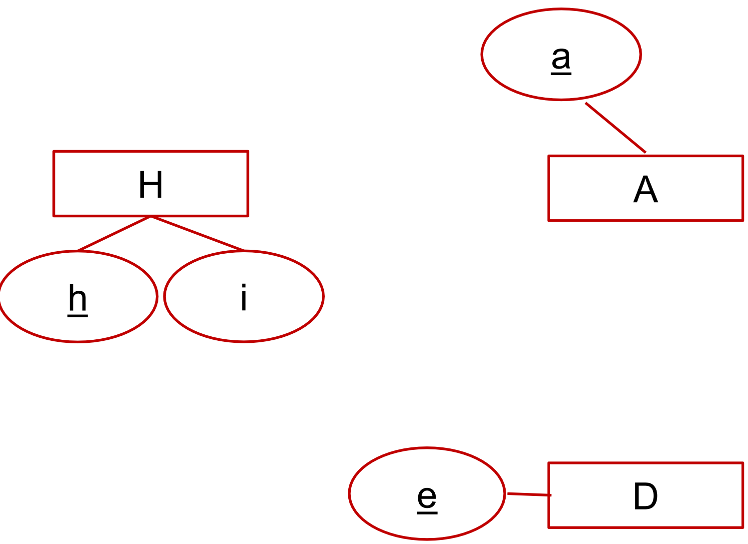
Tips for Reverse Engineering Schema to ER

- There are a few tips and tricks you can use to help identify what each relation is on the ER diagram.

1. Identify strong entities
2. Identify weak entities
3. Identify 1:1 or 1:N relationships
4. Identify M:N relationships
5. Identify multi-valued attributes
6. Identify N-ary relationships
7. Identify super and subclasses

1. Identify Strong Entities

- If the Primary Key of the relation has no Foreign Keys referencing other tables then the relation will be a Strong Entity. Add all attributes that are not keys of other relations.

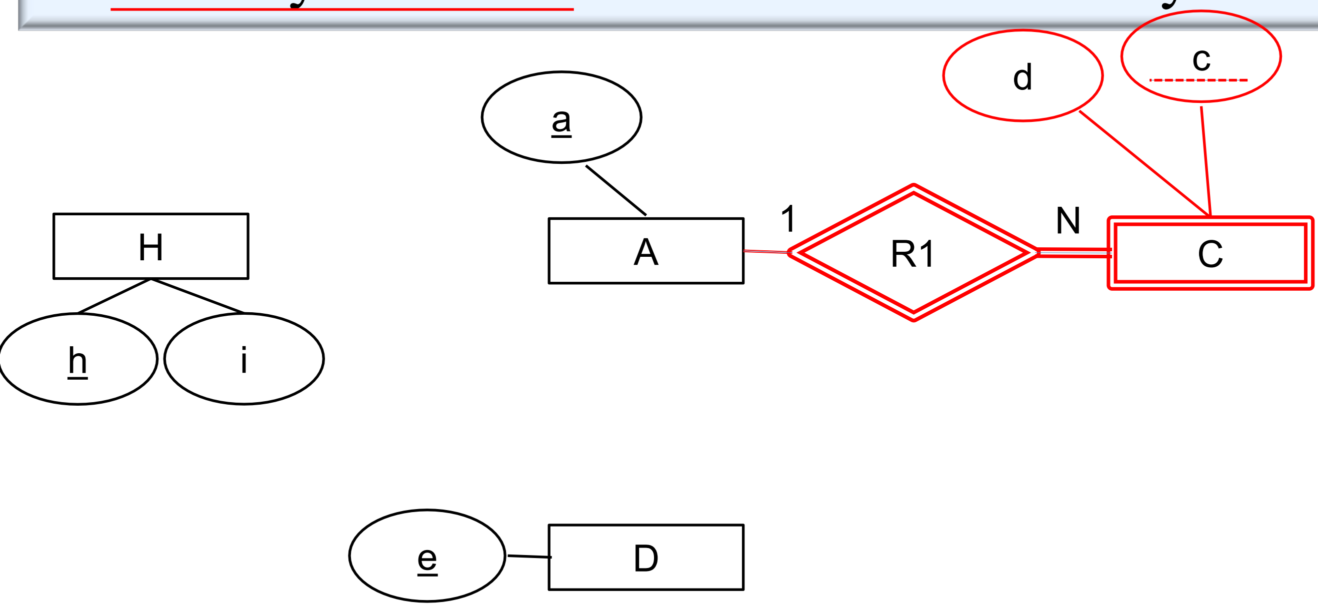


A[a, h]
B [a, b]
C [a, c, d]
D (e)
E(a, e)
F(e, f)
G(e, f)
H(h, i)

B.b references A.b
C.a references A.a
A.h references H.h
E.a references A.a
E.e references D.e
F.e reference D.e
G.e referencs D.e

2. Identify Weak Entities

- If the Primary Key of the relation combines an existing Primary Key (via foreign key link), adds new attributes to the key and has additional non-key attributes it will be a Weak Entity.

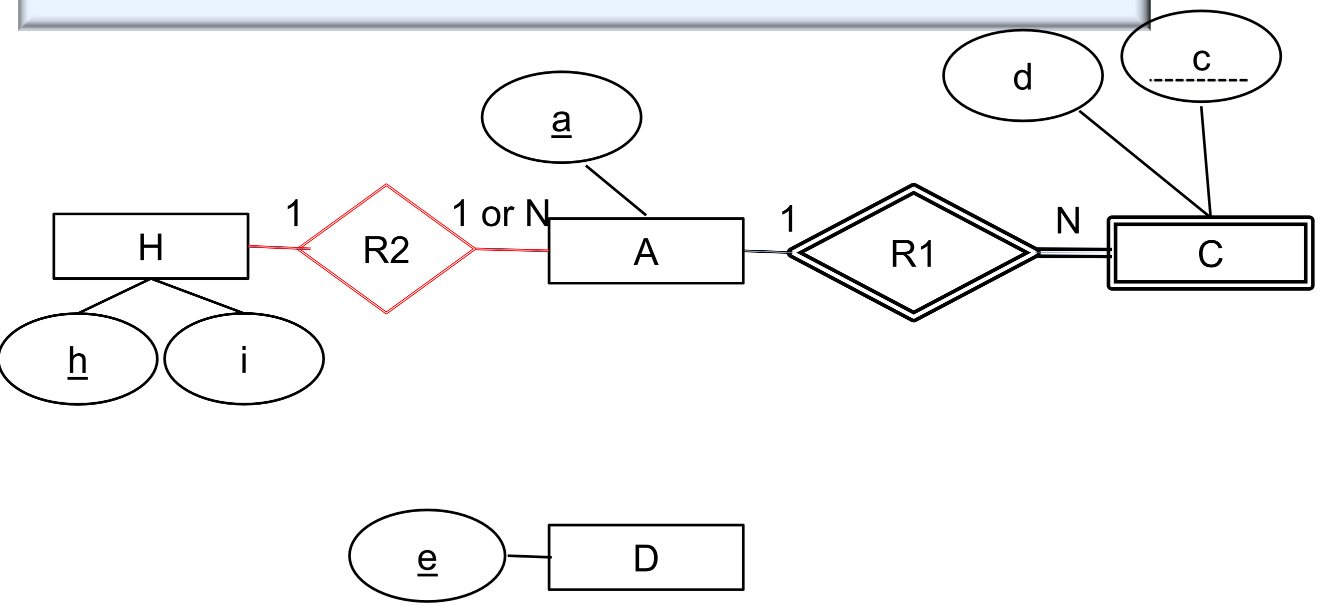


A[a, h]
B [a, b]
C [a, c, d]
D (e)
E(a, e)
F(e, f)
G(e, f)
H(h, i)

B.b references A.b
C.a references A.a
A.h references H.h
E.a references A.a
E.e references D.e
F.e reference D.e
G.e references D.e

3. Identify 1:1 or 1:N Relationships

- If a relation has a Foreign key that is not in the Primary key it indicates a 1:M or 1:1 Relationship between the two tables. This usually can be inferred from context.



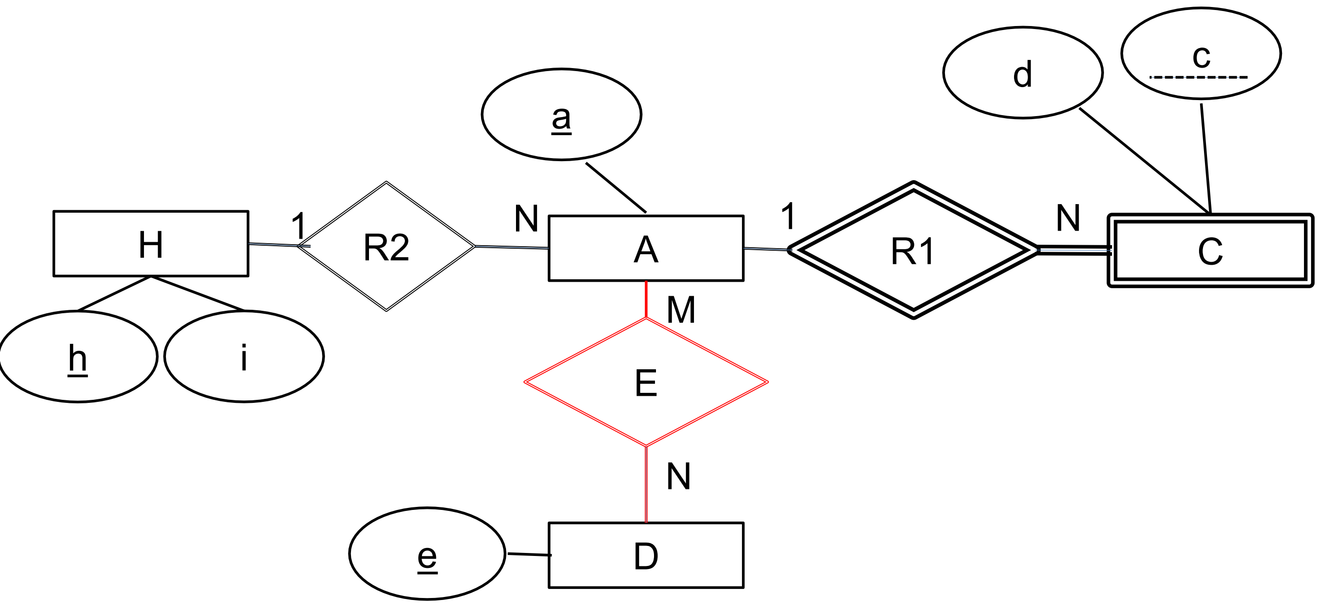
A[a, h]
B [a, b]
C [a, c, d]
D (e)
E(a, e)
F(e, f)
G(e, f)
H(h, i)

B.b references A.b
C.a references A.a
A.h references H.h
E.a references A.a
E.e references D.e
F.e reference D.e
G.e references D.e

4. Identify M:N Relationships

- If the Primary Key of the relation is a composition of existing Primary Keys (via foreign key links, the relation was created due to a M:N Relationship

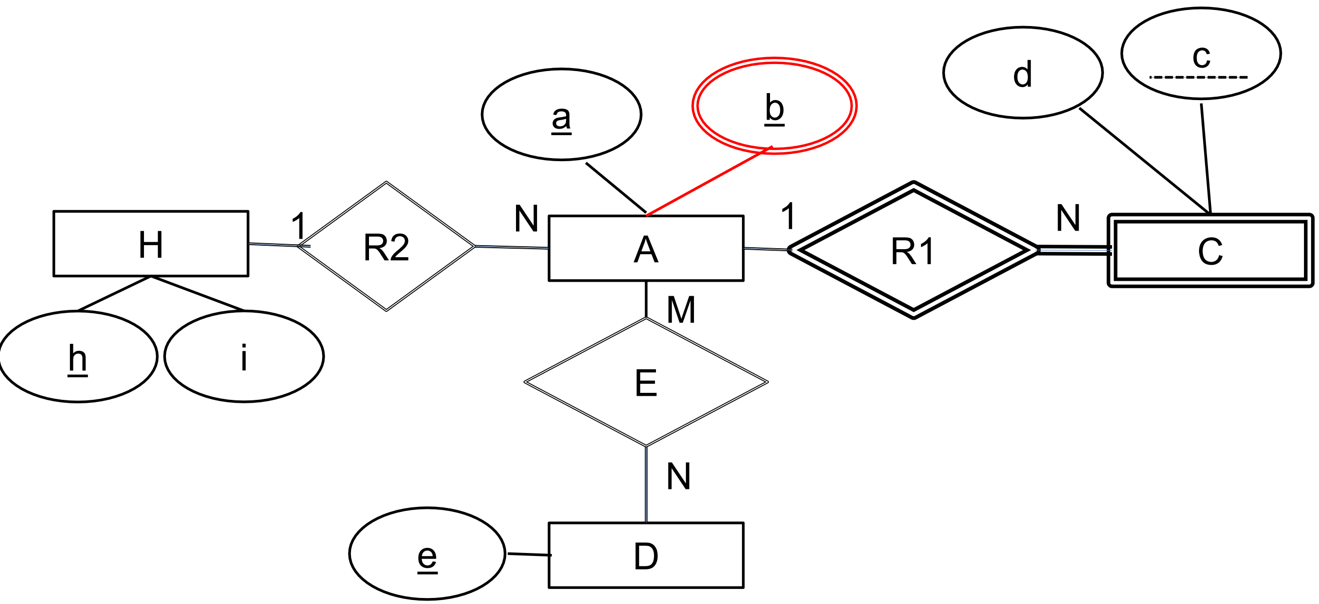
A[a, h]
B [a, b]
C [a, c, d]
D (e)
E(a, e)
F(e, f)
G(e, f)
H(h, i)



B.b references A.b
C.a A.h references H.h
references A.a
E.a references A.a
E.e references D.e
F.e reference D.e
G.e references D.e

5. Identify multi-valued attributes

- If the Primary Key of the relation combines an existing Primary Key (via foreign key link), adds new attributes to the key and has **no** additional non-key attributes it will be a multi-valued attribute

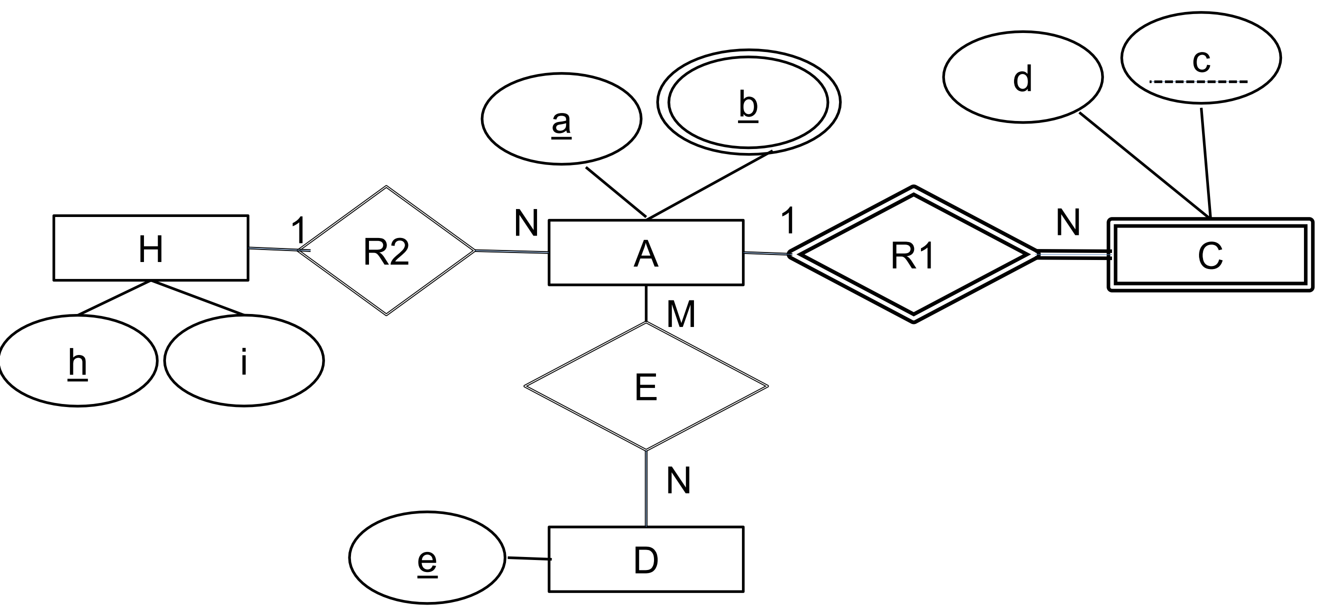


A[a, h]
B [a, b]
C [a, c, d]
D (e)
E(a, e)
F(e, f)
G(e, f)
H(h, i)

B.b references A.b
C.a references A.a
A.h references H.h
E.a references A.a
E.e references D.e
F.e reference D.e
G.e referencs D.e

6. Identify N-ary relationships

- If the Primary Key of the relation is a composition of existing of more than two Primary Keys then it is a N-ary

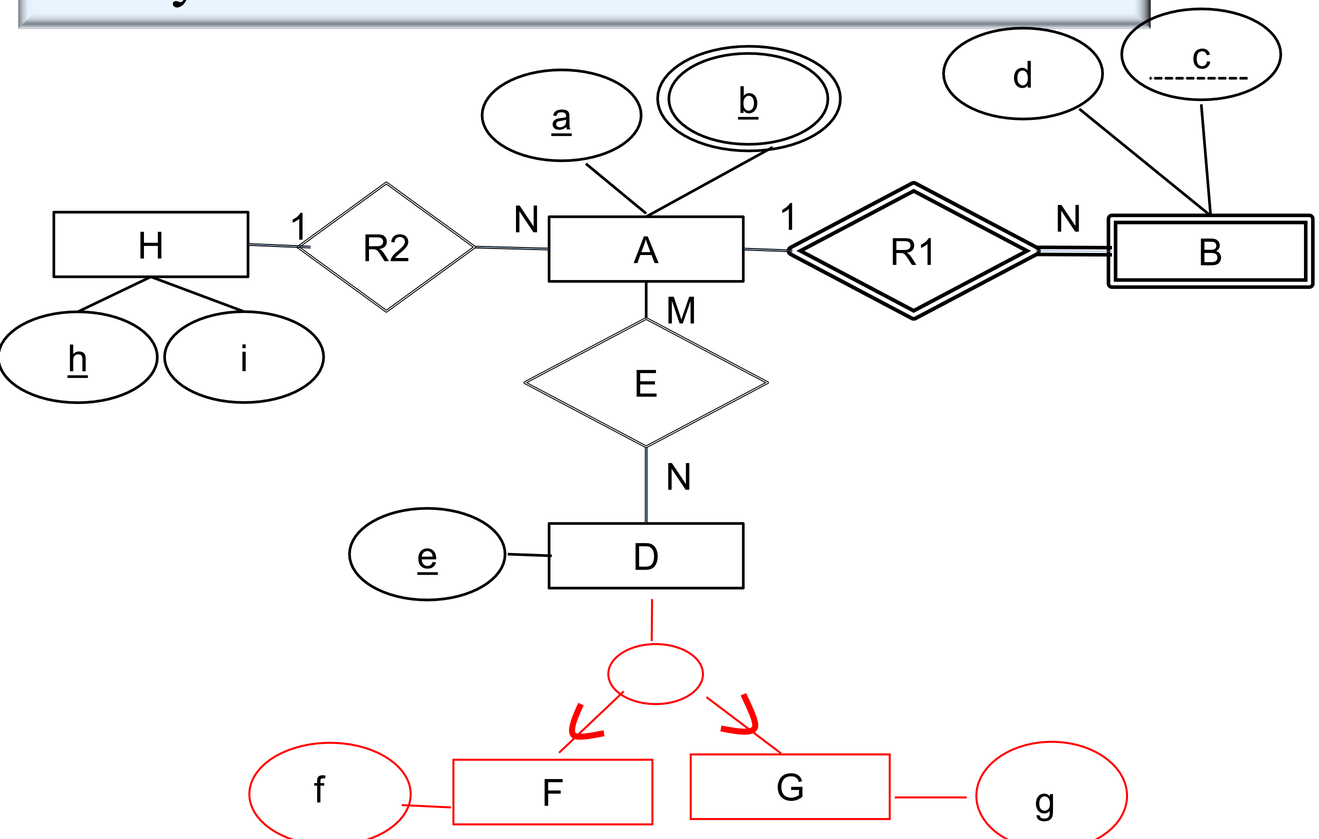


A[a, h]
B [a, b]
C [a, c, d]
D (e)
E(a, e)
F(e, f)
G(e, f)
H(h, i)

B.b references A.b
C.a references A.a
A.h references H.h
E.a references A.a
E.e references D.e
F.e reference D.e
G.e referencs D.e

7. Identify super and subclasses

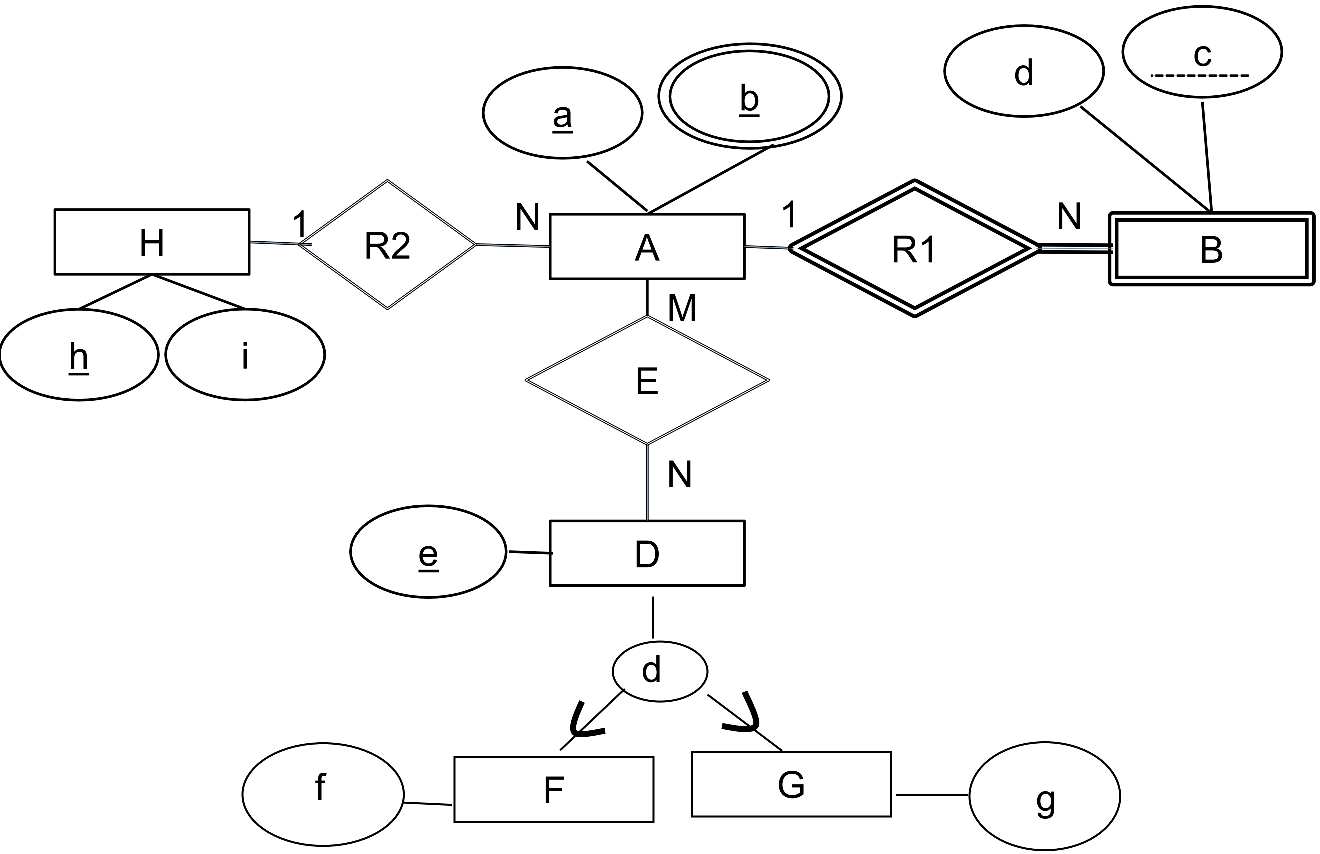
- If a relation has the same Primary Key as an entity it implies that is it a subclass of the entity that the foreign key references.



A[a, h]
B [a, b]
C [a, c, d]
D (e)
E(a, e)
F(e, f)
~~G(e, f)~~ G(e,g)
H(h, i)

B.b references A.b
C.a references A.a
A.h references H.h
E.a references A.a
E.e references D.e
F.e reference D.e
G.e references D.e

Final Results



A[a, h]
B [a, b]
C [a, c, d]
D (e)
E(a, e)
F(e, f)
G(e, f)
H(h, i)

B.b references A.b
C.a references A.a
A.h references H.h
E.a references A.a
E.e references D.e
F.e reference D.e
G.e referencs D.e

Example

Reverse engineering the following table schema to an ER Diagram.

- A[b, c, d, q]
- E[b, f, g]
- H[i, j]
- K[b, i, q l]
- M[q, o]
- P[q, r]

- E.b references A.b
- K.b references A.b
- K.i references H.i
- K.q references P.q
- A.q references P.q
- M.q references P.q

Learning Outcomes Revisited

Description	Tag
Explain the three-Schema architecture, separating internal level, conceptual level and external level.	DBMS
Explain the concept of physical and logical data independence.	
Define the main components of the relational model: Relations, Domains, Attributes and Tuples.	Relational-model
Explain and provide examples for domain constraints.	
Explain and provide examples for key constraints.	
Explain and provide examples for notion of superkeys.	
Explain and provide examples for notion of keys.	
Explain and provide examples for referential integrity constraints.	
Explain and provide examples for user-specified constraints.	
Map entities to relations.	Map-ER-Schema
Map relationships to relations.	
Map weak entities to relations.	
Map Super & Sub-classes to relations.	
Given an ER diagram, map it to a set of relations using the Relational Model.	
Given a relational schema, use the Relational Model to determine the underlying ER diagram.	Map-Schema-ER