CST 204 Database Management Systems

B.Tech. CSE

Semester IV

Viswajyothi College of Engineering and Technology

Sindhu Jose, AP, CSE, VJCET

MODULE 2

Relational Model

Syllabus of Module 2

- Structure of Relational Databases - Integrity Constraints, Synthesizing ER diagram to relational schema
- Introduction to Relational Algebra - select, project, cartesian product operations, join - Equi-join, natural join. query examples,
- Introduction to Structured Query Language (SQL), Data Definition Language (DDL), Table definitions and operations – CREATE, DROP, ALTER, INSERT, DELETE, UPDATE.

Relational data model

- Represent database as a collection of relations.
- Relation is a **table** which has values and rows in table is a collection of related data values.
- Each row in table is a fact.
- Row in relational table is called a **tuple**, column header is **attribute** and table is a relation.
- Every row in the table represents a collection of related data values. These rows in the table denote a real-world entity.
- It is used for data storage and processing.

Attributes

Relation name

EMPLOYEE

EMP_NO	Name	Address	Mobile number	Age	Salary
101	RAM	XYZ	9898989898	20	10000
102	SAM	CVF	9999999999	21	20000
103	SITA	FDFD	888888888	22	30000

Tuples

Components of relational database

The main components of relational database structure are as follows:

1. **Domain**
2. **Tuples (rows)**
3. **Attributes(columns)**
4. **Keys**
5. **Relations (Tables)**

- **Domain:** A set of possible values for a given attribute is known as domain of a relation.
  - Example: Employee_ages. Possible ages of employees in a company; each must be an integer value between 18 and 60.
- **Tuple:** A row in a table represents the record of a relation and known as a tuple of a relation.
- **Columns:** Columns in a table are also called **attributes** or **fields** of the relation.
- **Keys:** Each row has a value of a data item (or set of items) that uniquely identifies that row in the table called the key.
- **Relations:** Relation is a table of values. A relation may be thought of as a set of rows. A relation may alternately be thought of as a set of columns. That is a table is perceived as a two-dimensional structure composed of rows and columns.

Schema of a Relation

- It is basically an outline of how data is organized
- It is denoted by *R* (A1, A2, …..An)
  - Here R is relation name and
  - It has some attributes A1 to An
- Each attribute have some domain and it is represented by dom(Ai)
- Relation schema is used to describe a relation and R is name of the relation
- Each attribute has a **domain** or a set of valid values.
  - For example, the domain of Cust-id is 6 digit numbers.

Degree of a relation

- Degree of a relation is number of attributes in a relation
- Eg: STUDENT(Id, Name, Age, Deptno)
  - Has degree 4
- Using datatype of each the definition can be written as

  STUDENT(Id:Integer, Name:String, Age:integer, Deptno:integer)

Relation State

- The **relation state** is a subset of the Cartesian product of the domains of its attributes
  - each domain contains the set of all possible values the attribute can take.
- Example: attribute Cust-name is defined over the domain of character strings of maximum length 25
  - dom(Cust-name) is varchar(25)
- A relation state r(R) is a mathematical relation of degree n on the domains dom(A1), dom(A2)…, dom(An) which is a subset of Cartesian product(X) of domains that define R
- Cartesian product specifies all possible combination of values from underlying domains

Formal Definitions - Summary

- Formally,
  - Given R(A1, A2, ..., An) [Eg: STUDENT(Id, Name, Age, Deptno)]
  - r(R) ⊂ dom (A1) X dom (A2) X ....X dom(An)
- R(A1, A2, …, An) is the **schema** of the relation
- R is the **name** of the relation
- A1, A2, …, An are the **attributes** of the relation
- r(R): a specific **state** (or "value" or "population") of relation R – this is a *set of tuples* (rows)
  - r(R) = {t1, t2, …, tn} where each ti is an n-tuple
  - ti = <v1, v2, …, vn> where each vj *element-of* dom(Aj)
- We refer to **component values** of a tuple t by:
  - t[Ai] or t.Ai.

Relational Database Schema

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

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

DEPT_LOCATIONS

Dnumber	Dlocation

PROJECT

Pname	Pnumber	Plocation	Dnum

WORKS_ON

Essn	Pno	Hours

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship

Figure 5.5
Schema diagram for the COMPANY relational database schema.

Relational Integrity Constraints

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

Key Constraints

Superkey of R: A super key is a group of single or multiple keys which identifies rows in a table.

- It is a set of attributes SK of R with the following condition:
  - No two tuples in any valid relation state r(R) will have the same value for SK
  - That is, for any distinct tuples t1 and t2 in r(R), $t1[SK] \neq t2[SK]$

Key of R:

- It is a "minimal" superkey
- That is, a key is a superkey K such that removal of any attribute from K results in a set of attributes that is not a superkey (does not possess the superkey uniqueness property).
- Relation may have more than one key. In such cases each of the keys are called **candidate keys**.

- Example: Consider the CAR relation schema:
  - CAR(State, Reg#, SerialNo, Make, Model, Year)
  - CAR has two keys(or candidate keys):
    - Key1 = {State, Reg#}
    - Key2 = {SerialNo}
  - Both are also superkeys of CAR
  - {SerialNo, Make,Model} is also a **superkey** but ***not*** a **key**. A super key may contain extra attributes that are not necessary to uniquely identify records like {Make,Model} in {SerialNo, Make,Model} .

- In general:
  - Any *key* is a *superkey* (but not vice versa)
  - Any set of attributes that *includes a key* is a *superkey*
  - A *minimal* superkey is also a key

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

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

CAR

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

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

Entity Integrity constraint

- **Entity Integrity:**
  - The **primary key attributes** PK of each relation schema R in S **cannot have null values** in any tuple of r(R).
    - This is because primary key values are used to identify the individual tuples.
    - $t[PK] \neq null$ for any tuple t in r(R)
    - If PK has several attributes, null is not allowed in any of these attributes

Referential Integrity constraint(Foreign key)

- This constraint involves **two** relations
- Used to specify a **relationship** among tuples in **two** relations:
  - **1. Referencing relation** and **2.** Re**ferenced relation**.
- Tuples in the **referencing relation** R1 have attributes FK (called **foreign key** attributes) that reference the primary key attributes PK of the **referenced relation** R2.
  - A tuple t1 in R1 is said to **reference** a tuple t2 in R2 if t1[FK] = t2[PK].
- A referential integrity constraint can be displayed in a relational database schema as a directed arc from R1.FK to R2.

Example (Foreign key)

In this example Stud_id is the Primary key of Student and Foreign key of Department

Student — Primary Key

Stud_Id	Name	Course
101	John	Computer
105	Merry	AI
107	Sheero	Biology
108	Bisle	Maths

R1

Department — Foreign Key

Dept_name	Stud_Id
CS_Department	105
CS_Department	101
Science_Department	101
Maths_Department	108

R2

Update Operations on Relations

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

Dealing with constraint violation due to insertion,deletion and updation of database

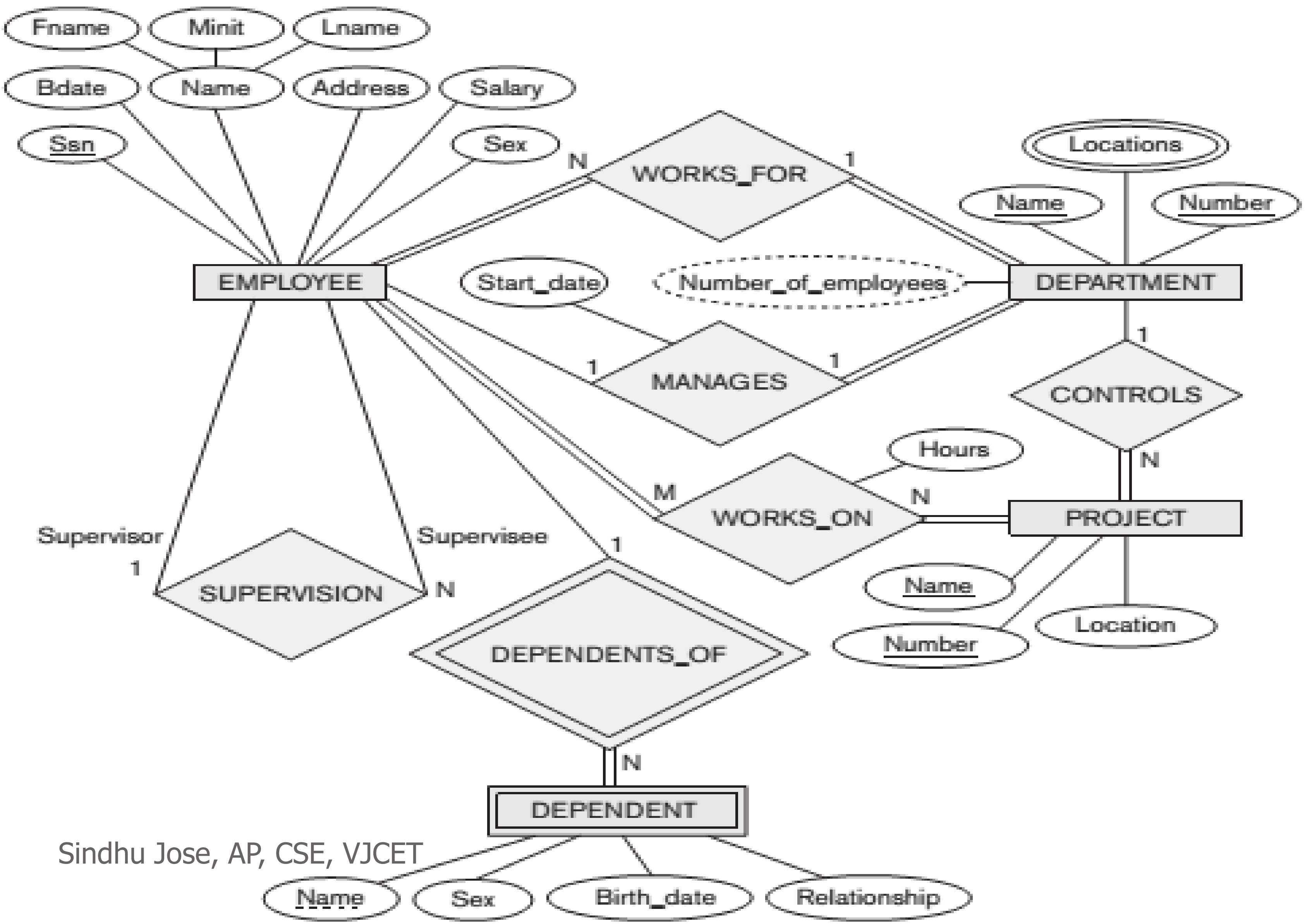
- **INSERT** may violate any of the constraints:
  - Domain constraint:
    - If one of the attribute values provided for the new tuple is not of the specified attribute domain
  - Key constraint:
    - If the value of a key attribute in the new tuple already exists in another tuple in the relation
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

- **DELETE** may violate only referential integrity:
  - If the primary key value of the tuple being deleted is referenced from other tuples in the database
    - Can be remedied by several actions: RESTRICT, CASCADE, SET NULL.
      - RESTRICT option: reject the deletion
      - CASCADE option: propagate deletion by deleting the tuples that reference the tuple that is being deleted.
      - SET NULL option: set the foreign keys of the referencing tuples to NULL

UPDATE : Updating an attribute that is neither part of a primary key nor of a foreign key usually causes no problems.

- Modifying a primary key value is similar to deleting one tuple and inserting another in its place. Hence the same measures can be taken as in INSERT and DELETE operation (Eg: RESTRICT, CASCADE etc)

Figure 9.1
The ER conceptual schema diagram for the COMPANY database.



Sindhu Jose, AP, CSE, VJCET

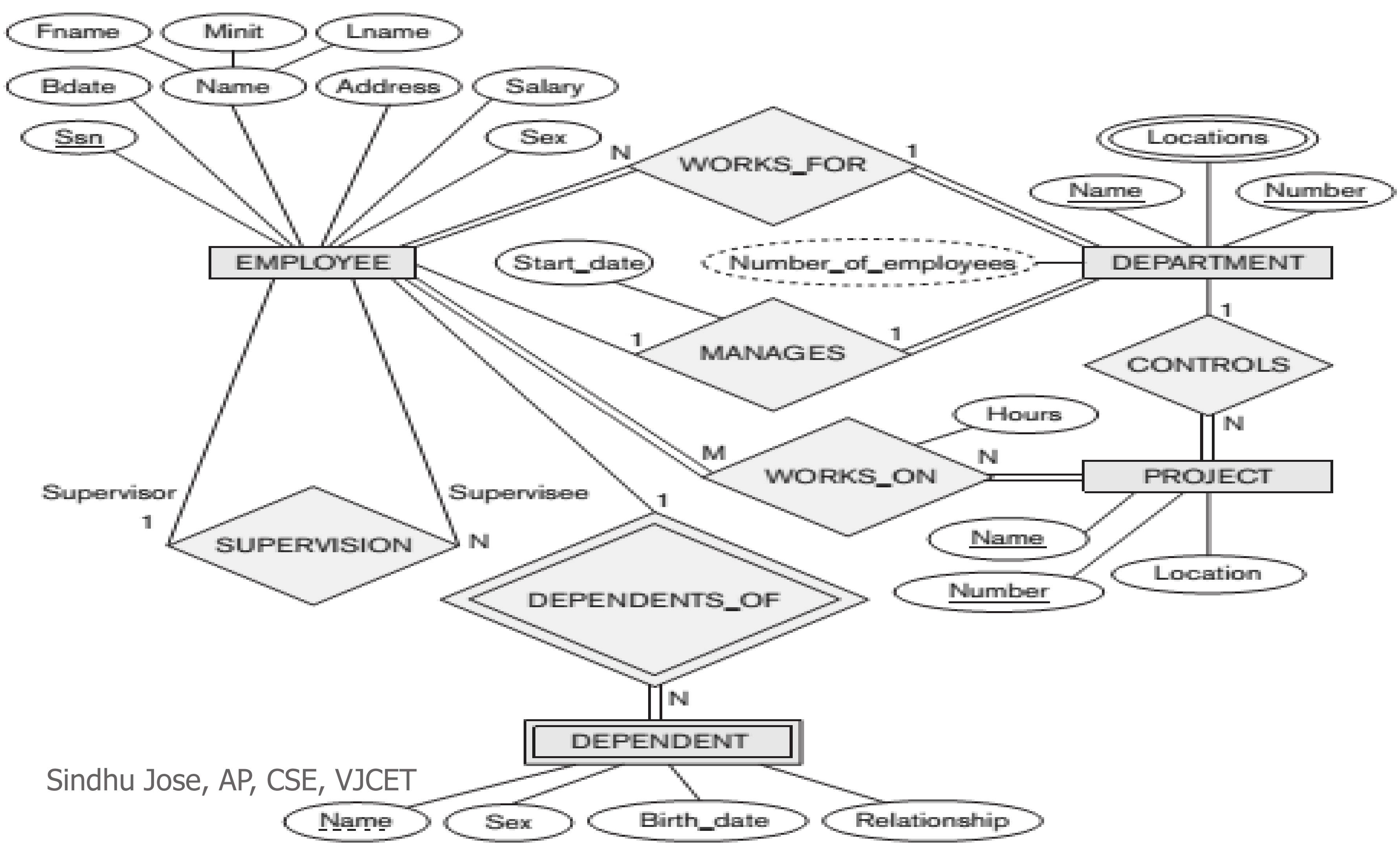
ER model to Relational model mapping

7-Step Process:

1. Map Regular Entity Types
2. Map Weak Entity Types
3. Map Binary 1:1 Relation Types
4. Map Binary 1:N Relationship Types.
5. Map Binary M:N Relationship Types.
6. Map Multivalued attributes.
7. Map N-ary Relationship Types.

Company ER schema

Figure 9.1
The ER conceptual schema diagram for the COMPANY database.



Sindhu Jose, AP, CSE, VJCET

Step 1: Map Regular Entity Types

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

(a) **EMPLOYEE**

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary

DEPARTMENT

Dname	<u>Dnumber</u>

PROJECT

Pname	<u>Pnumber</u>	Plocation

Step 2: Map Weak Entity Types

- For each weak entity type W in the ER schema with owner entity type E, create a relation R & include all simple attributes (or simple components of composite attributes) of W as attributes of R.
- Also, include as foreign key attributes of R the primary key attribute(s) of the relation(s) that correspond to the owner entity type(s).
- Weak Entity Relation after Step 2 is given below

(b) **DEPENDENT**

Essn	Dependent_name	Sex	Bdate	Relationship

Step 3: Map Binary 1:1 Relation Types

- For each binary 1:1 relationship type R in the ER schema, identify the relations S and T that correspond to the entity types participating in R.
- Choose one of the relations-say S-and include a foreign key in S the primary key of T. It is better to choose an entity type with total participation in R in the role of S.
- For eg: Manages is 1:1 relation between Department and Employee. So add Manager_ssn as foreign key and include simple attributes in Department table.Also add simple attributes of one to one relationship as attribute of S

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

Step 4 : Map Binary 1:N Relationship Types.

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

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

PROJECT

Pname	Pnumber	Plocation	Dnum

Step 5 : Map Binary M:N Relationship Types.

- For each regular binary M:N relationship type R, create a new relation S to represent R.
- Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types; their combination will form the primary key of S.
- Also include any simple attributes of the M:N relationship type (or simple components of composite attributes) as attributes of S.
- Ex: Relation WORKS_ON
- Include the primary keys of the PROJECT and EMPLOYEE relations as foreign keys in WORKS_ON.
- Include an attribute Hours in WORKS_ON to represent the Hours attribute of the relationship type.
- The primary key of the WORKS_ON relation is the combination of the foreign key attributes {Essn, Pno}.

- **Result after Step 5**

(c) WORKS_ON

Essn	Pno	Hours

Step 6: Map Multivalued attributes

- For each multivalued attribute A, create a new relation R. ◦ This relation R will include an attribute corresponding to A, plus the primary key attribute K-as a foreign key in R-of the relation that represents the entity type of relationship type that has A as an attribute.
- The primary key of R is the combination of A and K. If the multivalued attribute is composite, we include its simple components.
- For ex: Create a relation DEPT_LOCATIONS.
- The attribute Dlocation represents the multivalued attribute LOCATIONS of DEPARTMENT, whereas Dnumber—as foreign key—represents the primary key of the DEPARTMENT relation.
- The primary key of DEPT_LOCATIONS is the combination of {Dnumber, Dlocation} .

- **Result after Step 6**

(d) DEPT_LOCATIONS

<u>Dnumber</u>	<u>Dlocation</u>

Step 7: Map N-ary Relationship Types.

- For each n-ary relationship type R, where n>2, create a new relationship S to represent R.
- Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types.
- Also include any simple attributes of the n-ary relationship type (or simple components of composite attributes) as attributes of S.

Example

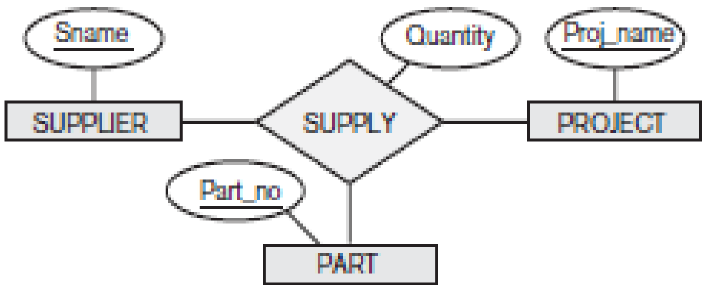
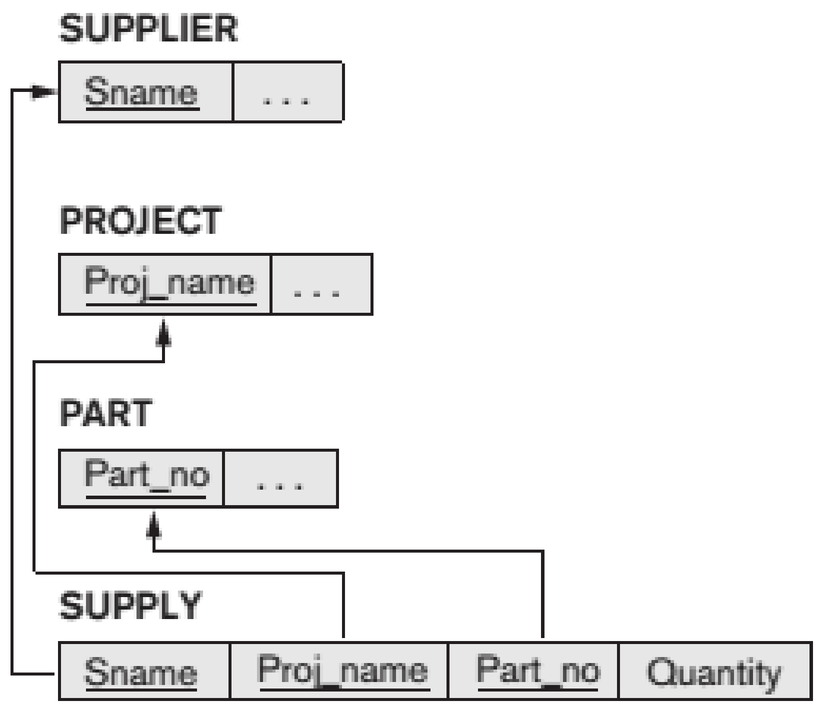


Figure 9.4
Mapping the *n*-ary relationship type SUPPLY from Figure 3.17(a).



Displaying a relational database schema and its constraints

- Each relation schema can be displayed as a row of attribute names
- The name of the relation is written above the attribute names
- The primary key attribute (or attributes) will be underlined
- A foreign key (referential integrity) constraints is displayed as a directed arc (arrow) from the foreign key attributes to the referenced table.
- Next slide shows the COMPANY **relational schema diagram**

Company Relational Database schema

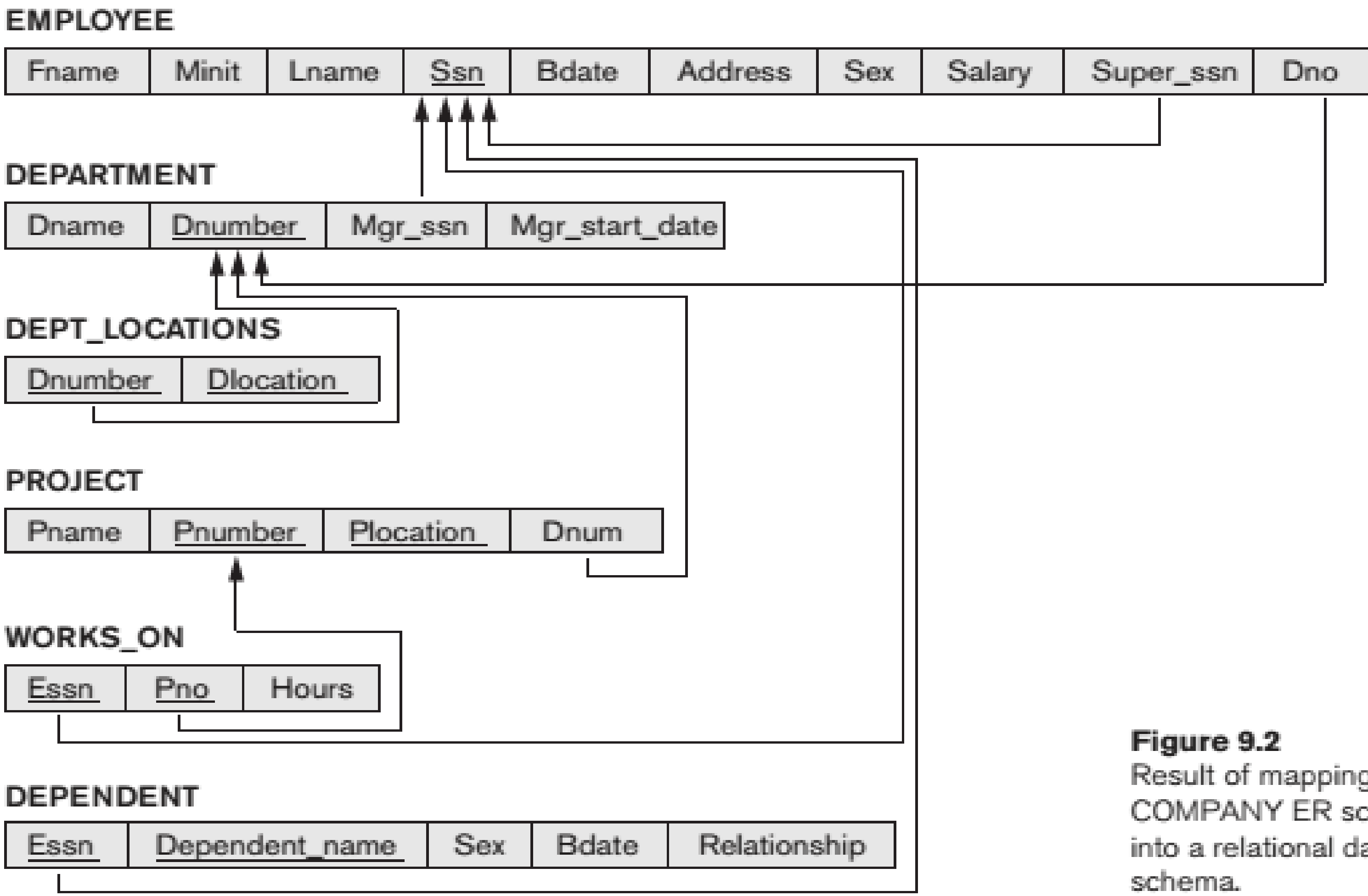


Figure 9.2
Result of mapping the COMPANY ER schema into a relational database schema.