

MODULE – 2

RELATIONAL MODEL

RELATIONAL ALGEBRA

MAPPING CONCEPTUAL DESIGN INTO A LOGICAL DESIGN

SQL

Relational Model Concepts

2

- 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.

Informal Definitions

3

□ RELATION: A table of values

- ▣ A relation may be thought of as a **set of rows**.
- ▣ A relation may alternately be thought of as a **set of columns**.
- ▣ Each row represents a fact that corresponds to a real-world **entity** or **relationship**.
- ▣ Each row has a value of an item or set of items that uniquely identifies that row in the table.
- ▣ Sometimes row-ids or sequential numbers are assigned to identify the rows in the table.
- ▣ Each column typically is called by its column name or column header or attribute name.

Domains, Attributes, Tuples and Relations

4

- Domain
 - ▣ A **domain** D is a set of atomic values.
 - ▣ Atomic means that each value in the domain is indivisible as far as the relational model is concerned.
- Relation Schema
 - ▣ **Relation schema** R , denoted by $R (A_1, A_2, \dots, A_n)$ is made up of a relation name R and a list of attributes A_1, A_2, \dots, A_n
 - ▣ Each **attribute** A_i is a name of a role played by some domain D in the relation schema R .
 - ▣ D is called the domain of A_i and is denoted by **dom**(A_i).
- Degree of the relation
 - ▣ The **degree** or **arity** of a relation is the number of attributes n of its relation schema.

Contd...

□ Relation

- A **relation** or **relation state** r of the relation schema $R (A_1, A_2, \dots, A_n)$, also denoted by $r(R)$, is a set of n – tuples $r = \{t_1, t_2, \dots, t_m\}$.
- Each **n – tuple** t is an ordered list of n values $t = \langle v_1, v_2, \dots, v_n \rangle$, where each $v_i, 1 \leq i \leq n$, is an element of $\text{dom}(A_i)$ or is a special NULL value.

- The terms relation intension for the relation schema R and relation extension for a relation state $r(R)$ are commonly used.

Mathematical Definition

6

- A relation schema or relation state $r(R)$ is a mathematical relation of degree n on the domains $\text{dom}(A_1), \text{dom}(A_2), \dots, \text{dom}(A_n)$ which is a subset of the Cartesian product of the domains that defines R :

$$r(R) \subseteq (\text{dom}(A_1) \times \text{dom}(A_2) \times \dots \times \text{dom}(A_n))$$

- The Cartesian product specifies all possible combinations of values from the underlying domains.
- The total number of values, or cardinality in a domain D by $|D|$, the total number of tuples in the Cartesian product is

$$|\text{dom}(A_1)| \times |\text{dom}(A_2)| \times \dots \times |\text{dom}(A_n)|$$

- Current relation state
 - A relation state at a given time is called the current relation state.

Example...

7

The diagram illustrates a relation table with the following structure and data:

STUDENT	Name	SSN	HomePhone	Address	OfficePhone	Age	GPA
	Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	null	19	3.21
	Katherine Ashly	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
	Charles Cooper	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

Annotations in the diagram:

- Relation name:** An arrow points from the label "Relation name" to the "STUDENT" header cell.
- Attributes:** An arrow points from the label "Attributes" to the header row, with multiple lines indicating the specific attributes: Name, SSN, HomePhone, Address, OfficePhone, Age, and GPA.
- Tuples:** An arrow points from the label "Tuples" to the data rows, indicating the individual records in the relation.

Characteristics of Relations

8

- Ordering of tuples in a relation
- Ordering of values within a tuple and an Alternative definition of a relation
- Values and NULLs in tuples
- Interpretation (Meaning) of a relation

Ordering of Tuples in a Relation

9

- A relation is a set of tuples.
 - ❖ The tuples are not considered to be ordered, even though they appear to be in the tabular form.
- Relation is not sensitive to the ordering of tuples.

Ordering of values within a tuple and an Alternative definition of a relation

10

- From the definition of the relation, an n – tuple is a ordered list of values, so the ordering of values in a tuple is important.

- Alternative Definition
 - ▣ A relation schema $R = \{A_1, A_2, \dots, A_n\}$ is a set of attributes, and a relation state $r(R)$ is a finite set of mappings $r = \{t_1, t_2, \dots, t_m\}$, where each tuple t_i is a mapping from R to D , and D is the union of the attribute domains; that is, $D = \text{dom}(A_1) \cup \text{dom}(A_2) \cup \dots \cup \text{dom}(A_n)$.

Values and NULLs in the Tuples

11

- All values are considered *atomic* (indivisible).
- This model sometimes called as the flat relational model.
- A special **NULL** value is used to represent values that are unknown or inapplicable to certain tuples.

Interpretation (Meaning) of a Relation

- The relation schema can be interpreted as a declaration or a type of **assertion**.
- For example, the schema of the student entity has a Name, SSN, HomePhone, Address, Officephone, Age and GPA. Each tuple in the relation can then be interpreted as a fact or a particular instance of the relation.

Example...

13

STUDENT	Name	SSN	HomePhone	Address	OfficePhone	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
	Charles Cooper	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
	Katherine Ashly	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

Relational Model Notations

14

- Following notations are used in the Relational Model:
 - ▣ A relation schema R of degree n is denoted by $R(A_1, A_2, \dots, A_n)$.
 - ▣ The letters Q, R, S denote relation names.
 - ▣ The letters q, r, s denote relation states.
 - ▣ The letters t, u, v denote tuples.

Relational Model Constraints and Relational Database Schemas

15

- Constraints on databases can generally be divided into three main categories:
 - ▣ Inherent model based or Implicit constraint
 - Constraints that are inherent to the data model
 - ▣ Schema based or Explicit constraint
 - Constraints that are directly expressed in schemas of the data model
 - ▣ Application based or Semantic constraint or Business rules
 - Constraints that can not be directly expressed in schemas of the data model and hence must be expressed and enforced by the application programs

Constraints in the Relational Model

16

- Domain Constraints
- Key Constraints and Constraint on NULL values
- Entity integrity constraints
- Referential integrity constraints

Domain Constraints

17

- These constraints specify that within each tuple, the value of each attribute A must be an atomic value from the domain $\text{dom}(A)$.
- The data types associated with domains typically include standard numeric data types for integers and real numbers.

Key Constraints and Constraints on NULL values

18

- Super Key
 - It is a subset of attributes SK where any two distinct tuples t_1 and t_2 in a relation state r of R , we have the constraint that $t_1[SK] \neq t_2[SK]$

- Key
 - A key K of a relation schema R is a super key of R with the additional property that removing any attribute A from K leaves set of attributes K' that is not a super key of R any more.

- Candidate Key
 - A relation schema may have more than one key. In this case, each of the keys is called a candidate key.

- Primary Key
 - This is the candidate key whose values are used to identify tuples in the relation.
 - The primary key of the relation schema are underlined.

Relational Databases and Relational Database Schemas

19

- Relational Database Schema
 - ▣ A relational database schema S is a set of relation schemas $S = \{R_1, R_2, \dots, R_m\}$ and a set of integrity constraints IC .

- Relational Database State
 - ▣ A relational database state DB of S is a set of relation states $DB = \{r_1, r_2, \dots, r_m\}$ 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 database state that does not obey all the integrity constraints is called an **invalid state**, and a state that satisfies all the constraints in IC is called a **valid state**.

Entity Integrity Constraints

20

- The entity integrity constraint states that no primary key can be NULL.
- This is because the primary key value is used to identify individual tuples in a relation.
- Having NULL values for the primary key implies that we can not identify some tuples.
- For example, if two or more tuples had NULL for their primary keys, we might not be able to distinguish them if we tried to reference them from other relations.

Referential Integrity Constraints

- The referential integrity constraint is specified between two relations and is used to maintain the consistency among tuples in the two relations.
- Informally, the referential integrity constraint states that a tuple in one relation that refers to another relation must refer to an existing tuple in that relation.
- For example, the DNO of EMPLOYEE gives the department number for which each employee works; hence, its value in every EMPLOYEE tuple must match the DNUMBER value of some tuple in the DEPARTMENT relation.

Foreign Key

22

- The condition for a foreign key specify a referential integrity constraint between the two relation schemas R_1 and R_2 .
- A set of attributes FK in the relation schema R_1 is a foreign key of R_1 that references relation R_2 if it satisfies the following rules:
 - ▣ 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 .
 - ▣ The 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 $t_1[\text{FK}] = t_2[\text{PK}]$, 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.

Update Operations and Dealing with Constraint Violations

23

- The operations of the relational model can be categorized into retrievals and update.

- There are three basic update operations:
 - Insert
 - Delete
 - Modify

The Insert Operation

- The insert operation provides a list of attribute values for a new tuple t that is to be inserted into a relation R .
- Insert can violate any of the four types of constraints.

Example...

25

1. Insert <'Cecilia', 'F', 'Kolonsky', null, '1960-04-05', '6357 Windy Lane, Katy, TX', F, 28000, null, 4> into EMPLOYEE.
2. Insert <'Alicia', 'J', 'Zelaya', '999887777', '1960-04-05', '6357 Windy Lane, Katy, TX', F, 28000, '987654321', 4> into EMPLOYEE.
3. Insert <'Cecilia', 'F', 'Kolonsky', '677678989', '1960-04-05', '6357 Windswept, Katy, TX', F, 28000, '987654321', 7> into EMPLOYEE.
4. Insert <'Cecilia', 'F', 'Kolonsky', '677678989', '1960-04-05', '6357 Windy Lane, Katy, TX', F, 28000, null, 4> into EMPLOYEE.

The Delete Operation

26

- The delete operation is used to delete the tuples from the relation.
- Eg:
- Delete the WORKS_ON tuple with ESSN = '999887777' and PNO = 10.
- Delete the EMPLOYEE tuple with SSN = '999887777'.
- Delete the EMPLOYEE tuple with SSN = '333445555'.

The Update Operation

27

- The update(or Modify) operation is used to change the values of one or more attributes in a tuple(or tuples) of some relation R.
- Eg:
- Update the SALARY of the EMPLOYEE tuple with SSN = '999887777' to 28000.
- Update the DNO of the EMPLOYEE tuple with SSN = '999887777' to 1.
- Update the DNO of the EMPLOYEE tuple with SSN = '999887777' to 7.
- Update the SSN of the EMPLOYEE tuple with SSN = '999887777' to '987654321'.

The Transaction Concept

28

- A database application program running against a relational database typically runs a series of transactions.
- A transaction involves reading from the database as well as doing insertions, deletions and updates to existing values in the database.
- These transactions must leave the database in a consistent state.
- A single transaction may involve any number of retrieval operations that reads from the database and any number of update operations.
- A large number of commercial applications running against relational databases in the Online Transaction Processing(OLTP) systems are executing transactions at rates several hundreds per second.

Relational Algebra Overview

29

- 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)

30

- 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)

Relational Algebra Overview

31

- 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)

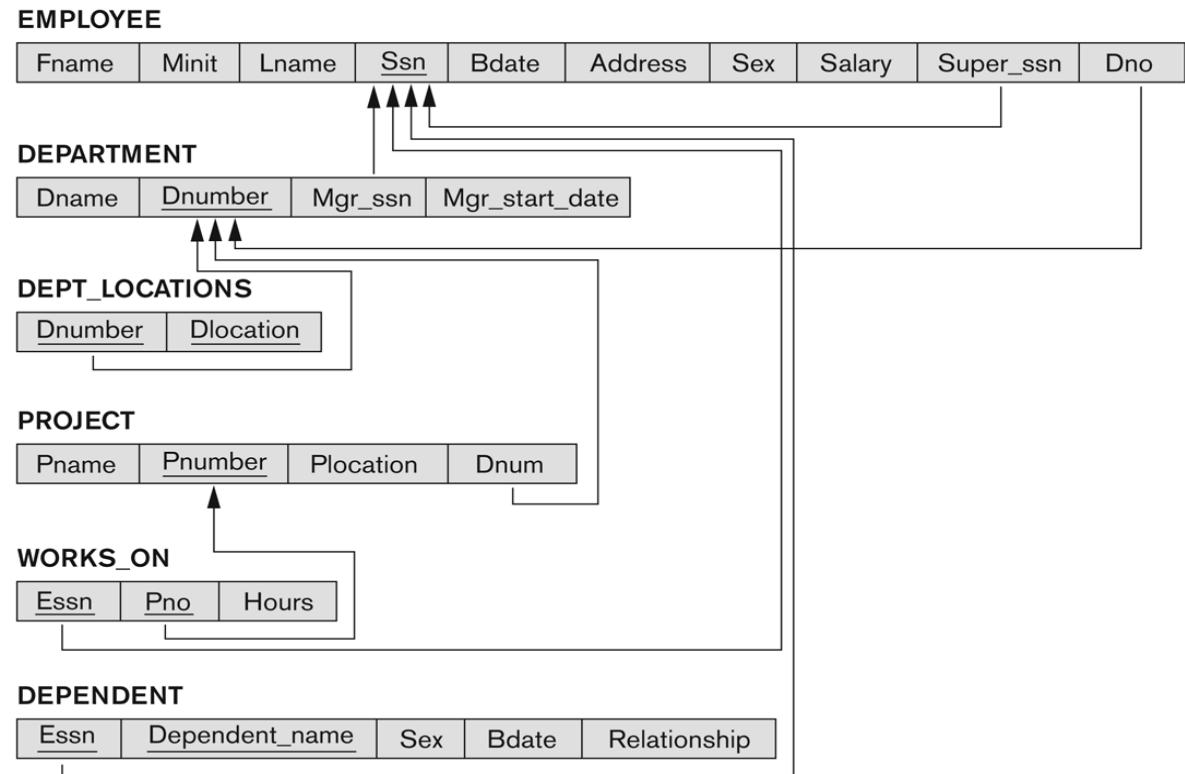
Database State for COMPANY

32

- All examples discussed below refer to the COMPANY database shown here.

Figure 5.7

Referential integrity constraints displayed on the COMPANY relational database schema.



Unary Relational Operations: SELECT

33

- 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:

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

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

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

Unary Relational Operations: SELECT

34

- In general, the *select* operation is denoted by σ $\langle \text{selection condition} \rangle (R)$ 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 (contd.)

35

□ 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:

$$\blacksquare \sigma_{\langle \text{condition1} \rangle}(\sigma_{\langle \text{condition2} \rangle}(R)) = \sigma_{\langle \text{condition2} \rangle}(\sigma_{\langle \text{condition1} \rangle}(R))$$

□ Because of commutativity property, a cascade (sequence) of SELECT operations may be applied in any order:

$$\blacksquare \sigma_{\langle \text{cond1} \rangle}(\sigma_{\langle \text{cond2} \rangle}(\sigma_{\langle \text{cond3} \rangle}(R))) = \sigma_{\langle \text{cond2} \rangle}(\sigma_{\langle \text{cond3} \rangle}(\sigma_{\langle \text{cond1} \rangle}(R)))$$

□ A cascade of SELECT operations may be replaced by a single selection with a conjunction of all the conditions:

$$\blacksquare \sigma_{\langle \text{cond1} \rangle}(\sigma_{\langle \text{cond2} \rangle}(\sigma_{\langle \text{cond3} \rangle}(R))) = \sigma_{\langle \text{cond1} \rangle \text{ AND } \langle \text{cond2} \rangle \text{ AND } \langle \text{cond3} \rangle}(R))$$

□ The number of tuples in the result of a SELECT is less than (or equal to) the number of tuples in the input relation R

The following query results refer to this database state

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

37

- 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 (cont.)

38

- 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 (contd.)

39

□ 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 $\langle \text{list1} \rangle$

Examples of applying SELECT and PROJECT operations

Figure 6.1

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

(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

41

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

Single expression versus sequence of relational operations (Example)

42

- 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

43

- 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.)

44

- 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.)

45

- 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:
 - $\text{RESULT (F, M, L, S, B, A, SX, SAL, SU, DNO)} \leftarrow \pi_{\text{FNAME, LNAME, SALARY}} (\text{DEP5_EMPS})$
 - 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

46

(a)

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

(b)

TEMP

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston,TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston,TX	M	40000	888665555	5
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 6.2

Results of a sequence of operations.

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

(b) Using intermediate relations and renaming of attributes.

Relational Algebra Operations from Set Theory: UNION

47

□ 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

48

□ Example:

- To retrieve the social security numbers of all employees who either work in department 5 (RESULT1 below) or directly supervise an employee who works in department 5 (RESULT2 below)
- We can use the UNION operation as follows:

$$\text{DEP5_EMPS} \leftarrow \sigma_{\text{DNO}=5} (\text{EMPLOYEE})$$
$$\text{RESULT1} \leftarrow \pi_{\text{SSN}}(\text{DEP5_EMPS})$$
$$\text{RESULT2}(\text{SSN}) \leftarrow \pi_{\text{SUPERSSN}}(\text{DEP5_EMPS})$$
$$\text{RESULT} \leftarrow \text{RESULT1} \cup \text{RESULT2}$$

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

Example of the result of a UNION operation

49

□ UNION Example

Figure 6.3

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

RESULT1

Ssn
123456789
333445555
666884444
453453453

RESULT2

Ssn
333445555
888665555

RESULT

Ssn
123456789
333445555
666884444
453453453
888665555

Relational Algebra Operations from Set Theory

50

- Type Compatibility of operands is required for the binary set operation UNION \cup , (also for INTERSECTION \cap , and SET DIFFERENCE $-$, see next slides)
- $R1(A1, A2, \dots, An)$ 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

51

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

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

52

- 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 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

Figure 6.4

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

54

- 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

55

- 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.)

56

- 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.)

57

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

Example of applying CARTESIAN PRODUCT

Figure 6.5
The CARTESIAN PRODUCT (CROSS PRODUCT) operation.

FEMALE_EMPS

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

EMP_NAMES

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

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	...

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

Binary Relational Operations: JOIN

59


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

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

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

Binary Relational Operations: JOIN (cont.)

60

- 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  operation.
 - ▣ $\text{DEPT_MGR} \leftarrow \text{DEPARTMENT} \underset{\text{MGRSSN=SSN}}{\bowtie} \text{EMPLOYEE}$
- MGRSSN=SSN is the join condition
 - ▣ Combines each department record with the employee who manages the department
 - ▣ The join condition can also be specified as DEPARTMENT.MGRSSN=EMPLOYEE.SSN

Example of applying the JOIN operation

61

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	...

Figure 6.6

Result of the JOIN operation

Some properties of JOIN

62


- Consider the following JOIN operation:

$$\begin{array}{ccc} \square & R(A_1, A_2, \dots, A_n) & \bowtie & S(B_1, B_2, \dots, B_m) \\ & & & R.A_i = S.B_j \end{array}$$

- 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

63

- The general case of JOIN operation is called a Theta-join: $R \bowtie_{\theta} S$

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

Binary Relational Operations: EQUIJOIN

64

- 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

65

- 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.)

66

- 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:
 - $R.C = S.C$ 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

67

(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

Figure 6.7

Results of two NATURAL JOIN operations.

(a) PROJ_DEPT ← PROJECT * DEPT.

(b) DEPT_LOCS ← DEPARTMENT * DEPT_LOCATIONS.

Complete Set of Relational Operations

68

- 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:
 - $R \cap S = (R \cup S) - ((R - S) \cup (S - R))$
 - $R \bowtie_{\langle \text{join condition} \rangle} S = \sigma_{\langle \text{join condition} \rangle} (R \times S)$

Binary Relational Operations: DIVISION

69

□ 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

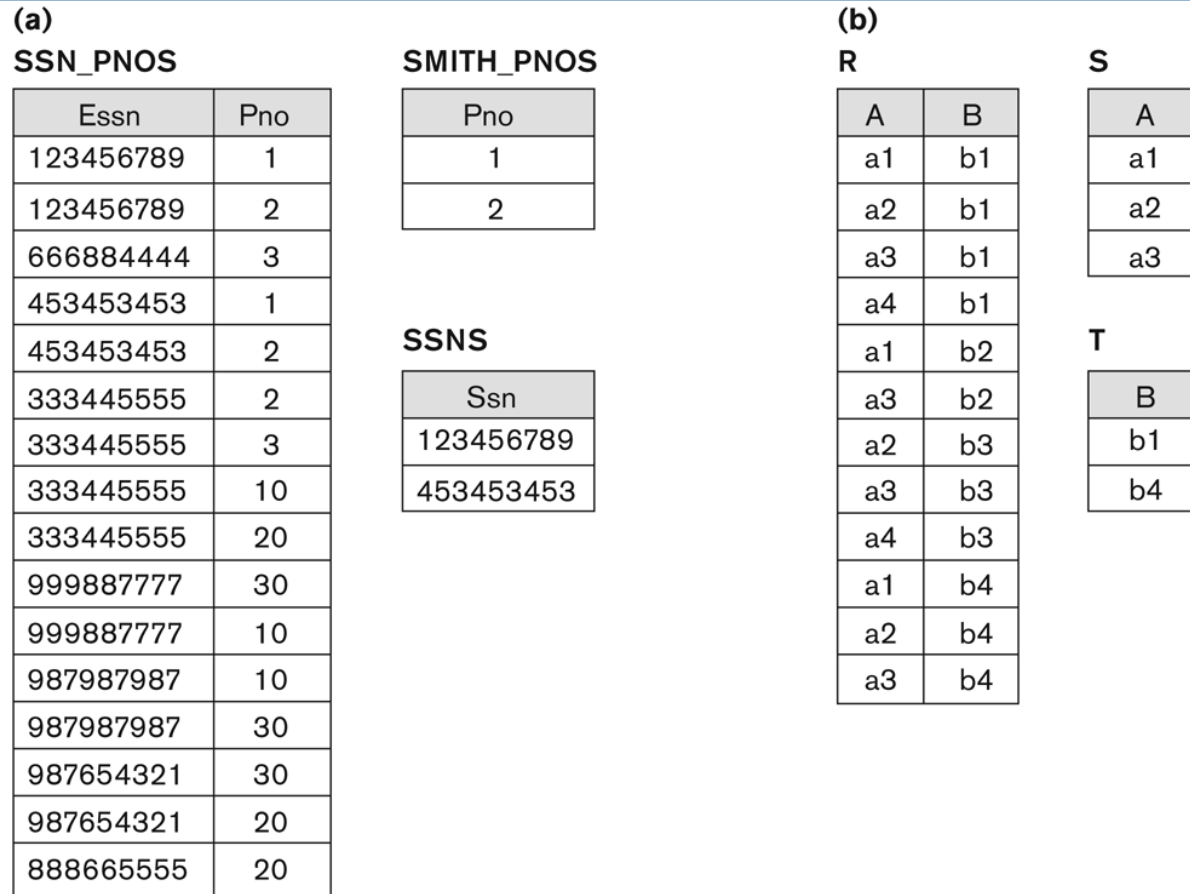


Figure 6.8

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

Recap of Relational Algebra Operations

Table 6.1
Operations of Relational Algebra

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

Additional Relational Operations: Aggregate Functions and Grouping

72

- 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

73

- 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

74

- 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

Figure 6.10

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

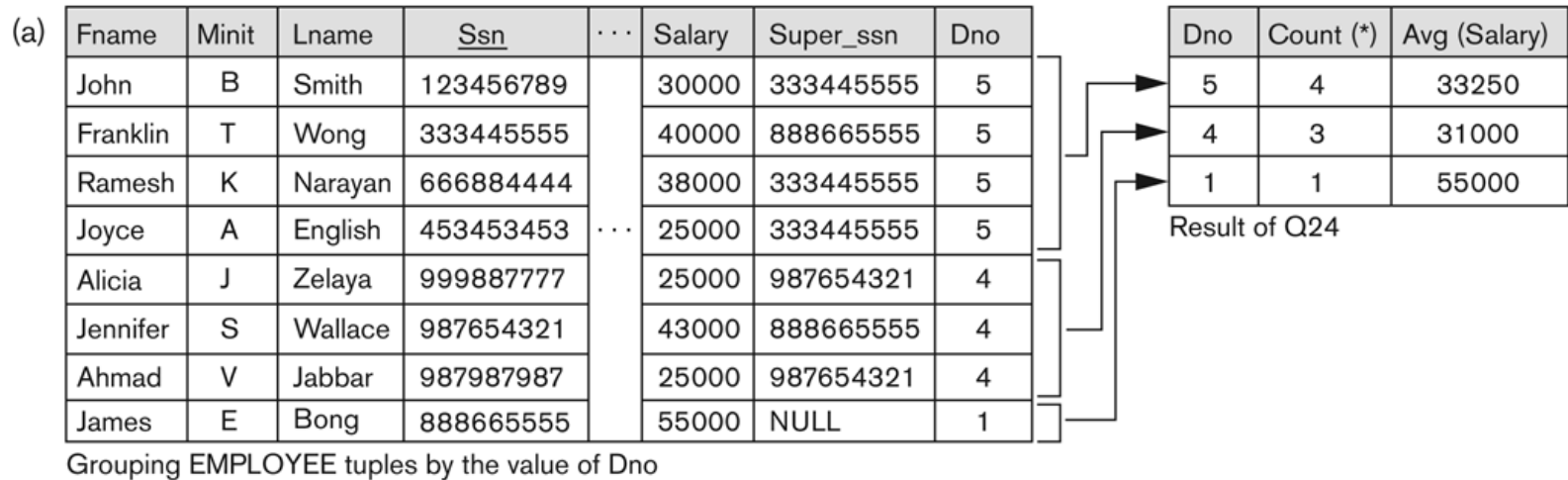
(c)

Count_ssn	Average_salary
8	35125

Illustrating aggregate functions and grouping

Figure 8.6

Results of GROUP BY and HAVING. (a) Q24. (b) Q26.



Additional Relational Operations (cont.)

77

- 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, 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.

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.)

80

- 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.)

81

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

Additional Relational Operations (cont.)

82

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

Figure 6.12
The result of a
LEFT OUTER JOIN
operation.

Additional Relational Operations (cont.)

83

□ 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.)

84

- 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)

Examples of Queries in Relational Algebra

85

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

$\text{RESEARCH_DEPT} \leftarrow \sigma_{\text{DNAME}='Research'}(\text{DEPARTMENT})$

$\text{RESEARCH_EMPS} \leftarrow (\text{RESEARCH_DEPT} \bowtie_{\text{DNUMBER}=\text{DNOEMPLOYEE}} \text{EMPLOYEE})$

$\text{RESULT} \leftarrow \pi_{\text{FNAME}, \text{LNAME}, \text{ADDRESS}}(\text{RESEARCH_EMPS})$

- **Q6: Retrieve the names of employees who have no dependents.**

$\text{ALL_EMPS} \leftarrow \pi_{\text{SSN}}(\text{EMPLOYEE})$

$\text{EMPS_WITH_DEPS}(\text{SSN}) \leftarrow \pi_{\text{ESSN}}(\text{DEPENDENT})$

$\text{EMPS_WITHOUT_DEPS} \leftarrow (\text{ALL_EMPS} - \text{EMPS_WITH_DEPS})$

$\text{RESULT} \leftarrow \pi_{\text{LNAME}, \text{FNAME}}(\text{EMPS_WITHOUT_DEPS} * \text{EMPLOYEE})$

Relational Database Design Using ER – to – Relational Mapping

86

1. Mapping of Regular Entity Types
2. Mapping of Weak Entity Types
3. Mapping of Binary 1:1 Relationship Types
4. Mapping of Binary 1:N Relationship Types
5. Mapping of Binary M:N Relationship Types
6. Mapping of Multivalued Attributes
7. Mapping of N – ary Relationship Types

Mapping of Regular Entity Types

- For each regular(strong) entity type E in the ER schema, create a relation R that includes all simple attributes of E.
- Include only the simple component attributes of a composite attribute.
- Choose one of the key attributes of E as the primary key of R.
- If the chosen key of E is a composite, then the set of simple attributes that form it will together form the primary key of R.
- If multiple keys were identified for E during the conceptual design, the information describing the attributes that form each additional key is kept in order to specify secondary keys of relation R.

Mapping of Weak Entity Types

88

- For each weak entity type W in the ER schema with owner entity type E , create a relation R and include all simple attributes of W as attributes of R .
- In addition, include as foreign key attributes of R , the primary key attribute(s) of the relation(s) that correspond to the owner entity type(s).
- The primary key of R is the combination of the primary key(s) of the owner(s) and the partial key of the weak entity type W , if any.
- If there is a weak entity type E_2 whose owner is also a weak entity type E_1 , then E_1 should be mapped before E_2 to determine its primary key first.

Mapping of Binary 1:1 Relationship Types

- For each binary 1:1 relationship type R in ER schema, identify the relations S and T that correspond to the entity types participating in R .

- There are three possible approaches:
 - ▣ The foreign key approach
 - ▣ The merged relationship approach
 - ▣ The cross reference or relationship relation approach

Mapping of Binary 1:N Relationship Types

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

Mapping of Binary M:N Relationship Types

- For each binary M:N relationship type R, create a new relation S to represent R.
- Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types.
- Their combination will form the primary key of S.
- Also include any simple attributes of the M:N relationship type as attributes of S.

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 foreign key in R – of the relation that represents the entity type or the relationship type that has A as an attribute.
- The primary key of R is the combination of A and K .

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 as attributes of S.
- The primary key of S is usually a combination of all the foreign keys that reference the relations representing the participating entity types.

Data Definition, Constraints, and Schema Changes

94

- Used to CREATE, DROP, and ALTER the descriptions of the tables (relations) of a database

CREATE TABLE

95

- Specifies a new base relation by giving it a name, and specifying each of its attributes and their data types (INTEGER, FLOAT, DECIMAL(i,j), CHAR(n), VARCHAR(n))
- A constraint NOT NULL may be specified on an attribute

```
CREATE TABLE DEPARTMENT (  
    DNAME                VARCHAR(10)    NOT  
NULL,  
    DNUMBER              INTEGER        NOT  
NULL,  
    MGRSSN               CHAR(9) ,  
    MGRSTARTDATE        CHAR(9)    ) ;
```

CREATE TABLE

96

- In SQL2, can use the CREATE TABLE command for specifying the primary key attributes, secondary keys, and referential integrity constraints (foreign keys).
- Key attributes can be specified via the PRIMARY KEY and UNIQUE phrases

```
CREATE TABLE DEPT (  
    DNAME                VARCHAR(10)        NOT  
    NULL,  
    DNUMBER              INTEGER            NOT  
    NULL,  
    MGRSSN               CHAR(9) ,  
    MGRSTARTDATE        CHAR(9) ,  
    PRIMARY KEY (DNUMBER) ,  
    UNIQUE (DNAME) ,  
    FOREIGN KEY (MGRSSN) REFERENCES EMP (ENUM)  
);
```


DATA TYPES

97

- Numeric – INTEGER, INT, SMALLINT,
FLOAT, REAL, DOUBLE PRECISION
- DECIMAL(i,j), NUMERIC(i,j)
- Character – CHAR(n), VARCHAR(n)
- Bit string – BIT(n), BIT VARYING(n)
- Boolean – TRUE, FALSE
- Other – DATE, TIME, TIMESTAMP

DROP TABLE

98

- Used to remove a relation (base table) and its definition
- The relation can no longer be used in queries, updates, or any other commands since its description no longer exists
- Example:

```
DROP TABLE DEPENDENT ;
```

```
DROP TABLE DEPENDENT CASCADE ;
```

```
DROP TABLE DEPT RESTRICT ;
```

ALTER TABLE

99

- Used to add an attribute to one of the base relations
 - ▣ The new attribute will have NULLs in all the tuples of the relation right after the command is executed; hence, the NOT NULL constraint is not allowed for such an attribute
- Example:
**ALTER TABLE EMPLOYEE ADD JOB
VARCHAR(12);**
- The database users must still enter a value for the new attribute JOB for each EMPLOYEE tuple.
 - ▣ This can be done using the UPDATE command.

- ❑ ALTER TABLE EMPLOYEE DROP COLUMN JOB CASCADE;
- ❑ ALTER TABLE EMPLOYEE ALTER COLUMN EMP_ID SET DEFAULT '1111';
- ❑ ALTER TABLE EMPLOYEE ALTER COLUMN EMP_ID DROP DEFAULT;
- ❑ ALTER TABLE EMPLOYEE DROP CONSTRAINT EMP_PK1 CASCADE;

Features Added in SQL2 and SQL-99

101

- Create schema
- Referential integrity options

CREATE SCHEMA

102

- Specifies a new database schema by giving it a name

REFERENTIAL INTEGRITY OPTIONS

103

- We can specify RESTRICT, CASCADE, SET NULL or SET DEFAULT on referential integrity constraints (foreign keys)

```
CREATE TABLE DEPT (  
    DNAME          VARCHAR(10)          NOT NULL,  
    DNUMBER        INTEGER              NOT  
NULL,  
    MGRSSN         CHAR(9) ,  
    MGRSTARTDATE   CHAR(9) ,  
    PRIMARY KEY (DNUMBER) ,  
    UNIQUE (DNAME) ,  
    FOREIGN KEY (MGRSSN) REFERENCES EMP  
ON DELETE SET DEFAULT ON UPDATE  
CASCADE) ;
```

REFERENTIAL INTEGRITY OPTIONS

(continued)

104

```
CREATE TABLE EMP (  
    ENAME          VARCHAR(30)      NOT NULL,  
    ESSN           CHAR(9) ,  
    BDATE          DATE ,  
    DNO            INTEGER  DEFAULT 1 ,  
    SUPERSSN       CHAR(9) ,  
    PRIMARY KEY   (ESSN) ,  
    FOREIGN KEY   (DNO) REFERENCES DEPT  
        ON DELETE SET DEFAULT ON UPDATE  
        CASCADE ,  
    FOREIGN KEY   (SUPERSSN) REFERENCES EMP  
        ON DELETE SET NULL ON UPDATE CASCADE) ;
```


Constraints

105

- CHECK
- DNUM INT NOT NULL CHECK(DNUM>0 AND DNUM<5)
- DOMAIN
- CREATE DOMAIN DN AS INTERGER CHECK (DN >0 AND DN<10)

Additional Data Types in SQL2 and SQL-99

106

Has DATE, TIME, and TIMESTAMP data types

- **DATE:**

- Made up of year-month-day in the format yyyy-mm-dd

- **TIME:**

- Made up of hour:minute:second in the format hh:mm:ss

- **TIME(i):**

- Made up of hour:minute:second plus i additional digits specifying fractions of a second
- format is hh:mm:ss:ii...i

Additional Data Types in SQL2 and SQL-99 (contd.)

107

□ **TIMESTAMP:**

- Has both DATE and TIME components

□ **INTERVAL:**

- Specifies a relative value rather than an absolute value
- Can be DAY/TIME intervals or YEAR/MONTH intervals
- Can be positive or negative when added to or subtracted from an absolute value, the result is an absolute value

Retrieval Queries in SQL

- SQL has one basic statement for retrieving information from a database; the **SELECT** statement
 - ▣ This is *not the same* as the SELECT operation of the relational algebra
- Important distinction between SQL and the formal relational model:
 - ▣ SQL allows a table (relation) to have two or more tuples that are identical in all their attribute values
 - ▣ Hence, an SQL relation (table) is a **multi-set** (sometimes called a **bag**) of tuples; it is *not* a set of tuples
- SQL relations can be constrained to be sets by specifying PRIMARY KEY or UNIQUE attributes, or by using the DISTINCT option in a query

Retrieval Queries in SQL (contd.)

109

- A **bag** or **multi-set** is like a set, but an element may appear more than once.
 - ▣ Example: $\{A, B, C, A\}$ is a bag. $\{A, B, C\}$ is also a bag that also is a set.
 - ▣ Bags also resemble lists, but the order is irrelevant in a bag.
- Example:
 - ▣ $\{A, B, A\} = \{B, A, A\}$ as bags
 - ▣ However, $[A, B, A]$ is not equal to $[B, A, A]$ as lists

Retrieval Queries in SQL (contd.)

110

- Basic form of the SQL SELECT statement is called a *mapping* or a SELECT-FROM-WHERE *block*

SELECT <attribute list>
FROM <table list>
WHERE <condition>

- <attribute list> is a list of attribute names whose values are to be retrieved by the query
- <table list> is a list of the relation names required to process the query
- <condition> is a conditional (Boolean) expression that identifies the tuples to be retrieved by the query

Relational Database Schema--Figure 5.5

111

EMPLOYEE

FNAME	MINIT	LNAME	<u>SSN</u>	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
-------	-------	-------	------------	-------	---------	-----	--------	----------	-----

DEPARTMENT

DNAME	<u>DNUMBER</u>	MGRSSN	MGRSTARTDATE
-------	----------------	--------	--------------

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
-------------	-----------------------	-----	-------	--------------

Populated Database--Fig.5.6

EMPLOYEE	FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	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	1968-07-19	3321 Castle, Spring, TX	F	25000	987654321	4
	Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
	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

DEPT_LOCATIONS	DNUMBER	DLOCATION
	1	Houston
	4	Stafford
	5	Bellaire
	5	Sugarland
	5	Houston

DEPARTMENT	DNAME	DNUMBER	MGRSSN	MGRSTARTDATE
	Research	5	333445555	1988-05-22
	Administration	4	987654321	1995-01-01
	Headquarters	1	888665555	1981-06-19

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

Simple SQL Queries

113

- Basic SQL queries correspond to using the following operations of the relational algebra:
 - SELECT
 - PROJECT
 - JOIN
- All subsequent examples use the COMPANY database

Simple SQL Queries (contd.)

114

- Example of a simple query on one relation
- Query 0: Retrieve the birthdate and address of the employee whose name is 'John B. Smith'.

```
Q0: SELECT BDATE, ADDRESS
      FROM      EMPLOYEE
      WHERE FNAME='John' AND MINIT='B'
      AND      LNAME='Smith'
```

- ▣ Similar to a SELECT-PROJECT pair of relational algebra operations:
 - The SELECT-clause specifies the projection attributes and the WHERE-clause specifies the selection condition
- ▣ However, the result of the query may contain duplicate tuples

Simple SQL Queries (contd.)

115

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

```
Q1: SELECT FNAME, LNAME, ADDRESS
      FROM      EMPLOYEE, DEPARTMENT
      WHERE DNAME='Research' AND DNUMBER=DNO
```

- ▣ Similar to a SELECT-PROJECT-JOIN sequence of relational algebra operations
- ▣ (DNAME='Research') is a selection condition (corresponds to a SELECT operation in relational algebra)
- ▣ (DNUMBER=DNO) is a join condition (corresponds to a JOIN operation in relational algebra)

Simple SQL Queries (contd.)

116

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

```
Q2: SELECT      PNUMBER, DNUM, LNAME, BDATE, ADDRESS
      FROM        PROJECT, DEPARTMENT, EMPLOYEE
      WHERE  DNUM=DNUMBER AND MGRSSN=SSN
            AND PLOCATION='Stafford'
```

- ▣ In Q2, there are two join conditions
- ▣ The join condition `DNUM=DNUMBER` relates a project to its controlling department
- ▣ The join condition `MGRSSN=SSN` relates the controlling department to the employee who manages that department

Aliases, * and DISTINCT, Empty WHERE-clause

117

- In SQL, we can use the same name for two (or more) attributes as long as the attributes are in *different relations*
- A query that refers to two or more attributes with the same name must *qualify* the attribute name with the relation name by *prefixing* the relation name to the attribute name
- Example:
- **EMPLOYEE.LNAME, DEPARTMENT.DNAME**

ALIASES

118

- Some queries need to refer to the same relation twice
 - ▣ In this case, *aliases* are given to the relation name
- Query 8: For each employee, retrieve the employee's name, and the name of his or her immediate supervisor.

```
Q8:  SELECT  E.FNAME, E.LNAME, S.FNAME, S.LNAME
      FROM    EMPLOYEE E S
      WHERE   E.SUPERSSN=S.SSN
```

- ▣ In Q8, the alternate relation names E and S are called *aliases* or *tuple variables* for the EMPLOYEE relation
- ▣ We can think of E and S as two different *copies* of EMPLOYEE; E represents employees in role of *supervisees* and S represents employees in role of *supervisors*

ALIASES (contd.)

119

- Aliasing can also be used in any SQL query for convenience
- Can also use the AS keyword to specify aliases

```
Q8:      SELECT      E.FNAME, E.LNAME,  
            S.FNAME, S.LNAME  
FROM EMPLOYEE AS E,  
EMPLOYEE AS S  
WHERE     E.SUPERSSN=S.SSN
```

UNSPECIFIED WHERE-clause

120

- A *missing WHERE-clause* indicates no condition; hence, all tuples of the relations in the FROM-clause are selected
 - ▣ This is equivalent to the condition WHERE TRUE
- Query 9: Retrieve the SSN values for all employees.

- ▣ Q9: SELECT SSN
 FROM EMPLOYEE

- If more than one relation is specified in the FROM-clause *and* there is no join condition, then the *CARTESIAN PRODUCT* of tuples is selected

UNSPECIFIED

WHERE-clause (contd.)

121

□ Example:

```
Q10:      SELECT      SSN, DNAME  
          FROM EMPLOYEE, DEPARTMENT
```

- It is extremely important not to overlook specifying any selection and join conditions in the WHERE-clause; otherwise, incorrect and very large relations may result

USE OF *

122

- To retrieve all the attribute values of the selected tuples, a * is used, which stands for *all the attributes*

Examples:

```
Q1C:    SELECT *
        FROM      EMPLOYEE
        WHERE DNO=5
```

```
Q1D:    SELECT *
        FROM      EMPLOYEE, DEPARTMENT
        WHERE DNAME='Research' AND
              DNO=DNUMBER
```

USE OF DISTINCT

123

- SQL does not treat a relation as a set; duplicate tuples can appear
- To eliminate duplicate tuples in a query result, the keyword **DISTINCT** is used
- For example, the result of Q11 may have duplicate SALARY values whereas Q11A does not have any duplicate values

```
Q11:      SELECT SALARY
          FROM          EMPLOYEE
Q11A:     SELECT DISTINCT SALARY
          FROM          EMPLOYEE
```

SET OPERATIONS

124

- SQL has directly incorporated some set operations
- There is a union operation (UNION), and in *some versions* of SQL there are set difference (MINUS) and intersection (INTERSECT) operations
- The resulting relations of these set operations are sets of tuples; *duplicate tuples are eliminated from the result*
- The set operations apply only to *union compatible relations*; the two relations must have the same attributes and the attributes must appear in the same order

SET OPERATIONS (contd.)

125

- Query 4: Make a list of all project numbers for projects that involve an employee whose last name is 'Smith' as a worker or as a manager of the department that controls the project.

```
Q4:          (SELECT PNAME
              FROM          PROJECT, DEPARTMENT,
                          EMPLOYEE
              WHERE DNUM=DNUMBER AND
                    MGRSSN=SSN AND LNAME='Smith')
              UNION
              (SELECT          PNAME
              FROM          PROJECT, WORKS_ON, EMPLOYEE
              WHERE PNUMBER=PNO AND
                    ESSN=SSN AND NAME='Smith')
```

NESTING OF QUERIES

126

- A complete SELECT query, called a *nested query*, can be specified within the WHERE-clause of another query, called the *outer query*
 - ▣ Many of the previous queries can be specified in an alternative form using nesting
- Query 1: Retrieve the name and address of all employees who work for the 'Research' department.

```
Q1: SELECT FNAME, LNAME, ADDRESS
      FROM      EMPLOYEE
      WHERE DNO IN (SELECT DNUMBER
                    FROM      DEPARTMENT
                    WHERE DNAME='Research' )
```

NESTING OF QUERIES (contd.)

127

- The nested query selects the number of the 'Research' department
- The outer query select an EMPLOYEE tuple if its DNO value is in the result of either nested query
- The comparison operator IN compares a value v with a set (or multi-set) of values V , and evaluates to TRUE if v is one of the elements in V
- In general, we can have several levels of nested queries
- A reference to an *unqualified attribute* refers to the relation declared in the *innermost nested query*
- In this example, the nested query is *not correlated* with the outer query

CORRELATED NESTED QUERIES

128

- If a condition in the WHERE-clause of a *nested query* references an attribute of a relation declared in the *outer query*, the two queries are said to be *correlated*
 - ▣ The result of a correlated nested query is different for each tuple (or combination of tuples) of the relation(s) the outer query
- Query 12: Retrieve the name of each employee who has a dependent with the same first name as the employee.

```
Q12: SELECT  E.FNAME, E.LNAME
        FROM      EMPLOYEE AS E
        WHERE  E.SSN IN
                (SELECT  ESSN
                 FROM      DEPENDENT
                 WHERE  ESSN=E.SSN AND
                        E.FNAME=DEPENDENT_NAME)
```


CORRELATED NESTED QUERIES (contd.)

129

- In Q12, the nested query has a different result in the outer query
- A query written with nested SELECT... FROM... WHERE... blocks and using the = or IN comparison operators can *always* be expressed as a single block query. For example, Q12 may be written as in Q12A

```
Q12A:  SELECT E.FNAME, E.LNAME
        FROM      EMPLOYEE E, DEPENDENT D
        WHERE E.SSN=D.ESSN AND
              E.FNAME=D.DEPENDENT_NAME
```

CORRELATED NESTED QUERIES (contd.)

130

- The original SQL as specified for SYSTEM R also had a **CONTAINS** comparison operator, which is used in conjunction with nested correlated queries
 - ▣ This operator was *dropped from the language*, possibly because of the difficulty in implementing it efficiently
 - ▣ Most implementations of SQL do not have this operator
 - ▣ The CONTAINS operator compares *two sets of values*, and returns TRUE if one set contains all values in the other set
 - Reminiscent of the division operation of algebra

CORRELATED NESTED QUERIES (contd.)

131

- Query 3: Retrieve the name of each employee who works on all the projects controlled by department number 5.

```
Q3:      SELECT FNAME, LNAME
          FROM      EMPLOYEE
          WHERE (    (SELECT PNO
                     FROM      WORKS_ON
                     WHERE SSN=ESSN)
                 CONTAINS
                 (SELECT PNUMBER
                  FROM      PROJECT
                  WHERE DNUM=5) )
```

CORRELATED NESTED QUERIES (contd.)

- In Q3, the second nested query, which is *not correlated* with the outer query, retrieves the project numbers of all projects controlled by department 5
- The first nested query, which is correlated, retrieves the project numbers on which the employee works, which is *different for each employee tuple* because of the correlation

THE EXISTS FUNCTION

133

- EXISTS is used to check whether the result of a correlated nested query is empty (contains no tuples) or not
 - ▣ We can formulate Query 12 in an alternative form that uses EXISTS as Q12B

THE EXISTS FUNCTION (contd.)

134

- Query 12: Retrieve the name of each employee who has a dependent with the same first name as the employee.

```
Q12B:      SELECT      FNAME, LNAME
            FROM        EMPLOYEE
            WHERE EXISTS (SELECT *
                          FROM          DEPENDENT
                          WHERE SSN=ESSN
                          AND
                          FNAME=DEPENDENT_NAME)
```

THE EXISTS FUNCTION (contd.)

135

- Query 6: Retrieve the names of employees who have no dependents.

```
Q6:      SELECT      FNAME, LNAME
          FROM        EMPLOYEE
          WHERE NOT EXISTS (SELECT *
                           FROM DEPENDENT
                           WHERE SSN=ESSN)
```

- In Q6, the correlated nested query retrieves all DEPENDENT tuples related to an EMPLOYEE tuple. If *none exist*, the EMPLOYEE tuple is selected
 - ▣ EXISTS is necessary for the expressive power of SQL

EXPLICIT SETS

136

- It is also possible to use an **explicit (enumerated) set of values** in the WHERE-clause rather than a nested query
- Query 13: Retrieve the social security numbers of all employees who work on project number 1, 2, or 3.

```
Q13:      SELECT      DISTINCT ESSN
           FROM WORKS_ON
           WHERE       PNO IN (1, 2, 3)
```


NULLS IN SQL QUERIES

137

- SQL allows queries that check if a value is **NULL** (missing or undefined or not applicable)
- SQL uses **IS** or **IS NOT** to compare NULLs because it considers each NULL value distinct from other NULL values, so *equality comparison is not appropriate*.
- Query 14: Retrieve the names of all employees who do not have supervisors.

```
Q14:      SELECT      FNAME, LNAME
          FROM        EMPLOYEE
          WHERE SUPERSSN IS NULL
```

- ▣ Note: If a join condition is specified, tuples with NULL values for the join attributes are not included in the result

Joined Relations Feature in SQL2

138

- Can specify a "joined relation" in the FROM-clause
 - ▣ Looks like any other relation but is the result of a join
 - ▣ Allows the user to specify different types of joins (regular "theta" JOIN, NATURAL JOIN, LEFT OUTER JOIN, RIGHT OUTER JOIN, CROSS JOIN, etc)

Joined Relations Feature in SQL2 (contd.)

139

- Examples:

```
Q8: SELECT E.FNAME, E.LNAME, S.FNAME, S.LNAME
      FROM          EMPLOYEE E S
      WHERE E.SUPERSSN=S.SSN
```

- can be written as:

```
Q8: SELECT E.FNAME, E.LNAME, S.FNAME, S.LNAME
      FROM          (EMPLOYEE E LEFT OUTER JOIN
                     EMPLOYEES ON E.SUPERSSN=S.SSN)
```

Joined Relations Feature in SQL2 (contd.)

140

- Examples:

```
Q1: SELECT FNAME, LNAME, ADDRESS
      FROM EMPLOYEE, DEPARTMENT
      WHERE DNAME='Research' AND DNUMBER=DNO
```

- could be written as:

```
Q1: SELECT FNAME, LNAME, ADDRESS
      FROM      (EMPLOYEE JOIN DEPARTMENT
                 ON DNUMBER=DNO)
      WHERE DNAME='Research'
```

- or as:

```
Q1: SELECT FNAME, LNAME, ADDRESS
      FROM      (EMPLOYEE NATURAL JOIN DEPARTMENT
                 AS DEPT(DNAME, DNO, MSSN, MSDATE))
      WHERE DNAME='Research'
```

Joined Relations Feature in SQL2 (contd.)

141

- Another Example: Q2 could be written as follows; this illustrates multiple joins in the joined tables

```
Q2: SELECT      PNUMBER, DNUM, LNAME,  
                BDATE, ADDRESS  
FROM (PROJECT JOIN  
      DEPARTMENT ON  
      DNUM=DNUMBER) JOIN  
EMPLOYEE ON  
MGRSSN=SSN) )  
WHERE          PLOCATION='Stafford'
```

AGGREGATE FUNCTIONS

142

- Include **COUNT, SUM, MAX, MIN, and AVG**
- Query 15: Find the maximum salary, the minimum salary, and the average salary among all employees.

```
Q15:      SELECT      MAX(SALARY),  
                MIN(SALARY), AVG(SALARY)  
          FROM EMPLOYEE
```

- Some SQL implementations *may not allow more than one function* in the SELECT-clause

AGGREGATE FUNCTIONS (contd.)

143

- Query 16: Find the maximum salary, the minimum salary, and the average salary among employees who work for the 'Research' department.

```
Q16:      SELECT      MAX(SALARY),
                MIN(SALARY), AVG(SALARY)
FROM EMPLOYEE, DEPARTMENT
WHERE      DNO=DNUMBER AND
                DNAME='Research'
```

AGGREGATE FUNCTIONS (contd.)

144

- Queries 17 and 18: Retrieve the total number of employees in the company (Q17), and the number of employees in the 'Research' department (Q18).

```
Q17:      SELECT      COUNT (*)
          FROM        EMPLOYEE
```

```
Q18:      SELECT      COUNT (*)
          FROM        EMPLOYEE, DEPARTMENT
          WHERE DNO=DNUMBER AND
          DNAME='Research'
```


GROUPING

- In many cases, we want to apply the aggregate functions to *subgroups of tuples* in a relation
- Each subgroup of tuples consists of the set of tuples that have the *same value* for the *grouping attribute(s)*
- The function is applied to each subgroup independently
- SQL has a **GROUP BY**-clause for specifying the grouping attributes, which *must also appear in the SELECT-clause*

GROUPING (contd.)

146

- Query 20: For each department, retrieve the department number, the number of employees in the department, and their average salary.

```
Q20:      SELECT DNO, COUNT (*), AVG (SALARY)
           FROM      EMPLOYEE
           GROUP BY  DNO
```

- In Q20, the EMPLOYEE tuples are divided into groups-
 - Each group having the same value for the grouping attribute DNO
- The COUNT and AVG functions are applied to each such group of tuples separately
- The SELECT-clause includes only the grouping attribute and the functions to be applied on each group of tuples
- A join condition can be used in conjunction with grouping

GROUPING (contd.)

147

- Query 21: For each project, retrieve the project number, project name, and the number of employees who work on that project.

```
Q21:      SELECT PNUMBER, PNAME, COUNT (*)
           FROM          PROJECT, WORKS_ON
           WHERE PNUMBER=PNO
           GROUP BY     PNUMBER, PNAME
```

- In this case, the grouping and functions are applied after the joining of the two relations

THE HAVING-CLAUSE

148

- Sometimes we want to retrieve the values of these functions for only those *groups that satisfy certain conditions*
- The **HAVING**-clause is used for specifying a selection condition on groups (rather than on individual tuples)

THE HAVING-CLAUSE (contd.)

149

- Query 22: For each project *on which more than two employees work*, retrieve the project number, project name, and the number of employees who work on that project.

```
Q22:      SELECT      PNUMBER, PNAME,
              COUNT(*)
FROM PROJECT, WORKS_ON
WHERE      PNUMBER=PNO
GROUP BY  PNUMBER, PNAME
HAVING    COUNT (*) > 2
```

SUBSTRING COMPARISON

150

- The **LIKE** comparison operator is used to compare partial strings
- Two reserved characters are used: '%' (or '*' in some implementations) replaces an arbitrary number of characters, and '_' replaces a single arbitrary character

SUBSTRING COMPARISON (contd.)

151

- Query 25: Retrieve all employees whose address is in Houston, Texas. Here, the value of the ADDRESS attribute must contain the substring 'Houston,TX' in it.

```
Q25:      SELECT      FNAME, LNAME
           FROM EMPLOYEE
           WHERE       ADDRESS LIKE
                    '%Houston,TX%'
```

SUBSTRING COMPARISON (contd.)

152

- Query 26: Retrieve all employees who were born during the 1950s.
 - ▣ Here, '5' must be the 8th character of the string (according to our format for date), so the BDATE value is '_____5_', with each underscore as a place holder for a single arbitrary character.

```
Q26:      SELECT FNAME, LNAME
           FROM      EMPLOYEE
           WHERE BDATE LIKE '_____5_'
```

- The LIKE operator allows us to get around the fact that each value is considered atomic and indivisible
 - ▣ Hence, in SQL, character string attribute values are not atomic

ARITHMETIC OPERATIONS

153

- The standard arithmetic operators '+', '-', '*', and '/' (for addition, subtraction, multiplication, and division, respectively) can be applied to numeric values in an SQL query result
- Query 27: Show the effect of giving all employees who work on the 'ProductX' project a 10% raise.

```
Q27:      SELECT FNAME, LNAME, 1.1*SALARY
           FROM          EMPLOYEE, WORKS_ON,
                       PROJECT
           WHERE SSN=ESSN AND PNO=PNUMBER
                 AND PNAME='ProductX'
```

ORDER BY

154

- The **ORDER BY** clause is used to sort the tuples in a query result based on the values of some attribute(s)
- Query 28: Retrieve a list of employees and the projects each works in, ordered by the employee's department, and within each department ordered alphabetically by employee last name.

```
Q28:      SELECT DNAME, LNAME, FNAME, PNAME
           FROM      DEPARTMENT, EMPLOYEE,
                    WORKS_ON, PROJECT
           WHERE DNUMBER=DNO AND SSN=ESSN
                    AND PNO=PNUMBER
           ORDER BY  DNAME, LNAME
```

ORDER BY (contd.)

155

- The default order is in ascending order of values
- We can specify the keyword **DESC** if we want a descending order; the keyword **ASC** can be used to explicitly specify ascending order, even though it is the default

Summary of SQL Queries

156

- A query in SQL can consist of up to six clauses, but only the first two, **SELECT** and **FROM**, are mandatory. The clauses are specified in the following order:

SELECT	<attribute list>
FROM	<table list>
[WHERE	<condition>]
[GROUP BY	<grouping attribute(s)>]
[HAVING	<group condition>]
[ORDER BY	<attribute list>]

Summary of SQL Queries (contd.)

157

- The SELECT-clause lists the attributes or functions to be retrieved
- The FROM-clause specifies all relations (or aliases) needed in the query but not those needed in nested queries
- The WHERE-clause specifies the conditions for selection and join of tuples from the relations specified in the FROM-clause
- GROUP BY specifies grouping attributes
- HAVING specifies a condition for selection of groups
- ORDER BY specifies an order for displaying the result of a query
 - A query is evaluated by first applying the WHERE-clause, then GROUP BY and HAVING, and finally the SELECT-clause

Specifying Updates in SQL

158

- There are three SQL commands to modify the database: **INSERT**, **DELETE**, and **UPDATE**

INSERT

159

- In its simplest form, it is used to add one or more tuples to a relation
- Attribute values should be listed in the same order as the attributes were specified in the **CREATE TABLE** command

INSERT (contd.)

160

- Example:

```
U1: INSERT INTO    EMPLOYEE
      VALUES ('Richard','K','Marini', '653298653', '30-DEC-52',
              '98 Oak Forest,Katy,TX', 'M', 37000,'987654321', 4 )
```

- An alternate form of INSERT specifies explicitly the attribute names that correspond to the values in the new tuple

- ▣ Attributes with NULL values can be left out

- Example: Insert a tuple for a new EMPLOYEE for whom we only know the FNAME, LNAME, and SSN attributes.

```
U1A: INSERT INTO  EMPLOYEE (FNAME, LNAME,
                          SSN)
      VALUES ('Richard', 'Marini', '653298653')
```


INSERT (contd.)

161

- Important Note: Only the constraints specified in the DDL commands are automatically enforced by the DBMS when updates are applied to the database
 - ▣ Another variation of INSERT allows insertion of *multiple tuples* resulting from a query into a relation

INSERT (contd.)

162

- Example: Suppose we want to create a temporary table that has the name, number of employees, and total salaries for each department.
 - ▣ A table DEPTS_INFO is created by U3A, and is loaded with the summary information retrieved from the database by the query in U3B.

```
U3A: CREATE TABLE DEPTS_INFO
      (DEPT_NAME          VARCHAR(10),
       NO_OF_EMPS        INTEGER,
       TOTAL_SAL         INTEGER);
```

```
U3B: INSERT INTO DEPTS_INFO (DEPT_NAME,
                             NO_OF_EMPS, TOTAL_SAL)
      SELECT DNAME, COUNT (*), SUM (SALARY)
      FROM DEPARTMENT, EMPLOYEE
      WHERE DNUMBER=DNO
      GROUP BY DNAME ;
```

INSERT (contd.)

163

- Note: The DEPTS_INFO table may not be up-to-date if we change the tuples in either the DEPARTMENT or the EMPLOYEE relations *after* issuing U3B. We have to create a view (see later) to keep such a table up to date.

DELETE

164

- Removes tuples from a relation
 - ▣ Includes a WHERE-clause to select the tuples to be deleted
 - ▣ Referential integrity should be enforced
 - ▣ Tuples are deleted from only *one table* at a time (unless CASCADE is specified on a referential integrity constraint)
 - ▣ A missing WHERE-clause specifies that *all tuples* in the relation are to be deleted; the table then becomes an empty table
 - ▣ The number of tuples deleted depends on the number of tuples in the relation that satisfy the WHERE-clause

DELETE (contd.)

165

□ Examples:

```
U4A: DELETE FROM EMPLOYEE
      WHERE      LNAME='Brown'
```

```
U4B: DELETE FROM EMPLOYEE
      WHERE      SSN='123456789'
```

```
U4C: DELETE FROM EMPLOYEE
      WHERE      DNO IN
                (SELECT DNUMBER
                 FROM DEPARTMENT
                 WHERE DNAME='Research')
```

```
U4D: DELETE FROM EMPLOYEE
```

UPDATE

166

- Used to modify attribute values of one or more selected tuples
- A WHERE-clause selects the tuples to be modified
- An additional SET-clause specifies the attributes to be modified and their new values
- Each command modifies tuples *in the same relation*
- Referential integrity should be enforced

UPDATE (contd.)

167

- Example: Change the location and controlling department number of project number 10 to 'Bellaire' and 5, respectively.

```
U5: UPDATE      PROJECT
           SET          PLOCATION = 'Bellaire',
                       DNUM = 5
           WHERE        PNUMBER=10
```

UPDATE (contd.)

168

- Example: Give all employees in the 'Research' department a 10% raise in salary.

```
U6: UPDATE      EMPLOYEE
      SET        SALARY = SALARY *1.1
      WHERE DNO IN (SELECT  DNUMBER
                        FROM    DEPARTMENT
                        WHERE   DNAME='Research')
```

- In this request, the modified SALARY value depends on the original SALARY value in each tuple
 - ▣ The reference to the SALARY attribute on the right of = refers to the old SALARY value before modification
 - ▣ The reference to the SALARY attribute on the left of = refers to the new SALARY value after modification

Questions

169

1. Define relation. Explain the various characteristics of relations.
2. Explain various types of relational model constraints.
3. Explain the DIVISION operation with example.
4. Explain the E-R to relational mapping algorithm with examples for each step.
5. Write a note on data types available in SQL.
6. How to we can add constraint to a relation? Explain.