



Database Management System

(18IS5DCDBM)

Mrs. Bhavani K
Assistant Professor
Dept. of ISE, DSCE

UNIT 2

- **Relational Model:**
 - Relational Model Concepts
 - Relational Model constraints and Relational Database Schemas
 - update operations, Transactions and dealing with constraint violations.
- **Relational Algebra:**
 - Unary Relational Operations: SELECT and PROJECT
 - Relational Algebra Operations from Set Theory
 - Binary Relational Operations: JOIN and DIVISION
 - Additional Relational Operations
 - Relational Database Design Using ER-to-Relational mapping

Relational Model

Relational Model Concepts

- The relational Model of Data is based on the concept of a Relation.
 - A Relation is a mathematical concept based on the ideas of sets.
- The strength of the relational approach to data management comes from the formal foundation provided by the theory of relations.

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 popular ACM Turing Award

Informal Definitions

- Informally, a **relation** is 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

Diagram illustrating the structure of a relation:

- Relation Name:** STUDENT
- Attributes:** Name, Ssn, Home_phone, Address, Office_phone, Age, Gpa
- Tuples:** The rows of data in the table.

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

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
 - 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
 - *artificial key or surrogate key*

Formal Definitions - Schema

- The **Schema** (or description) of a Relation:
 - Denoted by $R(A_1, A_2, \dots, A_n)$
 - R is the **name** of the relation
 - The **attributes** of the relation are A_1, A_2, \dots, A_n
- Example:
CUSTOMER (Cust-id, Cust-name, Address, PhoneNo)
 - CUSTOMER is the relation name
 - Defined over the four attributes: Cust-id, Cust-name, Address, PhoneNo
- Each attribute has a **domain** or a set of valid values.
 - For example, the domain of Cust-id is 6 digit numbers.

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, "Ram", "101, BSK 3rd Stage , Bangalore", 9882756799>
 - 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: “India_phone_numbers” are the set of 10 digit phone numbers valid in India.
- A domain also has a data-type or a format defined for it.
 - The India_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.
- 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
 - $\text{dom}(\text{Cust-name})$ 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 $R(A_1, A_2, \dots, A_n)$
 - $r(R) \subseteq \text{dom}(A_1) \times \text{dom}(A_2) \times \dots \times \text{dom}(A_n)$
- $R(A_1, A_2, \dots, A_n)$ is the **schema** of the relation
- R is the **name** of the relation
- A_1, A_2, \dots, A_n 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)
 - $r(R) = \{t_1, t_2, \dots, t_n\}$ where each t_i is an n -tuple
 - $t_i = \langle v_1, v_2, \dots, v_n \rangle$ where each v_j *element-of* $\text{dom}(A_j)$

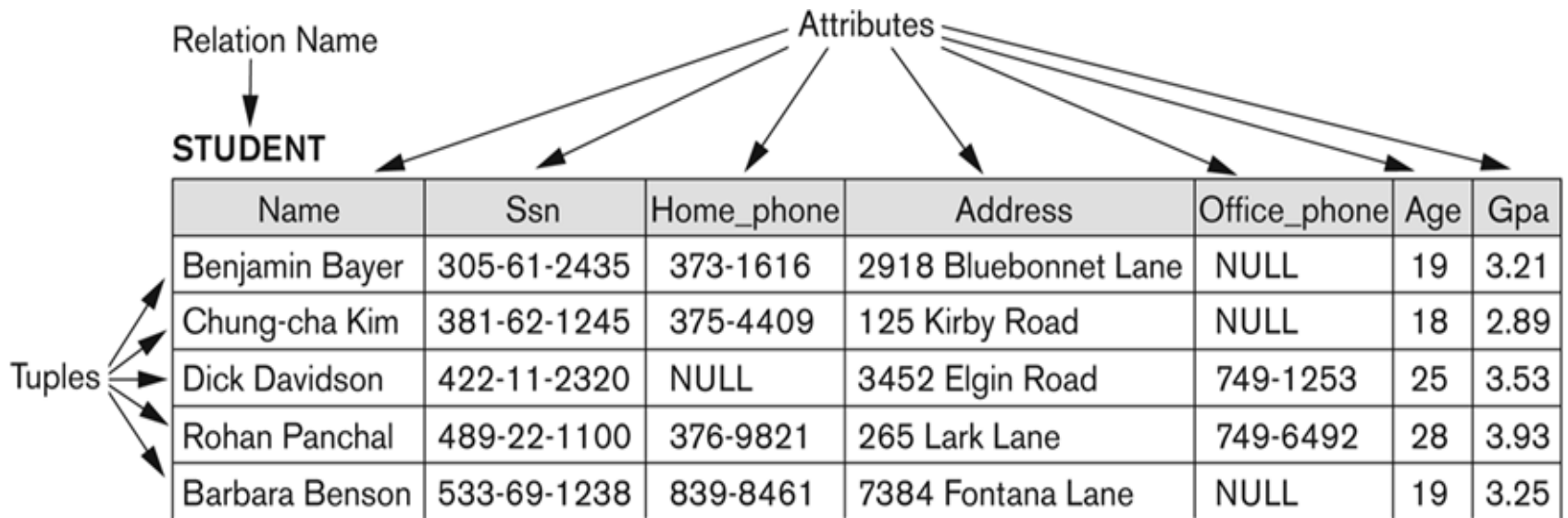
Formal Definitions - Example

- Let $R(A1, A2)$ be a relation schema:
 - Let $\text{dom}(A1) = \{0,1\}$
 - Let $\text{dom}(A2) = \{a,b,c\}$
- Then: $\text{dom}(A1) \times \text{dom}(A2)$ is all possible combinations:
 $\{ \langle 0,a \rangle, \langle 0,b \rangle, \langle 0,c \rangle, \langle 1,a \rangle, \langle 1,b \rangle, \langle 1,c \rangle \}$
- The relation state $r(R) \subset \text{dom}(A1) \times \text{dom}(A2)$
- For example: $r(R)$ could be $\{ \langle 0,a \rangle, \langle 0,b \rangle, \langle 1,c \rangle \}$
 - this is one possible state (or “population” or “extension”) r of the relation R , defined over $A1$ and $A2$.
 - It has three 2-tuples: $\langle 0,a \rangle, \langle 0,b \rangle, \langle 1,c \rangle$

Definition Summary

<u>Informal Terms</u>	<u>Formal Terms</u>
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



Relation Name

STUDENT

Attributes

Tuples

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

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 $R(A_1, A_2, \dots, A_n)$ and the values in $t = \langle v_1, v_2, \dots, v_n \rangle$ to be ordered .
 - (However, a more general alternative definition of relation does not require this ordering).

Same state as previous Figure (but with different order of tuples)

The relation STUDENT from Figure — 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 $t = \langle v_1, v_2, \dots, v_n \rangle$ is a tuple (row) in the relation state r of $R(A_1, A_2, \dots, A_n)$
 - Then each v_i must be a value from $dom(A_i)$
 - A special **null** value is used to represent values that are unknown or inapplicable to certain tuples.

Relational Model Concepts (Summary)

Table also called **Relation**

Primary Key

Domain
Ex: NOT NULL

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive

Tuple OR Row
Total # of rows is **Cardinality**

Column OR Attributes
Total # of column is **Degree**

Characteristics Of Relations

- Notation:
 - We refer to **component values** of a tuple t by:
 - $t[A_i]$ or $t.A_i$
 - This is the value v_i of attribute A_i for tuple t
 - Similarly, $t[A_u, A_v, \dots, A_w]$ refers to the subtuple of t containing the values of attributes A_u, A_v, \dots, A_w , respectively in t

Relational Integrity Constraints

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

Key Constraints

- **Superkey of R:**
 - 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 t_1 and t_2 in $r(R)$, $t_1[SK] \neq t_2[SK]$
 - This condition must hold in *any valid state* $r(R)$
- **Key of R:**
 - A "minimal" superkey
 - That is, a key is a superkey K such that removal of any attribute from K results in a set of attributes that is not a superkey (does not possess the superkey uniqueness property)

Key Constraints (continued)

- 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.
- 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

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

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

CAR

<u>License_number</u>	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

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

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**
 - $S = \{R_1, R_2, \dots, R_n\}$
 - R_1, R_2, \dots, R_n are the names of the individual **relation schemas** within the database S
- Following slide shows a COMPANY database schema with 6 relation schemas

COMPANY Database Schema

EMPLOYEE

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

DEPARTMENT

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

DEPT_LOCATIONS

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

PROJECT

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

WORKS_ON

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

DEPENDENT

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

Schema diagram for
the COMPANY
relational database
schema.

Entity Integrity

- **Entity Integrity:**
 - The *primary key attributes* PK of each relation schema R in S cannot have null values in any tuple of $r(R)$.
 - This is because primary key values are used to *identify* the individual tuples.
 - $t[PK] \neq \text{null}$ 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

- Statement of the constraint
 - The value in the foreign key column (or columns) FK of the 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

Referential Integrity (or foreign key) Constraint

Employee

EID	Name	DNO
101	Radha	11
102	Krishna	22
103	Sham	33

Department

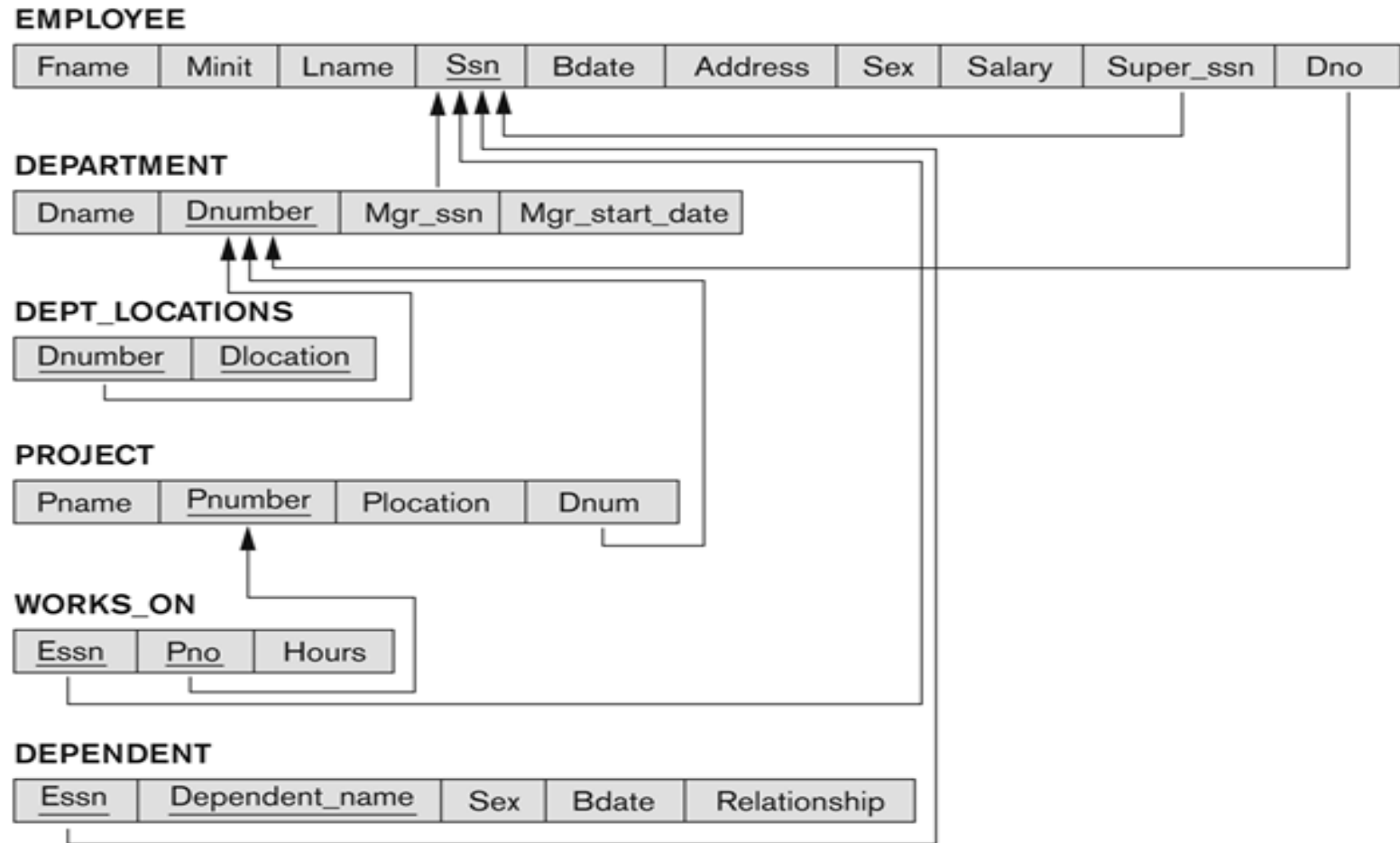
DNO	Place
11	Bangalore
22	Hyderabad
33	Jaipur

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 to the primary key of the referenced relation for clarity
- Next slide shows the **COMPANY relational schema diagram**

Referential Integrity Constraints for COMPANY database

Referential integrity constraints displayed on the COMPANY relational database schema



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 allows triggers and **ASSERTIONS** to express for some of these

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
- Next slide shows an example state for the COMPANY database

Populated database state for COMPANY

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

Update Operations on Relations

- 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

- 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 each 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
 - 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

Possible violations for each 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 (see Chapter 8 for more details)
 - 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 each 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):
 - Similar to a 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

Relational Algebra

Relational Algebra

- 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
 - all objects in relational algebra are relations
- 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)

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 (\times)
 - Binary Relational Operations
 - JOIN (several variations of JOIN exist)
 - DIVISION
 - Additional Relational Operations
 - OUTER JOINS, OUTER UNION
 - AGGREGATE FUNCTIONS (These compute summary of information: for example, SUM, COUNT, AVG, MIN, MAX)

Database Schema for COMPANY

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: 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:

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

Unary Relational Operations: SELECT

– In general, the *select* operation is denoted by

$\sigma_{\langle \text{selection condition} \rangle}(R)$ 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

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

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:

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

Unary Relational Operations: PROJECT

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

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

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

Unary Relational Operations: PROJECT

- PROJECT Operation Properties
 - The number of tuples in the result of projection $\pi_{\langle \text{list} \rangle}(R)$ is always less or equal to the number of tuples in R
 - 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
 - $\pi_{\langle \text{list1} \rangle}(\pi_{\langle \text{list2} \rangle}(R)) = \pi_{\langle \text{list1} \rangle}(R)$ as long as $\langle \text{list2} \rangle$ contains the attributes in $\langle \text{list1} \rangle$

Example of applying SELECT and PROJECT operations

Results of SELECT and PROJECT operations. (a) $\sigma_{(Dno=4 \text{ AND } Salary>25000) \text{ OR } (Dno=5 \text{ AND } Salary>30000)}(EMPLOYEE)$.
(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

SELECTION & PROJECTION Example

Person

Id	Name	Address	Hobby
1123	John	123 Main	stamps
1123	John	123 Main	coins
5556	Mary	7 Lake Dr	hiking
9876	Bart	5 Pine St	stamps

σ *Hobby='stamps'(Person)*

Id	Name	Address	Hobby
1123	John	123 Main	stamps
9876	Bart	5 Pine St	stamps

Π *Name, Hobby(Person)*

Name	Hobby
John	stamps
John	coins
Mary	Hiking
Bart	stamps

Relational Algebra Expressions

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

Single expression versus sequence of relational operations (Example)

- To retrieve the first name, last name, and salary of all employees who work in department number 5, we must apply a select and a project operation
- We can write a *single relational algebra expression* as follows:
 - $\pi_{\text{FNAME, LNAME, SALARY}}(\sigma_{\text{DNO}=5}(\text{EMPLOYEE}))$
- OR We can explicitly show the *sequence of operations*, giving a name to each intermediate relation:
 - $\text{DEP5_EMPS} \leftarrow \sigma_{\text{DNO}=5}(\text{EMPLOYEE})$
 - $\text{RESULT} \leftarrow \pi_{\text{FNAME, LNAME, SALARY}}(\text{DEP5_EMPS})$

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 name or both
 - Useful when a query requires multiple operations
 - Necessary in some cases (see JOIN operation later)

Unary Relational Operations: RENAME (contd.)

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

Unary Relational Operations: RENAME (contd.)

- For convenience, we also use a *shorthand* for renaming attributes in an intermediate relation:
 - If we write:
 - $\text{RESULT} \leftarrow \pi_{\text{FNAME, LNAME, SALARY}}(\text{DEP5_EMPS})$
 - RESULT will have the *same attribute names* as DEP5_EMPS (same attributes as EMPLOYEE)
 - If we write:
 - $\text{RESULT}(\text{F, M, L, S, B, A, SX, SAL, SU, DNO}) \leftarrow \pi_{\text{FNAME, LNAME, SALARY}}(\text{DEP5_EMPS})$
 - The 10 attributes of DEP5_EMPS are *renamed* to F, M, L, S, B, A, SX, SAL, SU, DNO, respectively

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

Results of a sequence of operations.

(a) $\pi_{Fname, Lname, Salary}(\sigma_{Dno=5}(EMPLOYEE))$.

(b) Using intermediate relations and renaming of attributes.

Relational Algebra Operations from Set Theory: UNION

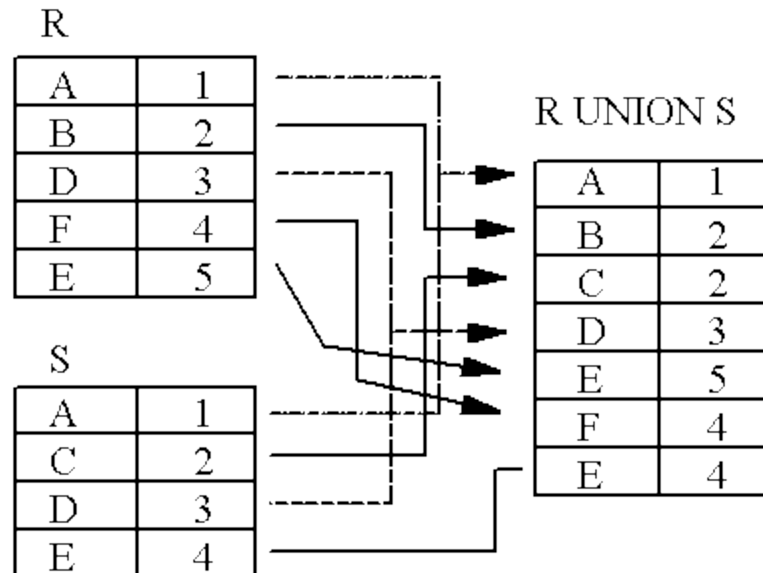
- UNION Operation
 - Binary operation, denoted by \cup
 - The result of $R \cup S$, is a relation that includes all tuples that are either in R or in S or in both R and S
 - Duplicate tuples are eliminated
 - The two operand relations R and S must be “type compatible” (or UNION compatible)
 - R and S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

Relational Algebra Operations from Set Theory: UNION

- Example:
 - To retrieve the social security numbers of all employees who either *work in department 5* (RESULT1 below) or *directly supervise an employee who works in department 5* (RESULT2 below)
 - We can use the UNION operation as follows:
$$\begin{aligned} \text{DEP5_EMPS} &\leftarrow \sigma_{\text{DNO}=5}(\text{EMPLOYEE}) \\ \text{RESULT1} &\leftarrow \pi_{\text{SSN}}(\text{DEP5_EMPS}) \\ \text{RESULT2}(\text{SSN}) &\leftarrow \pi_{\text{SUPERSSN}}(\text{DEP5_EMPS}) \\ \text{RESULT} &\leftarrow \text{RESULT1} \cup \text{RESULT2} \end{aligned}$$
 - The union operation produces the tuples that are in either RESULT1 or RESULT2 or both

Example of the result of a UNION operation

UNION Example



Relational Algebra Operations from Set Theory

- Type Compatibility of operands is required for the binary set operation UNION \cup , (also for INTERSECTION \cap , and SET DIFFERENCE $-$)
- $R1(A1, A2, \dots, An)$ and $R2(B1, B2, \dots, Bn)$ are type compatible if:
 - they have the same number of attributes, and
 - the domains of corresponding attributes are type compatible (i.e. $\text{dom}(Ai) = \text{dom}(Bi)$ for $i=1, 2, \dots, n$).
- The resulting relation for $R1 \cup R2$ (also for $R1 \cap R2$, or $R1 - R2$, see next slides) has the same attribute names as the *first* operand relation $R1$ (by convention)

Relational Algebra Operations from Set Theory:

INTERSECTION

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

Example of the result of a INTERSECTION

R

A	1
B	2
D	3
F	4
E	5

R INTERSECTION S

A	1
D	3

S

A	1
C	2
D	3
E	4

Relational Algebra Operations from Set Theory: SET DIFFERENCE

- 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”

Example of the result of a SET DIFFERENCE

R

A	1
B	2
D	3
F	4
E	5

S

A	1
C	2
D	3
E	4

R DIFFERENCE S

B	2
F	4
E	5

S DIFFERENCE R

C	2
E	4

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

(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

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

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

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

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

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

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

Example of applying CARTESIAN PRODUCT

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

Example of applying CARTESIAN PRODUCT

EMP_DEPENDENTS

Fname	Lname	San	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	...

Example of applying CARTESIAN PRODUCT

ACTUAL_DEPENDENTS

Fname	Lname	San	Esn	Dependent_name	Sex	Bdate	...
Jennifer	Wallace	987654321	987654321	Abner	M	1942-02-28	...

RESULT

Fname	Lname	Dependent_name
Jennifer	Wallace	Abner

CARTESIAN PRODUCT Example

R

A	1
B	2
D	3
F	4
E	5

S

A	1
C	2
D	3
E	4

R CROSS S

A	1	A	1
A	1	C	2
A	1	D	3
A	1	E	4
B	2	A	1
B	2	C	2
B	2	D	3
B	2	E	4
D	3	A	1
D	3	C	2
D	3	D	3
D	3	E	4

F	4	A	1
F	4	C	2
F	4	D	3
F	4	E	4
E	5	A	1
E	5	C	2
E	5	D	3
E	5	E	4

Binary Relational Operations: JOIN

- JOIN Operation (denoted by \bowtie)
 - 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 $R(A_1, A_2, \dots, A_n)$ and $S(B_1, B_2, \dots, B_m)$ is:
$$R \bowtie_{\langle \text{join condition} \rangle} S$$
 - where R and S can be any relations that result from general *relational algebra expressions*.

Binary Relational Operations: JOIN (cont.)

- Example: Suppose that we want to retrieve the name of the manager of each department.
 - To get the manager's name, we need to combine each DEPARTMENT tuple with the EMPLOYEE tuple whose SSN value matches the MGRSSN value in the department tuple.
 - We do this by using the join \bowtie operation.
 - $\text{DEPT_MGR} \leftarrow \text{DEPARTMENT} \bowtie_{\text{MGRSSN}=\text{SSN}} \text{EMPLOYEE}$
- MGRSSN=SSN is the join condition
 - Combines each department record with the employee who manages the department
 - The join condition can also be specified as $\text{DEPARTMENT.MGRSSN} = \text{EMPLOYEE.SSN}$

Example of applying the JOIN operation

DEPT_MGR

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

Result of the JOIN operation

JOIN Example

R *ColA* *ColB*

A	1
B	2
D	3
F	4
E	5

S *SColA* *SColB*

A	1
C	2
D	3
E	4

R JOIN_{R.ColA = S.SColA} S

A	1	A	1
D	3	D	3
E	5	E	4

R JOIN_{R.ColB = S.SColB} S

A	1	A	1
B	2	C	2
D	3	D	3
F	4	E	4

Some properties of JOIN

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

Some properties of JOIN

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

Binary Relational Operations: EQUIJOIN

- 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.

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.

Binary Relational Operations

NATURAL JOIN (contd.)

- Example: To apply a natural join on the DNUMBER attributes of DEPARTMENT and DEPT_LOCATIONS, it is sufficient to write:
 - $DEPT_LOCS \leftarrow DEPARTMENT * DEPT_LOCATIONS$
- 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: $Q \leftarrow R(A,B,C,D) * S(C,D,E)$
 - The implicit join condition includes *each pair* of attributes with the same name, “AND”ed together:
 - $R.C = S.C \text{ AND } R.D = S.D$
 - Result keeps only one attribute of each such pair:
 - $Q(A,B,C,D,E)$

Example of NATURAL JOIN operation

(a)

PROJ_DEPT

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

(b)

DEPT_LOCS

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

Results of two NATURAL JOIN operations.

(a) PROJ_DEPT \leftarrow PROJECT * DEPT.

(b) DEPT_LOCS \leftarrow DEPARTMENT * DEPT_LOCATIONS.

Complete Set of Relational Operations

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

Binary Relational Operations: DIVISION

- 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 .

Example of DIVISION

(a)

SSN_PNOS

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

SMITH_PNOS

Pno
1
2

SSNS

Ssn
123456789
453453453

(b)

R

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

S

A
a1
a2
a3

T

B
b1
b4

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

Recap of Relational Algebra Operations

Operations of Relational Algebra

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

Relational Algebra

- Banking Example

branch (branch-name, branch-city, assets)

customer (customer-name, customer-street, customer-city)

account (account-number, branch-name, balance)

loan (loan-number, branch-name, amount)

depositor (customer-name, account-number)

borrower (customer-name, loan-number)

Example Queries

- Find all loans of over 1200

$$\sigma_{amount > 1200} (loan)$$

- Find the loan number for each loan of an amount greater than 1200

$$\Pi_{loan-number} (\sigma_{amount > 1200} (loan))$$

Example Queries

Find the names of all customers who have a loan, an account, or both, from the bank

$$\Pi_{customer-name}(borrower) \cup \Pi_{customer-name}(depositor)$$

Find the names of all customers who have a loan and an account at bank.

$$\Pi_{customer-name}(borrower) \cap \Pi_{customer-name}(depositor)$$

Example Queries

Find the names of all customers who have a loan at the Banashankari branch.

$$\Pi_{customer-name} (\sigma_{branch-name="Banashankari"} (\sigma_{borrower.loan-number = loan.loan-number} (borrower \times loan)))$$

Find the names of all customers who have a loan at Banashankari branch but do not have an account at any other branch

$$\Pi_{customer-name} (\sigma_{branch-name = "Banashankari"} (\sigma_{borrower.loan-number = loan.loan-number} (borrower \times loan))) - \Pi_{customer-name} (depositor)$$

Example Queries

- **Find the largest account balance**
 1. Rename *account* relation as *d*
 2. The query is:

$$\Pi_{balance}(account) - \Pi_{account.balance}(\sigma_{account.balance < d.balance} (account \times \rho_d (account)))$$

Additional Relational Operations: Aggregate Functions and Grouping

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

Aggregate Function Operation

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

Using Grouping with Aggregation

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

Examples of applying aggregate functions and grouping

The aggregate function operation.

(a) $\rho_{R(Dno, No_of_employees, Average_sal)} (Dno \int COUNT Ssn, AVERAGE Salary (EMPLOYEE)).$

(b) $Dno \int COUNT Ssn, AVERAGE Salary (EMPLOYEE).$

(c) $\int COUNT Ssn, AVERAGE Salary (EMPLOYEE).$

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

Illustrating aggregate functions and grouping

(a)

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

Grouping EMPLOYEE tuples by the value of Dno

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

Result of Q24

Additional Relational Operations (cont.)

- 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 (cont.)

- Although it is possible to retrieve employees at each level and then take their union, one 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.

Additional Relational Operations (cont.)

(Borg's SSN is 888665555)

(SSN)

(SUPERSSN)

SUPERVISION	SSN1	SSN2
	123456789	333445555
	333445555	888665555
	999887777	987654321
	987654321	888665555
	666884444	333445555
	453453453	333445555
	987987987	987654321

RESULT 1	SSN
	333445555
	987654321

(Supervised by Borg)

RESULT 2	SSN
	123456789
	999887777
	666884444
	453453453
	987987987

(Supervised by Borg's subordinates)

RESULT	SSN
	123456789
	999887777
	666884444
	453453453
	987987987
	333445555
	987654321

(RESULT1 \cup RESULT2)

Additional Relational Operations (cont.)

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

Additional Relational Operations (cont.)

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

OUTER JOIN Example

R	ColA	ColB	R LEFT OUTER JOIN R.ColA = S.SColA	S
	A	1	A	1
	B	2	D	3
	D	3	E	5
	F	4	B	2
	E	5	F	4
			-	-
			-	-

S	SColA	SColB	R RIGHT OUTER JOIN R.ColA = S.SColA	S
	A	1	A	1
	C	2	D	3
	D	3	E	5
	E	4	-	-
			C	2

OUTER JOIN Example (cont.)

R	ColA	ColB	R FULL OUTER JOIN R.ColA = S.SColA	S
	A	1	A	1
	B	2	D	3
	D	3	E	5
	F	4	B	2
	E	5	F	4
			-	-
			-	-
			C	2
S	SColA	SColB		
	A	1		
	C	2		
	D	3		
	E	4		

Additional Relational Operations (cont.)

- 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 (cont.)

- 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)

OUTER UNION Example

R OUTER UNION S

R

X	A
1	a
1	a
1	b
2	c
3	v
4	e
6	g

S

X	B
1	x
2	y
3	z
3	v
5	w

X	A	B
1	a	
1	a	
1	b	
2	c	
3	v	
4	e	
6	g	
1		x
2		y
3		z
3		v
5		w

ER-to-Relational Mapping Algorithm

Step 1: Mapping of Regular Entity Types

Step 2: Mapping of Weak Entity Types

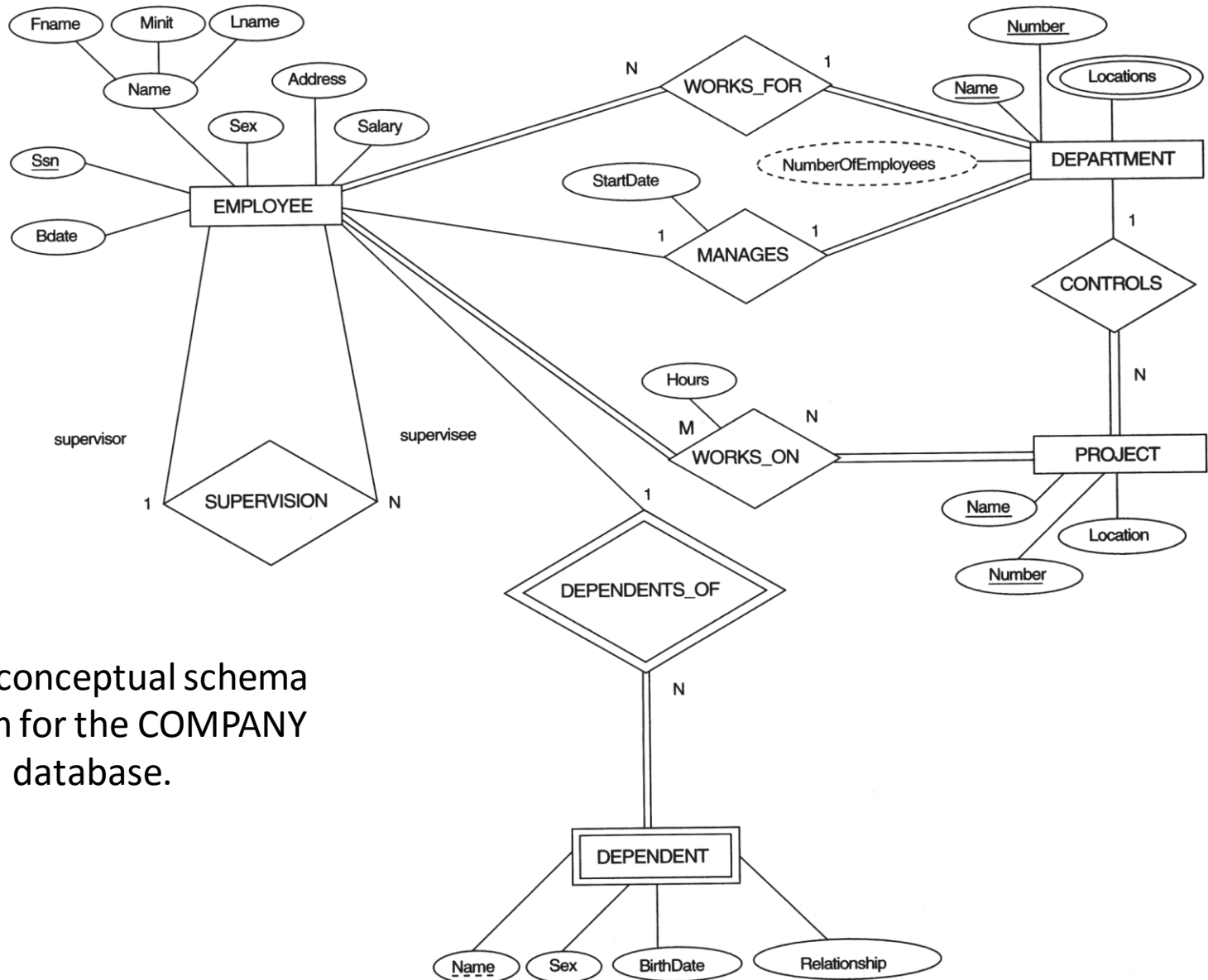
Step 3: Mapping of Binary 1:1 Relation Types

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



The ER conceptual schema diagram for the COMPANY database.

ER-to-Relational Mapping Algorithm

- 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:** 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.

ER-to-Relational Mapping Algorithm (contd.)

- Step 1 Result

EMPLOYEE

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

DEPARTMENT

Dname	<u>Dnumber</u>
-------	----------------

PROJECT

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

ER-to-Relational Mapping Algorithm (contd.)

- **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.
- **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*.

ER-to-Relational Mapping Algorithm (contd.)

- Step 2 Result

EMPLOYEE

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

DEPARTMENT

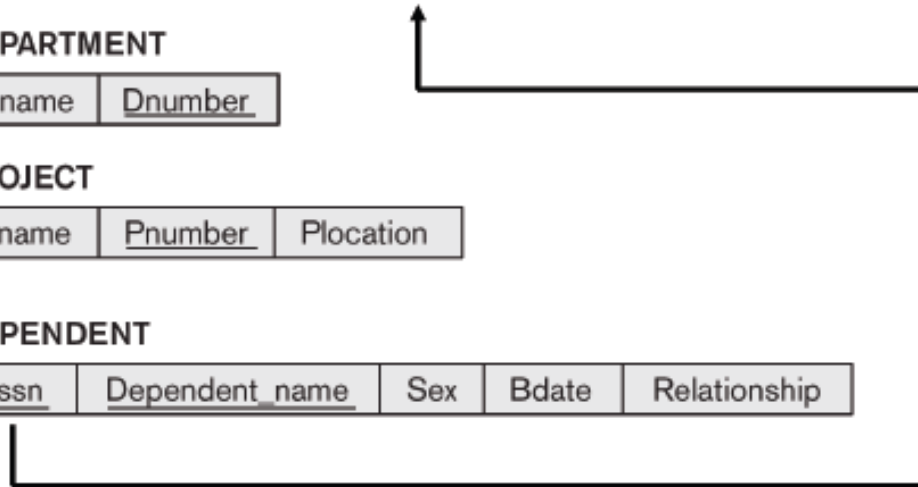
Dname	<u>Dnumber</u>
-------	----------------

PROJECT

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

DEPENDENT

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



ER-to-Relational Mapping Algorithm (contd.)

- **Step 3: Mapping of Binary 1:1 Relationship Types**
- Three approaches
 - **Foreign Key**
 - Usually appropriate
 - Merged Relation
 - Possible when both participations are total
 - Relationship Relation
 - Create a new relation for the relationship
- For each 1:1 relationship type R, identify the relations S and T participating in R.
- **Foreign key approach**
 - Choose one relation as *S*, the other *T*
Better if S has total participation (reduces number of NULL values)
 - Add to *S* all the simple attributes of the relationship
 - Add as a foreign key in *S* the primary key attributes of *T*

ER-to-Relational Mapping Algorithm (contd.)

- **Example:** No relation is created for the 1:1 relationship type MANAGES.
 - Include the primary key Ssn of the EMPLOYEE relation as a foreign key attribute of DEPARTMENT (renamed to Mgr_ssn).
 - Include the attribute Start_date of the MANAGES relationship in DEPARTMENT and rename it Mgr_start_date.

ER-to-Relational Mapping Algorithm (contd.)

- Step 2 Result

EMPLOYEE

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

DEPARTMENT

Dname	<u>Dnumber</u>
-------	----------------

- Step 3 Result

EMPLOYEE

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

DEPARTMENT

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

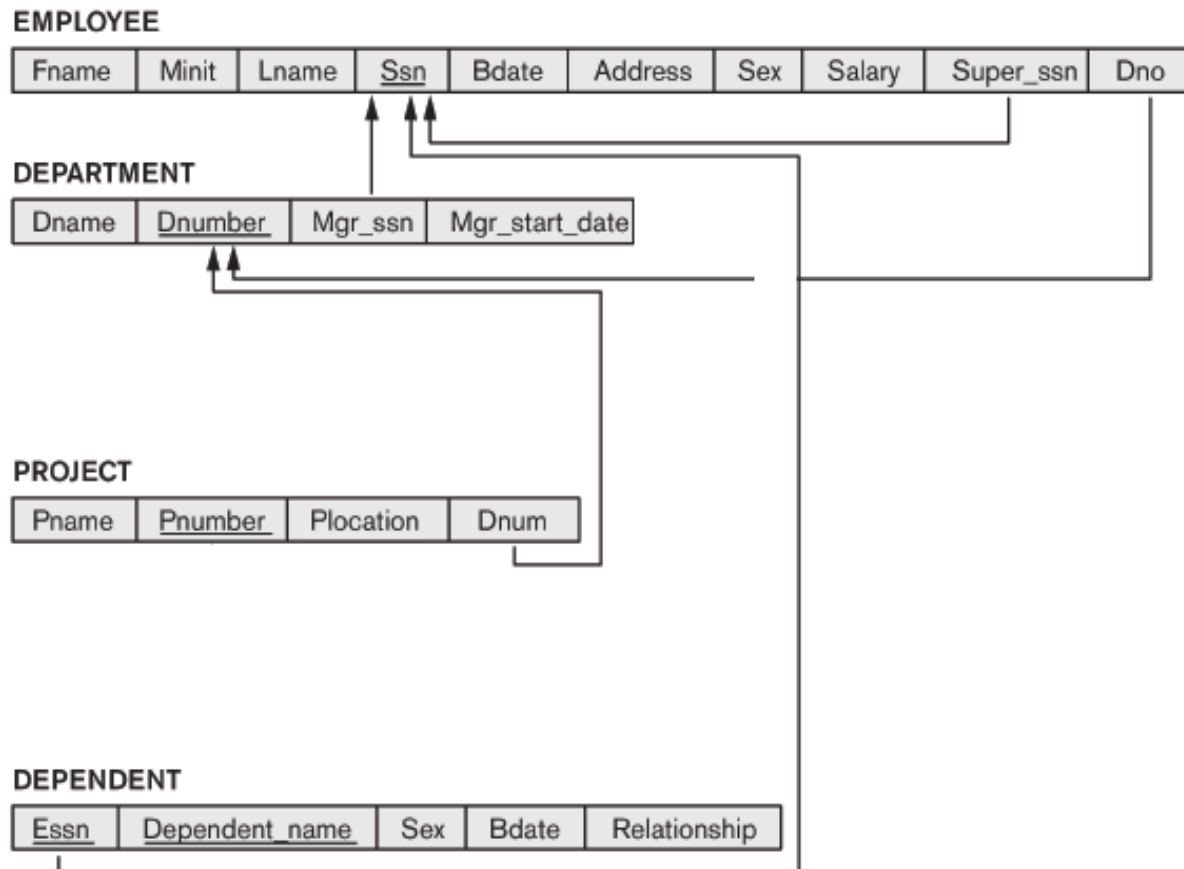


ER-to-Relational Mapping Algorithm (contd.)

- **Step 4: Mapping of Binary 1:N Relationship Types**
 - For each binary 1:N relationship type R, identify the relation S that represents the participating entity type at the N-side of the relationship. Let T denotes the other participating entity.
 - Include the primary key of T as a foreign key in S.
 - Include any simple attributes of the relationship as attributes of S.
- **Example:** Map the 1:N relationship types WORKS_FOR, CONTROLS, SUPERVISION.
 - WORKS_FOR: Include the primary key Dnumber of DEPARTMENT as foreign key in EMPLOYEE. Call it Dno.
 - SUPERVISION: Include the primary key of EMPLOYEE as foreign key in EMPLOYEE. Call it Super_ssn.
 - CONTROLS: Include the primary key Dnumber of DEPARTMENT in PROJECT. Call it Dnum.

ER-to-Relational Mapping Algorithm (contd.)

- Step 4 Result

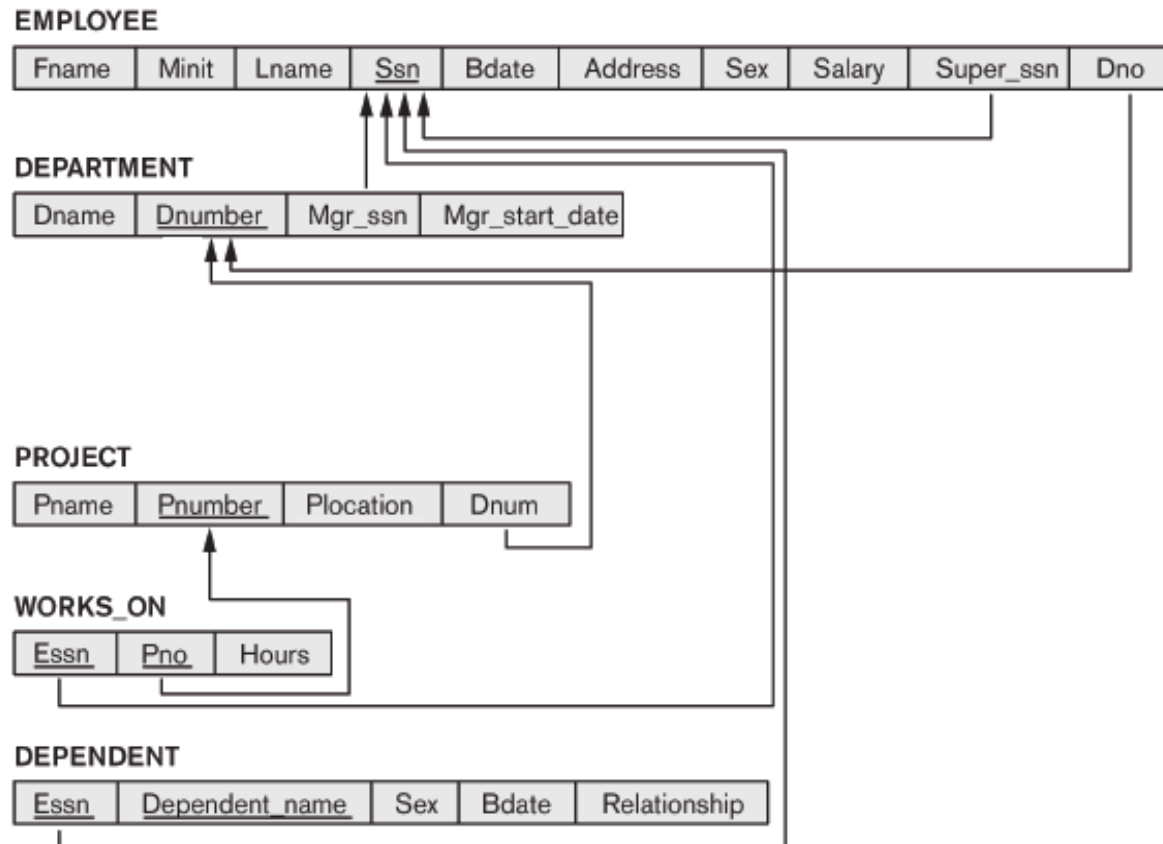


ER-to-Relational Mapping Algorithm (contd.)

- **Step 5: Mapping of Binary M:N Relationship Types**
 - For each binary M:N relationship type R, create a new relation S.
 - Include the primary keys of both participating entities as foreign keys.
 - Their combination will form the primary key of S.
 - Also include any simple attributes of R.
- **Example:** Create the relation WORKS_ON.
 - Include the primary keys of PROJECT and EMPLOYEE as foreign keys. Rename them Pno and Essn.
 - Also include an attribute Hours.
 - The primary key is the combination of {Essn, Pno}.

ER-to-Relational Mapping Algorithm (contd.)

- Step 5 Result

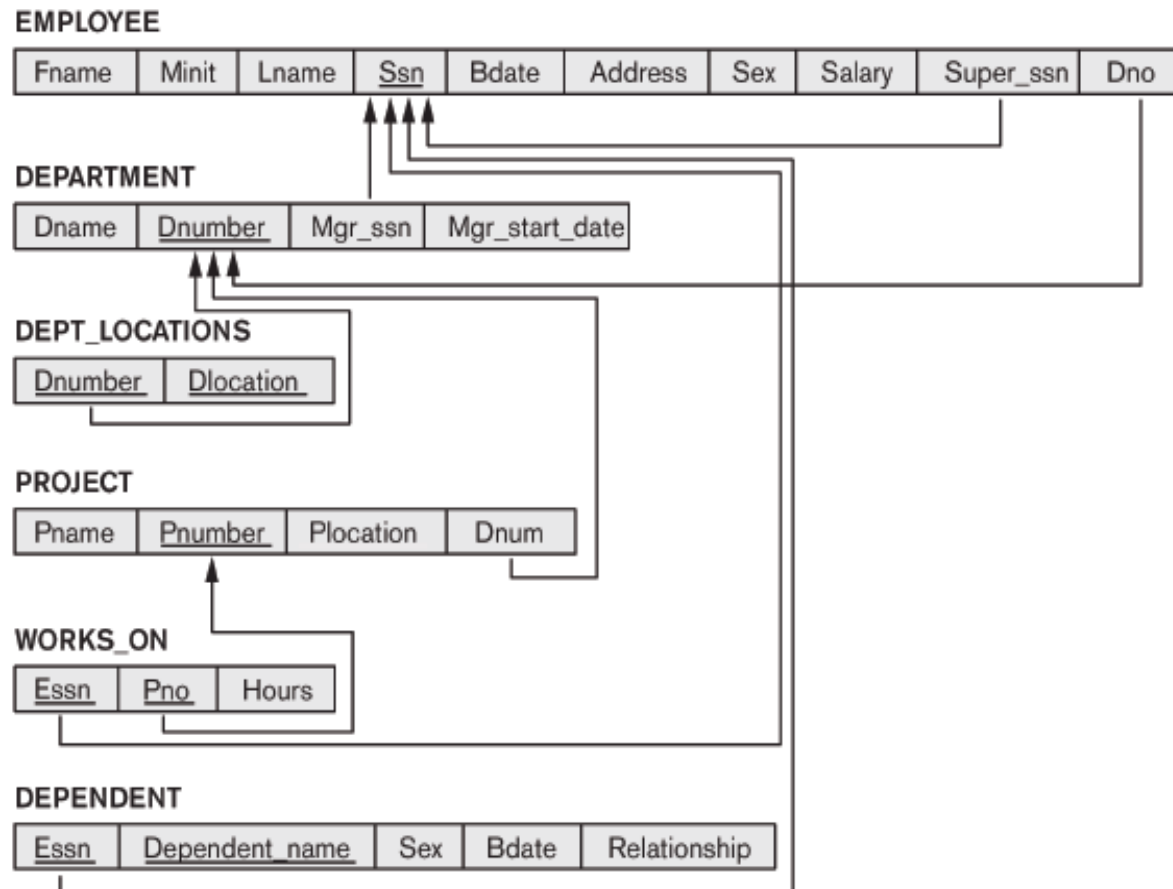


ER-to-Relational Mapping Algorithm (contd.)

- **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, include its simple components.
- **Example:** The relation DEPT_LOCATIONS is created.
 - The attribute DLOCATION represents the multivalued attribute Locations of DEPARTMENT, while Dnumber (as foreign key) representing the primary key of the DEPARTMENT relation.
 - The primary key of R is the combination of {Dnumber, Dlocation}.

ER-to-Relational Mapping Algorithm (contd.)

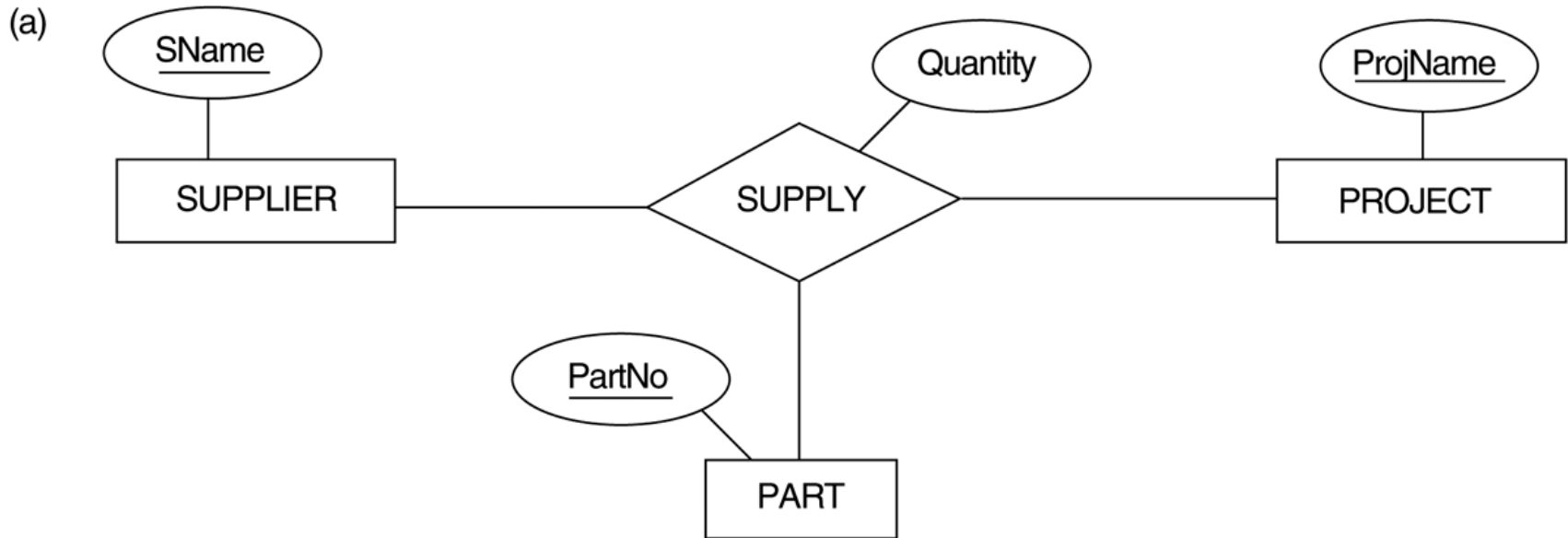
- Step 6 Result



ER-to-Relational Mapping Algorithm (contd.)

- **Step 7: Mapping of N-ary Relationship Types.**
 - For each n-ary relationship type R, where $n > 2$, create a new relation S to represent R.
 - Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types.
 - 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-diagram 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, Part_no, Proj_name}

ER-to-Relational Mapping Algorithm (contd.)



ER-to-Relational Mapping Algorithm (contd.)

SUPPLIER

<u>SNAME</u>	...
--------------	-----

PROJECT

<u>PROJNAME</u>	...
-----------------	-----

PART

<u>PARTNO</u>	...
---------------	-----

SUPPLY

<u>SNAME</u>	PROJNAME	<u>PARTNO</u>	QUANTITY
--------------	----------	---------------	----------

Summary of Mapping constructs and constraints

Correspondence between ER and Relational Models

ER Model

Entity type

1:1 or 1:N relationship type

M:N relationship type

n -ary relationship type

Simple attribute

Composite attribute

Multivalued attribute

Value set

Key attribute

Relational Model

“Entity” relation

Foreign key (or “relationship” relation)

“Relationship” relation and two foreign keys

“Relationship” relation and n foreign keys

Attribute

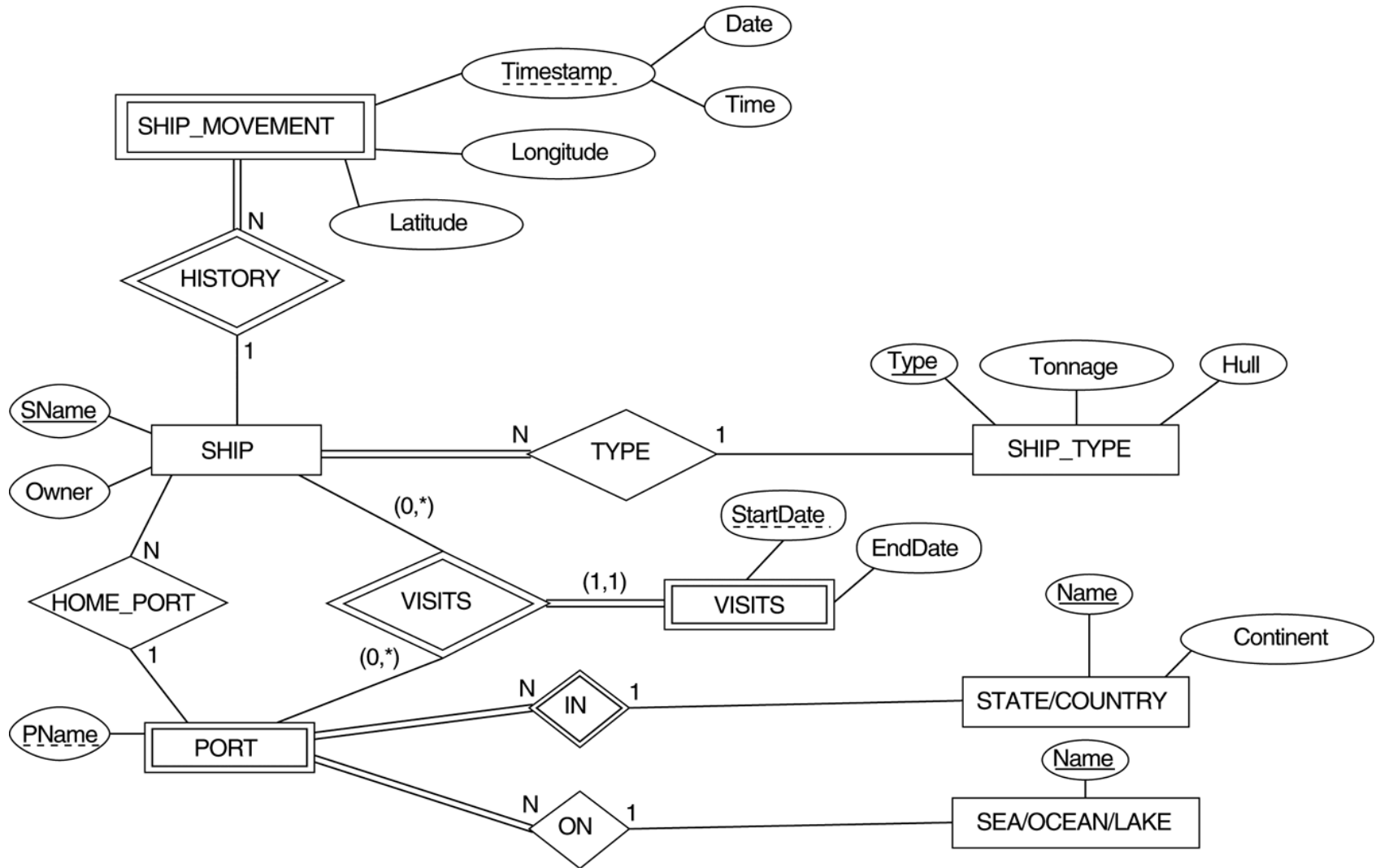
Set of simple component attributes

Relation and foreign key

Domain

Primary (or secondary) key

Exercise: An ER schema for a SHIP_TRACKING database. Convert to relational schema



Thank you