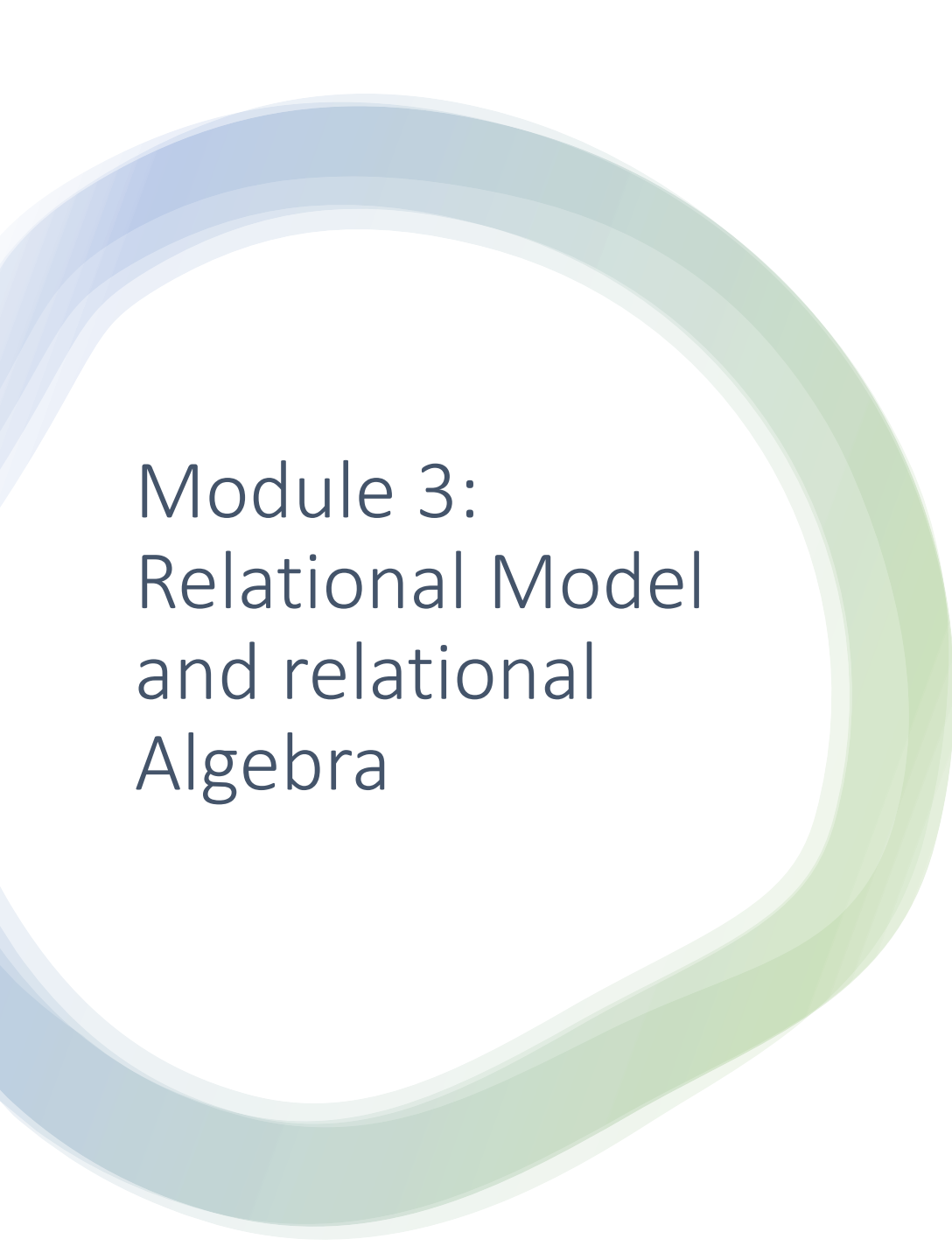


# DATABASE MANAGEMENT SYSTEMS

Couse code:CSC403

Prof. Juhi Janjua



## Module 3: Relational Model and relational Algebra

- Introduction to the Relational Model
- Relational schema and concept of keys.
- Mapping the ER and EER Model to the Relational Model
- Relational Algebra-operators
- Relational Algebra Queries.

# Introduction to Relational Model

- Relational Model was proposed by E.F. Codd to model data in the form of relations or tables.
- After designing the conceptual model of Database using ER diagram, we need to convert the conceptual model in the relational model which can be implemented using any RDBMS languages like Oracle SQL, MySQL etc.
- So, we will see what Relational Model is.

# Relational Model

- It represents how data is stored in Relational Databases.
- It stores data in the form of relations (tables).

**Example: STUDENT Relation**

NAME	ROLL_NO	PHONE_NO	ADDRESS	AGE
Ram	14795	7305758992	Noida	24
Shyam	12839	9026288936	Delhi	35
Laxman	33289	8583287182	Gurugram	20
Mahesh	27857	7086819134	Ghaziabad	27
Ganesh	17282	9028 9i3988	Delhi	40

# Terminologies

**Attribute**: properties that define a relation.

e.g.; ROLL\_NO, NAME

**Relation Schema**: represents name of the relation with its attributes.

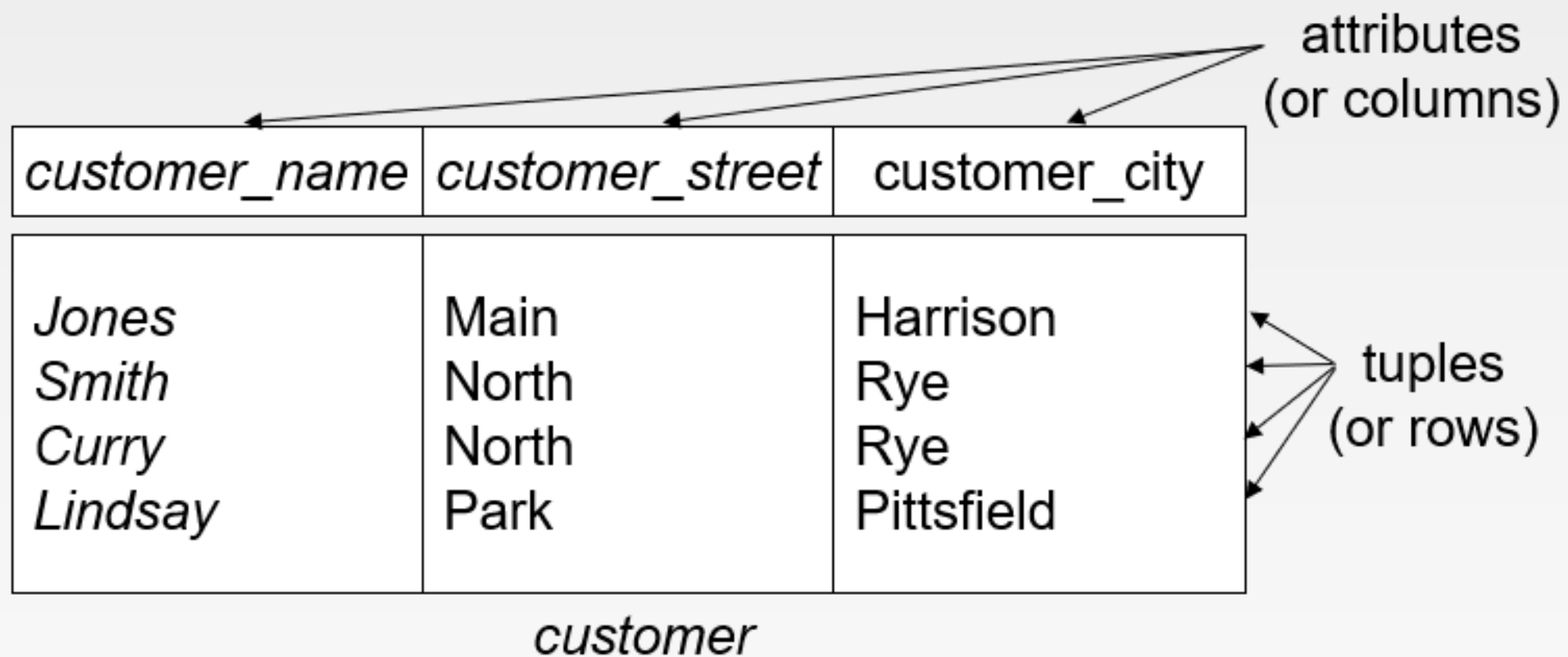
e.g.; STUDENT (ROLL\_NO, NAME, ADDRESS, PHONE and AGE) is relation schema for STUDENT.

If a schema has more than 1 relation, it is called Relational Schema.

**Tuple**: each row in the relation is known as tuple.

In student relation contains 5 tuples, one of which is shown as:

Ganesh	17282	9028 9i3988	Delhi	40
--------	-------	-------------	-------	----



The diagram illustrates a database table structure. The table is labeled *customer* at the bottom. It consists of three columns: *customer\_name*, *customer\_street*, and *customer\_city*. The first column contains four entries: *Jones*, *Smith*, *Curry*, and *Lindsay*. The second column contains three entries: *Main*, *North*, and *Park*. The third column contains three entries: *Harrison*, *Rye*, and *Pittsfield*. Annotations on the right side of the table include:
 

- An arrow pointing to the column headers with the text "attributes (or columns)".
- An arrow pointing to the data rows with the text "tuples (or rows)".

<i>customer_name</i>	<i>customer_street</i>	<i>customer_city</i>
<i>Jones</i>	<i>Main</i>	<i>Harrison</i>
<i>Smith</i>	<i>North</i>	<i>Rye</i>
<i>Curry</i>	<i>North</i>	<i>Rye</i>
<i>Lindsay</i>	<i>Park</i>	<i>Pittsfield</i>

*customer*

# Terminologies...

**Relation Instance:** set of tuples of a relation at a particular instance of time

Table shows the relation instance of STUDENT at a particular time.

It can change whenever there is insertion, deletion or update in the database.

**Degree:** no. of attributes in the relation is known as degree of the relation.

**STUDENT** relation defined above has degree 5.

**Cardinality:** no. of tuples in a relation is known as cardinality.

The **STUDENT** relation defined above has cardinality 5.

# Properties of Relations

- Name of the relation is distinct from all other relations.
- Each relation cell contains exactly one atomic (single) value
- Each attribute contains a distinct name
- Tuple has no duplicate value
- Order of tuples is irrelevant (tuples may be stored in an arbitrary order)



# Relation Schema in DBMS

- It defines the design and structure of the relation.
- It consists of the relation name & set of attributes.

Example:

There is a student named Smith, he is pursuing B.Tech, in the 4th year, and belongs to Computer department(deptno. 1) and has roll number 16047 She is mentored by Mrs. S Mohanty.

If we want to represent this using databases we would have to create a student table with name, degree, year, department, department number, roll number and mentor as the attributes.

student(rollNo,name,degree,year,deptNo,mentor)

# Relation schema...

This and other departments can be represented by the department table, having department ID, name and hod as attributes.

department(deptID,name,hod)

The course that a student has selected has a courseid, course name, credit and department number.

course(courseId,cname,credits,deptNo)

The professor would have an employee Id, name, department no. and phone number.

professor(empId,name,deptNo,phoneNo)

# Relation schema...

We can have another table named enrollment (relationship between course & student), which has roll no, course id, semester, year and grade as the attributes.

enrollment(rollNo,courseId,sem,year,grade)

Teaching (relationship between professor & course) can be another table, having employee id, course id, semester, year and classroom as attributes

teaching(empId,courseId,sem,year,classroom)

And so on...

Relations between them is represented through arrows

# Concept of keys

- **Keys** are defined to easily identify any row of data in a table.
- Let's try to understand about all the keys using a student table with fields student-id, name, phone & age

student_id	name	phone	age
1	Akon	9876723452	17
2	Akon	9991165674	19
3	Bkon	7898756543	18
4	Ckon	8987867898	19
5	Dkon	9990080080	17

# Super Key

- **Super Key** is defined as a set of attributes within a table that can uniquely identify each record within a table.
- It is a superset of Candidate key.

Example:

Considering student table

Super key: {student-id}, {student-id, name}, {phone}

# Candidate Key

- They are defined as the minimal set of fields which can uniquely identify each record in a table.
- It is an attribute or a set of attributes that can act as a primary Key for a table to uniquely identify each record in that table.

Example:

student-id & phone are candidate keys of student table

# Properties of candidate key

- A candidate key can never be NULL or empty.
- Its value should be unique.
- There can be more than one candidate keys for a table.
- A candidate key can be a combination of more than one columns(attributes).

# Primary Key

- There can be more than one candidate key in relation out of which one can be chosen as the primary key.
- For the table Student we can make the student\_id column as the primary key.

Primary Key for this table



student_id	name	age	phone



# Foreign Key

- It is a column that creates a relationship between two tables.
- The purpose of Foreign keys is to maintain data integrity and allow navigation between two different instances of an entity.
- It acts as a cross-reference between two tables as it references the primary key of another table.

# Example

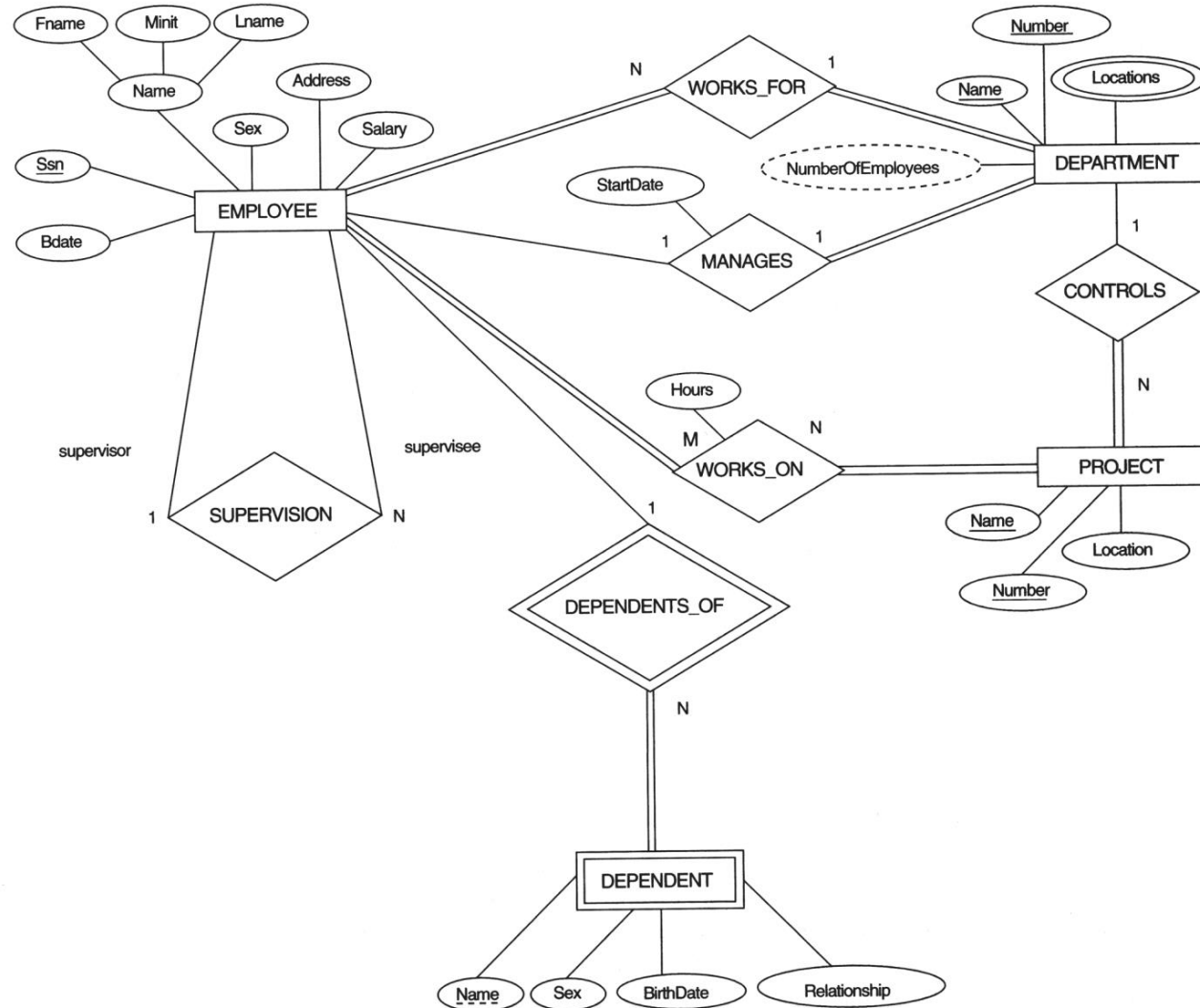
Example:

DeptCode	DeptName
001	Science
002	English
005	Computer

Teacher ID	Fname	Lname
B002	David	Warner
B017	Sara	Joseph
B009	Mike	Brunton

- In this example, we have two table, teach and department in a school. However, there is no way to see which teacher work in which department.
- In this table, adding the foreign key DeptCode to the teach table, we can create a relationship between the two tables.
- This concept is also known as Referential Integrity.

# Mapping the ER and EER Model to the Relational Model



# Steps for mapping

- **ER-to-Relational Mapping Algorithm**

Step 1: Mapping of Regular Entity Types

Step 2: Mapping of Weak Entity Types

Step 3: Mapping of Binary 1:1 Relation Types

Step 4: Mapping of Binary 1:N Relationship Types.

Step 5: Mapping of Binary M:N Relationship Types.

Step 6: Mapping of Multivalued attributes.

Step 7: Mapping of N-ary Relationship Types.

- **Mapping EER Model Constructs to Relations**

Step 8: Options for Mapping Specialization or Generalization.

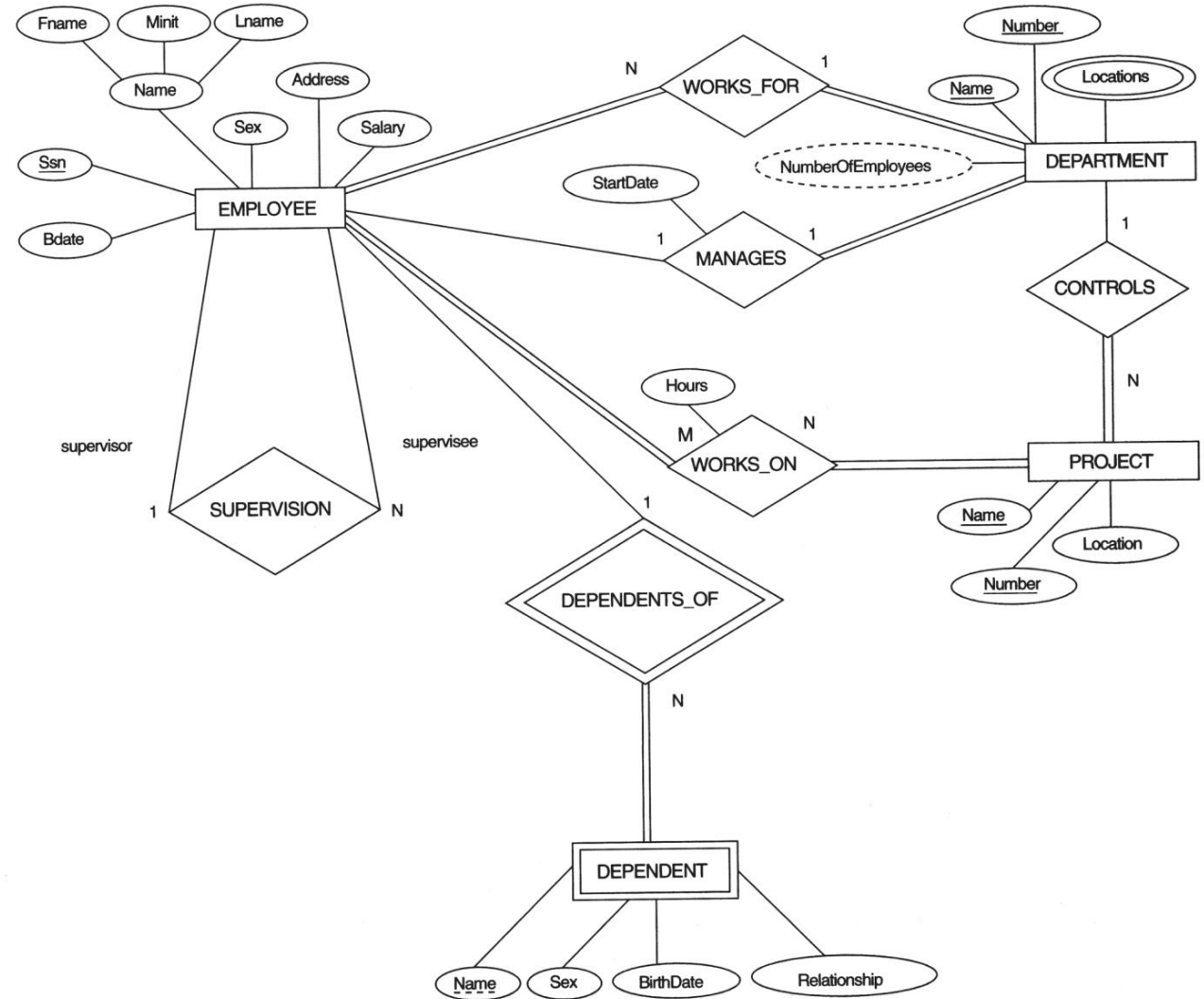
Step 9: Mapping of Union Types (Categories).

# Step 1: Mapping of 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  $R$ .

## Step 1

**Example:** We create the relations EMPLOYEE, DEPARTMENT, and PROJECT in the relational schema corresponding to the regular entities in the ER diagram. SSN, DNUMBER, and PNUMBER are the primary keys for the relations EMPLOYEE, DEPARTMENT, and PROJECT as shown.



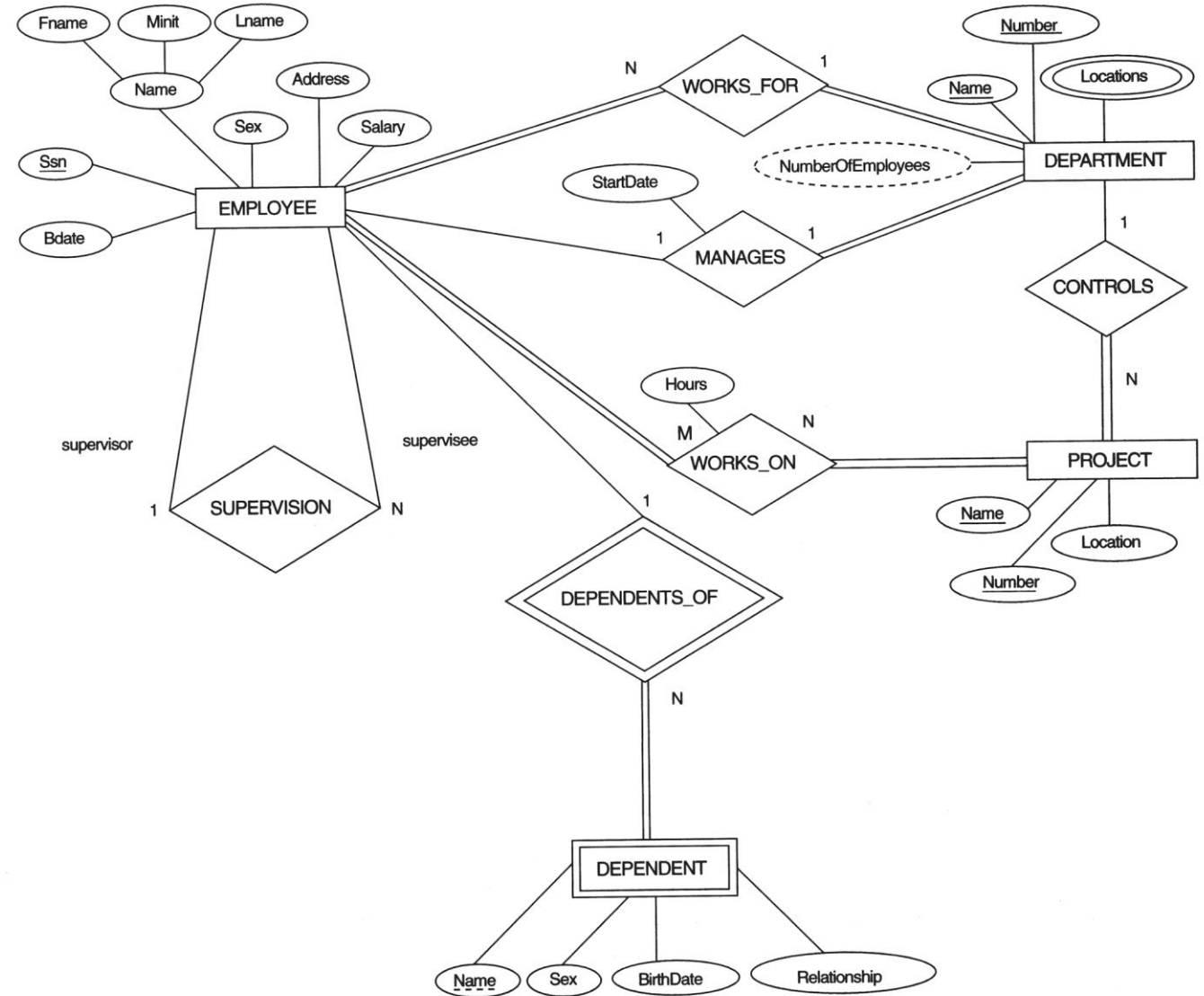
## Step 2: Mapping of Weak Entity Types

- For each weak entity type  $W$  in the ER schema with owner entity type  $E$ , create a relation  $R$  and include all simple attributes (or simple components of composite attributes) of  $W$  as attributes of  $R$ .
- In addition, include as foreign key attributes of  $R$  the primary key attribute(s) of the relation(s) that correspond to the owner entity type(s).
- The primary key of  $R$  is the *combination of* the primary key(s) of the owner(s) and the partial key of the weak entity type  $W$ , if any.

## Step 2

**Example:** Create the relation **DEPENDENT** in this step to correspond to the weak entity type **DEPENDENT**. Include the primary key **SSN** of the **EMPLOYEE** relation as a foreign key attribute of **DEPENDENT** (renamed to **ESSN**).

The primary key of the **DEPENDENT** relation is the combination {**ESSN**, **DEPENDENT\_NAME**} because **DEPENDENT\_NAME** is the partial key of **DEPENDENT**.





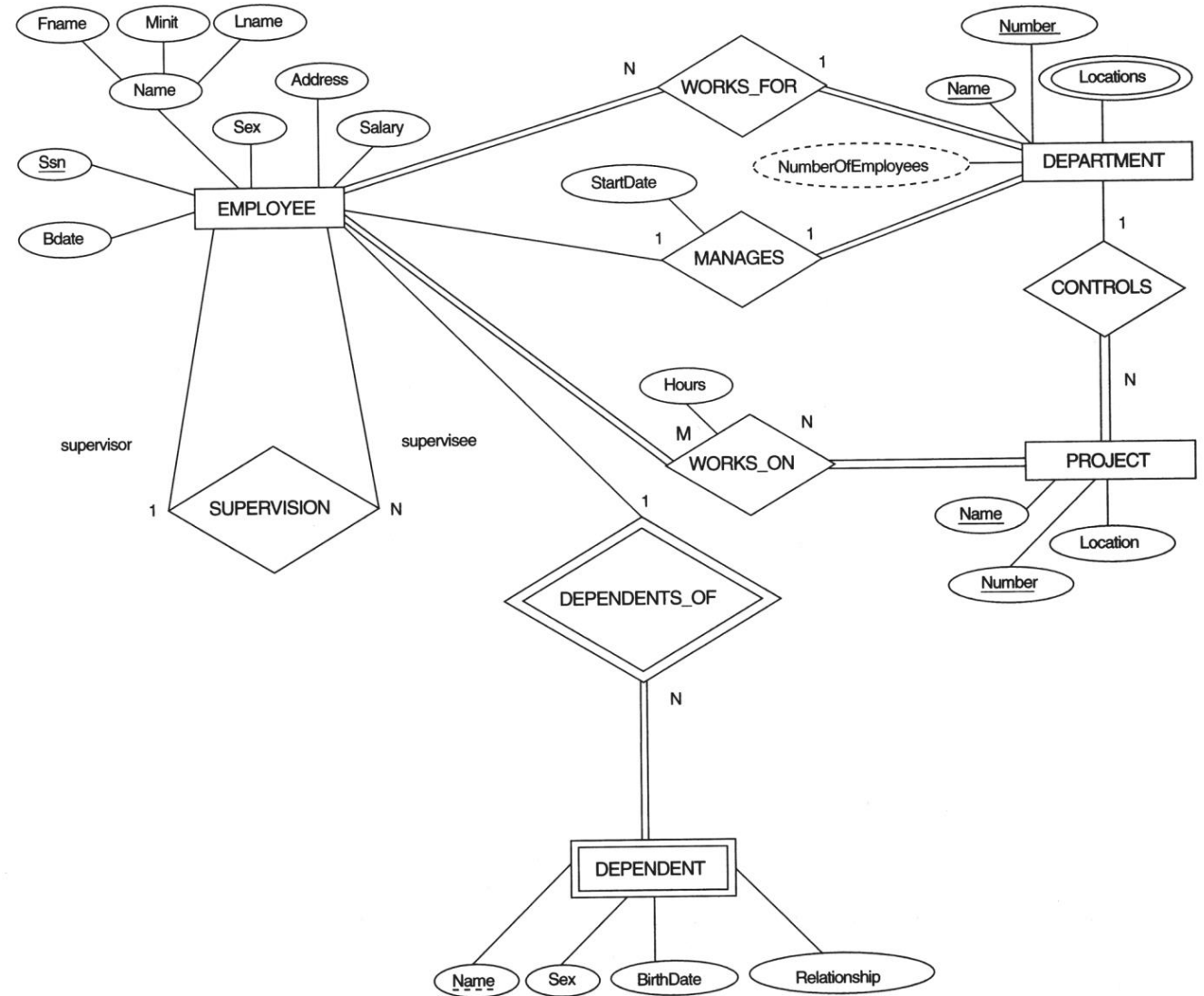
## Step 3: Mapping of Binary 1:1 Relation Types

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. There are three possible approaches:

- (1) Foreign Key approach: Choose the relations with ***total participation*** in R – say S-- and include T's primary key in S.
- (2) Merged relation option: An alternate mapping of a 1:1 relationship type is possible by merging the two entity types and the relationship into a single relation. This may be appropriate when ***both participations are total***.
- (3) Cross-reference or relationship relation option: The third alternative is to set up a third relation ***W(T.primarykey, S.primaryKey)*** for the purpose of cross-referencing the primary keys of the two relations S and T representing the entity types.

## Step 3 (Foreign key approach)

**Example:** 1:1 relation MANAGES is mapped by choosing the participating entity type DEPARTMENT to serve in the role of S, because its participation in the MANAGES relationship type is total.

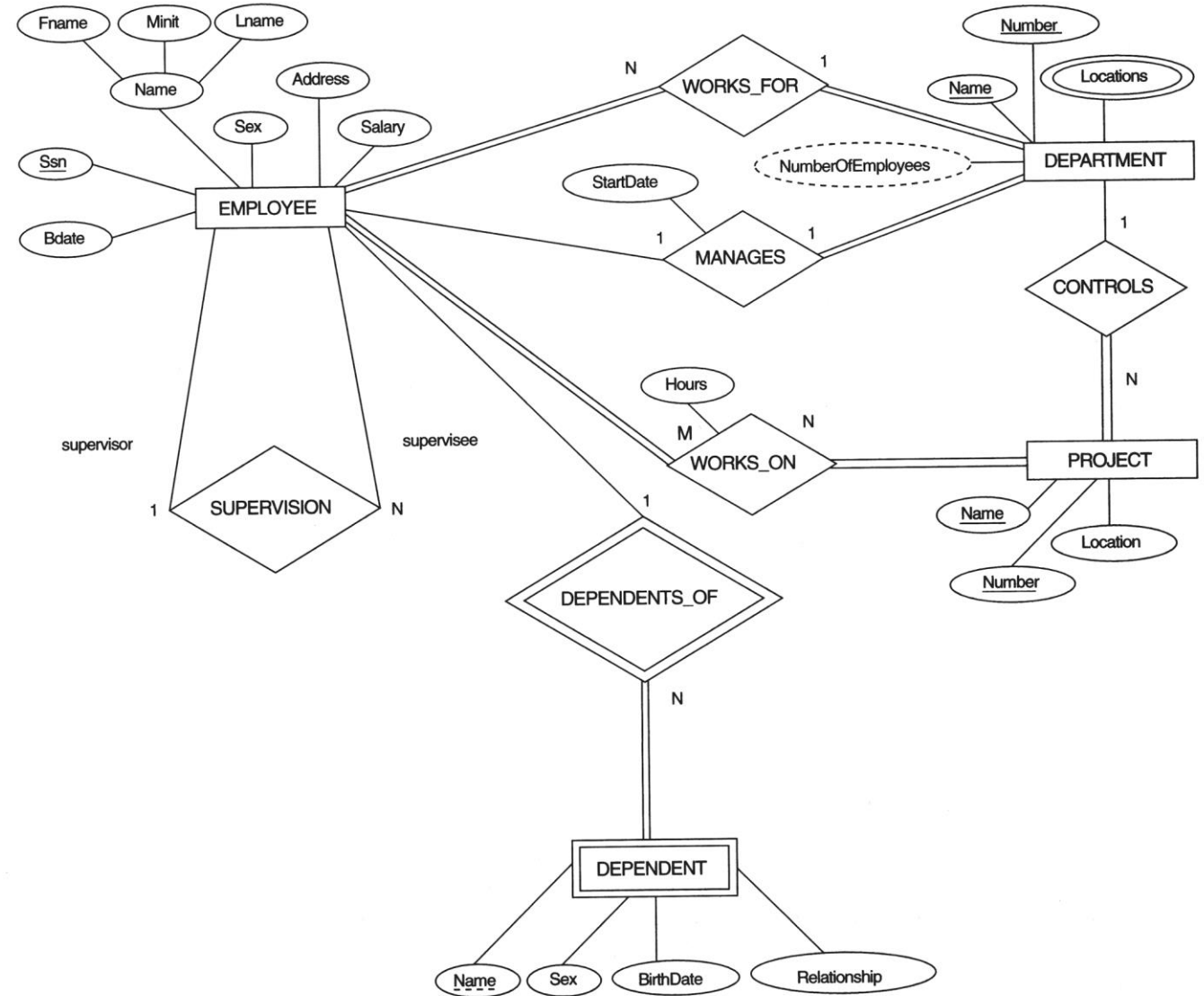


## Step 4: Mapping of Binary 1:N Relationship Types

- 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.

## Step 4

**Example:** 1:N relationship types WORKS\_FOR, CONTROLS, and SUPERVISION in the figure. For WORKS\_FOR we include the primary key DNUMBER of the DEPARTMENT relation as foreign key in the EMPLOYEE relation and call it DNO.



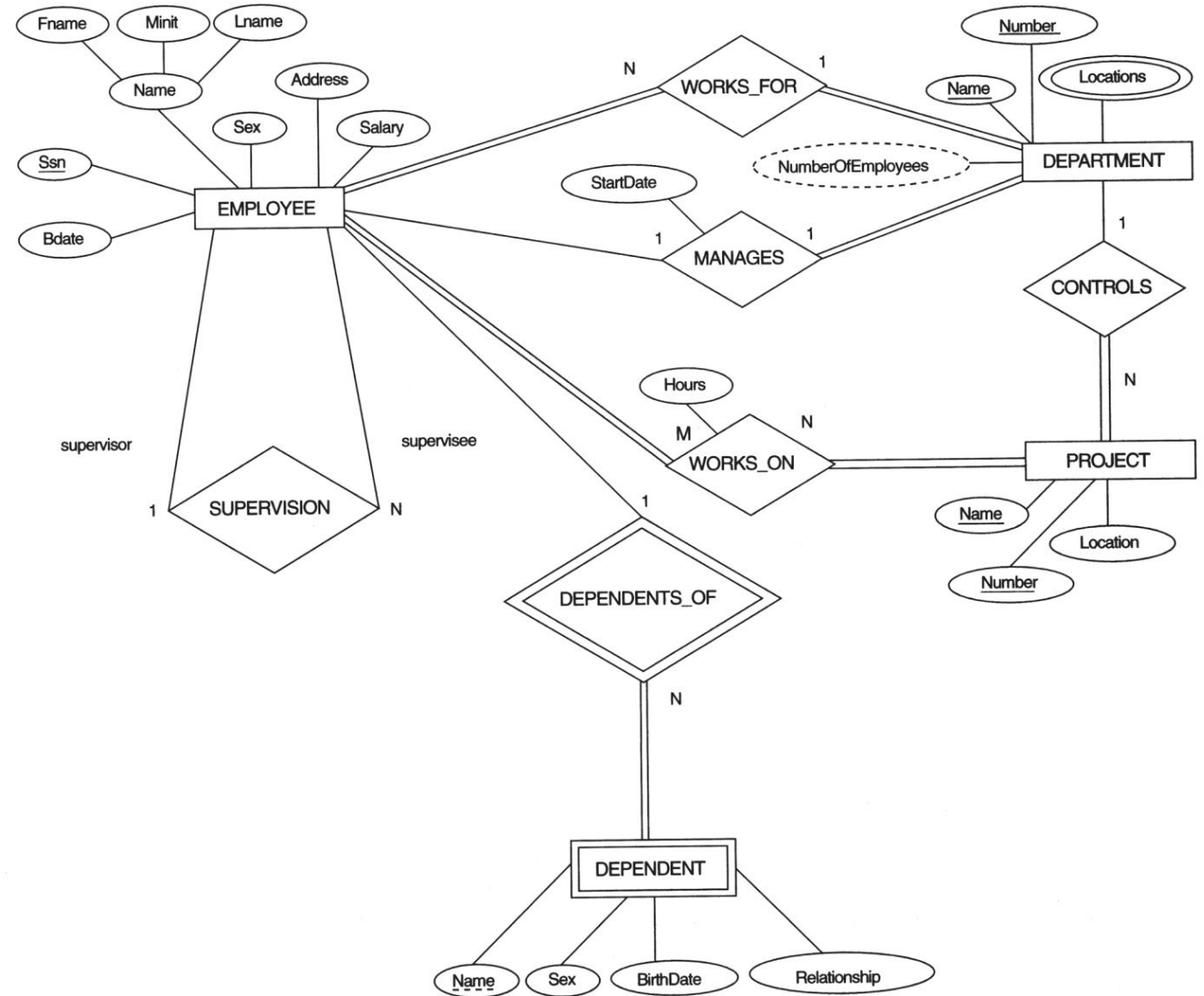
## Step 5: Mapping of Binary M:N Relationship Types

- 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

## Step 5

**Example:** The M:N relationship type WORKS\_ON from the ER diagram is mapped by creating a relation WORKS\_ON in the relational database schema. The primary keys of the PROJECT and EMPLOYEE relations are included as foreign keys in WORKS\_ON and renamed PNO and ESSN, respectively.

Attribute HOURS in WORKS\_ON represents the HOURS attribute of the relation type. The primary key of the WORKS\_ON relation is the combination of the foreign key attributes {ESSN, PNO}.

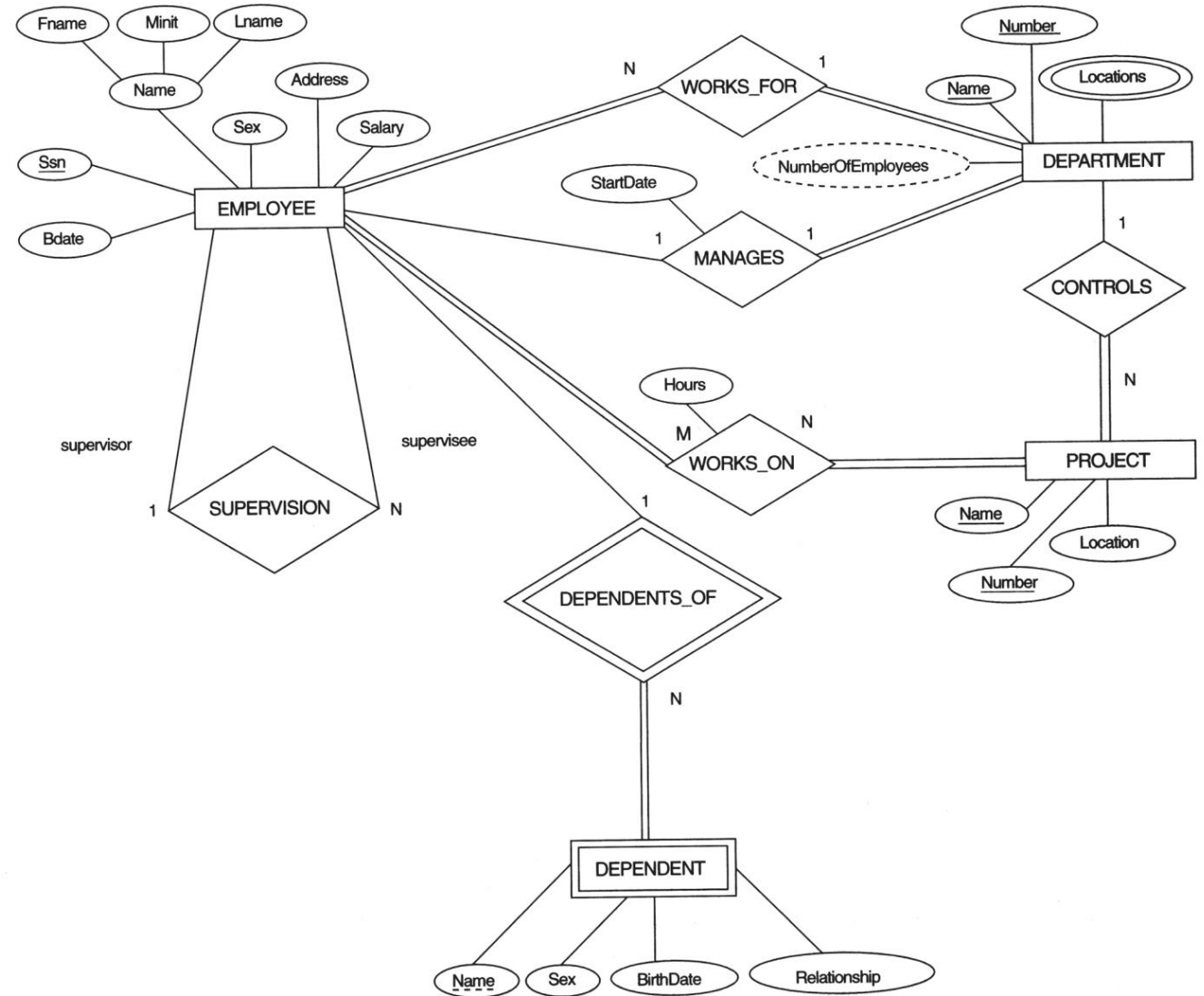


## Step 6: Mapping of Multivalued attributes

- 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 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.

## Step 6

**Example:** The relation DEPT\_LOCATIONS is created. The attribute DLOCATION represents the multivalued attribute LOCATIONS of DEPARTMENT, while DNUMBER-as foreign key-represents the primary key of the DEPARTMENT relation. The primary key of R is the combination of {DNUMBER, DLOCATION}.





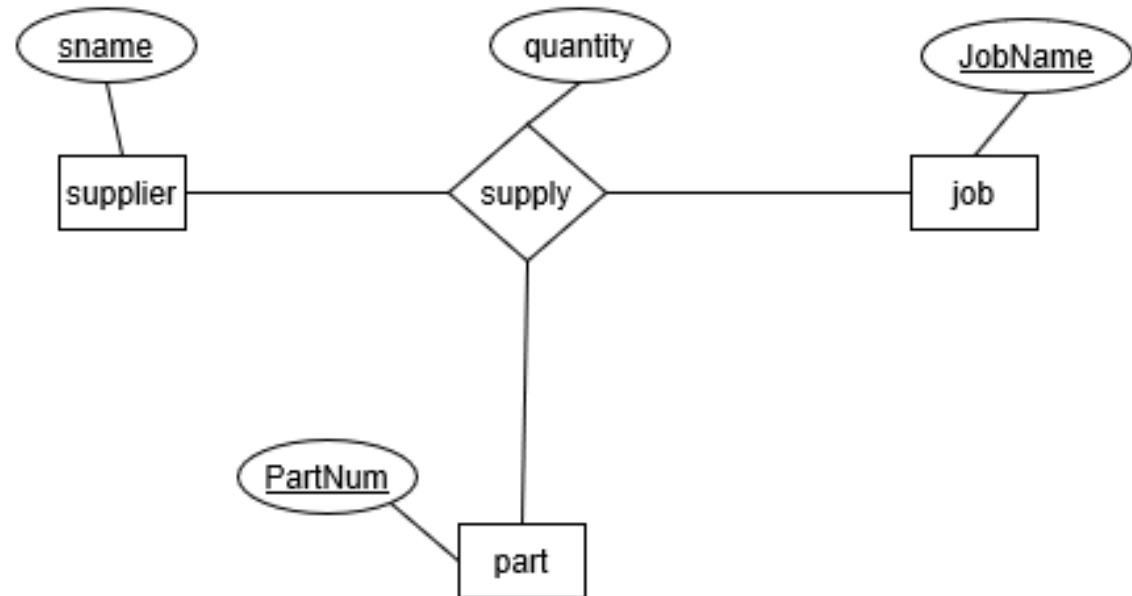
# Step 7: Mapping of N-ary Relationship Types

- 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.

## Step 7

**Example:** The relationship type SUPPLY in the ER

This can be mapped to the relation SUPPLY shown in the relational schema, whose primary key is the combination of the three foreign keys {SNAME, PARTNUM, JOBNAME}



# Solution to n-ary example

Tables that already exist:

**Supplier** (sname, ...)

**Job** (jname, ...)

**Part** (partNum, ...)

Add new Table:

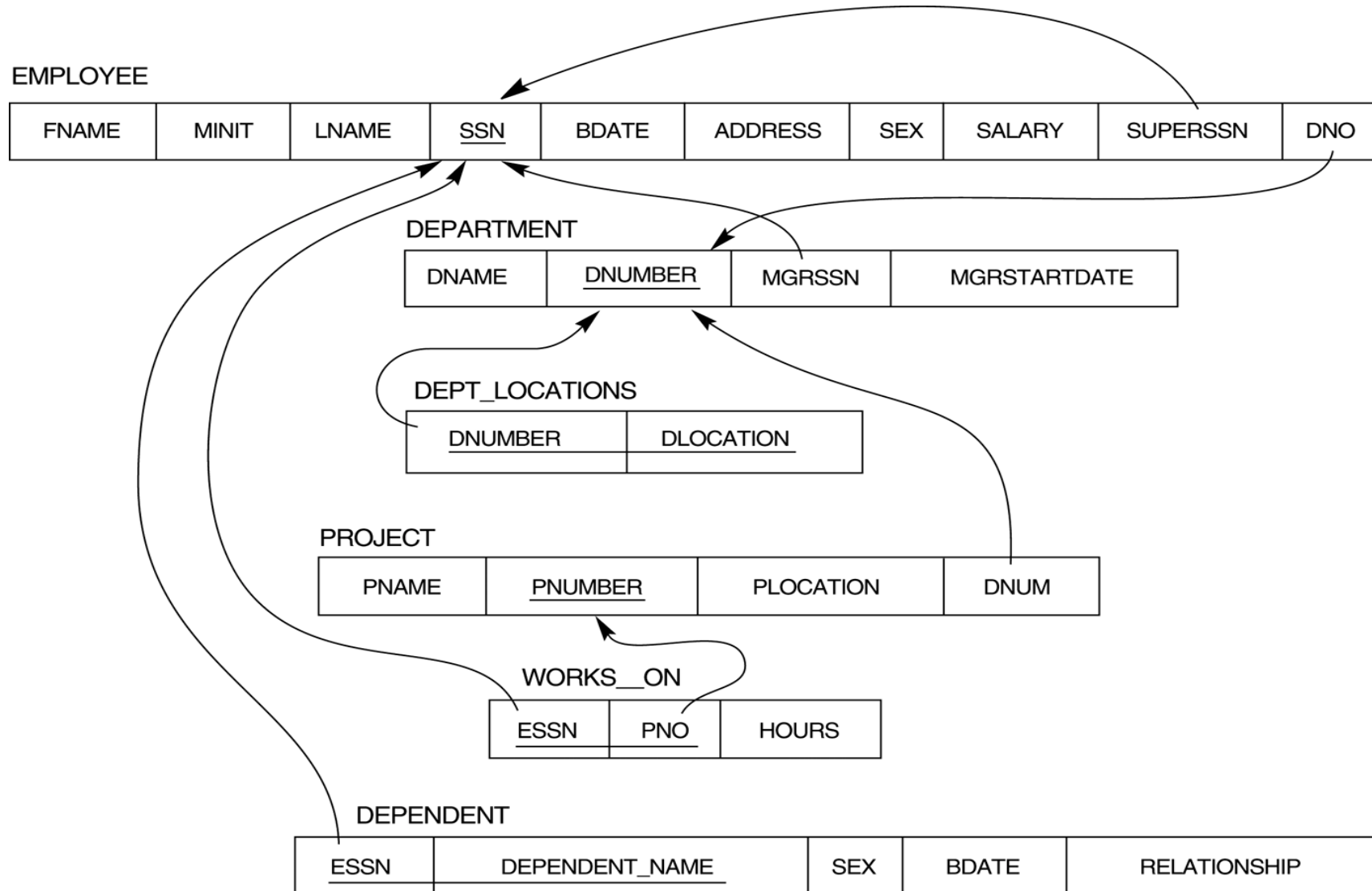
**Supply** (sname, jname, partNum, quantity, ...)

sname is FK -> Supplier (sname)

jname is FK -> Job (jname)

partNum is FK -> Part (partNum)

Result of mapping the COMPANY ER schema into a relational schema.



# Summary of mapping constructs & constraints

## *Correspondence between ER and Relational Models*

### **ER Model**

Entity type

1:1 or 1:N relationship type

M:N relationship type

$n$ -ary relationship type

Simple attribute

Composite attribute

Multivalued attribute

### **Relational Model**

“Entity” relation

Foreign key (or “relationship” relation)

“Relationship” relation and two foreign keys

“Relationship” relation and  $n$  foreign keys

Attribute

Set of simple component attributes

Relation and foreign key

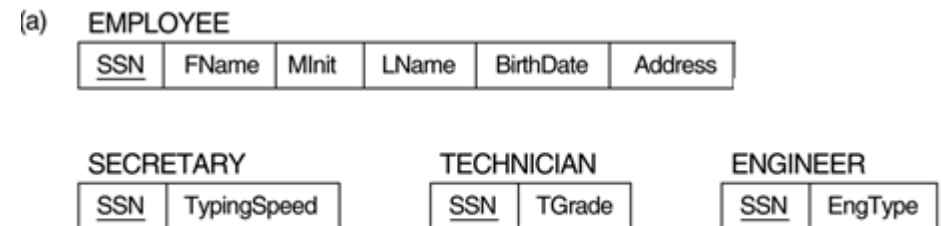
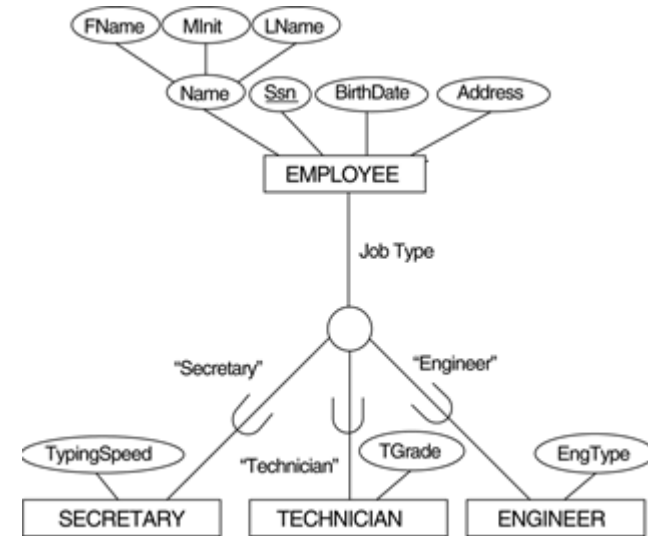
# Mapping EER Model Constructs to Relations

- **Step8: Options for Mapping Specialization or Generalization.**
  - Convert each specialization with  $m$  subclasses  $\{S_1, S_2, \dots, S_m\}$  and generalized superclass  $C$ , where the attributes of  $C$  are  $\{k, a_1, \dots, a_n\}$  and  $k$  is the (primary) key, into relational schemas using one of the four following options:
    - **Option 8A:** Multiple relations-Superclass and subclasses
    - **Option 8B:** Multiple relations-Subclass relations only
    - **Option 8C:** Single relation with one type attribute
    - **Option 8D:** Single relation with multiple type attributes

# Option 8A

## Option 8A: Multiple relations-Superclass and subclasses

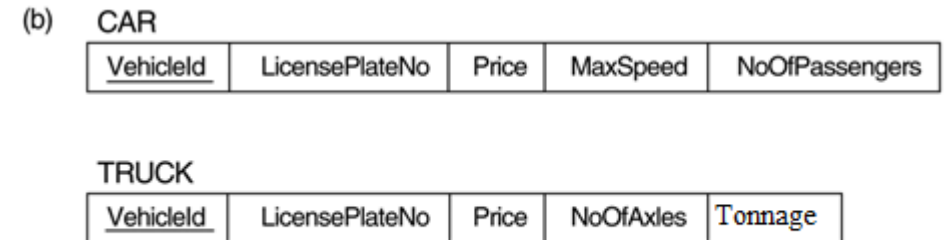
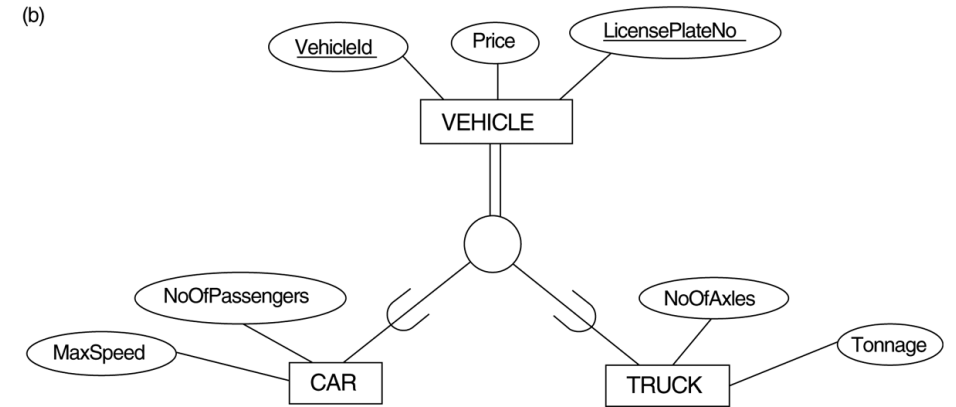
Create a relation  $L$  for  $C$  as  $L\{k, a_1, \dots, a_n\}$ , and  $PK(L) = k$ . Create a relation  $L_i$  for each subclass  $S_i$ ,  $1 < i < m$ ,  $L_i\{k, s_{i1}, \dots, s_{iik}\}$ , where  $s_{i1}, \dots, s_{iik}$  are the local attributes of  $S_i$ , and  $PK(L_i) = k$ . This option works for **any specialization** (total or partial, disjoint or over-lapping).



# Option 8B

## Option 8B: Multiple relations-Subclass relations only

Create a relation  $L_i$  for each subclass  $S_i$ ,  $1 < i < m$ , with the attributes  $\text{Attr}(L_i) = \{\text{attributes of } S_i\} \cup \{k, a_1, \dots, a_n\}$  and  $\text{PK}(L_i) = k$ . This option **only** works for a specialization whose subclasses are **total** (every entity in the superclass must belong to (at least) one of the subclasses).



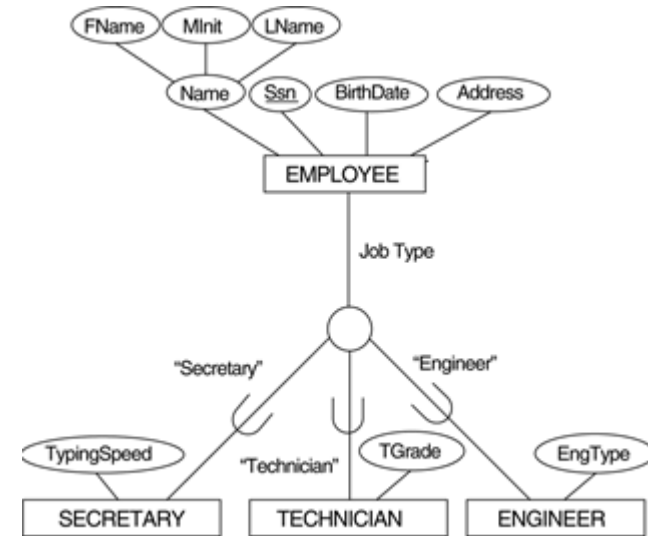


# Option 8C

## Option 8C: Single relation with one type attribute.

Create a single relation  $L$  with attributes  $\text{Attrs}(L) = \{k, a_1, \dots, a_n\} \cup \{\text{attributes of } S_1\} \cup \dots \cup \{\text{attributes of } S_m\} \cup \{t\}$  and  $\text{PK}(L) = k$ . The attribute  $t$  is called a type (or **discriminating**) attribute that indicates the subclass to which each tuple belongs.

This option works for **disjoint specilaization**.



(c) EMPLOYEE

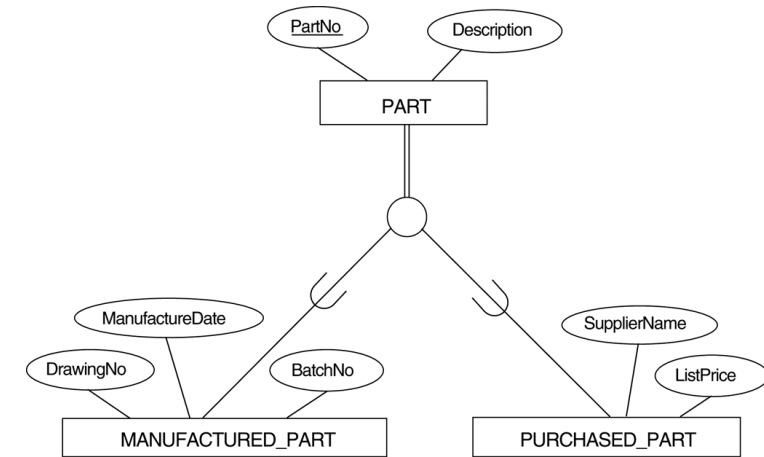
<u>SSN</u>	FName	Minit	LName	BirthDate	Address	JobType	TypingSpeed	TGrade	EngType
------------	-------	-------	-------	-----------	---------	---------	-------------	--------	---------

# Option 8D

## Option 8D: Single relation with multiple type attributes.

Create a single relation schema  $L$  with attributes  $\text{Attrs}(L) = \{k, a_1, \dots, a_n\} \cup \{\text{attributes of } S_1\} \cup \dots \cup \{\text{attributes of } S_m\} \cup \{t_1, t_2, \dots, t_m\}$  and  $\text{PK}(L) = k$ . Each  $t_i$ ,  $1 < i < m$ , is a Boolean type attribute indicating whether a tuple belongs to the subclass  $S_i$ .

This option works for **overlapping specialization**.



(d) PART

<u>PartNo</u>	Description	MFlag	DrawingNo	ManufactureDate	BatchNo	PFlag	SupplierName	ListPrice
---------------	-------------	-------	-----------	-----------------	---------	-------	--------------	-----------

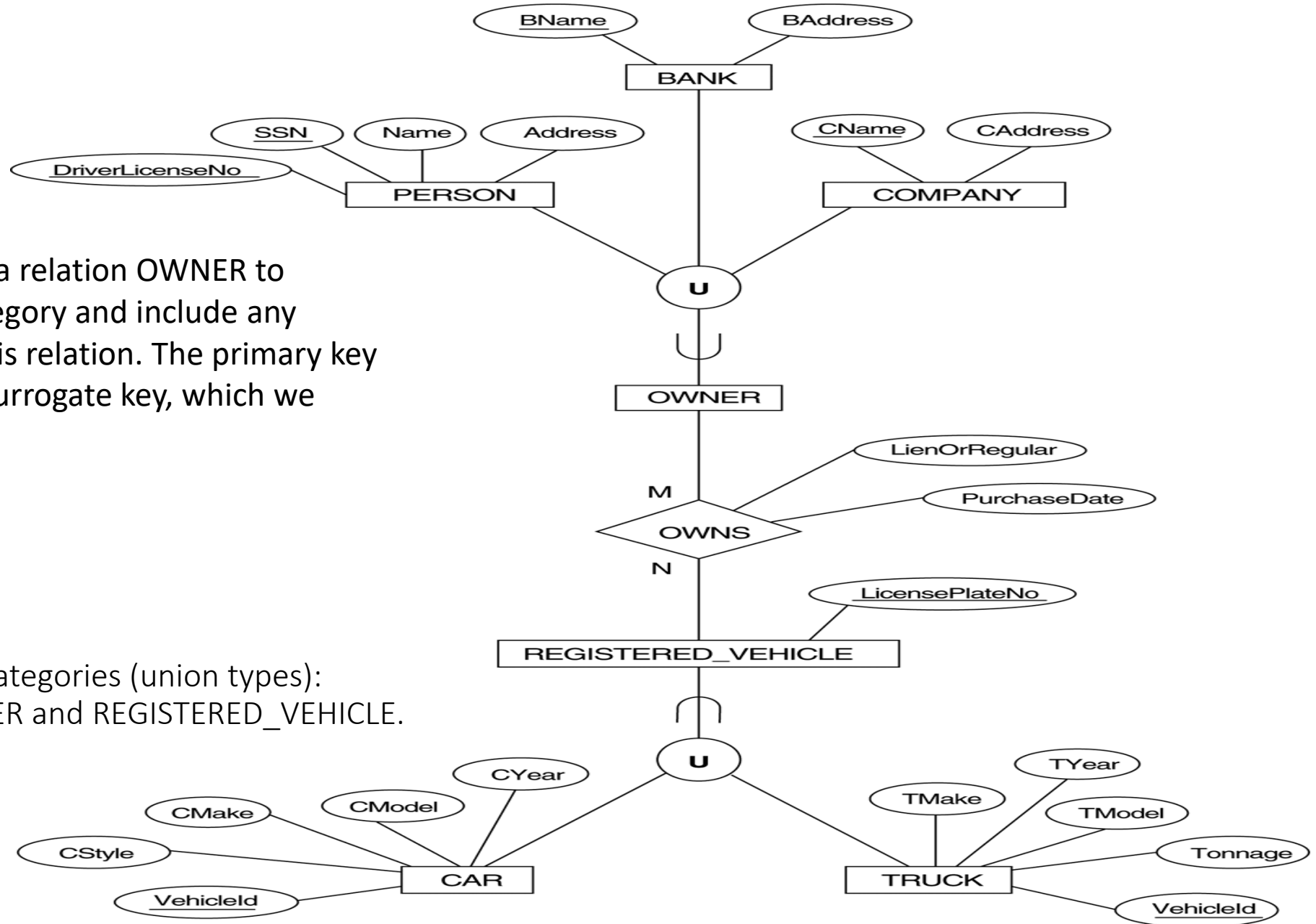
# Mapping EER Model Constructs to Relations (cont)

## Step 9: Mapping of Union Types (Categories).

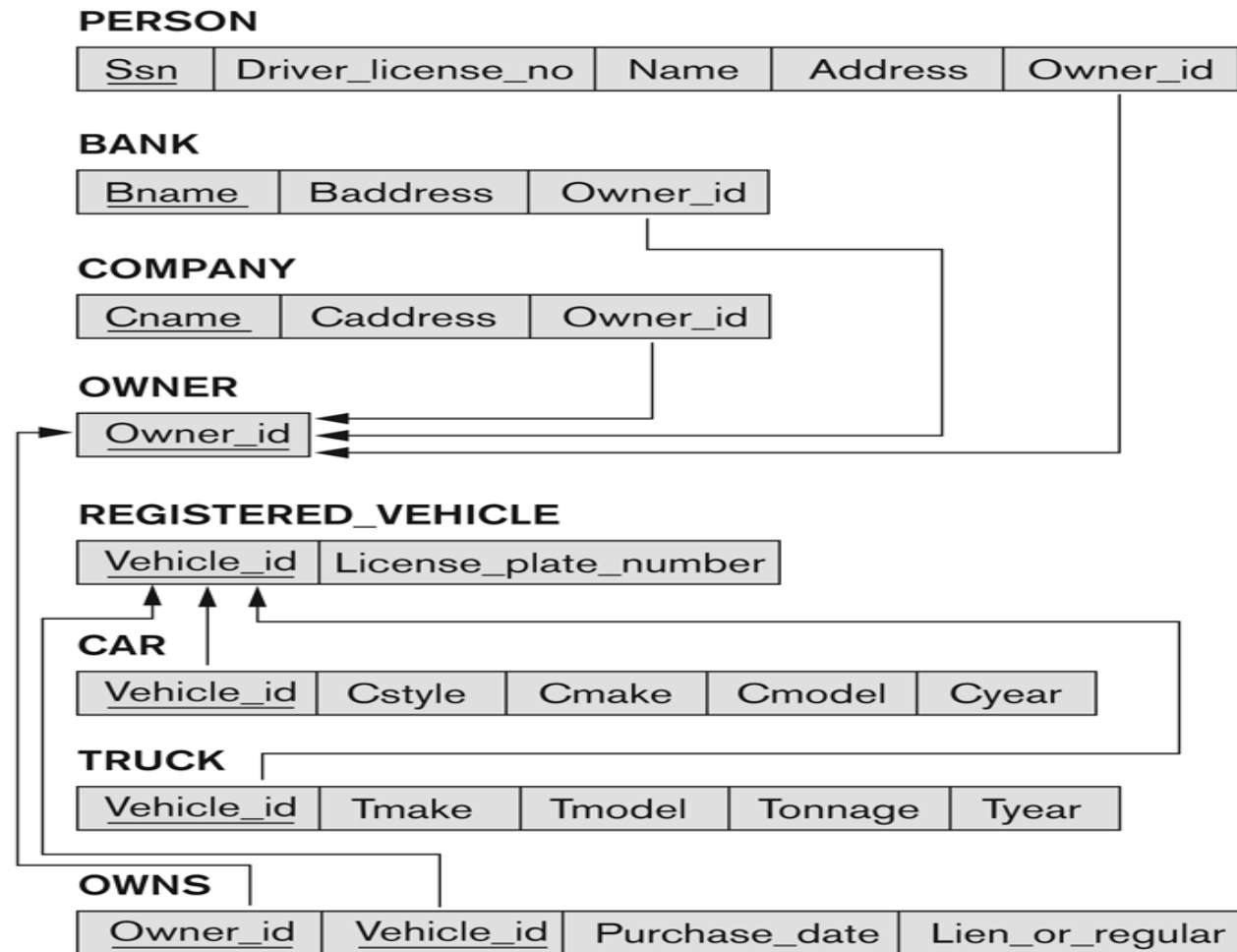
- For mapping a category whose defining superclass have different keys, it is customary to specify a new key attribute, called a **surrogate key**, when creating a relation to correspond to the category.

In the example, we can create a relation OWNER to correspond to the OWNER category and include any attributes of the category in this relation. The primary key of the OWNER relation is the surrogate key, which we called OwnerId.

Two categories (union types):  
OWNER and REGISTERED\_VEHICLE.



Mapping the EER categories (union types) to relations.



# Relational Algebra Overview

Relational Algebra consists of several groups of operations:

- **Unary Relational Operations**

- SELECT (symbol:  $\sigma$  (sigma))
- PROJECT (symbol:  $\pi$  (pi))
- RENAME (symbol:  $\rho$  (rho))

- **Relational Algebra Operations From Set Theory**

- UNION (  $\cup$  ), INTERSECTION (  $\cap$  ), DIFFERENCE (or MINUS,  $-$  )
- CARTESIAN PRODUCT (  $\times$  )

- **Binary Relational Operations**

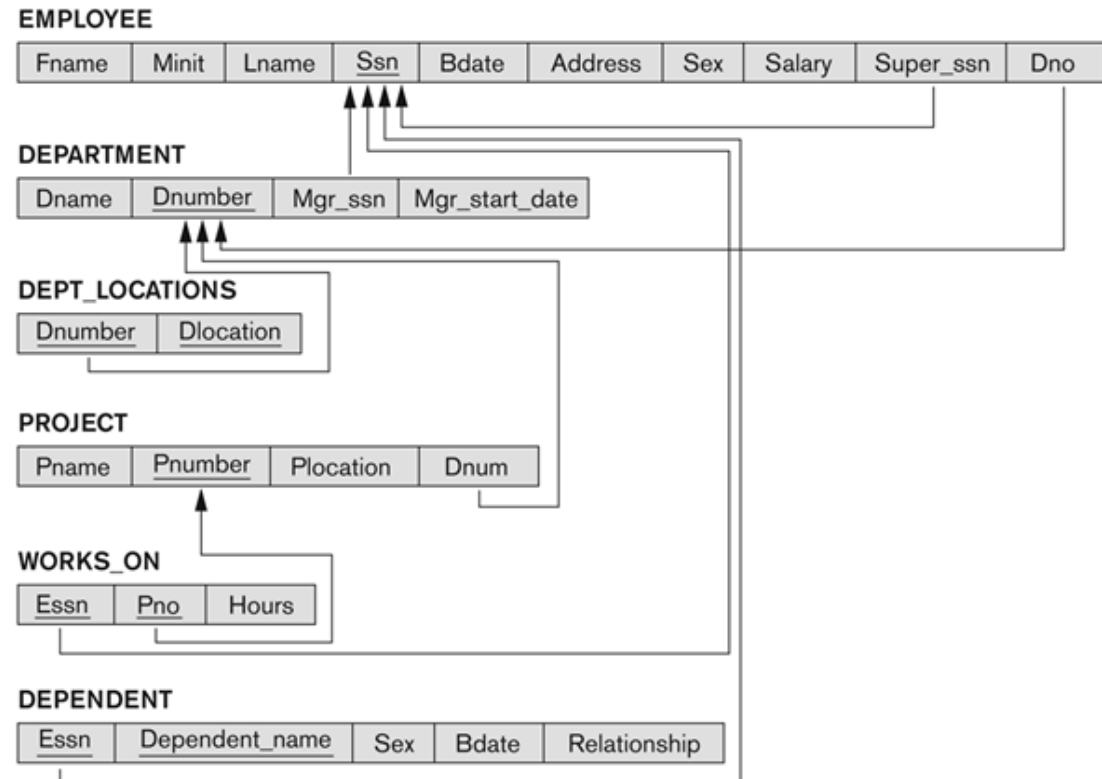
- JOIN (several variations of JOIN exist)
- DIVISION

- **Additional Relational Operations**

- OUTER JOINS, OUTER UNION
- AGGREGATE FUNCTIONS (These compute summary of information: for example, SUM, COUNT, AVG, MIN, MAX)

# Database State for COMPANY

- All examples discussed in this topic refer to the COMPANY database shown here.



# Unary Relational Operations: SELECT

- The SELECT operation (denoted by  $\sigma$  (sigma)) is used to select a *subset* of the tuples from a relation based on a **selection condition**.
  - The selection condition acts as a **filter**
  - Keeps only those tuples that satisfy the qualifying condition
  - Tuples satisfying the condition are *selected* whereas the other tuples are discarded (*filtered out*)
- Examples:
  - Select the EMPLOYEE tuples whose department number is 4:

$$\sigma_{DNO = 4} (EMPLOYEE)$$

- Select the employee tuples whose salary is greater than \$30,000:

$$\sigma_{SALARY > 30,000} (EMPLOYEE)$$



# Unary Relational Operations: SELECT

- In general, the *select* operation is denoted by  $\sigma_{\langle \text{selection condition} \rangle}(R)$  where
  - the symbol  $\sigma$  (sigma) is used to denote the *select* operator
  - the selection condition is a Boolean (conditional) expression specified on the attributes of relation R
  - tuples that make the condition **true** are selected
    - appear in the result of the operation
  - tuples that make the condition **false** are filtered out
    - discarded from the result of the operation

# Unary Relational Operations: SELECT (contd.)

- SELECT Operation Properties
  - The SELECT operation  $\sigma_{\langle \text{selection condition} \rangle}(R)$  produces a relation S that has the same schema (same attributes) as R
  - SELECT  $\sigma$  is commutative:
    - $\sigma_{\langle \text{condition1} \rangle}(\sigma_{\langle \text{condition2} \rangle}(R)) = \sigma_{\langle \text{condition2} \rangle}(\sigma_{\langle \text{condition1} \rangle}(R))$
  - Because of commutativity property, a cascade (sequence) of SELECT operations may be applied in any order:
    - $\sigma_{\langle \text{cond1} \rangle}(\sigma_{\langle \text{cond2} \rangle}(\sigma_{\langle \text{cond3} \rangle}(R))) = \sigma_{\langle \text{cond2} \rangle}(\sigma_{\langle \text{cond3} \rangle}(\sigma_{\langle \text{cond1} \rangle}(R)))$
  - A cascade of SELECT operations may be replaced by a single selection with a conjunction of all the conditions:
    - $\sigma_{\langle \text{cond1} \rangle}(\sigma_{\langle \text{cond2} \rangle}(\sigma_{\langle \text{cond3} \rangle}(R))) = \sigma_{\langle \text{cond1} \rangle \text{ AND } \langle \text{cond2} \rangle \text{ AND } \langle \text{cond3} \rangle}(R))$
  - The number of tuples in the result of a SELECT is less than (or equal to) the number of tuples in the input relation R

# The following query results refer to this database state

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT\_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS\_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

# Unary Relational Operations: SELECT (contd.)

- Examples:

Select the EMPLOYEE tuples whose department number is 4 with salary greater than 25000 or department number is 5 with salary greater than 30000 :

$\sigma_{(DNO = 4 \text{ AND } Salary > 25000) \text{ OR } (DNO = 5 \text{ AND } Salary > 30000)} (EMPLOYEE)$

EMPLOYEE

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary	Super_ssn	Dno
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5

# Unary Relational Operations: PROJECT

- PROJECT Operation is denoted by  $\pi$  (pi)
- This operation keeps certain *columns* (attributes) from a relation and discards the other columns.
  - PROJECT creates a vertical partitioning
    - The list of specified columns (attributes) is kept in each tuple
    - The other attributes in each tuple are discarded
- Example: To list each employee's first and last name and salary, the following is used:

$\pi_{\text{LNAME, FNAME, SALARY}}(\text{EMPLOYEE})$

# Unary Relational Operations: PROJECT (cont.)

- The general form of the *project* operation is:

$$\pi_{\langle \text{attribute list} \rangle}(R)$$

- $\pi$  (pi) is the symbol used to represent the *project* operation
  - $\langle \text{attribute list} \rangle$  is the desired list of attributes from relation R.
- The project operation *removes any duplicate tuples*
  - This is because the result of the *project* operation must be a *set of tuples*
    - Mathematical sets *do not allow* duplicate elements.

# Unary Relational Operations: PROJECT (contd.)

- PROJECT Operation Properties

- The number of tuples in the result of projection  $\pi_{\langle \text{list} \rangle}(R)$  is always less or equal to the number of tuples in  $R$
- $\pi_{\langle \text{list1} \rangle}(\pi_{\langle \text{list2} \rangle}(R)) = \pi_{\langle \text{list1} \rangle}(R)$  as long as  $\langle \text{list2} \rangle$  contains the attributes in  $\langle \text{list1} \rangle$
- PROJECT is *not* commutative

# Unary Relational Operations: PROJECT (contd.)

- Example: To list each employee's first and last name and salary, the following is used:

$\pi_{\text{LNAME, FNAME, SALARY}}(\text{EMPLOYEE})$

EMPLOYEE

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

Lname	Fname	Salary
Smith	John	30000
Wong	Franklin	40000
Zelaya	Alicia	25000
Wallace	Jennifer	43000
Narayan	Ramesh	38000
English	Joyce	25000
Jabbar	Ahmad	25000
Borg	James	55000



# Relational Algebra Expressions

- We may want to apply several relational algebra operations one after the other
  - Either we can write the operations as a single **relational algebra expression** by nesting the operations, or
  - We can apply one operation at a time and create **intermediate result relations**.
- In the latter case, we must give names to the relations that hold the intermediate results.

# Single expression versus sequence of relational operations (Example)

- To retrieve the first name, last name, and salary of all employees who work in department number 5
- we must apply a select and a project operation
- We can write a *single relational algebra expression* as follows:
  - $\pi_{\text{FNAME, LNAME, SALARY}}(\sigma_{\text{DNO}=5}(\text{EMPLOYEE}))$
- OR We can explicitly show the *sequence of operations*, giving a name to each intermediate relation:
  - $\text{DEP5\_EMPS} \leftarrow \sigma_{\text{DNO}=5}(\text{EMPLOYEE})$
  - $\text{RESULT} \leftarrow \pi_{\text{FNAME, LNAME, SALARY}}(\text{DEP5\_EMPS})$

Fname	Lname	Salary
John	Smith	30000
Franklin	Wong	40000
Ramesh	Narayan	38000
Joyce	English	25000

# Unary Relational Operations: RENAME

- The RENAME operator is denoted by  $\rho$  (rho)
- In some cases, we may want to *rename* the attributes of a relation or the relation name or both
  - Useful when a query requires multiple operations
  - Necessary in some cases (see JOIN operation later)

# Unary Relational Operations: RENAME (contd.)

- The general RENAME operation  $\rho$  can be expressed by any of the following forms:
  - $\rho_{S(B_1, B_2, \dots, B_n)}(R)$  changes both:
    - the relation name to  $S$ , *and*
    - the column (attribute) names to  $B_1, B_2, \dots, B_n$
  - $\rho_S(R)$  changes:
    - the *relation name* only to  $S$
  - $\rho_{(B_1, B_2, \dots, B_n)}(R)$  changes:
    - the *column (attribute) names* only to  $B_1, B_2, \dots, B_n$

# Relational Algebra Operations from Set Theory:

## UNION

- Binary operation, denoted by  $\cup$
- The result of  $R \cup S$ , is a relation that includes all tuples that are either in R or in S or in both R and S
- Duplicate tuples are eliminated
- The two-operand relations R and S must be “type compatible” (or UNION compatible)
  - R and S must have same number of attributes
  - Each pair of corresponding attributes must be type compatible (have same or compatible domains)

# Relational Algebra Operations from Set Theory: UNION

## Example:

To retrieve the social security numbers of all employees who either *work in department 5* (RESULT1 below) or *directly supervise an employee who works in department 5* (RESULT2 below)

We can use the UNION operation as follows:

$$\text{DEP5\_EMPS} \leftarrow \sigma_{\text{DNO}=5} (\text{EMPLOYEE})$$
$$\text{RESULT1} \leftarrow \pi_{\text{SSN}}(\text{DEP5\_EMPS})$$
$$\text{RESULT2} \leftarrow \pi_{\text{SUPERSSN}}(\text{DEP5\_EMPS})$$
$$\text{RESULT} \leftarrow \text{RESULT1} \cup \text{RESULT2}$$

The union operation produces the tuples that are in either RESULT1 or RESULT2 or both

# Example of the result of a UNION operation

Result of the  
UNION operation  
 $\text{RESULT} \leftarrow \text{RESULT1} \cup \text{RESULT2}.$

**RESULT1**

Ssn
123456789
333445555
666884444
453453453

**RESULT2**

Ssn
333445555
888665555

**RESULT**

Ssn
123456789
333445555
666884444
453453453
888665555

# Relational Algebra Operations from Set Theory: INTERSECTION

- INTERSECTION is denoted by  $\cap$
- The result of the operation  $R \cap S$ , is a relation that includes all tuples that are in both R and S
- The attribute names in the result will be the same as the attribute names in R
- The two operand relations R and S must be “type compatible”



# Relational Algebra Operations from Set Theory: SET DIFFERENCE (cont.)

- SET DIFFERENCE (also called MINUS or EXCEPT) is denoted by  $-$
- The result of  $R - S$ , is a relation that includes all tuples that are in  $R$  but not in  $S$
- The attribute names in the result will be the same as the attribute names in  $R$
- The two operand relations  $R$  and  $S$  must be “type compatible”

The set operations UNION, INTERSECTION, and MINUS. (a) Two union-compatible relations.  
 (b)  $\text{STUDENT} \cup \text{INSTRUCTOR}$ . (c)  $\text{STUDENT} \cap \text{INSTRUCTOR}$ . (d)  $\text{STUDENT} - \text{INSTRUCTOR}$ .  
 (e)  $\text{INSTRUCTOR} - \text{STUDENT}$ .

(a) STUDENT

Fn	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

INSTRUCTOR

Fname	Lname
John	Smith
Ricardo	Browne
Susan	Yao
Francis	Johnson
Ramesh	Shah

(b)

Fn	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert
John	Smith
Ricardo	Browne
Francis	Johnson

(c)

Fn	Ln
Susan	Yao
Ramesh	Shah

Example to  
illustrate the  
result of UNION,  
INTERSECT, and  
DIFFERENCE

(d)

Fn	Ln
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

(e)

Fname	Lname
John	Smith
Ricardo	Browne
Francis	Johnson

# Some properties of UNION, INTERSECT, and DIFFERENCE

- Notice that both union and intersection are *commutative* operations; that is
  - $R \cup S = S \cup R$ , and  $R \cap S = S \cap R$
- Both union and intersection can be treated as n-ary operations applicable to any number of relations as both are *associative* operations; that is
  - $R \cup (S \cup T) = (R \cup S) \cup T$
  - $(R \cap S) \cap T = R \cap (S \cap T)$
- The minus operation is not commutative; that is, in general
  - $R - S \neq S - R$

# Relational Algebra Operations from Set Theory: CARTESIAN PRODUCT

- This operation is used to combine tuples from two relations in a combinatorial fashion.
- Denoted by  $R(A_1, A_2, \dots, A_n) \times S(B_1, B_2, \dots, B_m)$
- Result is a relation  $Q$  with degree  $n + m$  attributes:
  - $Q(A_1, A_2, \dots, A_n, B_1, B_2, \dots, B_m)$ , in that order.
- The resulting relation state has one tuple for each combination of tuples—one from  $R$  and one from  $S$ .
- Hence, if  $R$  has  $n_R$  tuples (denoted as  $|R| = n_R$ ), and  $S$  has  $n_S$  tuples, then  $R \times S$  will have  $n_R * n_S$  tuples.
- The two operands do NOT have to be "type compatible"

# Relational Algebra Operations from Set Theory: CARTESIAN PRODUCT (cont.)

- Generally, CROSS PRODUCT is not a meaningful operation
  - Can become meaningful when followed by other operations
- Example (not meaningful):
  - $\text{FEMALE\_EMPS} \leftarrow \sigma_{\text{SEX}='F'}(\text{EMPLOYEE})$
  - $\text{EMP\_NAMES} \leftarrow \pi_{\text{FNAME, LNAME, SSN}}(\text{FEMALE\_EMPS})$
  - $\text{EMP\_DEPENDENTS} \leftarrow \text{EMP\_NAMES} \times \text{DEPENDENT}$
- EMP\_DEPENDENTS will contain every combination of EMP\_NAMES and DEPENDENT
  - whether or not they are actually related

# Relational Algebra Operations from Set Theory: CARTESIAN PRODUCT (cont.)

- To keep only combinations where the DEPENDENT is related to the EMPLOYEE, we add a SELECT operation as follows
- Example (meaningful):
  - $\text{FEMALE\_EMPS} \leftarrow \sigma_{\text{SEX}='F'}(\text{EMPLOYEE})$
  - $\text{EMP\_NAMES} \leftarrow \pi_{\text{FNAME, LNAME, SSN}}(\text{FEMALE\_EMPS})$
  - $\text{EMP\_DEPENDENTS} \leftarrow \text{EMP\_NAMES} \times \text{DEPENDENT}$
  - $\text{ACTUAL\_DEPS} \leftarrow \sigma_{\text{SSN}=\text{ESSN}}(\text{EMP\_DEPENDENTS})$
  - $\text{RESULT} \leftarrow \pi_{\text{FNAME, LNAME, DEPENDENT\_NAME}}(\text{ACTUAL\_DEPS})$
- RESULT will now contain the name of female employees and their dependents

# Example of applying CARTESIAN PRODUCT

The CARTESIAN PRODUCT (CROSS PRODUCT) operation.

FEMALE\_EMPS

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
Alicia	J	Zelaya	999887777	1988-07-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Joyce	A	English	453453453	1972-07-31	5831 Rice, Houston, TX	F	25000	333445555	5

EMPNames

Fname	Lname	Ssn
Alicia	Zelaya	999887777
Jennifer	Wallace	987654321
Joyce	English	453453453

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

ACTUAL\_DEPENDENTS

Fname	Lname	Ssn	Essn	Dependent_name	Sex	Bdate	...
Jennifer	Wallace	987654321	987654321	Abner	M	1942-02-28	...

EMP\_DEPENDENTS

Fname	Lname	Ssn	Essn	Dependent_name	Sex	Bdate	...
Alicia	Zelaya	999887777	333445555	Alice	F	1986-04-05	...
Alicia	Zelaya	999887777	333445555	Theodore	M	1983-10-25	...
Alicia	Zelaya	999887777	333445555	Joy	F	1958-05-03	...
Alicia	Zelaya	999887777	987654321	Abner	M	1942-02-28	...
Alicia	Zelaya	999887777	123456789	Michael	M	1988-01-04	...
Alicia	Zelaya	999887777	123456789	Alice	F	1988-12-30	...
Alicia	Zelaya	999887777	123456789	Elizabeth	F	1967-05-05	...
Jennifer	Wallace	987654321	333445555	Alice	F	1986-04-05	...
Jennifer	Wallace	987654321	333445555	Theodore	M	1983-10-25	...
Jennifer	Wallace	987654321	333445555	Joy	F	1958-05-03	...
Jennifer	Wallace	987654321	987654321	Abner	M	1942-02-28	...
Jennifer	Wallace	987654321	123456789	Michael	M	1988-01-04	...
Jennifer	Wallace	987654321	123456789	Alice	F	1988-12-30	...
Jennifer	Wallace	987654321	123456789	Elizabeth	F	1967-05-05	...
Joyce	English	453453453	333445555	Alice	F	1986-04-05	...
Joyce	English	453453453	333445555	Theodore	M	1983-10-25	...
Joyce	English	453453453	333445555	Joy	F	1958-05-03	...
Joyce	English	453453453	987654321	Abner	M	1942-02-28	...
Joyce	English	453453453	123456789	Michael	M	1988-01-04	...
Joyce	English	453453453	123456789	Alice	F	1988-12-30	...
Joyce	English	453453453	123456789	Elizabeth	F	1967-05-05	...

RESULT

Fname	Lname	Dependent_name
Jennifer	Wallace	Abner

# Binary Relational Operations: JOIN

- JOIN Operation (denoted by  $\bowtie$ )
  - The sequence of CARTESIAN PRODUCT followed by SELECT is used quite commonly to identify and select related tuples from two relations
  - A special operation, called JOIN combines this sequence into a single operation
  - This operation is very important for any relational database with more than a single relation, because it allows us *combine related tuples* from various relations
  - The general form of a join operation on two relations  $R(A_1, A_2, \dots, A_n)$  and  $S(B_1, B_2, \dots, B_m)$  is:

$$R \bowtie_{\langle \text{join condition} \rangle} S$$

where  $R$  and  $S$  can be any relations that result from general *relational algebra expressions*.



# Binary Relational Operations: JOIN (cont.)

**Example:** Suppose that we want to retrieve the name of the manager of each department.

- To get the manager's name, we need to combine each DEPARTMENT tuple with the EMPLOYEE tuple whose SSN value matches the MGRSSN value in the department tuple.

$$\text{DEPT\_MGR} \leftarrow \text{DEPARTMENT} \bowtie_{\text{MGRSSN}=\text{SSN}} \text{EMPLOYEE}$$

- MGRSSN=SSN is the join condition
  - Combines each department record with the employee who manages the department
  - The join condition can also be specified as DEPARTMENT.MGRSSN= EMPLOYEE.SSN

# Example of applying the JOIN operation

**DEPT\_MGR**

Dname	Dnumber	Mgr_ssn	...	Fname	Minit	Lname	Ssn	...
Research	5	333445555	...	Franklin	T	Wong	333445555	...
Administration	4	987654321	...	Jennifer	S	Wallace	987654321	...
Headquarters	1	888665555	...	James	E	Borg	888665555	...

Result of the JOIN operation

# Some properties of JOIN

- Consider the following JOIN operation:
  - $R(A_1, A_2, \dots, A_n) \bowtie_{R.A_i=S.B_j} S(B_1, B_2, \dots, B_m)$
  - Result is a relation Q with degree  $n + m$  attributes:
    - $Q(A_1, A_2, \dots, A_n, B_1, B_2, \dots, B_m)$ , in that order.
  - The resulting relation state has one tuple for each combination of tuples— $r$  from  $R$  and  $s$  from  $S$ , but *only if they satisfy the join condition*  $r[A_i]=s[B_j]$
  - Hence, if  $R$  has  $n_R$  tuples, and  $S$  has  $n_S$  tuples, then the join result will generally have *less than*  $n_R * n_S$  tuples.
  - Only related tuples (based on the join condition) will appear in the result

# Some properties of JOIN

- The general case of JOIN operation is called a Theta-join:  $R \bowtie_{\text{theta}} S$
- The join condition is called *theta*
- *Theta* can be any general boolean expression on the attributes of R and S; for example:  
$$R.A_i < S.B_j \text{ AND } (R.A_k = S.B_l \text{ OR } R.A_p < S.B_q)$$
- Most join conditions involve one or more equality conditions “AND”ed together; for example:  
$$R.A_i = S.B_j \text{ AND } R.A_k = S.B_l \text{ AND } R.A_p = S.B_q$$

# Binary Relational Operations: EQUIJOIN

- The most common use of join involves join conditions with *equality comparisons* only
- Such a join, where the only comparison operator used is =, is called an EQUIJOIN.
  - In the result of an EQUIJOIN we always have one or more pairs of attributes (whose names need not be identical) that have identical values in every tuple.
  - The JOIN seen in the previous example was an EQUIJOIN.

# Binary Relational Operations: NATURAL JOIN Operation

- Another variation of JOIN called NATURAL JOIN — denoted by  $*$  — was created to get rid of the second (superfluous) attribute in an EQUIJOIN condition.
  - because one of each pair of attributes with identical values is superfluous
- The standard definition of natural join requires that the two join attributes, or each pair of corresponding join attributes, *have the same name* in both relations.
- If this is not the case, a renaming operation is applied first.

# Binary Relational Operations NATURAL JOIN (contd.)

- **Example:** To apply a natural join on the DNUMBER attributes of DEPARTMENT and DEPT\_LOCATIONS, it is sufficient to write:

$\text{DEPT\_LOCS} \leftarrow \text{DEPARTMENT} * \text{DEPT\_LOCATIONS}$

- Only attribute with the same name is DNUMBER
- An implicit join condition is created based on this attribute:

$\text{DEPARTMENT.DNUMBER} = \text{DEPT\_LOCATIONS.DNUMBER}$

- Another example:  $Q \leftarrow R(A,B,C,D) * S(C,D,E)$ 
  - The implicit join condition includes *each pair* of attributes with the same name, “AND”ed together:
    - $R.C = S.C \text{ AND } R.D = S.D$
  - Result keeps only one attribute of each such pair:
    - $Q(A,B,C,D,E)$

# Example of NATURAL JOIN operation

(a)

**PROJ\_DEPT**

Pname	<u>Pnumber</u>	Plocation	Dnum	Dname	Mgr_ssn	Mgr_start_date
ProductX	1	Bellaire	5	Research	333445555	1988-05-22
ProductY	2	Sugarland	5	Research	333445555	1988-05-22
ProductZ	3	Houston	5	Research	333445555	1988-05-22
Computerization	10	Stafford	4	Administration	987654321	1995-01-01
Reorganization	20	Houston	1	Headquarters	888665555	1981-06-19
Newbenefits	30	Stafford	4	Administration	987654321	1995-01-01

(b)

**DEPT\_LOCS**

Dname	Dnumber	Mgr_ssn	Mgr_start_date	Location
Headquarters	1	888665555	1981-06-19	Houston
Administration	4	987654321	1995-01-01	Stafford
Research	5	333445555	1988-05-22	Bellaire
Research	5	333445555	1988-05-22	Sugarland
Research	5	333445555	1988-05-22	Houston

Results of two NATURAL JOIN operations.

(a) PROJ\_DEPT  $\leftarrow$  PROJECT \* DEPT.

(b) DEPT\_LOCS  $\leftarrow$  DEPARTMENT \* DEPT\_LOCATIONS.



# Complete Set of Relational Operations

- The set of operations including SELECT  $\sigma$ , PROJECT  $\pi$ , UNION  $\cup$ , DIFFERENCE  $-$ , RENAME  $\rho$ , and CARTESIAN PRODUCT  $\times$  is called a *complete set* because any other relational algebra expression can be expressed by a combination of these five operations.
- For example:
  - $R \cap S = (R \cup S) - ((R - S) \cup (S - R))$
  - $R \bowtie_{\langle \text{join condition} \rangle} S = \sigma_{\langle \text{join condition} \rangle} (R \times S)$

# Binary Relational Operations: DIVISION

- The division operation is applied to two relations
- $R(Z) \div S(X)$ , where  $X$  subset  $Z$ .
- For a tuple  $t$  to appear in the result  $T$  of the DIVISION, the values in  $t$  must appear in  $R$  in combination with *every* tuple in  $S$ .

# Example of DIVISION

(a)

**SSN\_PNOS**

Essn	Pno
123456789	1
123456789	2
666884444	3
453453453	1
453453453	2
333445555	2
333445555	3
333445555	10
333445555	20
999887777	30
999887777	10
987987987	10
987987987	30
987654321	30
987654321	20
888665555	20

**SMITH\_PNOS**

Pno
1
2

**SSNS**

Ssn
123456789
453453453

(b)

**R**

A	B
a1	b1
a2	b1
a3	b1
a4	b1
a1	b2
a3	b2
a2	b3
a3	b3
a4	b3
a1	b4
a2	b4
a3	b4

**T**

B
b1
b4

**S**

A
a1
a2
a3

The DIVISION operation. (a) Dividing SSN\_PNOS by SMITH\_PNOS. (b)  $T \leftarrow R \div S$ .

# Recap of Relational Algebra Operations

Operations of Relational Algebra

Operation	Purpose	Notation
SELECT	Selects all tuples that satisfy the selection condition from a relation $R$ .	$\sigma_{\langle \text{selection condition} \rangle}(R)$
PROJECT	Produces a new relation with only some of the attributes of $R$ , and removes duplicate tuples.	$\pi_{\langle \text{attribute list} \rangle}(R)$
THETA JOIN	Produces all combinations of tuples from $R_1$ and $R_2$ that satisfy the join condition.	$R_1 \bowtie_{\langle \text{join condition} \rangle} R_2$
EQUIJOIN	Produces all the combinations of tuples from $R_1$ and $R_2$ that satisfy a join condition with only equality comparisons.	$R_1 \bowtie_{\langle \text{join condition} \rangle} R_2$ OR $R_1 \bowtie_{(\langle \text{join attributes 1} \rangle), (\langle \text{join attributes 2} \rangle)} R_2$
NATURAL JOIN	Same as EQUIJOIN except that the join attributes of $R_2$ are not included in the resulting relation; if the join attributes have the same names, they do not have to be specified at all.	$R_1 *_{\langle \text{join condition} \rangle} R_2$ OR $R_1 *_{(\langle \text{join attributes 1} \rangle), (\langle \text{join attributes 2} \rangle)} R_2$ OR $R_1 * R_2$
UNION	Produces a relation that includes all the tuples in $R_1$ or $R_2$ or both $R_1$ and $R_2$ ; $R_1$ and $R_2$ must be union compatible.	$R_1 \cup R_2$
INTERSECTION	Produces a relation that includes all the tuples in both $R_1$ and $R_2$ ; $R_1$ and $R_2$ must be union compatible.	$R_1 \cap R_2$
DIFFERENCE	Produces a relation that includes all the tuples in $R_1$ that are not in $R_2$ ; $R_1$ and $R_2$ must be union compatible.	$R_1 - R_2$
CARTESIAN PRODUCT	Produces a relation that has the attributes of $R_1$ and $R_2$ and includes as tuples all possible combinations of tuples from $R_1$ and $R_2$ .	$R_1 \times R_2$
DIVISION	Produces a relation $R(X)$ that includes all tuples $t[X]$ in $R_1(Z)$ that appear in $R_1$ in combination with every tuple from $R_2(Y)$ , where $Z = X \cup Y$ .	$R_1(Z) \div R_2(Y)$

# Additional Relational Operations: Aggregate Functions and Grouping

- A type of request that cannot be expressed in the basic relational algebra is to specify mathematical **aggregate functions** on collections of values from the database.
- Examples of such functions include retrieving the average or total salary of all employees or the total number of employee tuples.
  - These functions are used in simple statistical queries that summarize information from the database tuples.
- Common functions applied to collections of numeric values include
  - **SUM, AVERAGE, MAXIMUM, and MINIMUM.**
- The **COUNT** function is used for counting tuples or values.

# Aggregate Function Operation

- Use of the Aggregate Functional operation  $\mathcal{F}$ 
  - $\mathcal{F}_{\text{MAX Salary}}(\text{EMPLOYEE})$  retrieves the maximum salary value from the EMPLOYEE relation
  - $\mathcal{F}_{\text{MIN Salary}}(\text{EMPLOYEE})$  retrieves the minimum Salary value from the EMPLOYEE relation
  - $\mathcal{F}_{\text{SUM Salary}}(\text{EMPLOYEE})$  retrieves the sum of the Salary from the EMPLOYEE relation
  - $\mathcal{F}_{\text{COUNT SSN, AVERAGE Salary}}(\text{EMPLOYEE})$  computes the count (number) of employees and their average salary
    - Note: count just counts the number of rows, without removing duplicates

# Using Grouping with Aggregation

- The previous examples all summarized one or more attributes for a set of tuples
  - Maximum Salary or Count (number of) Ssn
- Grouping can be combined with Aggregate Functions
- **Example:** For each department, retrieve the DNO, COUNT SSN, and AVERAGE SALARY
- A variation of aggregate operation  $\mathcal{F}$  allows this:
  - Grouping attribute placed to left of symbol
  - Aggregate functions to right of symbol
  - $\text{DNO } \mathcal{F}_{\text{COUNT SSN, AVERAGE Salary}} (\text{EMPLOYEE})$
- Above operation groups employees by DNO (department number) and computes the count of employees and average salary per department

# Examples of applying aggregate functions and grouping

The aggregate function operation.

- (a)  $\rho_{R(Dno, No\_of\_employees, Average\_sal)} (Dno \text{ } \mathcal{S} \text{ COUNT Ssn, AVERAGE Salary (EMPLOYEE))}.$   
 (b)  $Dno \text{ } \mathcal{S} \text{ COUNT Ssn, AVERAGE Salary (EMPLOYEE)}.$   
 (c)  $\mathcal{S} \text{ COUNT Ssn, AVERAGE Salary (EMPLOYEE)}.$

(a)

Dno	No_of_employees	Average_sal
5	4	33250
4	3	31000
1	1	55000

(c)

Count_ssn	Average_salary
8	35125

Fname	Minit	Lname	<u>Ssn</u>	...	Salary	Super_ssn	Dno
John	B	Smith	123456789		30000	333445555	5
Franklin	T	Wong	333445555		40000	888665555	5
Ramesh	K	Narayan	666884444		38000	333445555	5
Joyce	A	English	453453453	...	25000	333445555	5
Alicia	J	Zelaya	999887777		25000	987654321	4
Jennifer	S	Wallace	987654321		43000	888665555	4
Ahmad	V	Jabbar	987987987		25000	987654321	4
James	E	Bong	888665555		55000	NULL	1

Grouping EMPLOYEE tuples by the value of Dno

(b)

Dno	Count_ssn	Average_salary
5	4	33250
4	3	31000
1	1	55000



# Additional Relational Operations

- The OUTER JOIN Operation
  - In NATURAL JOIN and EQUIJOIN, tuples without a *matching* (or *related*) tuple are eliminated from the join result
    - Tuples with null in the join attributes are also eliminated
    - This amounts to loss of information.
  - A set of operations, called OUTER joins, can be used when we want to keep all the tuples in R, or all those in S, or all those in both relations in the result of the join, regardless of whether or not they have matching tuples in the other relation.

— —  
— —

# Additional Relational Operations (cont.)

- The **left outer join** operation keeps every tuple in the first or left relation  $R$  in  $R \bowtie_{\text{LOJ}} S$ ; if no matching tuple is found in  $S$ , then the attributes of  $S$  in the join result are filled or “padded” with null values.
- A similar operation, **right outer join**, keeps every tuple in the second or right relation  $S$  in the result of  $R \bowtie_{\text{ROJ}} S$ .
- A third operation, **full outer join**, denoted by  $\bowtie_{\text{FOJ}}$  keeps all tuples in both the left and the right relations when no matching tuples are found, padding them with null values as needed.

# Additional Relational Operations (cont.)

## RESULT

Fname	Minit	Lname	Dname
John	B	Smith	NULL
Franklin	T	Wong	Research
Alicia	J	Zelaya	NULL
Jennifer	S	Wallace	Administration
Ramesh	K	Narayan	NULL
Joyce	A	English	NULL
Ahmad	V	Jabbar	NULL
James	E	Borg	Headquarters

The result of a  
LEFT OUTER JOIN  
operation.

# Examples on following state

## EMPLOYEE

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

## DEPARTMENT

Dname	<u>Dnumber</u>	Mgr_ssn	Mgr_start_date
-------	----------------	---------	----------------

## DEPT\_LOCATIONS

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

## PROJECT

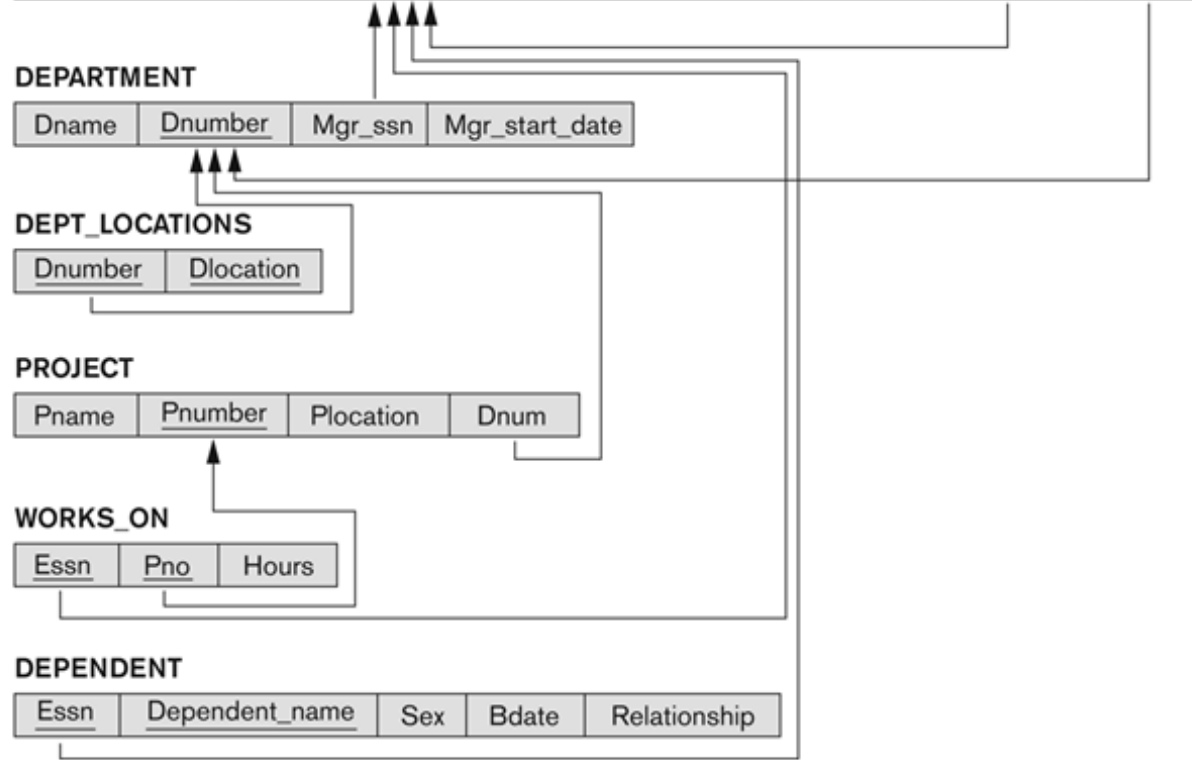
Pname	<u>Pnumber</u>	Plocation	Dnum
-------	----------------	-----------	------

## WORKS\_ON

<u>Essn</u>	<u>Pno</u>	Hours
-------------	------------	-------

## DEPENDENT

<u>Essn</u>	<u>Dependent_name</u>	Sex	Bdate	Relationship
-------------	-----------------------	-----	-------	--------------



Query 1. Retrieve the name and address of all employees who work for the 'Research' department.

$\text{RESEARCH\_DEPT} \leftarrow \sigma_{\text{Dname}='Research'}(\text{DEPARTMENT})$

$\text{RESEARCH\_EMPS} \leftarrow (\text{RESEARCH\_DEPT} \bowtie_{\text{Dnumber}=\text{Dno}} \text{EMPLOYEE})$

$\text{RESULT} \leftarrow \pi_{\text{Fname}, \text{Lname}, \text{Address}}(\text{RESEARCH\_EMPS})$

Can also be written in single line:

$\pi_{\text{Fname}, \text{Lname}, \text{Address}}(\sigma_{\text{Dname}='Research'}(\text{DEPARTMENT} \bowtie_{\text{Dnumber}=\text{Dno}} (\text{EMPLOYEE})))$

Query 2. For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, address, and birth date.

STAFFORD\_PROJS  $\leftarrow \sigma_{Plocation='Stafford'}(PROJECT)$

CONTR\_DEPTS  $\leftarrow (STAFFORD\_PROJS \bowtie_{Dnum=Dnumber} DEPARTMENT)$

PROJ\_DEPT\_MGRS  $\leftarrow (CONTR\_DEPTS \bowtie_{Mgr\_ssn=Ssn} EMPLOYEE)$

RESULT  $\leftarrow \pi_{Pnumber, Dnum, Lname, Address, Bdate}(PROJ\_DEPT\_MGRS)$