DATABASE MANAGEMENT SYSTEM

S Nagasundari

Department of Computer Science and Engineering

DATABASE MANAGEMENT SYSTEM

Relational Model Concepts

Slides adapted from Author Slides of Fundamentals of Database Systems", Ramez Elamsri, Shamkant B Navathe, Pearson, 7th Edition, 2017.

S Nagasundari

Department of Computer Science and Engineering

DATABASE MANAGEMENT SYSTEM

Unit 2 : Relational Model

1. **Relational Model Concepts**
2. Relational Model Constraints and Relational Database Schemas
3. Update Operations, Transactions and dealing with constraint violations
4. Relational Database Design Using ER-to Relational Mapping
5. Unary Relational Operations
6. Set Theory Operations
7. Binary Relational Operations
8. Aggregate Functions and Grouping
9. Examples of Queries in Relational Algebra

This unit covers

- The operations of the relational algebra which is one of the formal languages associated with the relational model.
- Relates the relational model data structures to the constructs of the ER models
- presents algorithms for designing a relational database schema by mapping a conceptual schema in the ER model into a relational representation

T1 : CHAPTER 5

The Relational Data Model Concepts

Relational Model Concepts

- A Relation is a mathematical concept based on the ideas of sets
- The model was first proposed by Dr. E.F. Codd of IBM Research in 1970 in the following paper:
  - "A Relational Model for Large Shared Data Banks," Communications of the ACM, June 1970
- The above paper caused a major revolution in the field of database management and earned Dr. Codd the coveted ACM Turing Award

Why Relational Model???

What is a flat file?

- The flat file refers to any file stored on a local file system, as opposed to a more complex set of files, such as those in a structured database.
- The data records in a flat file are typically stored sequentially and without any metadata, such as the indexes, keys, and relationships, that you would find in database storage.

CSV format

```
Mr,Smith,Ottawa,ON
Mrs,Jones,Winnipeg,MB
```

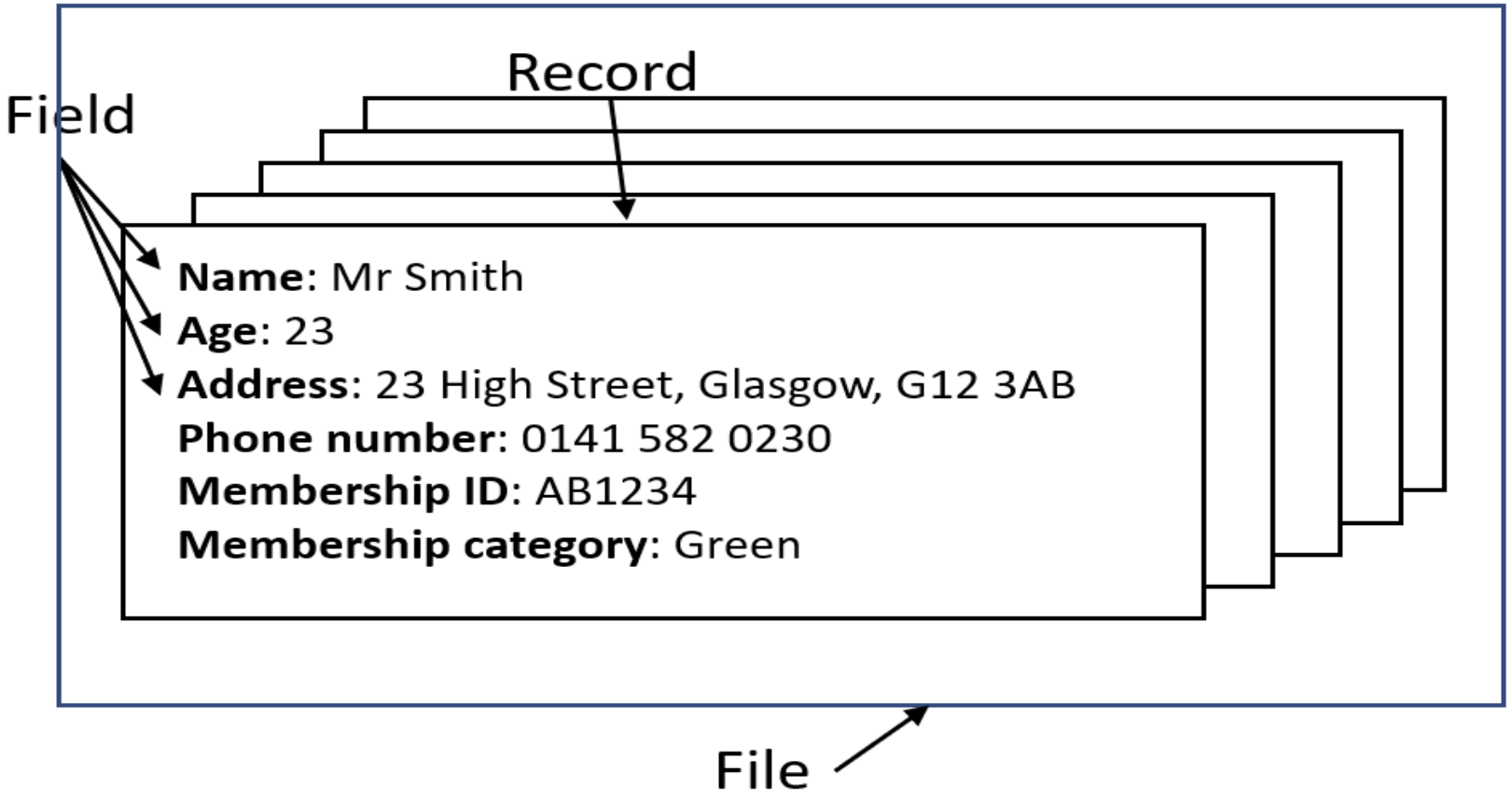
File/Database Structure

Data is organised into **fields**, **records** and **files**:

- **Field** contains an individual piece of data
- **Record** contains one or more fields about a particular person or thing
- **File** is a collection of records

DATABASE MANAGEMENT SYSTEM

File/Database Structure



DATABASE MANAGEMENT SYSTEM

Database Structure (Flat file)

- A **flat file database** organises its data using a **single table**

Flat File Model

	Route No.	Miles	Activity
Record 1	I-95	12	Overlay
Record 2	I-495	05	Patching
Record 3	SR-301	33	Crack seal

DATABASE MANAGEMENT SYSTEM

What is wrong with this database?

Customer ID	Forename	Surname	Pet's Name	Type of Animal
2653	Brian	Davies	Scruff	Labradoodle
1293	Sue	Townes	Phoebe	Syrian Hamster
1293	Sue	Townes	Jango	Syrian Hamster
1293	Sue	Townes	Sandy	Russian Dwarf Hamster
9735	Barry	Grey	Killer	English Bulldog
9735	Barry	Grey	Psyco	French Bulldog
6312	Courtney	Heaven	Jasmine	Chihuahua
2653	Brian	Davies	Jessie	Labrador Retriever
2653	Brian	Davies	Sid	Labradoodle
9935	Tia	Smith	Napa	Beagle
9935	Tia	Smith	Fajita	Cocker Spaniel
1190	Zak	Cassells	Scamper	Rex Rat
1745	Chloe	Wyse	Taboo	Siamese
6689	Jack	Lewis	Dawn	Common Raccoon

DATABASE MANAGEMENT SYSTEM

What is wrong with this database?

Customer ID	Forename	Surname	Pet's Name	Type of Animal
2653	Brian	Davies	Scruff	Labradoodle
1293	Sue	Townes	Phoebe	Syrian Hamster
1293	Sue	Townes	Jango	Syrian Hamster
1293	Sue	Townes	Sandy	Russian Dwarf Hamster
9735	Barry	Grey	Killer	English Bulldog
9735	Barry	Grey	Psyco	French Bulldog
6312	Courtney	Heaven	Jasmine	Chihuahua
2653	Brian	Davies	Jessie	Labrador Retriever
2653	Brian	Davies	Sid	Labradoodle
9935	Tia	Smith	Napa	Beagle
9935	Tia	Smith	Fajita	Cocker Spaniel
1190	Zak	Cassells	Scamper	Rex Rat
1745	Chloe	Wyse	Taboo	Siamese
6689	Jack	Lewis	Dawn	Common Raccoon

Why is that a problem?

Customer ID	Forename	Surname	Pet's Name	Type of Animal
2653	Brian	Davies	Scruff	Labradoodle
1293	Sue	Jones	Phoebe	Syrian Hamster
1293	Sue	Jones	Jango	Syrian Hamster
1293	Sue	Townes	Sandy	Russian Dwarf Hamster
9735	Barry	Grey	Killer	English Bulldog
9735	Barry	Grey	Psyco	French Bulldog
6312	Courtney	Heaven	Jasmine	Chihuahua
2653	Brian	Davies	Jessie	Labrador Retriever
2653	Brian	Davies	Sid	Labradoodle
9935	Tia	Smith	Napa	Beagle
9935	Tia	Smith	Fajita	Cocker Spaniel
1190	Zak	Cassells	Scamper	Rex Rat
1745	Chloe	Wyse	Taboo	Siamese
6689	Jack	Lewis	Dawn	Common Raccoon

Why is that a problem?

Customer ID	Forename	Surname	Pet's Name	Type of Animal
2653	Brian	Davies	Scruff	Labradoodle
1293	Sue	Townes	Sandy	Russian Dwarf Hamster
9735	Barry	Grey	Killer	English Bulldog
9735	Barry	Grey	Psyco	French Bulldog
6312	Courtney	Heaven	Jasmine	Chihuahua
2653	Brian	Davies	Jessie	Labrador Retriever
2653	Brian	Davies	Sid	Labradoodle
9935	Tia	Smith	Napa	Beagle
9935	Tia	Smith	Fajita	Cocker Spaniel
1190	Zak	Cassells	Scamper	Rex Rat
1745	Chloe	Wyse	Taboo	Siamese
6689	Jack	Lewis	Dawn	Common Raccoon

Flat file Database Problems

- Duplicate (repeated) data causes modification errors (update anomalies):
  - might miss a record when you **update** details
  - might miss a record when you need to **delete** details
  - cannot **insert** a new animal into the database without a customer or vice-versa

DATABASE MANAGEMENT SYSTEM

Database Structure (relational)

- A **relational database** stores data in more than 1 table to stop duplication
- Splitting a single table db (flat file) into a relational db (multiple tables) is called **normalisation**

Normalising the Vet table

Customer ID	Forename	Surname
2653	Brian	Davies
1293	Sue	Townes
9735	Barry	Grey
6312	Courtney	Heaven
9935	Tia	Smith
1190	Zak	Cassells
1745	Chloe	Wyse
6689	Jack	Lewis

Customer ID	Pet's Name	Type of Animal
2653	Scruff	Labradoodle
1293	Phoebe	Syrian Hamster
1293	Jango	Syrian Hamster
1293	Sandy	Russian Dwarf Hamster
9735	Killer	English Bulldog
9735	Psyco	French Bulldog
6312	Jasmine	Chihuahua
2653	Jessie	Labrador Retriever
2653	Sid	Labradoodle
9935	Napa	Beagle
9935	Fajita	Cocker Spaniel
1190	Scamper	Rex Rat
1745	Taboo	Siamese
6689	Dawn	Common Raccoon

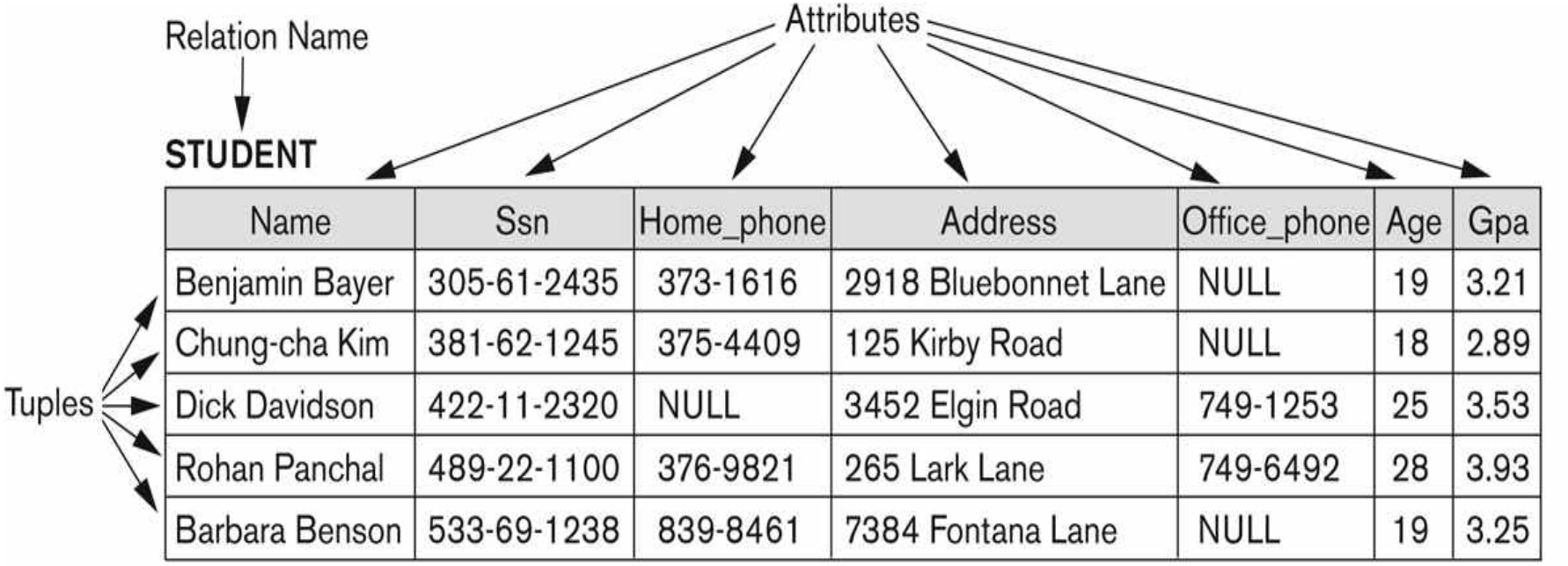
How does this solve the problem?

- Splitting the table into two tables means:
  - less data storage required - information is now only stored once
  - updates are less error prone as not multiple records of same information to update

Informal Definitions

- Informally, a **relation** looks like a **table** of values.
- A relation typically contains a **set of rows**.
- The data elements in each **row** represent certain facts that correspond to a real-world **entity** or **relationship**
  - In the formal model, rows are called **tuples**
- Each **column** has a column header that gives an indication of the meaning of the data items in that column
  - In the formal model, the column header is called an **attribute name** (or just **attribute**)

Example of a Relation



Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	749-1253	25	3.53
Rohan Panchal	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	NULL	19	3.25

Figure 5.1
The attributes and tuples of a relation STUDENT.

Informal Definitions

Key of a Relation:

- Each row has a value of a data item (or set of items) that uniquely identifies that row in the table
  - Called the *key*
- In the STUDENT table, SSN is the key
- Sometimes row-ids or sequential numbers are assigned as keys to identify the rows in a table
  - Called *artificial key* or *surrogate key*

Formal Definitions - Schema

The **Schema** (or description) of a Relation:

- Denoted by R(A1, A2, ...An)
- R is the **name** of the relation
- The **attributes** of the relation are A1, A2, ..., An

Each attribute has a **domain** or a set of valid values.

- For example, the domain of Cust-id is 6 digit numbers.

Example:

CUSTOMER (Cust-id, Cust-name, Address, Phone#)

CUSTOMER is the relation name

Defined over the four attributes: Cust-id, Cust-name, Address, Phone#

Formal Definitions - Tuple

- A **tuple** is an ordered set of values (enclosed in angled brackets '< … >')
- Each value is derived from an appropriate *domain*.
- A row in the CUSTOMER relation is a 4-tuple and would consist of four values, for example:
  - <632895, "John Smith", "101 Main St. Atlanta, GA 30332", "(404) 894-2000">
  - This is called a 4-tuple as it has 4 values
  - A tuple (row) in the CUSTOMER relation.
- A relation is a **set** of such tuples (rows)

Formal Definitions - Domain

- A **domain** has a logical definition:
  - Example: “USA_phone_numbers” are the set of 10 digit phone numbers valid in the U.S.
- A domain also has a data-type or a format defined for it.
  - The USA_phone_numbers may have a format: (ddd)ddd-dddd where each d is a decimal digit.
  - Dates have various formats such as year, month, date formatted as yyyy-mm-dd, or as dd mm,yyyy etc.

Formal Definitions - Domain

- The attribute name designates the role played by a domain in a relation:
  - Used to interpret the meaning of the data elements corresponding to that attribute
  - Example: The domain Date may be used to define two attributes named “Invoice-date” and “Payment-date” with different meanings

Formal Definitions - 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)
- The role these strings play in the CUSTOMER relation is that of the *name of a customer*.

Formal Definitions - Summary

Formally,

- Given R(A1, A2, ..., An)
- 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)

Formal Definitions - Example

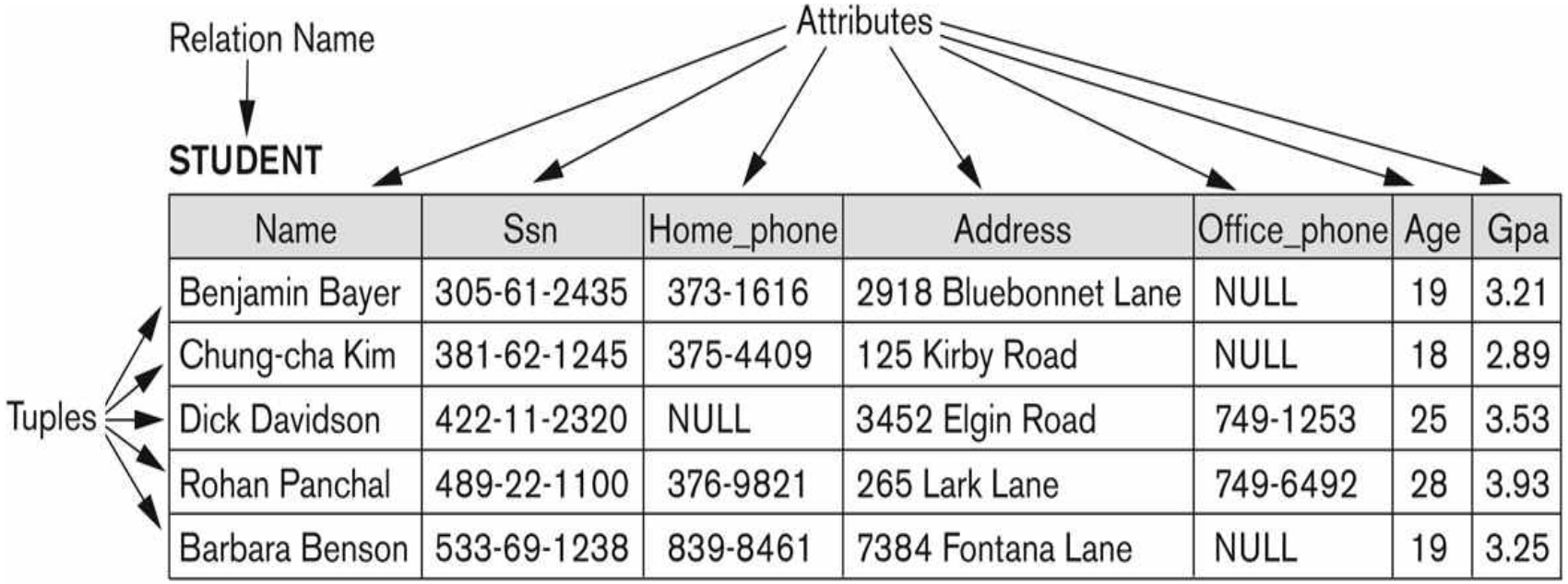
- Let R(A1, A2) be a relation schema:
  - Let dom(A1) = {0,1}
  - Let dom(A2) = {a,b,c}
- Then: dom(A1) X dom(A2) is all possible combinations:

  {<0,a> , <0,b> , <0,c>, <1,a>, <1,b>, <1,c> }
- The relation state r(R) ⊂ dom(A1) X dom(A2)
- For example: r(R) could be {<0,a> , <0,b> , <1,c> }
  - this is one possible state (or “population” or “extension”) r of the relation R, defined over A1 and A2.
  - It has three 2-tuples: <0,a> , <0,b> , <1,c>

Definition Summary

Informal Terms	Formal Terms
Table	Relation
Column Header	Attribute
All possible Column Values	Domain
Row	Tuple
Table Definition	Schema of a Relation
Populated Table	State of the Relation

Example-A Relation Student



Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	749-1253	25	3.53
Rohan Panchal	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	NULL	19	3.25

Figure 5.1
The attributes and tuples of a relation STUDENT.

Characteristics of Relations

Ordering of tuples in a relation r(R):

- The tuples are *not considered to be ordered*, even though they appear to be in the tabular form.

Ordering of attributes in a relation schema R

(and of values within each tuple):

- We will consider the attributes in R(A1, A2, ..., An) and the values in t=<v1, v2, ..., vn> to be ordered .
  - (However, a more general alternative definition of relation does not require this ordering. It includes both the name and the value for each of the attributes).
  - Example: t= { <name, “John” >, <SSN, 123456789> }
  - This representation may be called as “self-describing”.

Same state as previous figure(But with different order of tuples)

Figure 5.2

The relation STUDENT from Figure 5.1 with a different order of tuples.

STUDENT

Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	749-1253	25	3.53
Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	NULL	19	3.25
Rohan Panchal	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
Chung-cha Kim	381-62-1245	375-4409	125 Kirby Road	NULL	18	2.89
Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	NULL	19	3.21

Characteristics of Relations

Values in a tuple:

- All values are considered atomic (indivisible).
- Each value in a tuple must be from the domain of the attribute for that column
  - If tuple t = <v1, v2, …, vn> is a tuple (row) in the relation state r of R(A1, A2, …, An)
  - Then each *vi* must be a value from *dom(Ai)*
- A special **null** value is used to represent values that are unknown or not available or inapplicable in certain tuples.

Characteristics of Relations

Notation:

- We refer to **component values** of a tuple t by:
  - t[Ai] or t.Ai
  - This is the value vi of attribute Ai for tuple t
- Similarly, t[Au, Av, ..., Aw] refers to the subtuple of t containing the values of attributes Au, Av, ..., Aw, respectively in t

THANK YOU

S Nagasundari
Department of Computer Science and Engineering

nagasundaris@pes.edu

DATABASE MANAGEMENT SYSTEM

S Nagasundari

Department of Computer Science and Engineering

DATABASE MANAGEMENT SYSTEM

Relational Model Constraints and Database Schema

Slides adapted from Author Slides of Fundamentals of Database Systems", Ramez Elamsri, Shamkant B Navathe, Pearson, 7th Edition, 2017.

S Nagasundari

Department of Computer Science and Engineering

DATABASE MANAGEMENT SYSTEM

Unit 2 : Relational Model

1. Relational Model Concepts
2. **Relational Model Constraints and Relational Database Schemas**
3. Update Operations, Transactions and dealing with constraint violations
4. Relational Database Design Using ER-to Relational Mapping
5. Unary Relational Operations
6. Set Theory Operations
7. Binary Relational Operations
8. Aggregate Functions and Grouping
9. Examples of Queries in Relational Algebra

PES UNIVERSITY ONLINE

CHAPTER 5

The Relational Data Model and Relational Database Constraints

Constraints

Constraints determine which values are permissible and which are not in the database. They are of three main types:

1. **Inherent or Implicit Constraints**:
   - Based on the data model itself. (E.g., relational model does not allow a list as a value for any attribute)

2. **Schema-based or Explicit Constraints**:
   - Expressed in the schema by using the facilities provided by the model. (E.g., max. cardinality ratio constraint in the ER model)

3. **Application based or semantic constraints**:
   - Beyond the expressive power of the model and must be specified and enforced by the application programs.

Relational Integrity Constraints

- Constraints are **conditions** that must hold on **all** valid relation states.
- *The Main types* of (explicit schema-based) constraints that can be expressed in the relational model:
  - **Domain** constraint.
  - **Key** constraints and Constraints on NULL values
  - **Integrity C**onstraints
    - **Entity integrity** constraints
    - **Referential integrity** constraints

Domain Constraints

Domain Constraint specifies that

- Within each tuple, the value of each attribute A must be an atomic value from the domain dom(A)
- Every value in a tuple must be from the *domain of its attribute* (or it could be **null**, if allowed for that attribute)

DATABASE MANAGEMENT SYSTEM

Domain Constraints

The data types associated with domains include

- Standard numeric data types for integers (such as short integer, integer, and long integer)
- Real numbers (float and double-precision float).
- Characters,
- Booleans,
- Fixed-length strings, Variable-length strings, as are date, time, timestamp, and other special data types.
- Domains can also be described by a subrange of values from a data type or
- as an enumerated data type in which all possible values are explicitly listed

DATABASE MANAGEMENT SYSTEM

Key Constraints

Superkey of R:

- 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] ≠ t2[SK]
  - This condition must hold in *any valid state* r(R)

Key of R satisfies two properties:

1. Two distinct tuples in any state of the relation cannot have identical values for (all) the attributes in the key. This uniqueness property also applies to a superkey.
2. A "minimal" superkey
   - 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)

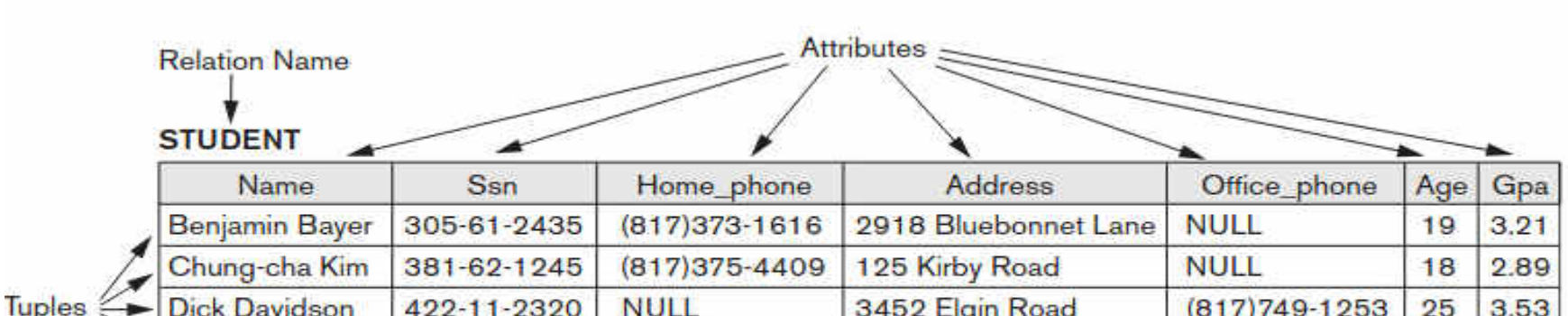
Key Constraints (Continued)

Example: Consider the STUDENT relation

- {Ssn} is a key of STUDENT
- {Ssn, Name, Age}—is a superkey.
- However, the superkey {Ssn, Name, Age} is not a key of STUDENT because removing Name or Age or both from the set still leaves us with a superkey.
- A key with multiple attributes must require all its attributes together to have the uniqueness property

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 and key is timeinvariant



Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	(817)373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	(817)375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	(817)749-1253	25	3.53

Key Constraints (Continued)

A relation schema may have more than one key.

In this case, each of the keys is called a candidate key.

Example: Consider the CAR relation schema:

- CAR(State, Reg#, SerialNo, Make, Model, Year)
- CAR has two keys:
  - Key1 = {State, Reg#}
  - Key2 = {SerialNo}
- Both are also superkeys of CAR
- {SerialNo, Make} is a superkey but *not* a key.

CAR Table with two candidate

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.

Key Constraints (Continued)

- If a relation has several **candidate keys**, one is chosen arbitrarily to be the **primary key**.
  - The primary key attributes are underlined.
- Example: Consider the CAR relation schema:
  - CAR(State, Reg#, SerialNo, Make, Model, Year)
  - We chose SerialNo as the primary key
- The primary key value is used to *uniquely identify* each tuple in a relation
  - Provides the tuple identity
- Also used to *reference* the tuple from another tuple

  General rule: Choose as primary key the smallest of the candidate keys (in terms of size)

  Not always applicable – choice is sometimes subjective

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} and a set IC of integrity constraints.
- R1, R2, …, Rn are the names of the individual **relation schemas** within the database S

COMPANY Database Schema

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

COMPANY database schema with six relation schemas

Figure 5.5
Schema diagram for the COMPANY relational database schema.

Relational Database State

- A **relational database state** DB of *S* is a set of relation states

  DB = $\{r_1, r_2, ..., r_m\}$ such that each r_i is a state of R_i and such that the r_i relation states satisfy the integrity constraints specified in IC.

- A relational database *state* is sometimes called a relational database *snapshot* or *instance*.

- A database state that does not meet the constraints is an invalid state

Note: Not use the term instance since it also applies to single tuples.

Relational Database State

- A database state that does not obey all the integrity constraints is called **not valid**, and
- A state that satisfies all the constraints in the defined set of integrity constraints IC is called a **valid state**

Populated Database State

- Each *relation* will have many tuples in its current relation state
- The *relational database state* is a union of all the individual relation states
- Whenever the database is changed, a new state arises
- **Basic operations for changing the database**:
  - INSERT a new tuple in a relation
  - DELETE an existing tuple from a relation
  - MODIFY an attribute of an existing tuple

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

Example state for the COMPANY database schema

THANK YOU

S Nagasundari
Department of Computer Science and Engineering

nagasundaris@pes.edu

DATABASE MANAGEMENT SYSTEM

S Nagasundari

Department of Computer Science and Engineering

DATABASE MANAGEMENT SYSTEM

Update Operations and Constraint Violations

Slides adapted from Author Slides of Fundamentals of Database Systems", Ramez Elamsri, Shamkant B Navathe, Pearson, 7th Edition, 2017.

S Nagasundari

Department of Computer Science and Engineering

DATABASE MANAGEMENT SYSTEM

Unit 2 : Relational Model

1. Relational Model Concepts
2. Relational Model Constraints and Relational Database Schemas
3. **Update Operations, Transactions and dealing with constraint violations**
4. Relational Database Design Using ER-to Relational Mapping
5. Unary Relational Operations
6. Set Theory Operations
7. Binary Relational Operations
8. Aggregate Functions and Grouping
9. Examples of Queries in Relational Algebra

PES UNIVERSITY ONLINE

CHAPTER 5

The Relational Data Model and Relational Database Constraints

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] ≠ null for any tuple t in r(R)
  - If PK has several attributes, null is not allowed in any of these attributes
- Note: Other attributes of R may be constrained to disallow null values, even though they are not members of the primary key.

Referential Integrity

- A constraint involving **two** relations
  - The previous constraints involve a single relation.
- Used to specify a **relationship** among tuples in two relations:
  - The **referencing relation** and the **referenced relation**.

Referential Integrity

- 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.

Referential Integrity (or foreign key) Constraint

- The conditions for a foreign key, given below, specify a referential integrity constraint between the two relation schemas *R*1 and *R*2.
- A set of attributes FK in relation schema *R*1 is a **foreign key** of *R*1 that **references** relation *R*2 if it satisfies the following rules:
    1. The attributes in FK have the same domain(s) as the primary key attributes PK of *R*2; the attributes FK are said to **reference** or **refer to** the relation *R*2.

Referential Integrity (or foreign key) Constraint

2. A value of FK in a tuple *t*1 of the current state *r*1(*R*1) either occurs as a value of PK for some tuple *t*2 in the current state *r*2(*R*2) *or is NULL*. In the former case, we have *t*1[FK] = *t*2[PK], and we say that the tuple *t*1 **references** or **refers to** the tuple *t*2.

- In this definition, *R*1 is called the **referencing relation** and *R*2 is the **referenced relation**. If these two conditions hold, a **referential integrity constraint** from *R*1 to *R*2 is said to hold.

Referential Integrity (or foreign key) Constraint

PES UNIVERSITY ONLINE

Statement of the constraint

- The value in the foreign key column (or columns) FK of the **referencing relation** R1 can be **either**:

  (1) a value of an existing primary key value of a corresponding primary key PK in the **referenced relation** R2, or

  (2) a **null**.

In case (2), the FK in R1 should **not** be a part of its own primary key.

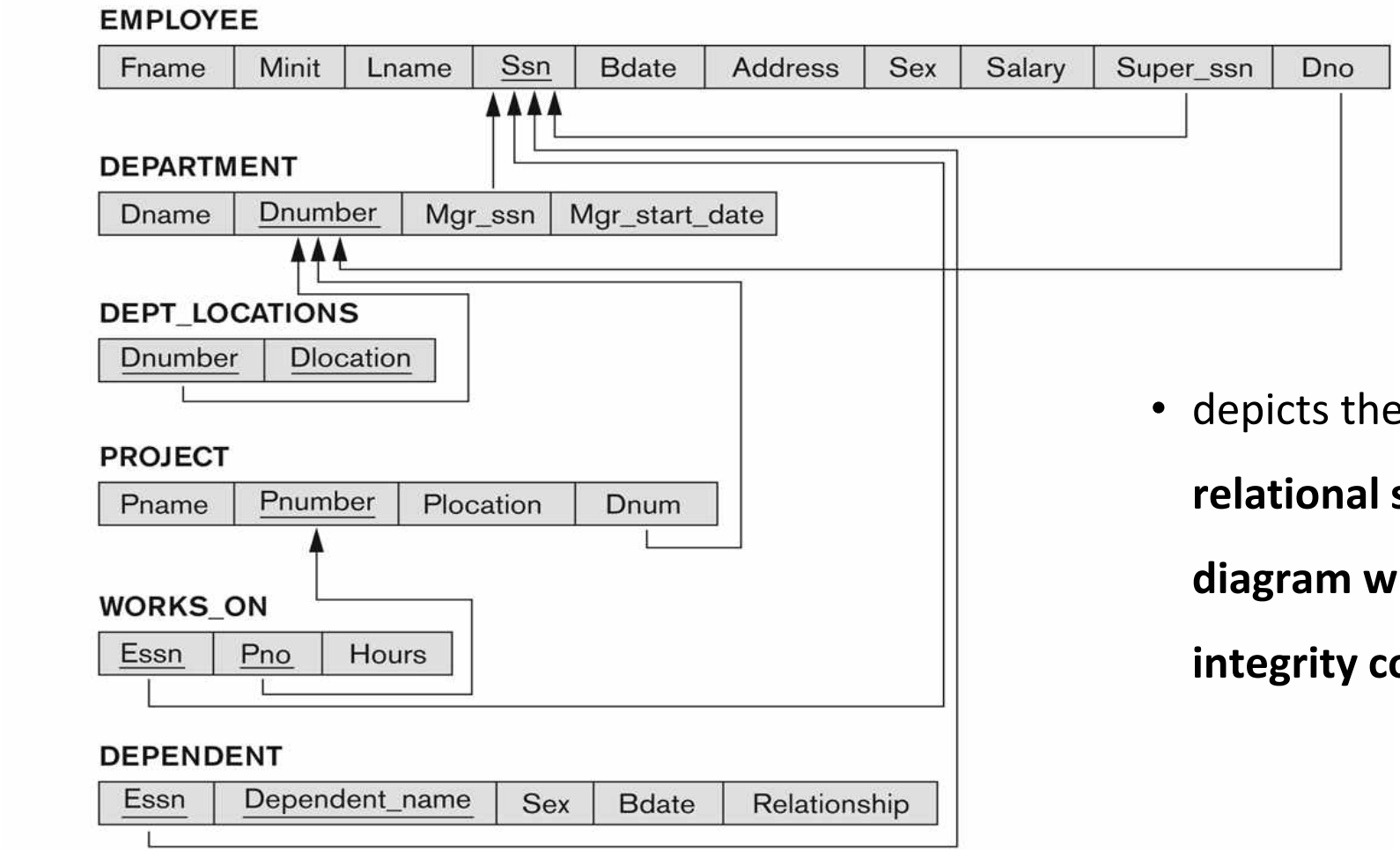
Displaying a relational database schema and its constraints

- 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
    - Can also point the primary key of the referenced relation for clarity

DATABASE MANAGEMENT SYSTEM

Referential Integrity Constraints for COMPANY database

Figure 5.7
Referential integrity constraints displayed on the COMPANY relational database schema.



- depicts the COMPANY **relational schema diagram with referential integrity constraints**

Other Types of Constraints

- Semantic Integrity Constraints:
  - based on application semantics and cannot be expressed by the model per se
  - Example: “the max. no. of hours per employee for all projects he or she works on is 56 hrs per week”
- A **constraint specification** language may have to be used to express these
- SQL-99 allows **CREATE TRIGGER** and **CREATE ASSERTION** to express some of these semantic constraints
- Keys, Permissibility of Null values, Candidate Keys (Unique in SQL), Foreign Keys, Referential Integrity etc. are expressed by the **CREATE TABLE** statement in SQL.

Other Types of Constraints

- **State constraints**

  because they define the constraints that a *valid state* of the database must satisfy.

- **Transition constraints**, can be defined to deal with state changes in the database.

  Example of a transition constraint is:

  “the salary of an employee can only increase.”

  Such constraints are typically enforced

  by the application programs or specified using active rules and triggers

Update Operations on Relations

Three basic operations that can change the states of relations in the database

- INSERT a tuple.
- DELETE a tuple.
- MODIFY a tuple.

- Integrity constraints should not be violated by the update operations.
- Several update operations may have to be grouped together.
- Updates may **propagate** to cause other updates automatically. This may be necessary to maintain integrity constraints.

Update Operations on Relations

Discuss the types of constraints that may be violated by each of the operations and the types of actions that may be taken if an operation causes a violation.

In case of integrity violation, several actions can be taken:

- Cancel the operation that causes the violation (RESTRICT or REJECT option)
- Perform the operation but inform the user of the violation
- Trigger additional updates so the violation is corrected (CASCADE option, SET NULL option)
- Execute a user-specified error-correction routine

Possible violations for Insert operation

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

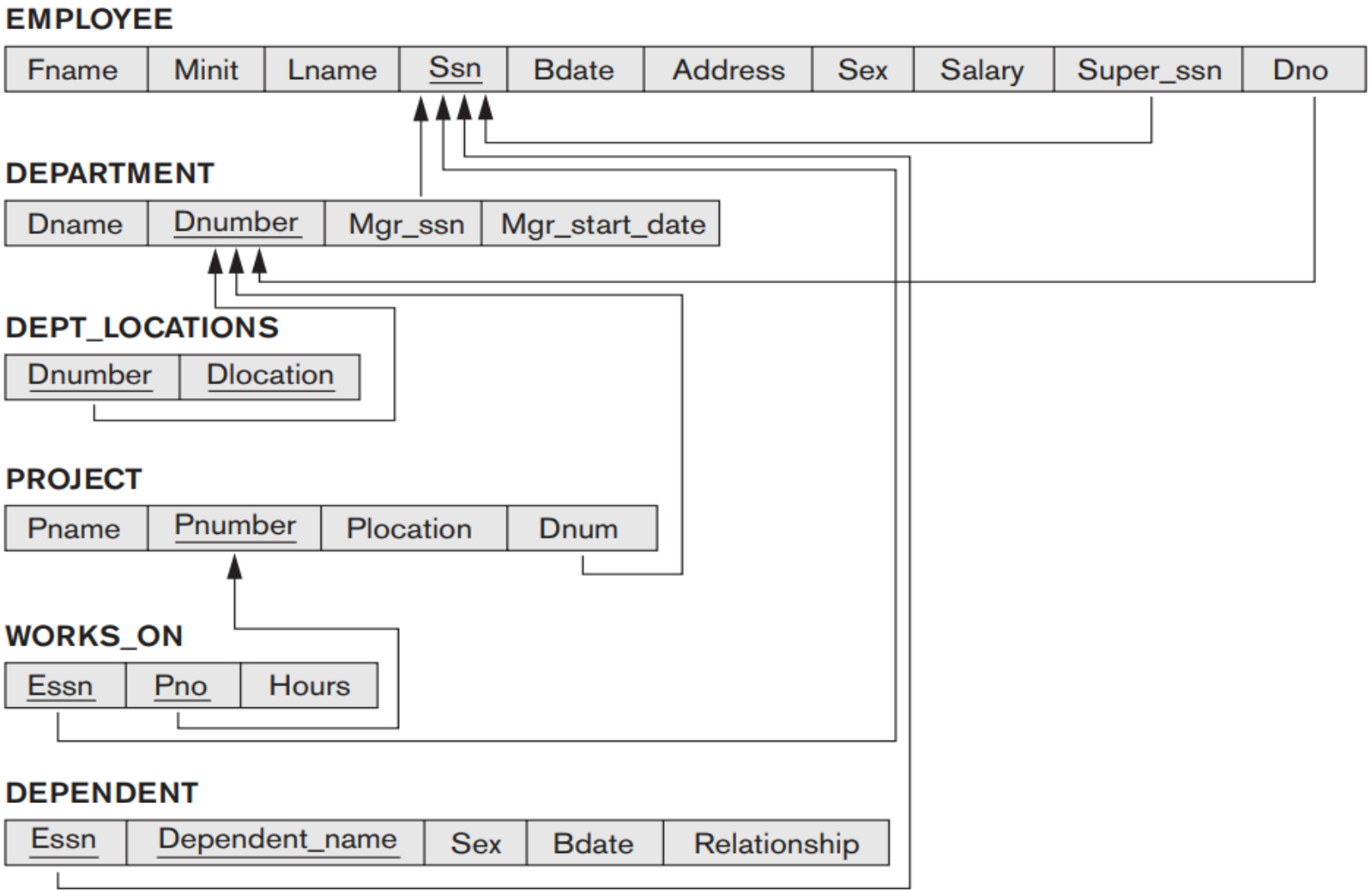
Possible violations for Insert operation

INSERT may violate any of the constraints:

- 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

DATABASE MANAGEMENT SYSTEM

Database schema



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

Example state for the COMPANY database schema

Possible Example violations for Insert operation

- *Operation*:
  Insert <‘Cecilia’, ‘F’, ‘Kolonsky’, NULL, ‘1960-04-05’, ‘6357 Windy Lane, Katy, TX’, F, 28000, NULL, 4> into EMPLOYEE.

  Result: This insertion violates the entity integrity constraint (NULL for the primary key Ssn), so it is rejected.

- *Operation*:
  Insert <‘Alicia’, ‘J’, ‘Zelaya’, ‘999887777’, ‘1960-04-05’, ‘6357 Windy Lane, Katy, TX’, F, 28000, ‘987654321’, 4> into EMPLOYEE.

  Result: This insertion violates the key constraint because another tuple with the same Ssn value already exists in the EMPLOYEE relation, and so it is rejected.

Possible Example violations for Insert operation

- *Operation*:
  Insert <‘Cecilia’, ‘F’, ‘Kolonsky’, ‘677678989’, ‘1960-04-05’, ‘6357 Windswept, Katy, TX’, F, 28000, ‘987654321’, 7> into EMPLOYEE.

  Result: This insertion violates the referential integrity constraint specified on Dno in EMPLOYEE because no corresponding referenced tuple exists in DEPARTMENT with Dnumber = 7.

- *Operation*:
  Insert <‘Cecilia’, ‘F’, ‘Kolonsky’, ‘677678989’, ‘1960-04-05’, ‘6357 Windy Lane, Katy, TX’, F, 28000, NULL, 4> into EMPLOYEE.

  Result: This insertion satisfies all constraints, so it is acceptable.

Possible violations for Insert operation

If an insertion violates one or more constraints, the default option is to *reject the insertion.*

It would be useful if the DBMS could

- provide a reason to the user as to why the insertion was rejected.
- attempt to *correct the reason for rejecting the insertion,* but this is *typically not used for violations caused by Insert*; rather, it is used more often in correcting violations for Delete and Update.

Possible violations for Insert operation

- Operation 1: ask the user to provide a value for Ssn, and could then accept the insertion if a valid Ssn value is provided.
- Operation 3: ask the user to change the value of Dno to some valid value (or set it to NULL), or it could ask the user to insert a DEPARTMENT tuple with Dnumber = 7 and could accept the original insertion only after such an operation was accepted.
- Notice that in the latter case the insertion violation can **cascade** back to the EMPLOYEE relation if the user attempts to insert a tuple for department 7 with a value for Mgr_ssn that does not exist in the EMPLOYEE relation.

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

Example state for the COMPANY database schema

Possible violations for Delete operation

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 the new primary key value into the foreign keys of the referencing tuples
    - SET NULL option: set the foreign keys of the referencing tuples to NULL
- One of the above options must be specified during database design for each foreign key constraint

Possible violations for Delete operation

- *Operation*:

  Delete the WORKS_ON tuple with Essn = '999887777' and Pno = 10.

  Result: This deletion is acceptable and deletes exactly one tuple.

- *Operation*:

  Delete the EMPLOYEE tuple with Ssn = '999887777'.

  Result: This deletion is not acceptable, because there are tuples in WORKS_ON that refer to this tuple. Hence, if the tuple in EMPLOYEE is deleted, referential integrity violations will result.

Possible violations for Delete operation

- *Operation*:

  Delete the EMPLOYEE tuple with Ssn = ‘333445555’.

  Result: This deletion will result in even worse referential integrity violations, because the tuple involved is referenced by tuples from the EMPLOYEE, DEPARTMENT, WORKS_ON, and DEPENDENT relations.

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

Example state for the COMPANY database schema

Possible violations for Update operation

UPDATE may violate domain constraint and NOT NULL constraint on an attribute being modified

- Any of the other constraints may also be violated, depending on the attribute being updated:
  - Updating the primary key (PK):
    - Like DELETE followed by an INSERT
    - Need to specify similar options to DELETE
  - Updating a foreign key (FK):
    - May violate referential integrity
  - Updating an ordinary attribute (neither PK nor FK):
    - Can only violate domain constraints

Possible violations for Update operation

- *Operation*:

  Update the salary of the EMPLOYEE tuple with Ssn = ‘999887777’ to 28000.

  Result: Acceptable.

- *Operation*:

  Update the Dno of the EMPLOYEE tuple with Ssn = ‘999887777’ to 1.

  Result: Acceptable.

Possible violations for Update operation

- *Operation*:

  Update the Dno of the EMPLOYEE tuple with Ssn = ‘999887777’ to 7.

  Result: Unacceptable, because it violates referential integrity.

- *Operation*:

  Update the Ssn of the EMPLOYEE tuple with Ssn = ‘999887777’ to ‘987654321’.

  Result: Unacceptable, because it violates primary key constraint by repeating a value that already exists as a primary key in another tuple; it violates referential integrity constraints because there are other relations that refer to the existing value of Ssn.

Summary

- Presented Relational Model Concepts
  - Definitions
  - Characteristics of relations
- Discussed Relational Model Constraints and Relational Database Schemas
  - Domain constraints
  - Key constraints
  - Entity integrity
  - Referential integrity
- Described the Relational Update Operations and Dealing with Constraint Violations

DATABASE MANAGEMENT SYSTEM

In-Class Exercise

(Taken from Exercise 5.15)

Consider the following relations for a database that keeps track of student enrollment in courses and the books adopted for each course:

STUDENT(SSN, Name, Major, Bdate)

COURSE(Course#, Cname, Dept)

ENROLL(SSN, Course#, Quarter, Grade)

BOOK_ADOPTION(Course#, Quarter, Book_ISBN)

TEXT(Book_ISBN, Book_Title, Publisher, Author)

Draw a relational schema diagram specifying the foreign keys for this schema.

THANK YOU

S Nagasundari
Department of Computer Science and Engineering

nagasundaris@pes.edu

DATABASE MANAGEMENT SYSTEM

S Nagasundari

Department of Computer Science and Engineering

DATABASE MANAGEMENT SYSTEM

Relational Database Design using ER to Relational mapping

Slides adapted from Author Slides of Fundamentals of Database Systems", Ramez Elamsri, Shamkant B Navathe, Pearson, 7th Edition, 2017.

S Nagasundari

Department of Computer Science and Engineering

DATABASE MANAGEMENT SYSTEM

Unit 2 : Relational Model

1. Relational Model Concepts
2. Relational Model Constraints and Relational Database Schemas
3. Update Operations, Transactions and dealing with constraint violations
4. Relational Database Design Using ER-to Relational Mapping
5. Unary Relational Operations
6. Set Theory Operations
7. Binary Relational Operations
8. Aggregate Functions and Grouping
9. Examples of Queries in Relational Algebra

CHAPTER 9

Relational Database Design by ER- to-Relational Mapping

Outline

ER-to-Relational Mapping Algorithm

Step 1: Mapping of Regular Entity Types

Step 2: Mapping of Weak Entity Types

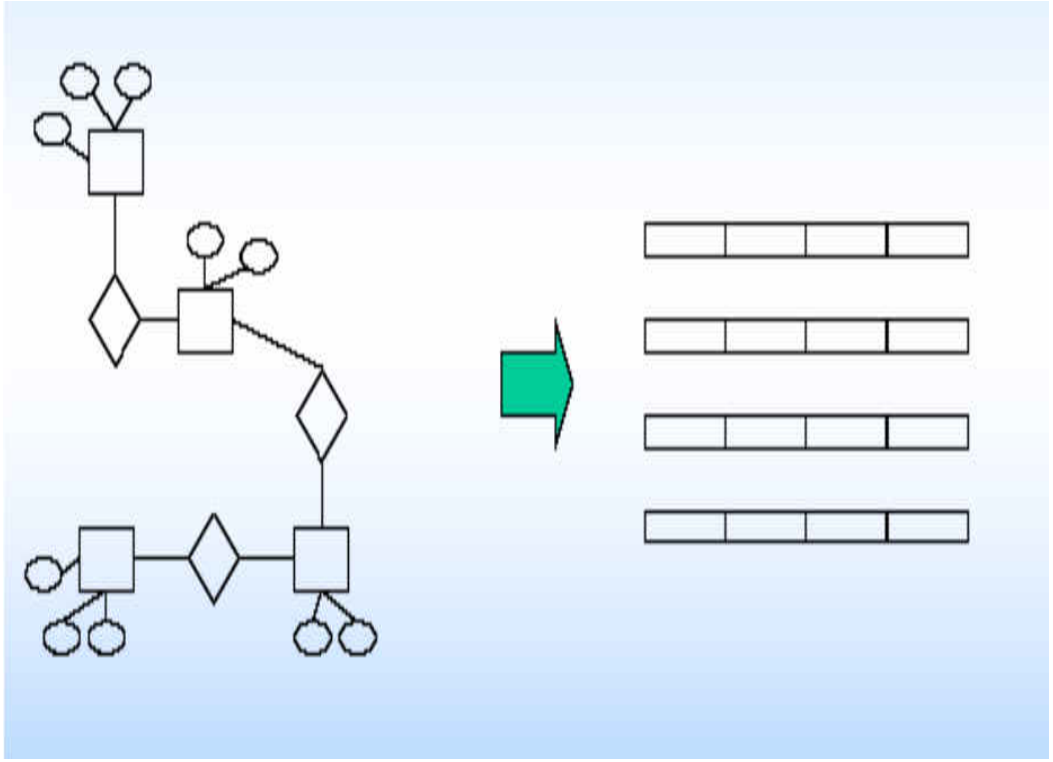
Step 3: Mapping of Binary 1:1 Relation Types

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.



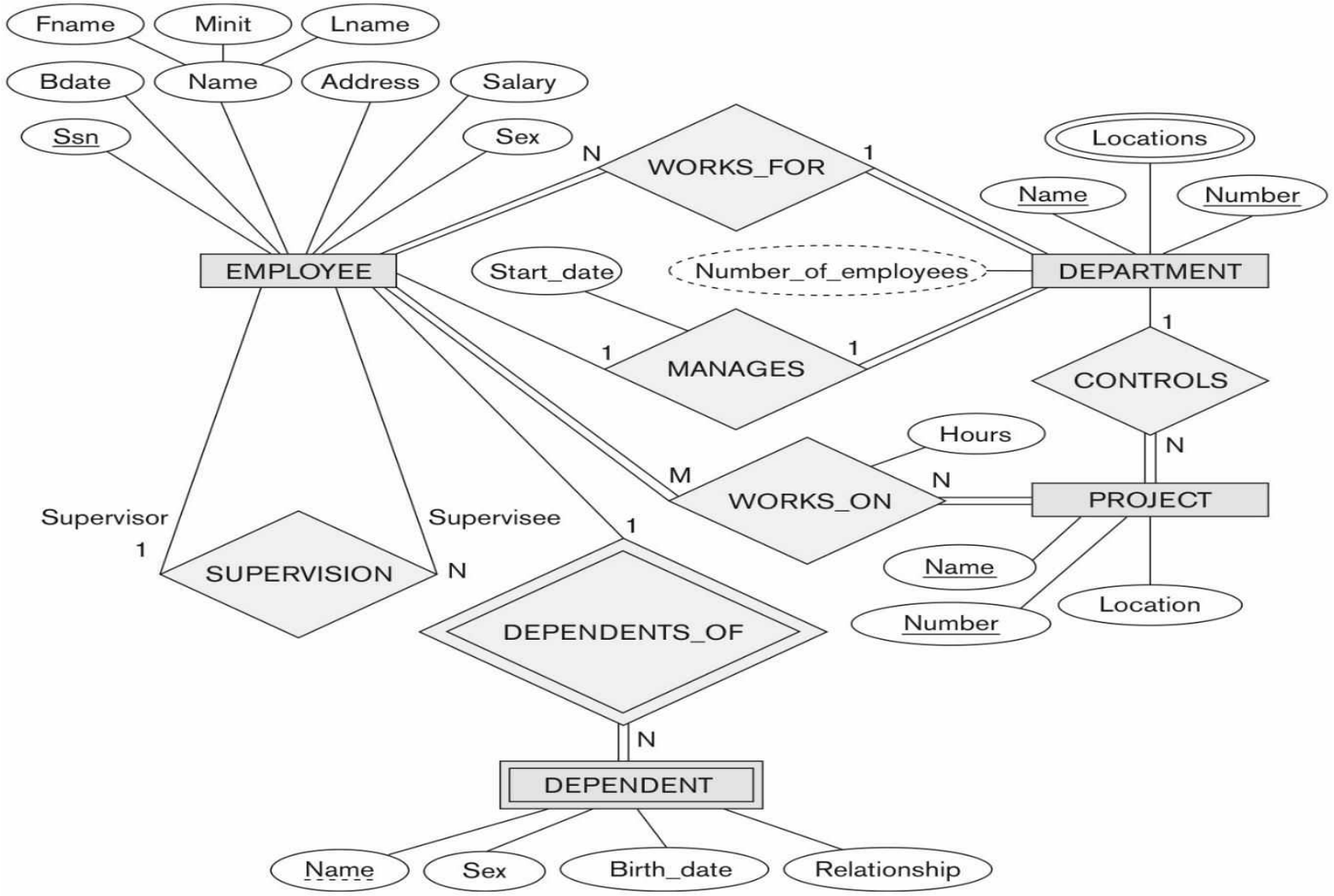
Goals during Mapping

- Preserve all information (that includes all attributes)
- Maintain the constraints to the extent possible (Relational Model cannot preserve all contstraints- e.g., max cardinality ratio such as 1:10 in ER; exhaustive classification into subtypes, e.g., STUDENTS are specialized into Domestic and Foreign)
- Minimize null values

The mapping procedure described has been implemented in many commercial tools.

DATABASE MANAGEMENT SYSTEM

ER Conceptual schema diagram for Company Database



ER to Relational Mapping Algorithm

PES UNIVERSITY ONLINE

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.

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 1 : Mapping of Regular Entity Types

Result

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary

DEPARTMENT

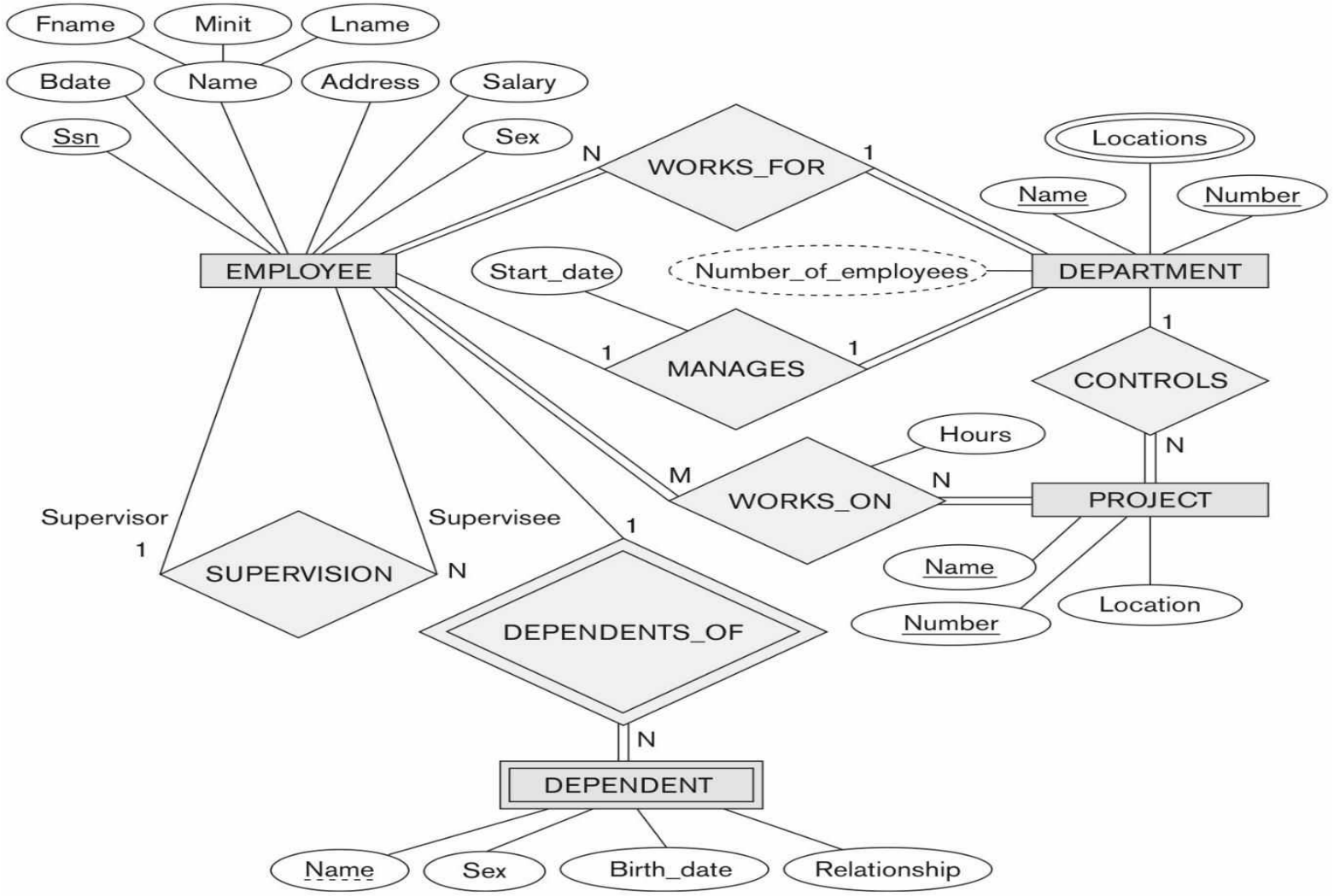
Dname	Dnumber

PROJECT

Pname	Pnumber	Plocation

DATABASE MANAGEMENT SYSTEM

ER Conceptual schema diagram for Company Database



ER to Relational Mapping Algorithm (Continued)

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 & 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).
- 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.

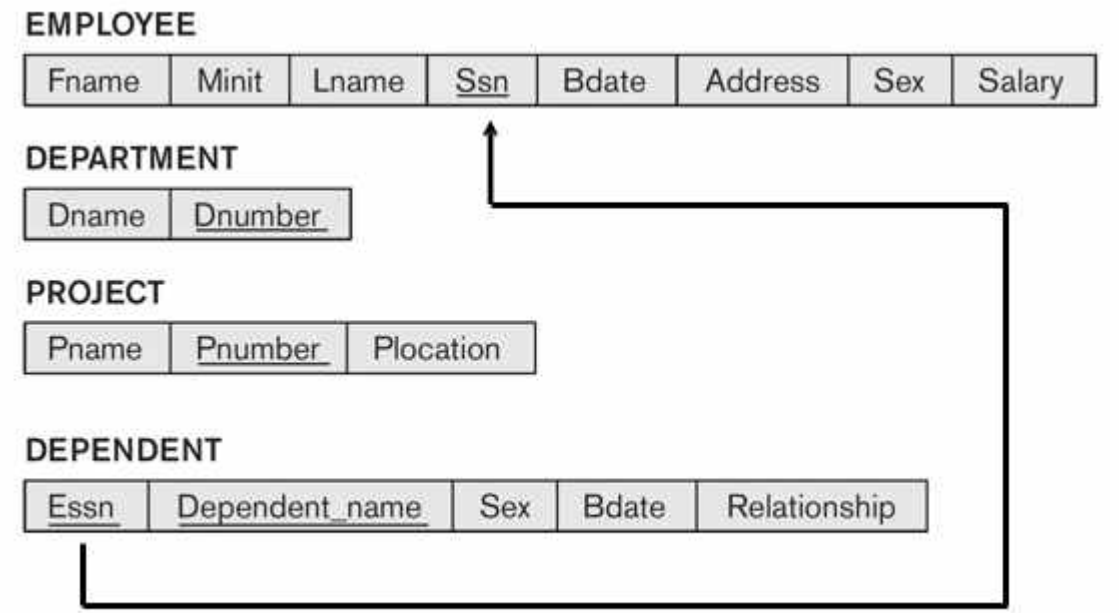
ER to Relational Mapping Algorithm (Continued)

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.

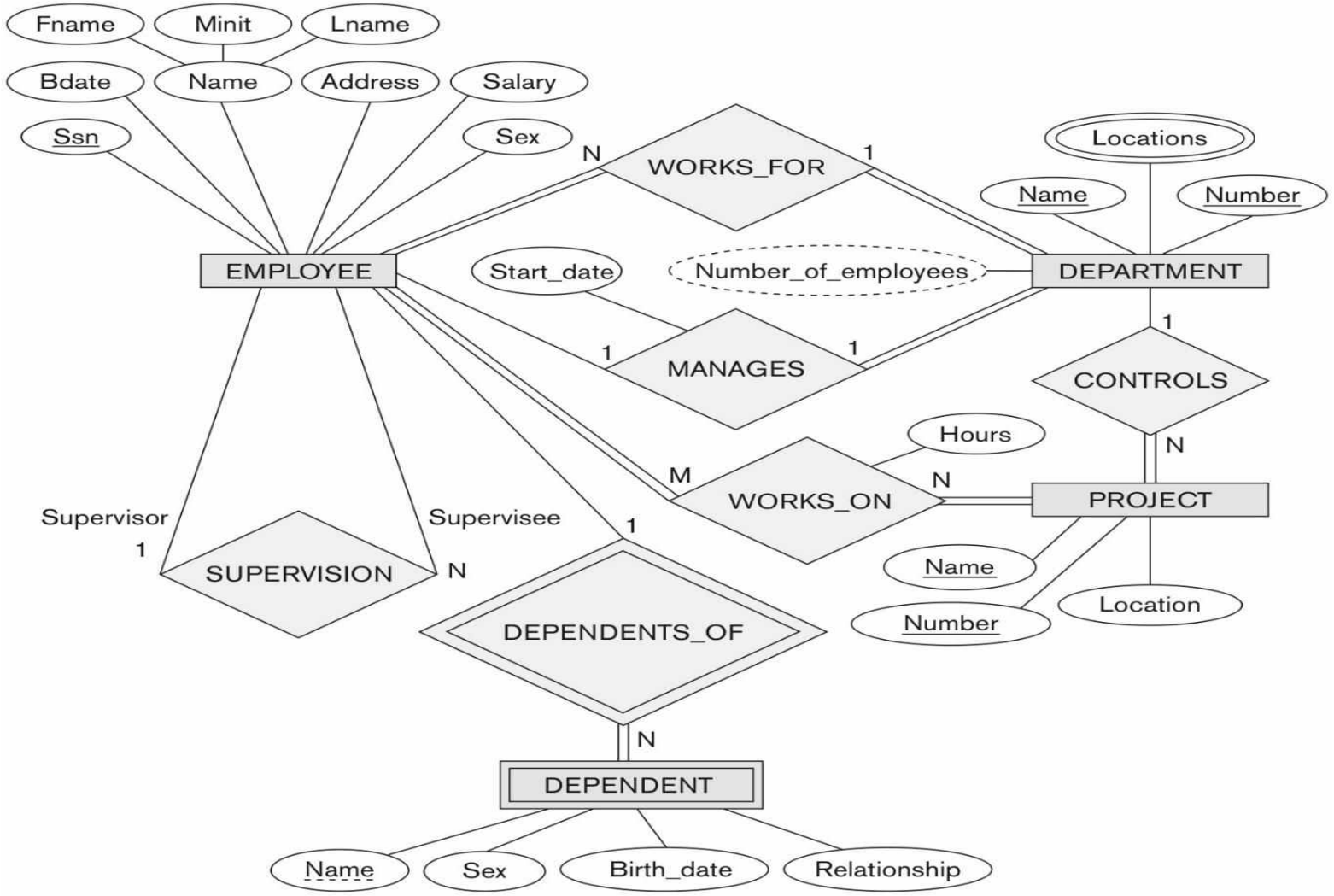
Step 2 : Mapping of Weak Entity Types

Result



DATABASE MANAGEMENT SYSTEM

ER Conceptual schema diagram for Company Database



ER to Relational Mapping Algorithm (Continued)

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 (2 relations) approach:** 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.

   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.

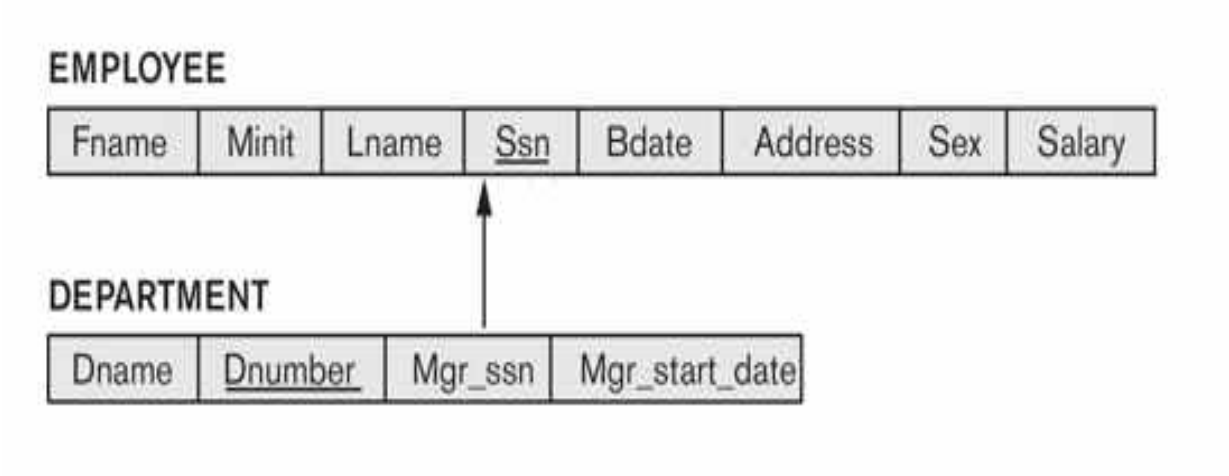
ER to Relational Mapping Algorithm (Continued)

Step 3: Mapping of Binary 1:1 Relation Types

2. **Merged relation (1 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 (3 relations) option:** The third alternative is to set up a third relation R for the purpose of cross-referencing the primary keys of the two relations S and T representing the entity types.

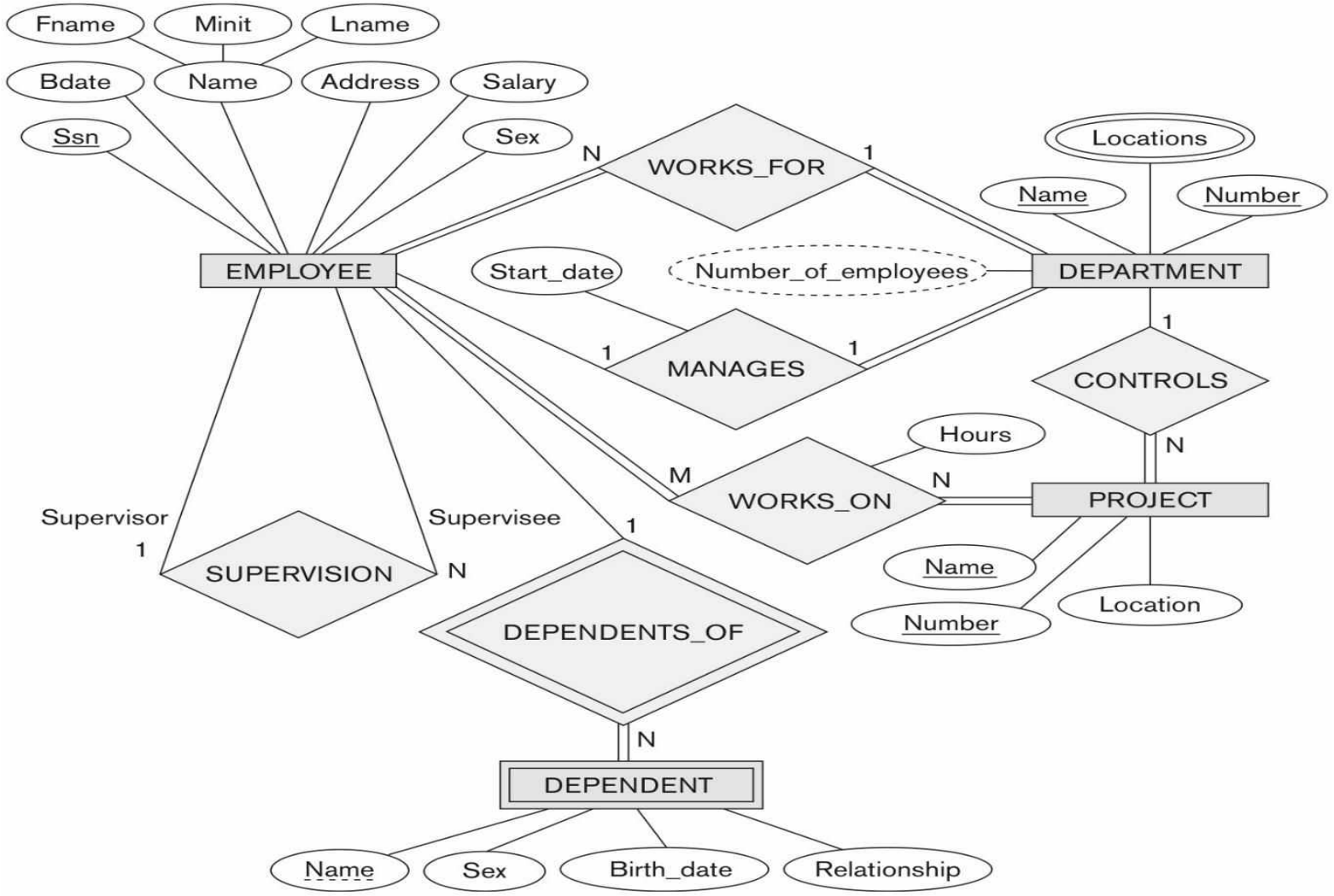
Step 3 : Mapping of Binary 1:1 Relation Types

Result



DATABASE MANAGEMENT SYSTEM

ER Conceptual schema diagram for Company Database



ER to Relational Mapping Algorithm (Continued)

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.

ER to Relational Mapping Algorithm (Continued)

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

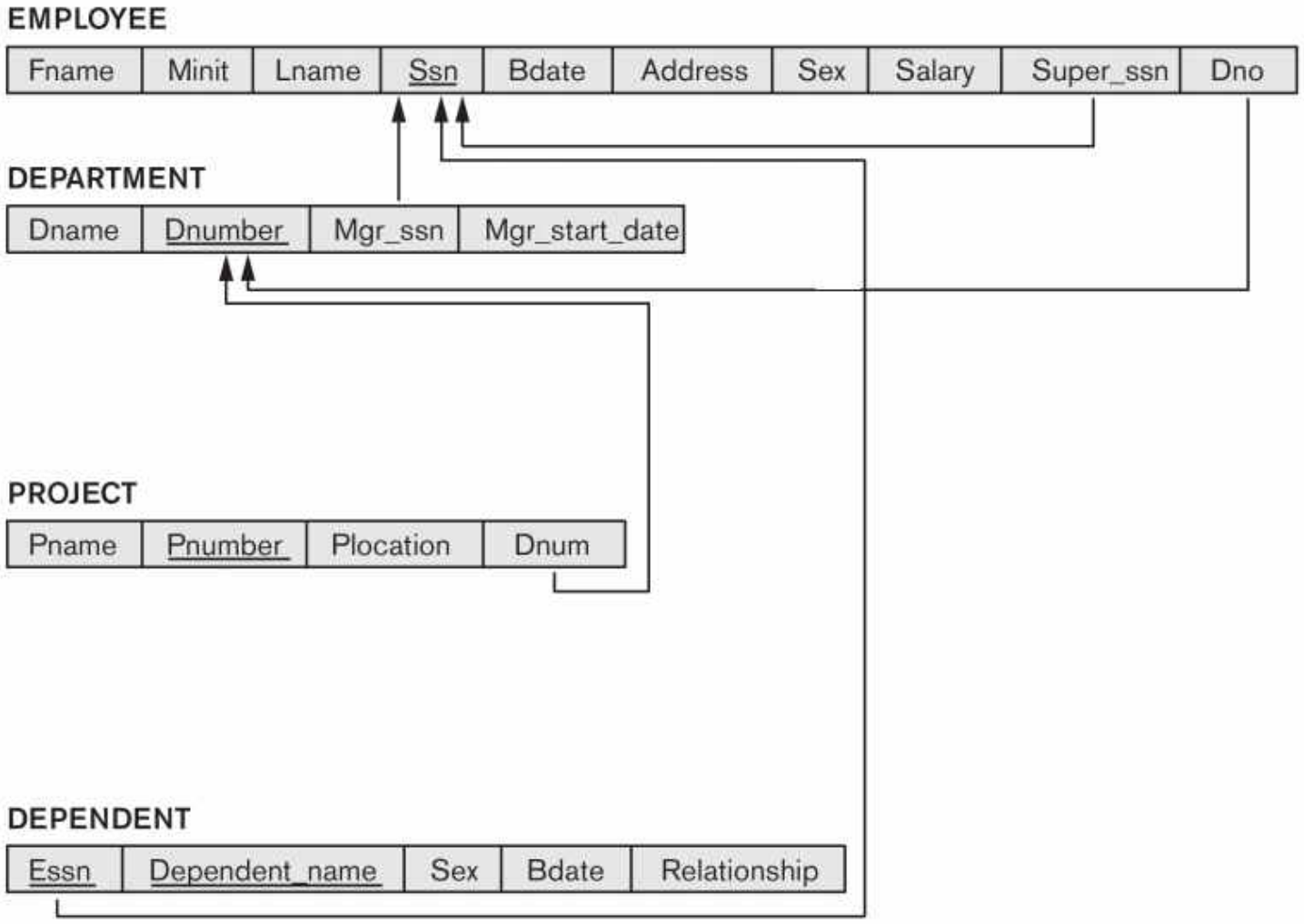
Example: 1:N relationship types WORKS_FOR, CONTROLS, and SUPERVISION in the figure.

- For WORKS_FOR we include the primary key DNUMBER of the DEPARTMENT relation as foreign key in the EMPLOYEE relation and call it DNO.

An alternative approach is to use a Relationship relation (cross referencing relation) – this is rarely done.

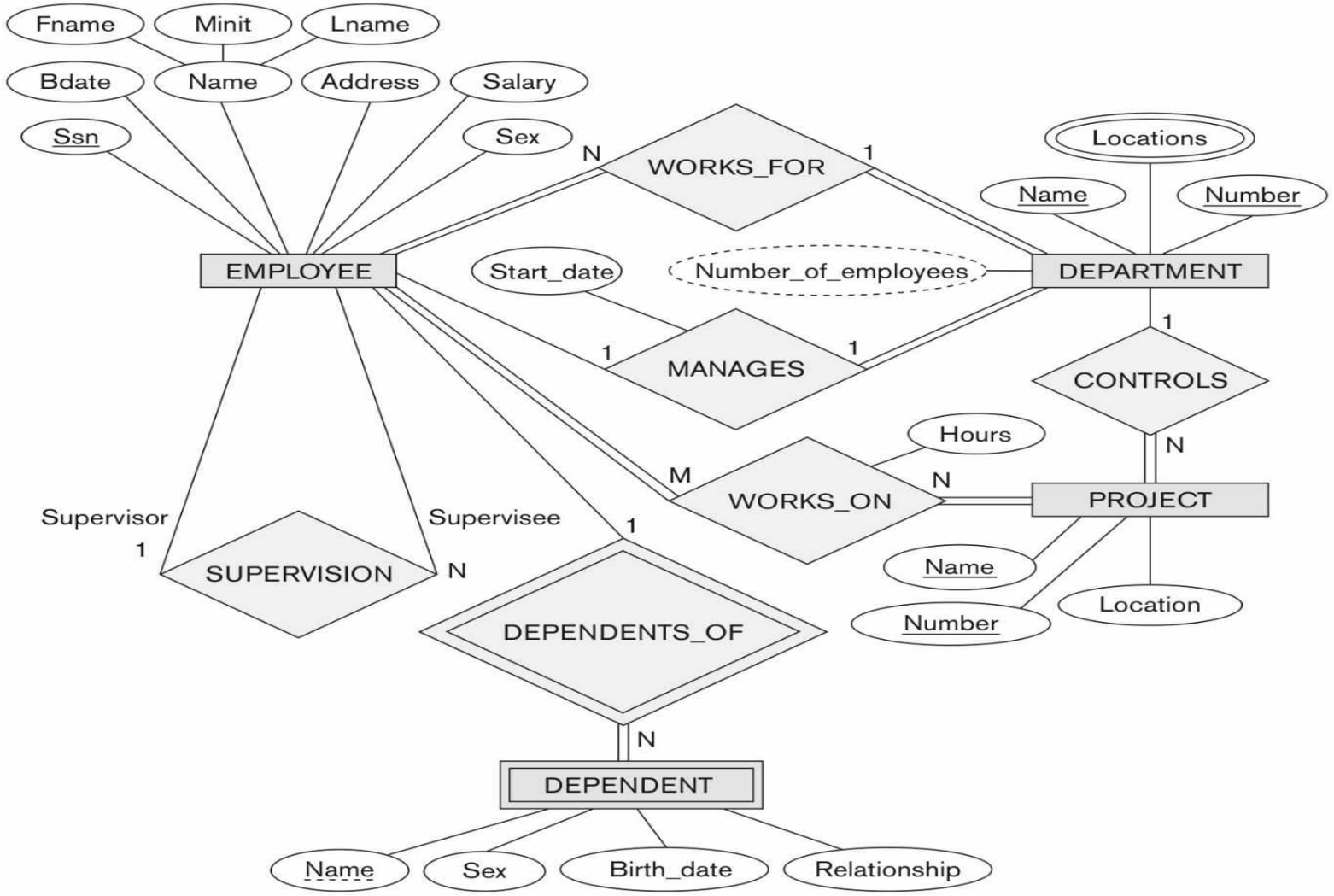
Step 4 : Mapping of Binary 1:N Relation Types

Result



DATABASE MANAGEMENT SYSTEM

ER Conceptual schema diagram for Company Database



ER to Relational Mapping Algorithm (Continued)

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. This is a *relationship relation.*
- 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.

ER to Relational Mapping Algorithm (Continued)

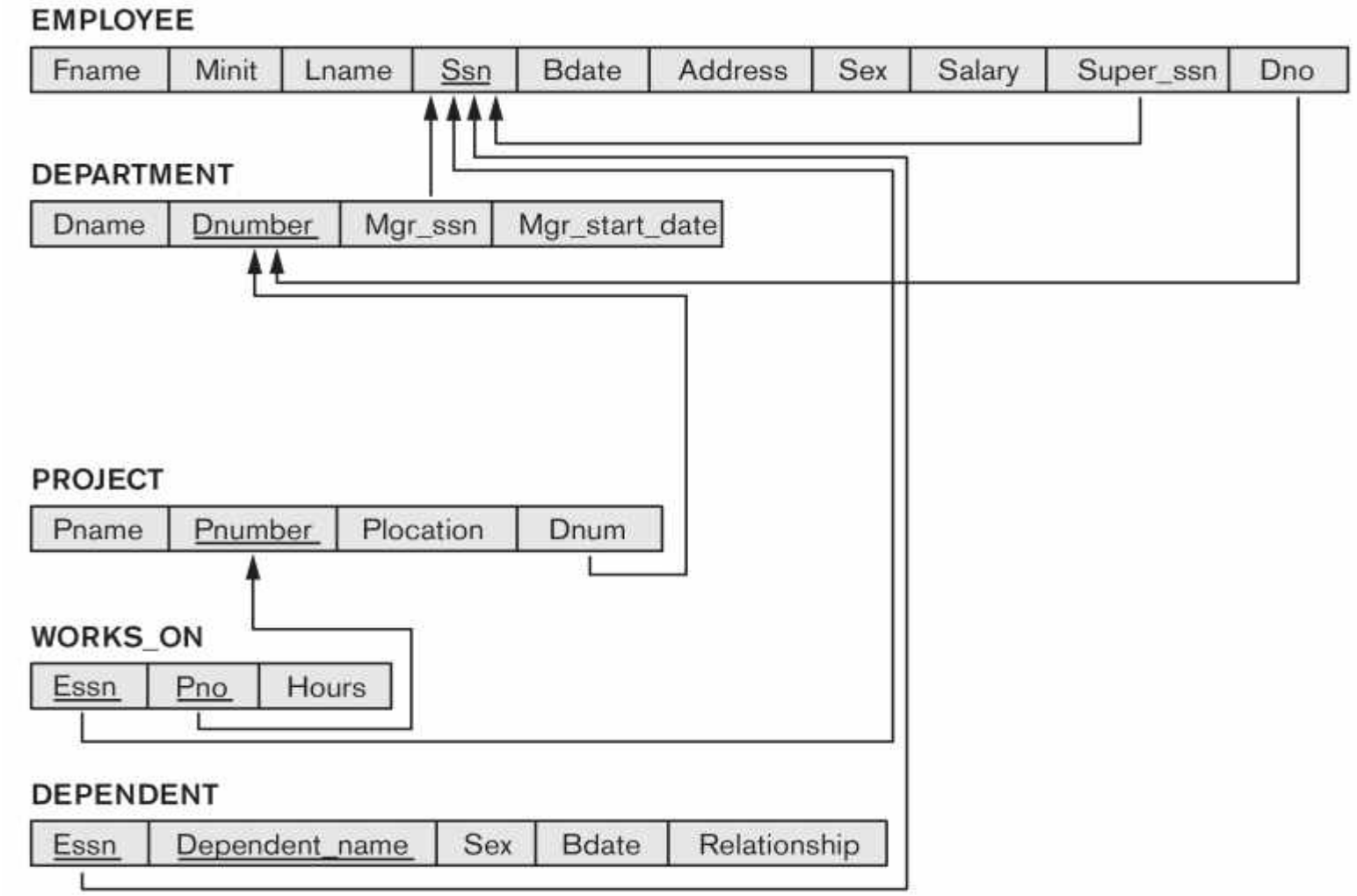
Step 5: Mapping of Binary M:N Relationship Types

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}.

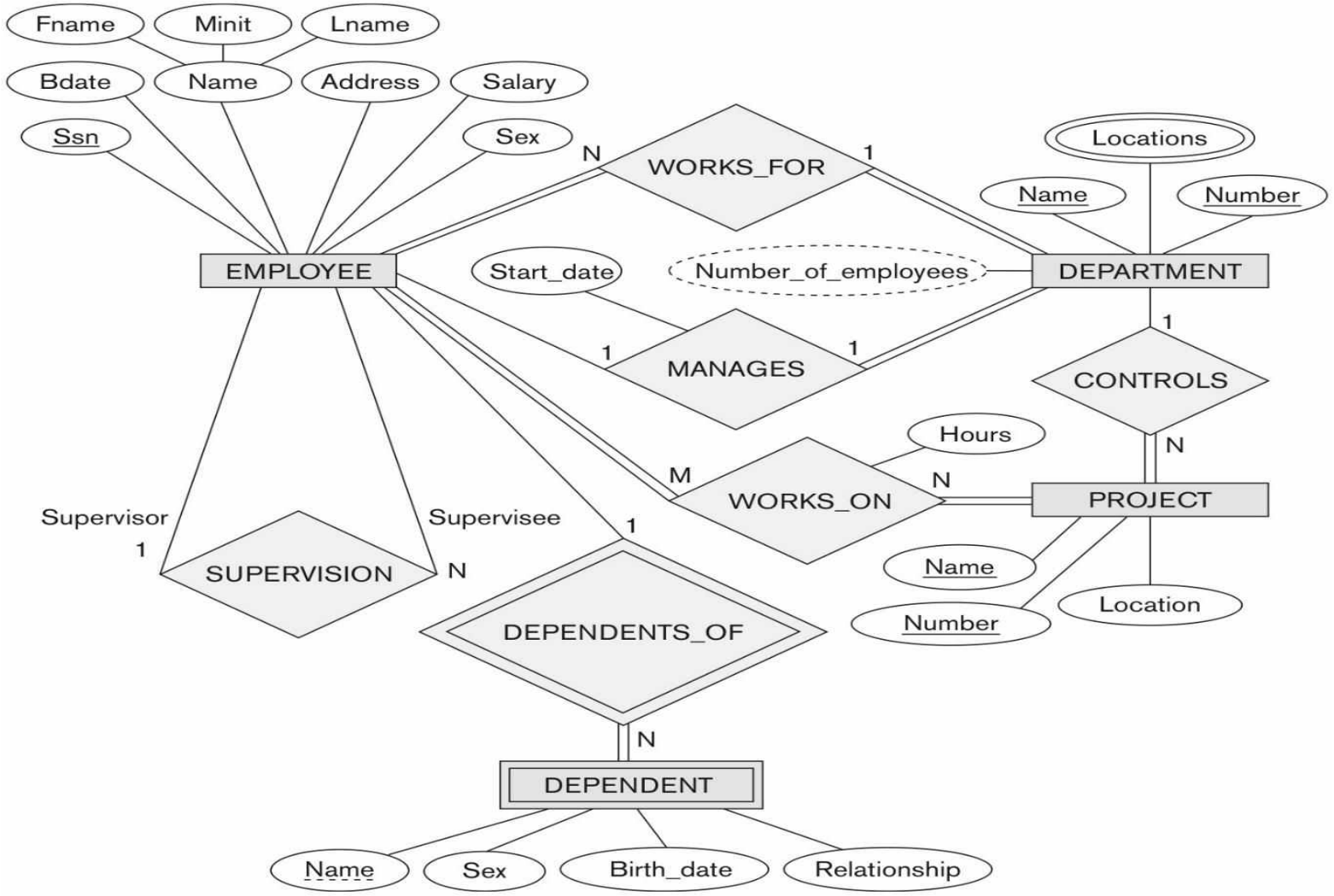
Step 5 : Mapping of Binary M:N Relation Types

Result



DATABASE MANAGEMENT SYSTEM

ER Conceptual schema diagram for Company Database



ER to Relational Mapping Algorithm (Continued)

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 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.

ER to Relational Mapping Algorithm (Continued)

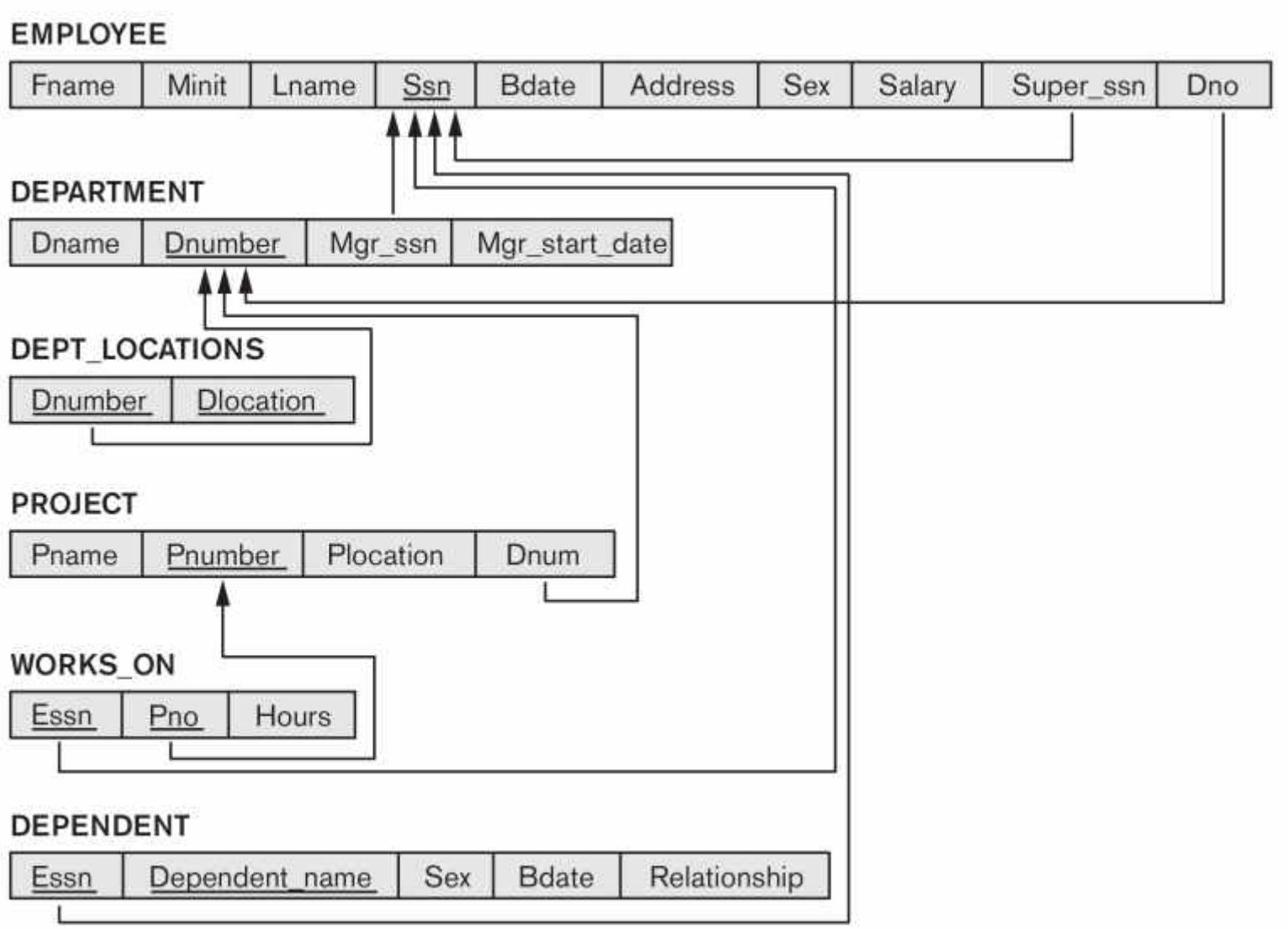
Step 6: Mapping of Multivalued attributes.

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 6 : Mapping of Multivalued Attributes

Result



ER to Relational Mapping Algorithm (Continued)

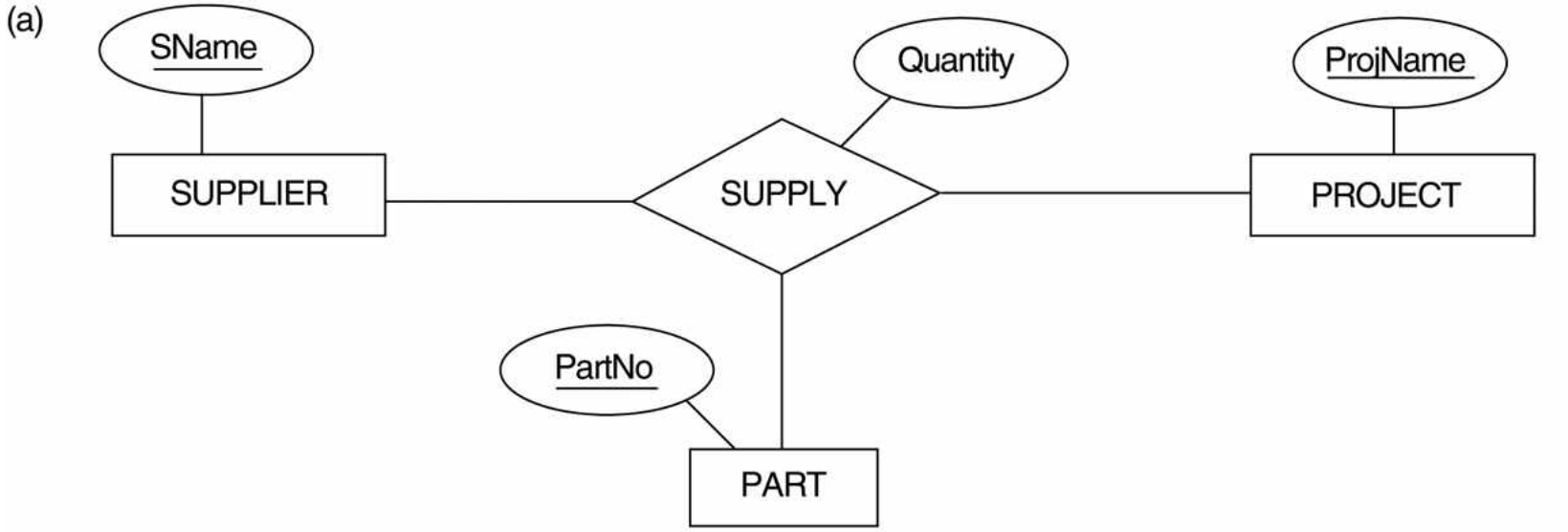
Step 7: Mapping of N-ary Relationship Types.

- 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.

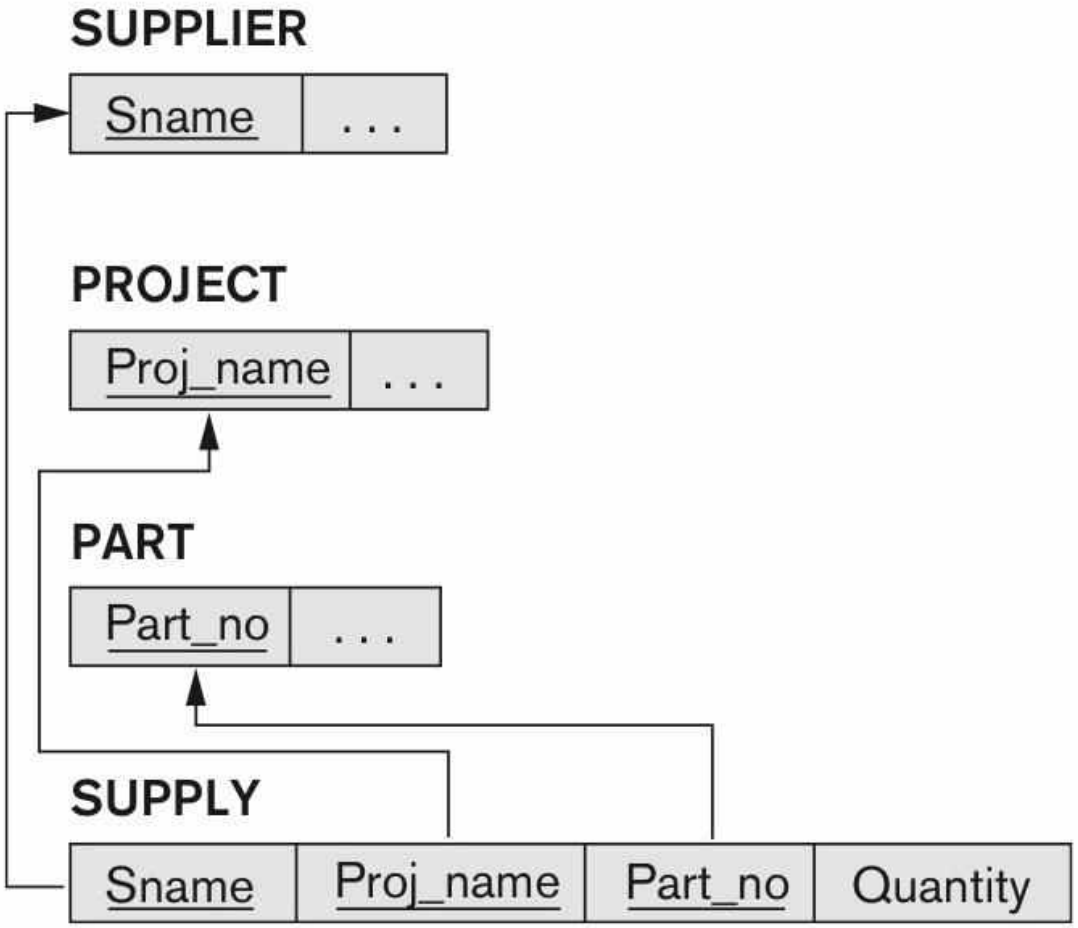
Example: The relationship type SUPPY in the ER on the next slide.

- 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, PARTNO, PROJNAME}

Ternary Relationship : Supply

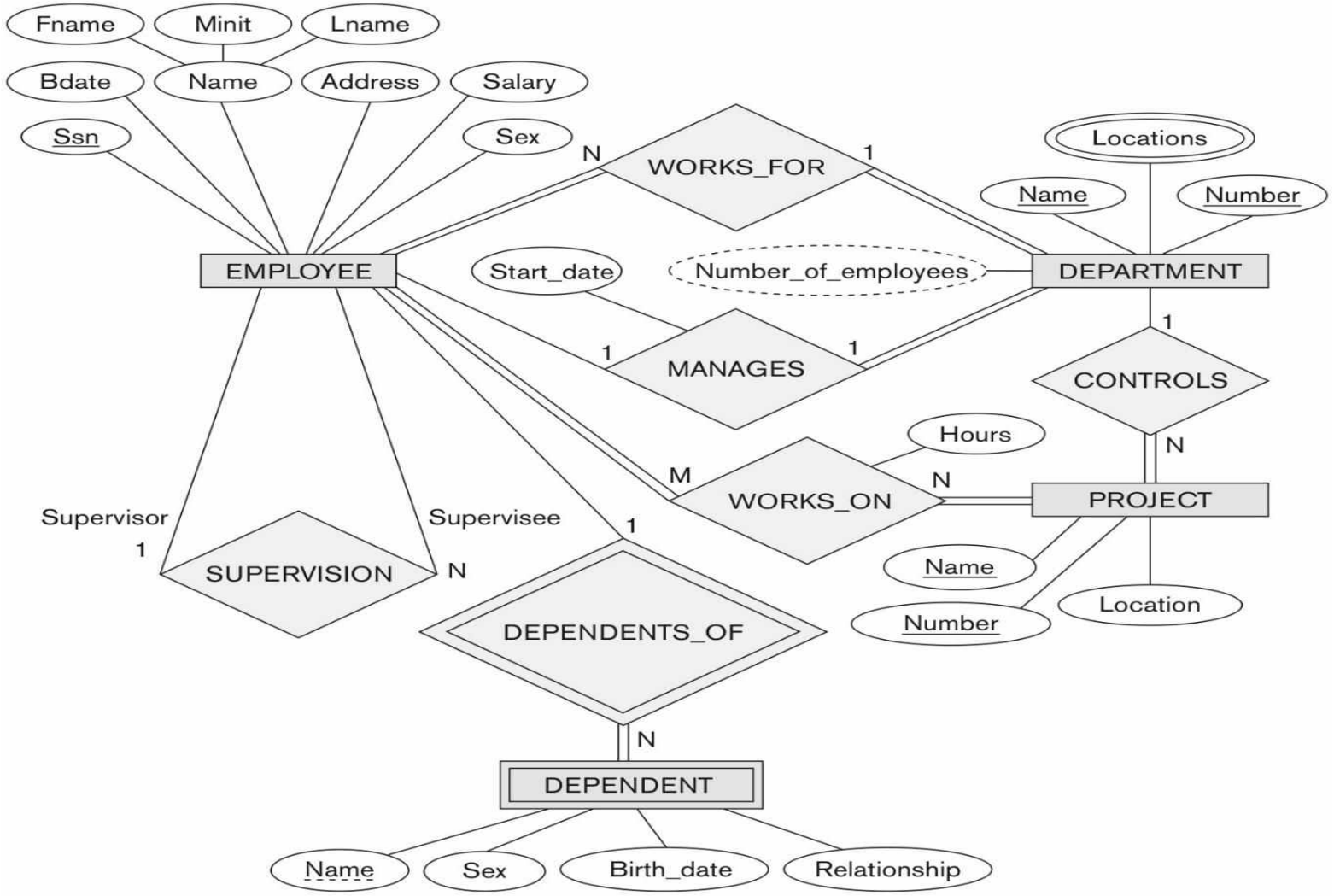


Mapping the n-ary relationship type Supply

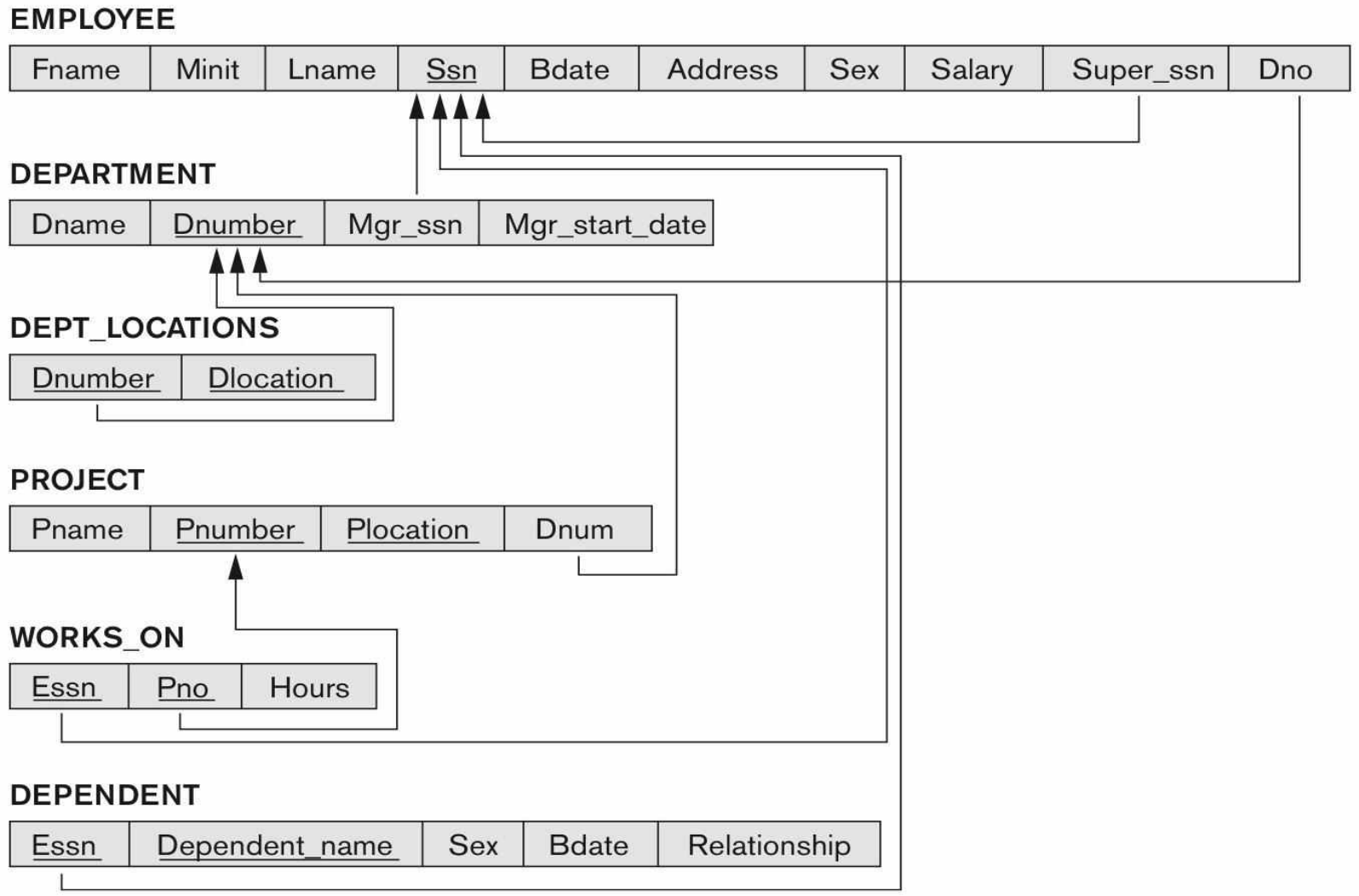


DATABASE MANAGEMENT SYSTEM

ER Conceptual schema diagram for Company Database



Result of mapping COMPANY ER schema into a relational database schema



Summary of Mapping constructs and constraints

Table 9.1 Correspondence between ER and Relational Models

ER MODEL	RELATIONAL MODEL
Entity type	*Entity* relation
1:1 or 1:N relationship type	Foreign key (or *relationship* relation)
M:N relationship type	*Relationship* relation and *two* foreign keys
n-ary relationship type	*Relationship* relation and *n* foreign keys
Simple attribute	Attribute
Composite attribute	Set of simple component attributes
Multivalued attribute	Relation and foreign key
Value set	Domain
Key attribute	Primary (or secondary) key

Summary

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.

THANK YOU

S Nagasundari
Department of Computer Science and Engineering

nagasundaris@pes.edu

DATABASE MANAGEMENT SYSTEM

S Nagasundari

Department of Computer Science and Engineering

DATABASE MANAGEMENT SYSTEM

Unary Relational Operations

Slides adapted from Author Slides of Fundamentals of Database Systems", Ramez Elamsri, Shamkant B Navathe, Pearson, 7th Edition, 2017.

S Nagasundari

Department of Computer Science and Engineering

DATABASE MANAGEMENT SYSTEM

Unit 2 : Relational Model

1. Relational Model Concepts
2. Relational Model Constraints and Relational Database Schemas
3. Update Operations, Transactions and dealing with constraint violations
4. Relational Database Design Using ER-to Relational Mapping
5. **Unary Relational Operations**
6. Set Theory Operations
7. Binary Relational Operations
8. Aggregate Functions and Grouping
9. Examples Queries in Relational Algebra

CHAPTER 8

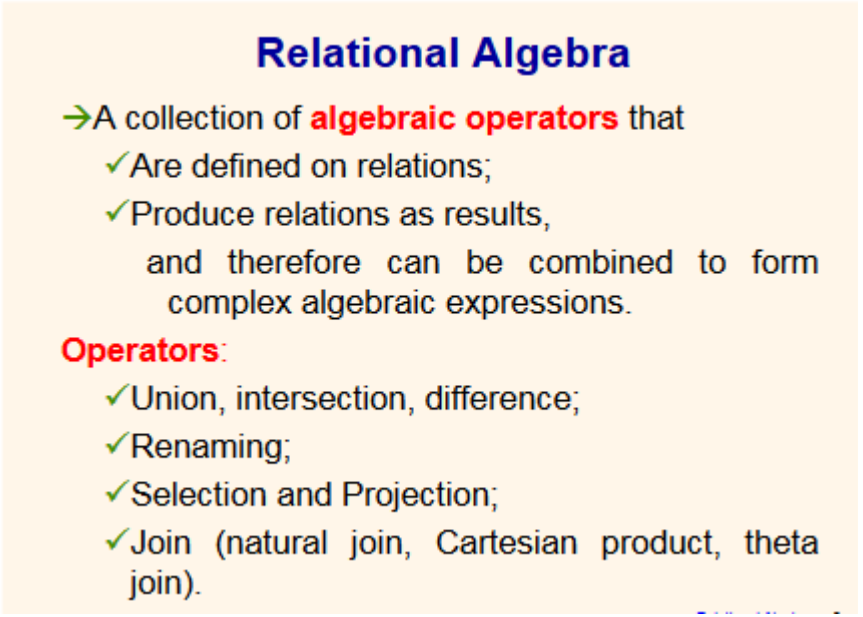
The Relational Algebra and The Relational Calculus

Relational Algebra

- **Unary Relational Operations**
- Relational Algebra Operations From Set Theory
- Binary Relational Operations
- Additional Relational Operations
- Examples of Queries in Relational Algebra

Relational Algebra Overview

- Relational algebra is the basic set of operations for the relational model
- These operations enable a user to specify **basic retrieval requests** (or **queries**)
- The result of an operation is a *new relation*, which may have been formed from one or more *input* relations
  - This property makes the algebra "closed" (all objects in relational algebra are relations)



Relational Algebra Overview(Continued)

- The **algebra operations** thus produce new relations
  - These can be further manipulated using operations of the same algebra
- A sequence of relational algebra operations forms a **relational algebra expression**
  - The result of a relational algebra expression is also a relation that represents the result of a database query (or retrieval request)

Brief History of Origins of Algebra

- Muhammad ibn Musa al-Khwarizmi (800-847 CE) – from Morocco wrote a book titled al-jabr about arithmetic of variables
  - Book was translated into Latin.
  - Its title (al-jabr) gave Algebra its name.

- Al-Khwarizmi called variables “shay”
  - “Shay” is Arabic for “thing”.
  - Spanish transliterated “shay” as “xay” (“x” was “sh” in Spain).
  - In time this word was abbreviated as x.

- Where does the word Algorithm come from?
  - Algorithm originates from “al-Khwarizmi"
  - Reference: PBS (http://www.pbs.org/empires/islam/innoalgebra.html)

Relational Algebra Overview

Relational Algebra consists of several groups of operations

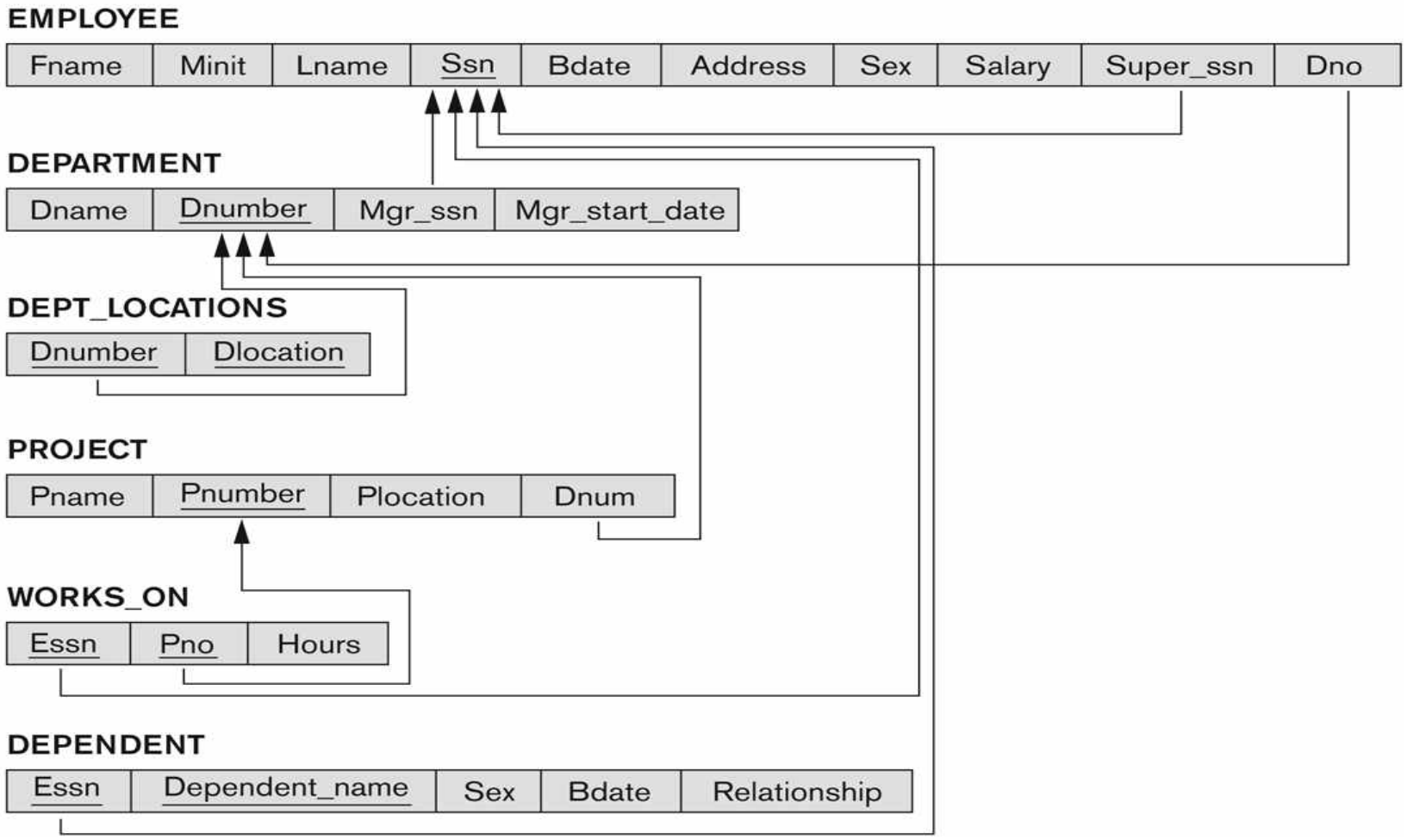
- Unary Relational Operations
  - SELECT (symbol: σ (sigma))
  - PROJECT (symbol: π (pi))
  - RENAME (symbol: ρ (rho))
- Relational Algebra Operations From Set Theory
  - UNION ($\cup$), INTERSECTION ($\cap$), DIFFERENCE (or MINUS, –)
  - CARTESIAN PRODUCT (**x**)
- Binary Relational Operations
  - JOIN (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

Figure 5.7
Referential integrity constraints displayed on the COMPANY relational database schema.



Unary Relational Operations : Select

- The SELECT operation (denoted by σ (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:

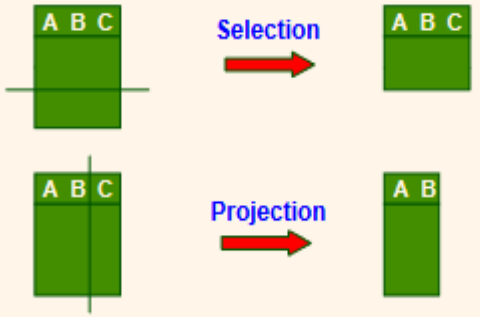
$$\sigma_{SALARY > 30,000} (EMPLOYEE)$$

Selection and Projection

→These are unary operators, in a sense orthogonal:

✓selection for "horizontal" decompositions;

✓projection for "vertical" decompositions.



Unary Relational Operations: SELECT

In general, the *select* operation is denoted by $\sigma_{\text{<selection condition>}}(R)$ where

- the symbol σ (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(Continued)

SELECT Operation Properties

- The SELECT operation $\sigma_{<\text{selection condition}>}(R)$ produces a relation S that has the same schema (same attributes) as R
- SELECT σ is commutative:
  - $\sigma_{<\text{condition1}>}(\sigma_{<\text{condition2}>}(R)) = \sigma_{<\text{condition2}>}(\sigma_{<\text{condition1}>}(R))$
- Because of commutativity property, a cascade (sequence) of SELECT operations may be applied in any order:
  - $\sigma_{<\text{cond1}>}(\sigma_{<\text{cond2}>}(\sigma_{<\text{cond3}>}(R)) = \sigma_{<\text{cond2}>}(\sigma_{<\text{cond3}>}(\sigma_{<\text{cond1}>}(R)))$

Citizens

Surname	FirstName	PlaceOfBirth	Residence
Smith	Mary	Rome	Milan
Black	Lucy	Rome	Rome
Verdi	Nico	Florence	Florence
Smith	Mark	Naples	Florence

$\sigma_{\text{PlaceOfBirth=Residence}}$ **(Citizens)**

Surname	FirstName	PlaceOfBirth	Residence
Black	Lucy	Rome	Rome
Verdi	Nico	Florence	Florence

Unary Relational Operations: SELECT(Continued)

SELECT Operation Properties

- A cascade of SELECT operations may be replaced by a single selection with a conjunction of all the conditions:
  - $\sigma_{<cond1>}(\sigma_{<cond2>}(\sigma_{<cond3>}(R)) = \sigma_{<cond1> \text{ AND } <cond2> \text{ AND } <cond3>}(R))$
- 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.

Employees

Surname	FirstName	Age	Salary
Smith	Mary	25	2000
Black	Lucy	40	3000
Verdi	Nico	36	4500
Smith	Mark	40	3900

$\sigma_{Age<30 \text{ v } Salary>4000}$ **(Employees)**

Surname	FirstName	Age	Salary
Smith	Mary	25	2000
Verdi	Nico	36	4500

The following query results refer to this database state

Figure 5.6
One possible database state for the COMPANY relational database schema.

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: PROJECT

- PROJECT Operation is denoted by π (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:

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

Employees

Surname	FirstName	Department	Head
Smith	Mary	Sales	De Rossi
Black	Lucy	Sales	De Rossi
Verdi	Mary	Personnel	Fox
Smith	Mark	Personnel	Fox

$\pi_{\text{Surname, FirstName}}$(Employees)

Surname	FirstName
Smith	Mary
Black	Lucy
Verdi	Mary
Smith	Mark

Unary Relational Operations: PROJECT

The general form of the *project* operation is:

$$\pi_{<\text{attribute list}>}(R)$$

- π (pi) is the symbol used to represent the *project* operation
- <attribute list> 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.

Employees

Surname	FirstName	Department	Head
Smith	Mary	Sales	De Rossi
Black	Lucy	Sales	De Rossi
Verdi	Mary	Personnel	Fox
Smith	Mark	Personnel	Fox

$\pi_{\text{Department, Head}}$ (Employees)

Department	Head
Sales	De Rossi
Personnel	Fox

Unary Relational Operations: PROJECT (contd.)

PES UNIVERSITY ONLINE

PROJECT Operation Properties

- The number of tuples in the result of projection $\pi_{<list>}(R)$ is always less or equal to the number of tuples in R
  - If the list of attributes includes a *key* of R, then the number of tuples in the result of PROJECT is *equal* to the number of tuples in R
- PROJECT is *not* commutative
  - $\pi_{<list1>}(\pi_{<list2>}(R)) = \pi_{<list1>}(R)$ as long as <list2> contains the attributes in <list1>

Examples of applying SELECT and PROJECT operations

Figure 8.1
Results of SELECT and PROJECT operations. (a) $\sigma_{(Dno=4\ \textbf{AND}\ Salary>25000)\ OR\ (Dno=5\ \textbf{AND}\ Salary>30000)}$ (EMPLOYEE). (b) $\pi_{Lname,\ Fname,\ Salary}$(EMPLOYEE). (c) $\pi_{Sex,\ Salary}$(EMPLOYEE).

(a)

Fname	Minit	Lname	Ssn	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

(b)

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

(c)

Sex	Salary
M	30000
M	40000
F	25000
F	43000
M	38000
M	25000
M	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 Vs 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:
  - $\pi_{FNAME, LNAME, SALARY}(\sigma_{DNO=5}(EMPLOYEE))$
- OR We can explicitly show the *sequence of operations*, giving a name to each intermediate relation:
  - DEP5_EMPS $\leftarrow \sigma_{DNO=5}(EMPLOYEE)$
  - RESULT $\leftarrow \pi_{FNAME, LNAME, SALARY}$ (DEP5_EMPS)

Unary Relational Operations: Rename

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

Renaming

→This is a unary operator which changes attribute names for a relation without changing any values.

→Renaming removes the limitations associated with set operators.

→Notation: $\rho_{OldName \rightarrow NewName}(r)$

→For example, $\rho_{Father \rightarrow Parent}(Paternity)$

→If there are two or more attributes involved in a renaming operation, then ordering is meaningful:

e.g., $\rho_{Branch,Salary \rightarrow Location,Pay}(Employees)$

Unary Relational Operations: RENAME(continued)

The general RENAME operation ρ can be expressed by any of the following forms:

- $\rho_{S\ (B1,\ B2,\ \ldots,\ Bn\ )}(R)$ changes both:
  - the relation name to S, *and*
  - the column (attribute) names to B1, B1, …..Bn
- $\rho_S(R)$ changes:
  - the *relation name* only to S
- $\rho_{(B1,\ B2,\ \ldots,\ Bn\ )}(R)$ changes:
  - the *column (attribute) names* only to B1, B1, …..Bn

Example of Renaming

Paternity

Father	Child
Adam	Cain
Adam	Abel
Abraham	Isaac
Abraham	Ishmael

$\rho_{Father \rightarrow Parent}(Paternity)$

Parent	Child
Adam	Cain
Adam	Abel
Abraham	Isaac
Abraham	Ishmael

- The textbook allows positions rather than attribute names, e.g., `1 → Parent`
- Textbook also allows renaming of the relation itself,e.g.,`Paternity,1→ Parenthood,Parent`

Unary Relational Operations: RENAME(continued)

For convenience, we also use a *shorthand* for renaming attributes in an intermediate relation:

If we write:

RESULT $\leftarrow \pi_{\text{FNAME, LNAME, SALARY}}$ (DEP5_EMPS)

RESULT will have the *same attribute names* as DEP5_EMPS (same attributes as EMPLOYEE)

If we write:

RESULT (F, M, L, S, B, A, SX, SAL, SU, DNO)

$\leftarrow \rho_{\text{RESULT (F.M.L.S.B,A,SX,SAL,SU, DNO)}}$(DEP5_EMPS)

The 10 attributes of DEP5_EMPS are *renamed* to

F, M, L, S, B, A, SX, SAL, SU, DNO, respectively

Note: the $\leftarrow$ symbol is an assignment operator

Example of applying multiple operations and RENAME

(a)

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

(b)

TEMP

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
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

R

First_name	Last_name	Salary
John	Smith	30000
Franklin	Wong	40000
Ramesh	Narayan	38000
Joyce	English	25000

Figure 8.2

Results of a sequence of operations. (a) $\pi_{Fname, Lname, Salary}(\sigma_{Dno=5}(EMPLOYEE))$.
(b) Using intermediate relations and renaming of attributes.

THANK YOU

S Nagasundari
Department of Computer Science and Engineering

nagasundaris@pes.edu

DATABASE MANAGEMENT SYSTEM

S Nagasundari

Department of Computer Science and Engineering

DATABASE MANAGEMENT SYSTEM

Set Theory Operations

Slides adapted from Author Slides of Fundamentals of Database Systems", Ramez Elamsri, Shamkant B Navathe, Pearson, 7th Edition, 2017.

S Nagasundari

Department of Computer Science and Engineering

DATABASE MANAGEMENT SYSTEM

Unit 2 : Relational Model

1. Relational Model Concepts
2. Relational Model Constraints and Relational Database Schemas
3. Update Operations, Transactions and dealing with constraint violations
4. Relational Database Design Using ER-to Relational Mapping
5. Unary Relational Operations
6. **Set Theory Operations**
7. Binary Relational Operations
8. Aggregate Functions and Grouping
9. Examples Queries in Relational Algebra

CHAPTER 8

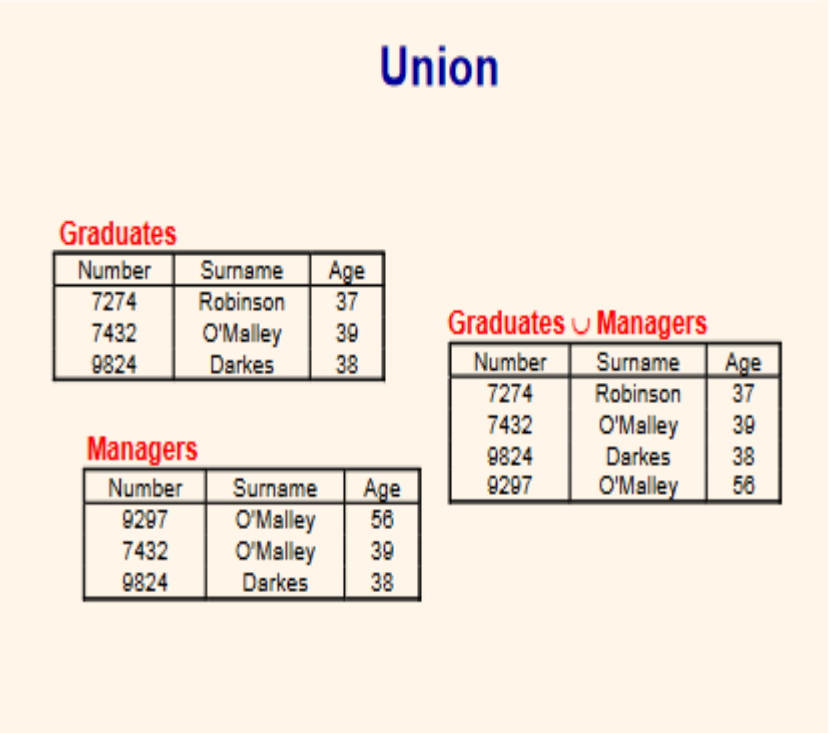
The Relational Algebra and The Relational Calculus

Relational Algebra

- Unary Relational Operations
- **Relational Algebra Operations From Set Theory**
- Binary Relational Operations
- Additional Relational Operations
- Examples of Queries in Relational Algebra

Relational Algebra Operations from UNION Operation

- Binary operation, denoted by ∪
- The result of R ∪ 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)



Source: http://www.cs.toronto.edu/~faye/343/f07/lectures/wk3/03_RAlgebra.pdf

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:

$$DEP5\_EMPS \leftarrow \sigma_{DNO=5}(EMPLOYEE)$$

$$RESULT1 \leftarrow \pi_{SSN}(DEP5\_EMPS)$$

$$RESULT2(SSN) \leftarrow \pi_{SUPERSSN}(DEP5\_EMPS)$$

$$RESULT \leftarrow RESULT1 \cup RESULT2$$

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

Result of the UNION operation

RESULT ← RESULT1 ∪ RESULT2.

RESULT1

Ssn
123456789
333445555
666884444
453453453

RESULT2

Ssn
333445555
888665555

RESULT

Ssn
123456789
333445555
666884444
453453453
888665555

Relational Algebra Operations from Set Theory

- Type Compatibility of operands is required for the binary set operation UNION ∪, (also for INTERSECTION ∩, and SET DIFFERENCE –)
- R1(A1, A2, ..., An) and R2(B1, B2, ..., Bn) are type compatible if:
  - they have the same number of attributes, and
  - the domains of corresponding attributes are type compatible (i.e. dom(Ai)=dom(Bi) for i=1, 2, ..., n).
- The resulting relation for R1∪R2

(also for R1∩R2, or R1–R2, has the same attribute names as the *first* operand relation R1 (by convention)

Relational Algebra Operations from Set Theory : Intersection

INTERSECTION is denoted by ∩

- The result of the operation R ∩ 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"

Intersection

Graduates

Number	Surname	Age
7274	Robinson	37
7432	O'Malley	39
9824	Darkes	38

Managers

Number	Surname	Age
9297	O'Malley	56
7432	O'Malley	39
9824	Darkes	38

Graduates ∩ Managers

Number	Surname	Age
7432	O'Malley	39
9824	Darkes	38

Relational Algebra Operations from Set Theory : SET DIFFERENCE(continued)

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”

Difference

Graduates

Number	Surname	Age
7274	Robinson	37
7432	O'Malley	39
9824	Darkes	38

Managers

Number	Surname	Age
9297	O'Malley	56
7432	O'Malley	39
9824	Darkes	38

Graduates - Managers

Number	Surname	Age
7274	Robinson	37

Example to illustrate the result of UNION, INTERSECT and DIFFERENCE

Figure 8.4
The set operations UNION, INTERSECTION, and MINUS. (a) Two union-compatible relations. (b) STUDENT ∪ INSTRUCTOR. (c) STUDENT ∩ INSTRUCTOR. (d) STUDENT − INSTRUCTOR. (e) INSTRUCTOR − 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

(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

CARTESIAN (or CROSS) PRODUCT Operation

- This operation is used to combine tuples from two relations in a combinatorial fashion.
- Denoted by R(A1, A2, . . ., An) x S(B1, B2, . . ., Bm)
- Result is a relation Q with degree n + m attributes:
  - Q(A1, A2, . . ., An, B1, B2, . . ., Bm), 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 x 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(contd.)

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

Relational Algebra Operations from Set Theory : Cartesian Product(contd.)

- To keep only combinations where the DEPENDENT is related to the EMPLOYEE, we add a SELECT operation as follows
- Example (meaningful):
  - FEMALE_EMPS $\leftarrow \sigma_{SEX='F'}$(EMPLOYEE)
  - EMPNAMES $\leftarrow \pi_{FNAME, LNAME, SSN}$ (FEMALE_EMPS)
  - EMP_DEPENDENTS $\leftarrow$ EMPNAMES x DEPENDENT
  - ACTUAL_DEPS $\leftarrow \sigma_{SSN=ESSN}$(EMP_DEPENDENTS)
  - RESULT $\leftarrow \pi_{FNAME, LNAME, DEPENDENT\_NAME}$ (ACTUAL_DEPS)
- RESULT will now contain the name of female employees and their dependents

The Cartesian Product (Cross Product)

FEMALE_EMPS

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
Alicia	J	Zelaya	999887777	1968-07-19	3321Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291Berry, Bellaire, TX	F	43000	888665555	4
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5

EMPNAMES

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

continued on next slide

The Cartesian Product (Cross Product)

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	...

continued on next slide

The Cartesian Product (Cross Product)

ACTUAL_DEPENDENTS

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

RESULT

Fname	Lname	Dependent_name
Jennifer	Wallace	Abner

The Cartesian Product (Cross Product) Example

Relation: Singers

Singer-id	Singer-name	Address	Age
0126	Helen Drummond	1 Thorley Street	42
0243	Katerina Christou	12 High Road	37
0247	Desmond Venables	27 Long Lane	55
0259	Anne Freeman	5 Tower Hill	40
0594	Alphonse Trieste	20 Longchamps	34
0628	Tamanna Patel	9 Crown Hill	23
0855	Swee Hor Tan	4 Long Lane	54
0876	Panos Constantinou	32 Mallet Road	49

Relation: Roles

Role-id	Role-name
0101	Figaro
0175	Mimi

In order to create a new relation which pairs each singer with each role, we need to use the relational operation Cartesian product.
Relational Algebra operation: Singers Cartesian product Roles giving Singers-Roles

Source: https://www.cs.uct.ac.za/mit_notes/database/htmls/chp02.html#cartesian-product

The Cartesian Product (Cross Product) Example

New relation: Singers-Roles

Singer-id	Name	Address	Age	Role-id	Role-name
0126	Helen Drummond	1 Thorley Street	42	0101	Figaro
0126	Helen Drummond	1 Thorley Street	42	0175	Mimi
0243	Katerina Christou	12 High Road	37	0101	Figaro
0243	Katerina Christou	12 High Road	37	0175	Mimi
0247	Desmond Venables	27 Long Lane	55	0101	Figaro
0247	Desmond Venables	27 Long Lane	55	0175	Mimi
0259	Anne Freeman	5 Tower Hill	40	0101	Figaro
0259	Anne Freeman	5 Tower Hill	40	0175	Mimi
0594	Alphonse Trieste	20 Longchamps	34	0101	Figaro
0594	Alphonse Trieste	20 Longchamps	34	0175	Mimi
0628	Tamanna Patel	9 Crown Hill	23	0101	Figaro
0628	Tamanna Patel	9 Crown Hill	23	0175	Mimi
0855	Swee Hor Tan	4 Long Lane	54	0101	Figaro
0855	Swee Hor Tan	4 Long Lane	54	0175	Mimi
0876	Panos Constantinou	32 Mallet Road	49	0101	Figaro
0876	Panos Constantinou	32 Mallet Road	49	0175	Mimi

- The result of Singers Cartesian product Roles gives a new relation Singers-Roles, showing each tuple from one relation with each tuple of the other, producing the tuples in the new relation.
- Each singer is associated with all roles; this produces a relation with 16 tuples, as there were 8 tuples in the relation Singers, and 2 tuples in the relation Roles.

Some Examples

- What will be the first Column Name after Union of Paternity and Maternity Relation?
- Is It Meaningful?

Paternity

Father	Child
Adam	Cain
Adam	Abel
Abraham	Isaac
Abraham	Ishmael

Maternity

Mother	Child
Eve	Cain
Eve	Seth
Sarah	Isaac
Hagar	Ishmael

Paternity ∪ Maternity ???

→The problem: **Father** and **Mother** are different names, but both represent a parent.

Renaming and Union

Paternity

Father	Child
Adam	Cain
Adam	Abel
Abraham	Isaac
Abraham	Ishmael

Maternity

Mother	Child
Eve	Cain
Eve	Seth
Sarah	Isaac
Hagar	Ishmael

$\rho_{Father \to Parent}(Paternity) \cup \rho_{Mother \to Parent}(Maternity)$

Parent	Child
Adam	Cain
Adam	Abel
Abraham	Isaac
Abraham	Ishmael
Eve	Cain
Eve	Seth
Sarah	Isaac
Hagar	Ishmael

Some Examples

Renaming and Union, with Several Attributes

Employees

Surname	Branch	Salary
Patterson	Rome	45
Trumble	London	53

Staff

Surname	Factory	Wages
Patterson	Rome	45
Trumble	London	53

$\rho_{Branch, Salary \rightarrow Location, Pay}(Employees) \cup \rho_{Factory, Wages \rightarrow Location, Pay}(Staff)$

Surname	Location	Pay
Patterson	Rome	45
Trumble	London	53
Cooke	Chicago	33
Bush	Monza	32

Some Examples

Question:

Consider the following relational database schema consisting of the four relation schemas:

passenger (pid, pname, pgender, pcity)

agency (aid, aname, acity)

flight (fid, fdate, time, src, dest)

booking (pid, aid, fid, fdate)

Answer the following questions using relational algebra queries;

Some Examples

a) Get the complete details of all flights to New Delhi.

$$\sigma_{destination = \text{“New Delhi”}}(\text{flight})$$

b) Get the details about all flights from Chennai to New Delhi.

$$\sigma_{src = \text{“Chennai”} \wedge dest = \text{“New Delhi”}}(\text{flight})$$

Some Examples

Get the details of flights that are scheduled on both dates 01/12/2020 and 02/12/2020 at 16:00 hours

$$(\sigma_{fdate = 01/12/2020 \wedge time = 16:00}(\text{flight})) \cap (\sigma_{fdate = 02/12/2020 \wedge time = 16:00}(\text{flight}))$$

Get the details of flights that are scheduled on either of the dates 01/12/2020 or 02/12/2020 or both at 16:00 hours.

$$(\sigma_{fdate = 01/12/2020 \wedge time = 16:00}(\text{flight})) \cup (\sigma_{fdate = 02/12/2020 \wedge time = 16:00}(\text{flight}))$$

THANK YOU

S Nagasundari

Department of Computer Science and Engineering

nagasundaris@pes.edu

DATABASE MANAGEMENT SYSTEM

S Nagasundari

Department of Computer Science and Engineering

DATABASE MANAGEMENT SYSTEM

Binary Relational Operations

Slides adapted from Author Slides of Fundamentals of Database Systems", Ramez Elamsri, Shamkant B Navathe, Pearson, 7th Edition, 2017.

S Nagasundari

Department of Computer Science and Engineering

DATABASE MANAGEMENT SYSTEM

Unit 2 : Relational Model

1. Relational Model Concepts
2. Relational Model Constraints and Relational Database Schemas
3. Update Operations, Transactions and dealing with constraint violations
4. Relational Database Design Using ER-to Relational Mapping
5. Unary Relational Operations
6. Set Theory Operations
7. Binary Relational Operations
8. Aggregate Functions and Grouping
9. Examples Queries in Relational Algebra

PES UNIVERSITY ONLINE

CHAPTER 8

The Relational Algebra and The Relational Calculus

PES UNIVERSITY ONLINE

Relational Algebra

- Unary Relational Operations
- Relational Algebra Operations From Set Theory
- **Binary Relational Operations**
- Additional Relational Operations
- Examples of Queries in Relational Algebra

Binary Relational Operations : JOIN

JOIN Operation (denoted by ⋈)

- The sequence of CARTESIAN PRODECT 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(A1, A2, . . ., An) and S(B1, B2, . . ., Bm) is:

  $R \bowtie_{\text{<join condition>}} S$

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

Binary Relational Operations : JOIN(contd.)

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.
- DEPT_MGR $\leftarrow$ DEPARTMENT $\bowtie_{MGRSSN=SSN}$ 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

Result of the JOIN operation

$$DEPT\_MGR \leftarrow DEPARTMENT \bowtie_{Mgr\_ssn=Ssn} EMPLOYEE$$

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	...

$$RESULT \leftarrow \pi_{Dname,\ Lname,\ Fname}(DEPT\_MGR)$$

JOIN operation can be replaced by (Recap)

- The JOIN operation can be specified as a CARTESIAN PRODUCT operation followed by a SELECT operation.

- Suppose if I want to retrieve the dependents of the Employees in EMPNAMES

$$EMP\_DEPENDENTS \leftarrow EMPNAMES \times DEPENDENT$$
$$ACTUAL\_DEPENDENTS \leftarrow \sigma_{Ssn=Essn}(EMP\_DEPENDENTS)$$

or

$$ACTUAL\_DEPENDENTS \leftarrow EMPNAMES \bowtie_{Ssn=Essn} DEPENDENT$$

DATABASE MANAGEMENT SYSTEM

JOIN operation can be replaced by (Recap)

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	...

EMPNAMES

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

Some properties of JOIN

Consider the following JOIN operation:

- $R(A1, A2, \ldots, An) \underset{R.Ai=S.Bj}{\bowtie} S(B1, B2, \ldots, Bm)$
- Result is a relation Q with degree n + m attributes:
  - Q(A1, A2, . . ., An, B1, B2, . . ., Bm), 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[Ai]=s[Bj]
- 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:

$$R \underset{theta}{\bowtie} S$$

- The join condition is called *theta*
- *Theta* can be any general boolean expression on the attributes of R and S; for example:
  - R.Ai<S.Bj AND (R.Ak=S.Bl OR R.Ap<S.Bq)
- Most join conditions involve one or more equality conditions “AND”ed together; for example:
  - R.Ai=S.Bj AND R.Ak=S.Bl AND R.Ap=S.Bq

Some properties of JOIN

A general join condition is of the form

<condition> AND <condition> AND … AND <condition>

where each <condition> is of the form *Ai* θ *Bj*,

- *Ai* is an attribute of *R*,
- *Bj* is an attribute of *S*,
- *Ai* and *Bj* have the same domain, and
- θ (theta) is one of the comparison operators {=, <, ≤, >, ≥, ≠}.

- Tuples whose join attributes are NULL or for which the join condition is FALSE *do not* appear in the result.
- In that sense, the JOIN operation does *not* necessarily preserve all the information in the participating relations, because tuples that do not get combined with matching ones in the other relation do not appear in the result.

DATABASE MANAGEMENT SYSTEM

Variations of JOINS

- EQUIJOIN
- NATURAL JOIN

Binary Relational Operations: EQUIJOIN

EQUIJOIN Operation

- 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.

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

Binary Relational Operations: Natural Join Operation

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.

COMPANY Relational Database Schema

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.

Example of Natural Join Operation

Suppose we want to retrieve the DEPARTMENT details that controls the project.

In the following example,

1. Rename the Dnumber attribute of DEPARTMENT to Dnum, so that it has the same name as the Dnum attribute in PROJECT

$$DEPT \leftarrow \rho_{(Dname,\ Dnum,\ Mgr\_ssn,\ Mgr\_start\_date)}(DEPARTMENT)$$

2. Apply NATURAL JOIN

$$PROJ\_DEPT \leftarrow PROJECT * DEPT$$

Example of Natural Join Operation

- If the attributes on which the natural join is specified already *have the same names in both relations*, renaming is unnecessary

- Suppose we want to retrieve the DEPARTMENT details and its location.

$$DEPT\_LOCS \leftarrow DEPARTMENT * DEPT\_LOCATIONS$$

Example of Natural Join Operation

(a)

PROJ_DEPT

Pname	Pnumber	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

Figure 8.7
Results of two natural join operations. (a) proj_dept ← project • dept.
(b) dept_locs ← department • dept_locations.

Binary Relational Operations: Natural Join Operation (contd.)

Example: To apply a natural join on the DNUMBER attributes of DEPARTMENT and DEPT_LOCATIONS, it is sufficient to write:

DEPT_LOCS ← DEPARTMENT * DEPT_LOCATIONS

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

DEPARTMENT.DNUMBER=DEPT_LOCATIONS.DNUMBER

Another example: Q ← 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 AND R.D.S.D

Result keeps only one attribute of each such pair:

Q(A,B,C,D,E)

Join Operation

- Notice that if no combination of tuples satisfies the join condition, the result of a JOIN is an empty relation with zero tuples.
- In general, if *R* has n_R tuples and *S* has n_S tuples, the result of a JOIN operation *R* <join condition> *S* will have between zero and $n_R * n_S$ tuples.
- The expected size of the join result divided by the maximum size $n_R * n_S$ leads to a ratio called **join selectivity**, which is a property of each join condition.
- If there is no join condition, all combinations of tuples qualify and the JOIN degenerates into a CARTESIAN PRODUCT, also called CROSS PRODUCT or CROSS JOIN

Join Operation

- The NATURAL JOIN or EQUIJOIN operation can also be specified among multiple tables, leading to an *n-way join*.
- For example, consider the following three-way join:

$$((\text{PROJECT} \bowtie_{\text{Dnum=Dnumber}} \text{DEPARTMENT}) \bowtie_{\text{Mgr\_ssn=Ssn}} \text{EMPLOYEE})$$

Complete Set of Relational Operations

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

Binary Relational Operations: Division Operation

DIVISION Operation

- The division operation is applied to two relations
- $R(Z) \div S(X)$, where X subset Z. Let $Y = Z - X$ (and hence $Z = X \cup Y$); that is, let Y be the set of attributes of R that are not attributes of S.
- The result of DIVISION is a relation T(Y) that includes a tuple t if tuples t_R appear in R with $t_R[Y] = t$, and with
  - $t_R[X] = t_s$ *for every tuple* t_s in S.
- 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.

COMPANY Relational Database Schema

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.

Example of Division

Say for example
*Retrieve the names of employees who work on **all** the projects that 'John Smith' works on*.

To express this query using the DIVISION operation, proceed as follows.

1. Retrieve the list of project numbers that 'John Smith' works on in the intermediate relation SMITH_PNOS:

$$SMITH \leftarrow \sigma_{Fname='John' \text{ AND } Lname='Smith'}(EMPLOYEE)$$
$$SMITH\_PNOS \leftarrow \pi_{Pno}(WORKS\_ON \bowtie_{Essn=Ssn} SMITH)$$

Example of Division

Say for example
*Retrieve the names of employees who work on **all** the projects that 'John Smith' works on*.

2. Create a relation that includes a tuple <Pno, Essn> whenever the employee whose Ssn is Essn works on the project whose number is Pno in the intermediate relation SSN_PNOS

$$SSN\_PNOS \leftarrow \pi_{Essn,\ Pno}(WORKS\_ON)$$

3. Apply the DIVISION operation to the two relations, which gives the desired employees' Social Security numbers

$$SSNS(Ssn) \leftarrow SSN\_PNOS \div SMITH\_PNOS$$
$$RESULT \leftarrow \pi_{Fname,\ Lname}(SSNS * EMPLOYEE)$$

Example of Division

Figure 8.8
The DIVISION operation. (a) Dividing SSN_PNOS by SMITH_PNOS. (b) $T \leftarrow R \div S$.

(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

S

A
a1
a2
a3

T

B
b1
b4

In general, the DIVISION operation is applied to two relations $R(Z) \div S(X)$, where the attributes of S are a subset of the attributes of R; that is, $X \subseteq Z$. Let Y be the set of attributes of R that are not attributes of S; that is, $Y = Z - X$ (and hence $Z = X \cup Y$).

Division Operation

The DIVISION operation can be expressed as a sequence of

π, ×, and –

$$T1 \leftarrow \pi_Y(R)$$
$$T2 \leftarrow \pi_Y((S \times T1) - R)$$
$$T \leftarrow T1 - T2$$

Operations of Relational Algebra

Table 8.1 Operations of Relational Algebra

OPERATION	PURPOSE	NOTATION
SELECT	Selects all tuples that satisfy the selection condition from a relation R.	$\sigma_{\text{<selection condition>}}(R)$
PROJECT	Produces a new relation with only some of the attributes of R, and removes duplicate tuples.	$\pi_{\text{<attribute list>}}(R)$
THETA JOIN	Produces all combinations of tuples from R_1 and R_2 that satisfy the join condition.	$R_1 \bowtie_{\text{<join condition>}} R_2$
EQUIJOIN	Produces all the combinations of tuples from R_1 and R_2 that satisfy a join condition with only equality comparisons.	$R_1 \bowtie_{\text{<join condition>}} R_2$, OR $R_1 \bowtie_{\text{(<join attributes 1>), (<join attributes 2>)}} R_2$
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.	$R_1 *_{\text{<join condition>}} R_2$, OR $R_1 *_{\text{(<join attributes 1>), (<join attributes 2>)}} R_2$ OR $R_1 * R_2$

continued on next slide

Operations of Relational Algebra (continued)

Table 8.1 Operations of Relational Algebra

OPERATION	PURPOSE	NOTATION
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 $Z = X \cup Y$.	$R_1(Z) \div R_2(Y)$

Query Tree Notation

PES UNIVERSITY ONLINE

Query Tree

- An internal data structure to represent a query in RDBMS
- The notation is called a *query tree* or sometimes it is known as a *query evaluation tree* or *query execution tree*
- Standard technique for estimating the work involved in executing the query, the generation of intermediate results, and the optimization of execution
- Nodes stand for operations like selection, projection, join, renaming, division, ….

Query Tree Notation (continued)

Query Tree

- Tree data structure corresponds to a relational algebra expression.
- It represents the input relations of the query as *leaf nodes* of the tree, represents the relational algebra operations as internal nodes.
- An execution of the query tree consists of executing an internal node operation whenever its operands (represented by its child nodes) are available, and then replacing that internal node by the relation that results from executing the operation.
- The execution terminates when the root node is executed and produces the result relation for the query.

Query Tree Notation (continued)

Query Tree

- A tree gives a good visual feel of the complexity of the query and the operations involved
- Algebraic Query Optimization consists of rewriting the query or modifying the query tree into an equivalent tree.
- *Consider the following scenario and construct the query tree*
- *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.*

Relational Algebra Expression- Query Tree (continued)

$$\pi_{\text{Pnumber, Dnum, Lname, Address, Bdate}}(((\sigma_{\text{Plocation='Stafford'}}(\text{PROJECT})) \bowtie_{\text{Dnum=Dnumber}}(\text{DEPARTMENT})) \bowtie_{\text{Mgr\_ssn=Ssn}}(\text{EMPLOYEE}))$$

Example of Query Tree

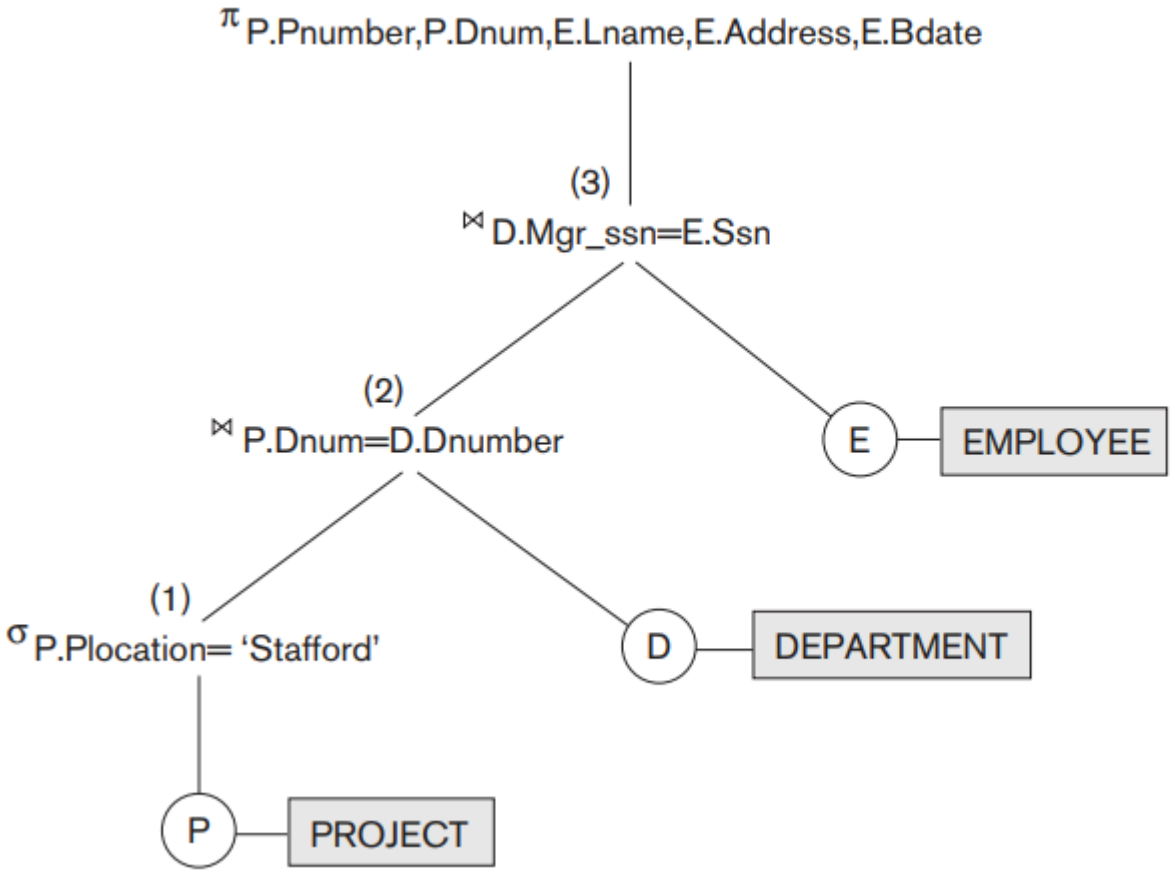


Figure 8.9
Query tree corresponding to the relational algebra expression for Q2.

THANK YOU

S Nagasundari
Department of Computer Science and Engineering

nagasundaris@pes.edu

DATABASE MANAGEMENT SYSTEM

S Nagasundari

Department of Computer Science and Engineering

DATABASE MANAGEMENT SYSTEM

Aggregate Functions and Grouping

Slides adapted from Author Slides of Fundamentals of Database Systems", Ramez Elamsri, Shamkant B Navathe, Pearson, 7th Edition, 2017.

S Nagasundari

Department of Computer Science and Engineering

DATABASE MANAGEMENT SYSTEM

Unit 2 : Relational Model

1. Relational Model Concepts
2. Relational Model Constraints and Relational Database Schemas
3. Update Operations, Transactions and dealing with constraint violations
4. Relational Database Design Using ER-to Relational Mapping
5. Unary Relational Operations
6. Set Theory Operations
7. Binary Relational Operations
8. Aggregate Functions and Grouping
9. Examples Queries in Relational Algebra

CHAPTER 8

The Relational Algebra and The Relational Calculus

Relational Algebra

- Unary Relational Operations
- Relational Algebra Operations From Set Theory
- Binary Relational Operations
- **Additional Relational Operations**
- Examples of Queries in Relational Algebra

Additional Relation Operation

- Generalized Projection

$$\pi_{F1,\ F2,\ \ldots,\ Fn}\ (R)$$

- where *F*1, *F*2, … , *Fn* are functions over the attributes in relation *R* and
- may involve arithmetic operations and constant values

Aggregate Function Operation

As an example, consider the relation

EMPLOYEE (Ssn, Salary, Deduction, Years_service)

A report may be required to show

Net Salary = Salary – Deduction,
Bonus = 2000 * Years_service, and
Tax = 0.25 * Salary

Then a generalized projection combined with renaming may be used as follows:

$$\text{REPORT} \leftarrow \rho_{(\text{Ssn, Net\_salary, Bonus, Tax})}(\pi_{\text{Ssn, Salary} - \text{Deduction, } 2000 * \text{Years\_service, } 0.25 * \text{Salary}}(\text{EMPLOYEE}))$$

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.

$$\langle \text{grouping attributes} \rangle \, \Im \, \langle \text{function list} \rangle \, (R)$$

- 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.

Additional Relational Operations: Aggregate Functions and Grouping

- 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

PES UNIVERSITY ONLINE

Use of the Aggregate Functional operation $\mathcal{F}$

- $\mathcal{F}_{\text{MAX Salary}}$ (EMPLOYEE) retrieves the maximum salary value from the EMPLOYEE relation
- $\mathcal{F}_{\text{MIN Salary}}$ (EMPLOYEE) retrieves the minimum Salary value from the EMPLOYEE relation
- $\mathcal{F}_{\text{SUM Salary}}$ (EMPLOYEE) retrieves the sum of the Salary from the EMPLOYEE relation
- $\mathcal{F}_{\text{COUNT SSN, AVERAGE Salary}}$ (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

Using Grouping with Aggregation

- A variation of aggregate operation $\mathcal{F}$ allows this:
  - Grouping attribute placed to left of symbol
  - Aggregate functions to right of symbol
  - $_{\text{DNO}}\mathcal{F}_{\text{COUNT SSN, AVERAGE Salary}}$ (EMPLOYEE)
- Above operation groups employees by DNO (department number) and computes the count of employees and average salary per department

The aggregate function operation

a. $\rho_{R(\text{Dno, No\_of\_employees, Average\_sal})}(\text{Dno } \Im \text{ COUNT Ssn, AVERAGE Salary } (\text{EMPLOYEE}))$.

b. $\text{Dno } \Im \text{ Count ssn,Average\_salary}(\text{EMPLOYEE})$.

c. $\Im \text{ COUNT ssn, AVERAGE Salary}(\text{EMPLOYEE})$.

R

(a)

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

(b)

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

(c)

Count_ssn	Average_salary
8	35125

Results of GROUP BY and HAVING

Fname	Minit	Lname	Ssn	...	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

Dno	Count (*)	Avg (Salary)
5	4	33250
4	3	31000
1	1	55000

Result of Q24

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

THANK YOU

S Nagasundari
Department of Computer Science and Engineering

nagasundaris@pes.edu

DATABASE MANAGEMENT SYSTEM

S Nagasundari

Department of Computer Science and Engineering

DATABASE MANAGEMENT SYSTEM

Aggregate Functions and Grouping

Slides adapted from Author Slides of Fundamentals of Database Systems", Ramez Elamsri, Shamkant B Navathe, Pearson, 7th Edition, 2017.

S Nagasundari

Department of Computer Science and Engineering

DATABASE MANAGEMENT SYSTEM

Unit 2 : Relational Model

1. Relational Model Concepts
2. Relational Model Constraints and Relational Database Schemas
3. Update Operations, Transactions and dealing with constraint violations
4. Relational Database Design Using ER-to Relational Mapping
5. Unary Relational Operations
6. Set Theory Operations
7. Binary Relational Operations
8. Aggregate Functions and Grouping
9. Examples Queries in Relational Algebra

CHAPTER 8

The Relational Algebra and The Relational Calculus

Relational Algebra

- Unary Relational Operations
- Relational Algebra Operations From Set Theory
- Binary Relational Operations
- **Additional Relational Operations**
- Examples of Queries in Relational Algebra

Additional Relational Operations

- Recursive Closure Operations
  - Another type of operation that, in general, cannot be specified in the basic original relational algebra is **recursive closure**.
    - This operation is applied to a **recursive relationship**.
  - An example of a recursive operation is to retrieve all SUPERVISEES of an EMPLOYEE **e** at all levels — that is, all EMPLOYEE **e'** directly supervised by **e**; all employees **e''** directly supervised by each employee **e'**; all employees **e'''** directly supervised by each employee **e''**; and so on.

Additional Relational Operations (continued)

- Although it is possible to retrieve employees at each level and then take their union, we cannot, in general, specify a query such as “retrieve the supervisees of ‘James Borg’ at all levels” without utilizing a looping mechanism.
    - The SQL3 standard includes syntax for recursive closure.

A two-level recursive query

Specify the Ssns of all employees directly supervised—*at level one*—by the employee whose name is 'James Borg'

$$\text{BORG\_SSN} \leftarrow \pi_{\text{Ssn}}(\sigma_{\text{Fname='James' AND Lname='Borg'}}(\text{EMPLOYEE}))$$
$$\text{SUPERVISION}(\text{Ssn1}, \text{Ssn2}) \leftarrow \pi_{\text{Ssn,Super\_ssn}}(\text{EMPLOYEE})$$
$$\text{RESULT1}(\text{Ssn}) \leftarrow \pi_{\text{Ssn1}}(\text{SUPERVISION} \bowtie_{\text{Ssn2=Ssn}} \text{BORG\_SSN})$$

To retrieve all employees supervised by Borg at level 2

$$\text{RESULT2}(\text{Ssn}) \leftarrow \pi_{\text{Ssn1}}(\text{SUPERVISION} \bowtie_{\text{Ssn2=Ssn}} \text{RESULT1})$$

To get both sets of employees supervised at levels 1 and 2 by 'James Borg'

$$\text{RESULT} \leftarrow \text{RESULT2} \cup \text{RESULT1}$$

SUPERVISION

(Borg's Ssn is 888665555)

(Ssn)	(Super_ssn)
Ssn1	Ssn2
123456789	333445555
333445555	888665555
999887777	987654321
987654321	888665555
666884444	333445555
453453453	333445555
987987987	987654321
888665555	null

RESULT1

Ssn
333445555
987654321

(Supervised by Borg)

RESULT2

Ssn
123456789
999887777
666884444
453453453
987987987

(Supervised by Borg's subordinates)

RESULT

Ssn
123456789
999887777
666884444
453453453
987987987
333445555
987654321

(RESULT1 ∪ RESULT2)

Additional Relational Operations (continued)

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 (continued)

- The left outer join operation keeps every tuple in the first or left relation R in

  R ⟕ 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 ⟖ S.

- A third operation, full outer join, denoted by ⟗ keeps all tuples in both the left and the right relations when no matching tuples are found, padding them with null values as needed.

Example

Suppose that we want a list of all employee names as well as the name of the departments they manage *if they happen to manage a department*; if they do not manage one, we can indicate it with a NULL value

$$TEMP \leftarrow (EMPLOYEE \bowtie_{Ssn=Mgr\_ssn} DEPARTMENT)$$
$$RESULT \leftarrow \pi_{Fname,\ Minit,\ Lname,\ Dname}(TEMP)$$

The result of a LEFT OUTER JOIN operation

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

Additional Relational Operations (continued)

OUTER UNION Operations

- The outer union operation was developed to take the union of tuples from two relations if the relations are *not type compatible*.
- This operation will take the union of tuples in two relations R(X, Y) and S(X, Z) that are **partially compatible**, meaning that only some of their attributes, say X, are type compatible.
- The attributes that are type compatible are represented only once in the result, and those attributes that are not type compatible from either relation are also kept in the result relation T(X, Y, Z).

Additional Relational Operations (continued)

Example: An outer union can be applied to two relations whose schemas are STUDENT(Name, SSN, Department, Advisor) and INSTRUCTOR(Name, SSN, Department, Rank).

- Tuples from the two relations are matched based on having the same combination of values of the shared attributes— Name, SSN, Department.
- If a student is also an instructor, both Advisor and Rank will have a value; otherwise, one of these two attributes will be null.
- The result relation STUDENT_OR_INSTRUCTOR will have the following attributes:

STUDENT_OR_INSTRUCTOR (Name, SSN, Department, Advisor, Rank)

THANK YOU

S Nagasundari
Department of Computer Science and Engineering

nagasundaris@pes.edu

DATABASE MANAGEMENT SYSTEM

S Nagasundari

Department of Computer Science and Engineering

DATABASE MANAGEMENT SYSTEM

Example Queries in Relational Algebra

Slides adapted from Author Slides of Fundamentals of Database Systems", Ramez Elamsri, Shamkant B Navathe, Pearson, 7th Edition, 2017.

S Nagasundari

Department of Computer Science and Engineering

DATABASE MANAGEMENT SYSTEM

Unit 2 : Relational Model

1. Relational Model Concepts
2. Relational Model Constraints and Relational Database Schemas
3. Update Operations, Transactions and dealing with constraint violations
4. Relational Database Design Using ER-to Relational Mapping
5. Unary Relational Operations
6. Set Theory Operations
7. Binary Relational Operations
8. Aggregate Functions and Grouping
9. **Examples Queries in Relational Algebra**

CHAPTER 8

The Relational Algebra and The Relational Calculus

Relational Algebra

- Unary Relational Operations
- Relational Algebra Operations From Set Theory
- Binary Relational Operations
- Additional Relational Operations
- **Examples of Queries in Relational Algebra**

Example Queries in Relational Algebra: Procedural Form

- **Q1: Retrieve the name and address of all employees who work for the 'Research' department.**

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

  $RESEARCH\_EMPS \leftarrow (RESEARCH\_DEPT \bowtie_{DNUMBER=DNOEMPLOYEE} EMPLOYEE)$

  $RESULT \leftarrow \pi_{FNAME, LNAME, ADDRESS}(RESEARCH\_EMPS)$

- **Q2: Retrieve the names of employees who have no dependents.**

  $ALL\_EMPS \leftarrow \pi_{SSN}(EMPLOYEE)$

  $EMPS\_WITH\_DEPS(SSN) \leftarrow \pi_{ESSN}(DEPENDENT)$

  $EMPS\_WITHOUT\_DEPS \leftarrow (ALL\_EMPS - EMPS\_WITH\_DEPS)$

  $RESULT \leftarrow \pi_{LNAME, FNAME}(EMPS\_WITHOUT\_DEPS * EMPLOYEE)$

Example Queries in Relational Algebra: Single Expression

As a single expression, these queries become:

- **Q1: Retrieve the name and address of all employees who work for the 'Research' department.**

  $\pi_{Fname, Lname, Address}$ (σ Dname= 'Research' (DEPARTMENT $\bowtie$ Dnumber=Dno(EMPLOYEE))

- **Q2: Retrieve the names of employees who have no dependents.**

  $\pi_{Lname, Fname}((\pi_{Ssn}$ (EMPLOYEE) − ρ_{Ssn} (π Essn (DEPENDENT))) $*$ EMPLOYEE)

Example Queries in Relational Algebra

Q3. Find the names of employees who work on *all* the projects controlled by department number 5.

$$\text{DEPT5\_PROJS} \leftarrow \rho_{(\text{Pno})}(\pi_{\text{Pnumber}}(\sigma_{\text{Dnum}=5}(\text{PROJECT})))$$
$$\text{EMP\_PROJ} \leftarrow \rho_{(\text{Ssn, Pno})}(\pi_{\text{Essn, Pno}}(\text{WORKS\_ON}))$$
$$\text{RESULT\_EMP\_SSNS} \leftarrow \text{EMP\_PROJ} \div \text{DEPT5\_PROJS}$$
$$\text{RESULT} \leftarrow \pi_{\text{Lname, Fname}}(\text{RESULT\_EMP\_SSNS} * \text{EMPLOYEE})$$

Example Queries in Relational Algebra

Q4. Make a list of project numbers for projects that involve an employee whose last name is 'Smith', either as a worker or as a manager of the department that controls the project.

$$\text{SMITHS(Essn)} \leftarrow \pi_{\text{Ssn}}\,(\sigma_{\text{Lname='Smith'}}(\text{EMPLOYEE}))$$
$$\text{SMITH\_WORKER\_PROJS} \leftarrow \pi_{\text{Pno}}(\text{WORKS\_ON} * \text{SMITHS})$$
$$\text{MGRS} \leftarrow \pi_{\text{Lname, Dnumber}}(\text{EMPLOYEE} \bowtie_{\text{Ssn=Mgr\_ssn}} \text{DEPARTMENT})$$
$$\text{SMITH\_MANAGED\_DEPTS(Dnum)} \leftarrow \pi_{\text{Dnumber}}\,(\sigma_{\text{Lname='Smith'}}(\text{MGRS}))$$
$$\text{SMITH\_MGR\_PROJS(Pno)} \leftarrow \pi_{\text{Pnumber}}(\text{SMITH\_MANAGED\_DEPTS} * \text{PROJECT})$$
$$\text{RESULT} \leftarrow (\text{SMITH\_WORKER\_PROJS} \cup \text{SMITH\_MGR\_PROJS})$$

$$\pi_{\text{Pno}}\,(\text{WORKS\_ON} \bowtie_{\text{Essn=Ssn}} (\pi_{\text{Ssn}}\,(\sigma_{\text{Lname='Smith'}}(\text{EMPLOYEE}))) \cup \pi_{\text{Pno}}$$
$$((\pi_{\text{Dnumber}}\,(\sigma_{\text{Lname='Smith'}}(\pi_{\text{Lname, Dnumber}}(\text{EMPLOYEE}))) \bowtie$$
$$_{\text{Ssn=Mgr\_ssn}}\text{DEPARTMENT})) \bowtie_{\text{Dnum-ber=Dnum}} \text{PROJECT})$$

Example Queries in Relational Algebra

Q5. List the names of all employees with two or more dependents

$$T1(\text{Ssn, No\_of\_dependents}) \leftarrow {}_{\text{Essn}}\Im_{\text{COUNT Dependent\_name}}(\text{DEPENDENT})$$
$$T2 \leftarrow \sigma_{\text{No\_of\_dependents}>2}(T1)$$
$$\text{RESULT} \leftarrow \pi_{\text{Lname, Fname}}(T2 * \text{EMPLOYEE})$$

Example Queries in Relational Algebra

Q7. List the names of managers who have at least one dependent.

$$MGRS(Ssn) \leftarrow \pi_{Mgr\_ssn}(DEPARTMENT)$$
$$EMPS\_WITH\_DEPS(Ssn) \leftarrow \pi_{Essn}(DEPENDENT)$$
$$MGRS\_WITH\_DEPS \leftarrow (MGRS \cap EMPS\_WITH\_DEPS)$$
$$RESULT \leftarrow \pi_{Lname,\ Fname}(MGRS\_WITH\_DEPS * EMPLOYEE)$$

Example Queries in Relational Algebra

Consider a database with the following schema:

Person (name, age, gender) **name is a key**

Frequents (name, pizzeria) (name, pizzeria) is a key

Eats (name, pizza) (name, pizza) is a key

Serves (pizzeria, pizza, price) (pizzeria, pizza) is a key

Write relational algebra expressions for the following queries

- Eats contains information about what type of pizza each person (customer) eats (likes)
- Name is always the customer's name
- Frequents record information about pizzerias that each customer frequents (visits)
- Serves contains information about all possible pizza types for each pizzeria

Source: http://openclassroom.stanford.edu/MainFolder/courses/cs145/old-site/docs/backup/ra-exercises.html

Example Queries in Relational Algebra

1. Find all pizzerias frequented by at least one person under the age of 18.
2. Find the names of all females who eat either mushroom or pepperoni pizza (or both).
3. Find the names of all females who eat both mushroom and pepperoni pizza.
4. Find all pizzerias that serve at least one pizza that Amy eats for less than $10.00.
5. Find all pizzerias that are frequented by only females or only males.
6. For each person, find all pizzas the person eats that are not served by any pizzeria the person frequents. Return all such person (name) / pizza pairs.
7. Find the names of all people who frequent only pizzerias serving at least one pizza they eat.
8. Find the names of all people who frequent every pizzeria serving at least one pizza they eat.
9. Find the pizzeria serving the cheapest pepperoni pizza. In the case of ties, return all of the cheapest-pepperoni pizzerias.

Exercise

Draw the Query tree for the queries given in the previous slides

Summary

Relational Algebra

- Unary Relational Operations
- Relational Algebra Operations From Set Theory
- Binary Relational Operations
- Additional Relational Operations
- Examples of Queries in Relational Algebra

THANK YOU

S Nagasundari
Department of Computer Science and Engineering

nagasundaris@pes.edu