

The Relational Algebra



CHAPTER OUTLINE

Relational Algebra

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



RELATIONAL ALGEBRA OVERVIEW

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

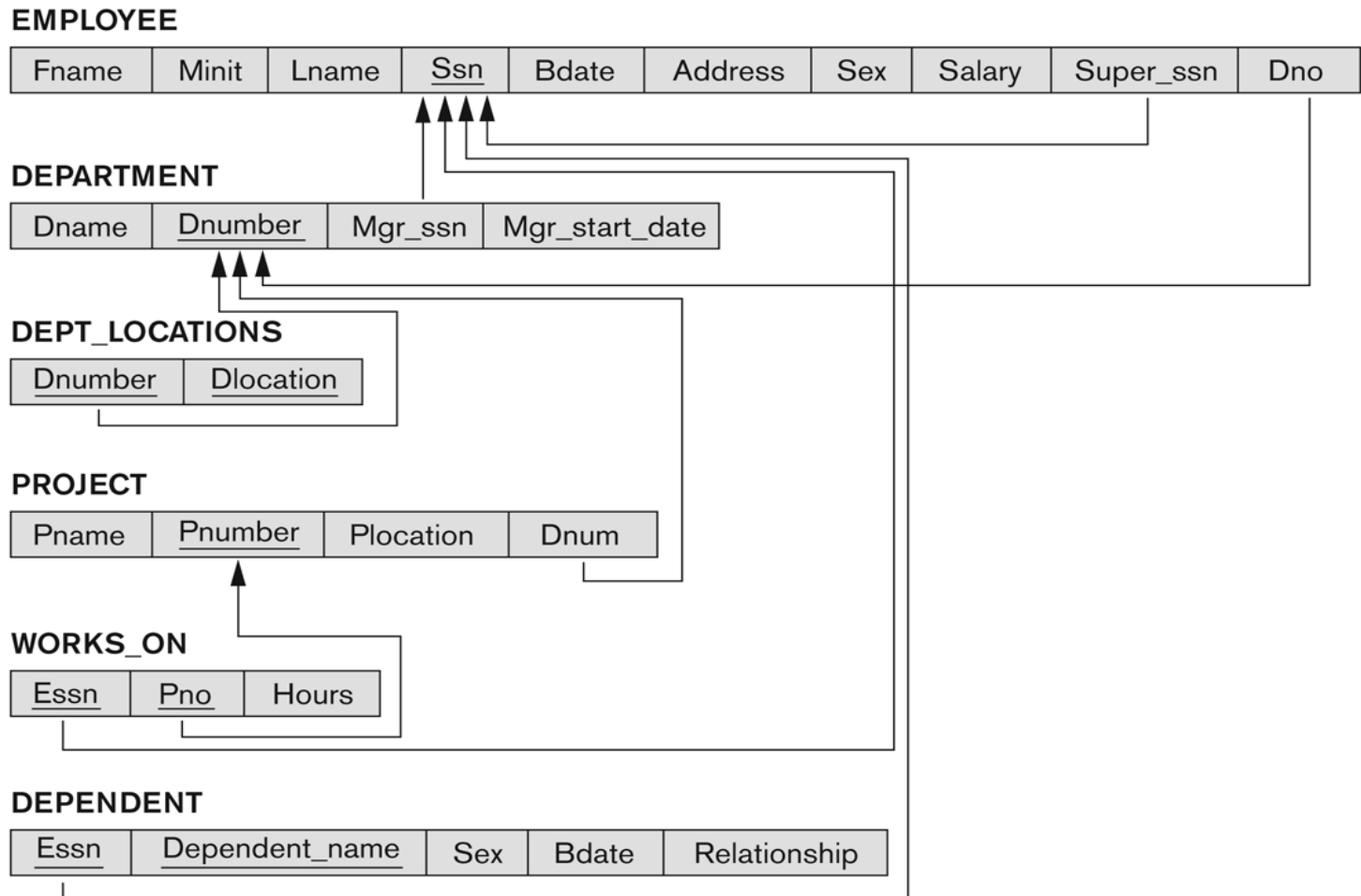
- Relational Algebra consists of several groups of operations
 - Unary Relational Operations
 - SELECT (symbol: σ (sigma))
 - PROJECT (symbol: π (pi))
 - RENAME (symbol: ρ (rho))
 - Relational Algebra Operations From Set Theory
 - UNION (\cup), INTERSECTION (\cap), DIFFERENCE (or MINUS, $-$)
 - CARTESIAN 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 STATE FOR COMPANY

Figure 5.7

Referential integrity constraints displayed on the COMPANY relational database schema.



UNARY RELATIONAL OPERATIONS: SELECT

- select a *subset* of the tuples from a relation based on a **selection condition**.
 - The selection condition acts as a **filter and keeps** only those tuples that satisfy the qualifying condition

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 (CONTINUED)

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

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

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

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

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



THE FOLLOWING QUERY RESULTS REFER TO THIS DATABASE STATE

Figure 5.6

One possible database state for the COMPANY relational database schema.

EMPLOYEE

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

DEPARTMENT

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

DEPT_LOCATIONS

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

WORKS_ON

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

PROJECT

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

DEPENDENT

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



UNARY RELATIONAL OPERATIONS: PROJECT

- PROJECT Operation is denoted by π (pi)
This operation keeps certain *columns* (attributes) from a relation and discards the other columns.
- PROJECT creates a vertical partitioning
The list of specified columns (attributes) is kept in each tuple
The other attributes in each tuple are discarded

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

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


UNARY RELATIONAL OPERATIONS: PROJECT (CONT.)

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

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

- 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.)

- 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 8.1 Results of SELECT and PROJECT operations. (a) $\sigma_{(Dno=4 \text{ AND } Salary > 25000) \text{ OR } (Dno=3 \text{ AND } Salary > 30000)}$ (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	888885555	5
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5

(b)

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

(c)

Sex	Salary
M	30000
M	40000
F	25000
F	43000
M	38000
M	25000
M	55000



RELATIONAL ALGEBRA EXPRESSIONS

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



SINGLE EXPRESSION 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 operation
- 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



EXAMPLE OF APPLYING MULTIPLE OPERATIONS AND RENAME

Figure 8.2 Results of a sequence of operations. (a) $\pi_{Fname, Lname, Salary}(\sigma_{Dno=3}(EMPLOYEE))$.
(b) Using intermediate relations and renaming of attributes.

(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	1982-09-15	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

Adapted from Navathe (7th Edition)



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:

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

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



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

- Type Compatibility of operands is required for the binary set operation UNION \cup , (also for INTERSECTION \cap , and SET DIFFERENCE $-$, see next slides)
- $R_1(A_1, A_2, \dots, A_n)$ and $R_2(B_1, B_2, \dots, B_n)$ are type compatible if:
 - they have the same number of attributes, and
 - the domains of corresponding attributes are type compatible (i.e. $\text{dom}(A_i) = \text{dom}(B_i)$ for $i=1, 2, \dots, n$).
- The resulting relation for $R_1 \cup R_2$ (also for $R_1 \cap R_2$, or $R_1 - R_2$, see next slides) has the same attribute names as the *first* operand relation R_1 (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”



RELATIONAL ALGEBRA OPERATIONS FROM SET THEORY: SET DIFFERENCE (CONT.)

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



EXAMPLE TO ILLUSTRATE THE RESULT OF UNION, INTERSECT, AND DIFFERENCE

Figure 8.4 The set operations UNION, INTERSECTION, and MINUS. (a) Two union-compatible relations. (b) $\text{STUDENT} \cup \text{INSTRUCTOR}$. (c) $\text{STUDENT} \cap \text{INSTRUCTOR}$. (d) $\text{STUDENT} - \text{INSTRUCTOR}$. (e) $\text{INSTRUCTOR} - \text{STUDENT}$.

(a) STUDENT

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

INSTRUCTOR

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

(b)

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

(c)

Fn	Ln
Susan	Yao
Ramesh	Shah

(d)

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

(e)

Fname	Lname
John	Smith
Ricardo	Browne
Francis	Johnson



SOME PROPERTIES OF UNION, INTERSECT, AND DIFFERENCE

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



BINARY RELATIONAL OPERATIONS: JOIN


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

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

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



BINARY RELATIONAL OPERATIONS: JOIN (CONT.)

- Example: Suppose that we want to retrieve the name of the manager of each department.
 - To get the manager's name, we need to combine each DEPARTMENT tuple with the EMPLOYEE tuple whose SSN value matches the MGRSSN value in the department tuple.
 - We do this by using the join  operation.
- $\text{DEPT_MGR} \leftarrow \text{DEPARTMENT} \underset{\text{MGRSSN=SSN}}{\text{MGRSSN=SSN}} \text{EMPLOYEE}$
- 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 $\text{DEPARTMENT.MGRSSN} = \text{EMPLOYEE.SSN}$



RESULT OF THE JOIN OPERATION DEPT_MGR \leftarrow DEPARTMENT \bowtie MGR_SSN=SSN EMPLOYEE.

DEPT_MGR

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



SOME PROPERTIES OF JOIN

- Consider the following JOIN operation:
 - $R(A_1, A_2, \dots, A_n) \quad S(B_1, B_2, \dots, B_m)$
 $R.A_i = S.B_j$
 - 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 \bowtie JOIN operation is called a Theta-join:
 $R \quad \textit{theta} \quad S$
- 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

- 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 (CONTINUED)

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



EXAMPLE OF NATURAL JOIN OPERATION

Figure 8.7 Results of two natural join operations. (a) proj_dept \leftarrow project * dept. (b) dept_locs \leftarrow department * dept_locations.

(a)

PROJ_DEPT

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

(b)

DEPT_LOCS

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



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



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

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

(a)

SSN_PNOS

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

SMITH_PNOS

Pno
1
2

SSNS

Sen
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



OPERATIONS OF RELATIONAL ALGEBRA

Table 8.1 Operations of Relational Algebra

OPERATION	PURPOSE	NOTATION
SELECT	Selects all tuples that satisfy the selection condition from a relation R .	$\sigma_{\langle \text{selection condition} \rangle}(R)$
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.	$R_1 \star_{\langle \text{join condition} \rangle} R_2$, OR $R_1 \star_{(\langle \text{join attributes 1} \rangle, \langle \text{join attributes 2} \rangle)} R_2$ OR $R_1 \star R_2$

continued on next slide



OPERATIONS OF RELATIONAL ALGEBRA (CONTINUED)

Table 8.1 Operations of Relational Algebra

OPERATION	PURPOSE	NOTATION
UNION	Produces a relation that includes all the tuples in R_1 or R_2 or both R_1 and R_2 ; R_1 and R_2 must be union compatible.	$R_1 \cup R_2$
INTERSECTION	Produces a relation that includes all the tuples in both R_1 and R_2 ; R_1 and R_2 must be union compatible.	$R_1 \cap R_2$
DIFFERENCE	Produces a relation that includes all the tuples in R_1 that are not in R_2 ; R_1 and R_2 must be union compatible.	$R_1 - R_2$
CARTESIAN PRODUCT	Produces a relation that has the attributes of R_1 and R_2 and includes as tuples all possible combinations of tuples from R_1 and R_2 .	$R_1 \times R_2$
DIVISION	Produces a relation $R(X)$ that includes all tuples $t[X]$ in $R_1(Z)$ that appear in R_1 in combination with every tuple from $R_2(Y)$, where $Z = X \cup Y$.	$R_1(Z) \div R_2(Y)$



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
 - $\text{DNO } \mathcal{F} \text{ COUNT SSN, AVERAGE Salary (EMPLOYEE)}$
- Above operation groups employees by DNO (department number) and computes the count of employees and average salary per department



THE AGGREGATE FUNCTION OPERATION.

- $\rho_{R(Dno, No_of_employees, Average_sal)}(Dno \bowtie COUNT Ssn, AVERAGE Salary (EMPLOYEE)).$
- $Dno \bowtie salary (EMPLOYEE).$
- $\bowtie COUNT Ssn, AVERAGE Salary (EMPLOYEE).$

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



RESULTS OF GROUP BY AND HAVING (IN SQL)

Fname	Minit	Lname	<u>Ssn</u>	...	Salary	Super_ssn	Dno		Dno	Count (*)	Avg (Salary)
John	B	Smith	123456789		30000	333445555	5		5	4	33250
Franklin	T	Wong	333445555		40000	888665555	5		4	3	31000
Ramesh	K	Narayan	666884444		38000	333445555	5		1	1	55000
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				

Result of Q24

Grouping EMPLOYEE tuples by the value of Dno

continued on next slide



ADDITIONAL RELATIONAL OPERATIONS

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



ADDITIONAL RELATIONAL OPERATIONS (CONTINUED)

- 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 $R \bowtie_{full} S$ keeps all tuples in both the left and the right relations when no matching tuples are found, padding them with null values as needed.



THE RESULT OF A LEFT OUTER JOIN OPERATION.

RESULT

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

Adapted from Navathe (7th Edition)



EXAMPLES OF QUERIES IN RELATIONAL ALGEBRA : PROCEDURAL FORM

- **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.**

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

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

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

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



EXAMPLES OF QUERIES IN RELATIONAL ALGEBRA – SINGLE EXPRESSIONS

As a single expression, these queries become:

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

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

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

$\pi_{\text{Lname, Fname}} ((\pi_{\text{Ssn}} (\text{EMPLOYEE}) - \rho_{\text{Ssn}} (\pi_{\text{Essn}} (\text{DEPENDENT}))) * \text{EMPLOYEE})$



QUESTIONS

Consider the following collection of relation schemes:

professor(**profname**, deptname)

department(**deptname**, **building**)

committee(**profname**, **commname**)

- a. Find all the professors who are in any one of the committees that Professor Smith is in.
- b. Find all the professors who are in at least all those committees that Professor Smith is in.
- c. Find all the professors who are in exactly (i.e., no more and no less) all those committees that Professor Smith is in.
- d. Find all the professors who have offices in at least all those buildings that Professor Smith has offices in.



QUESTIONS

- Consider the following GRADEBOOK relational schema describing the data for a grade book of a particular instructor. (*Note: The attributes A, B, C, and D of COURSES store grade cutoffs.*)

CATALOG(Cno, Ctitle)

STUDENTS(Sid, Fname, Lname, Minit)

COURSES(Term, Sec_no, Cno, A, B, C, D)

ENROLLS(Sid, Term, Sec_no)

Specify and execute the following queries using the RA interpreter on the GRADEBOOK database schema.

- a. Retrieve the names of students enrolled in the Automata class during the fall 2009 term.
- b. Retrieve the Sid values of students who have enrolled in CSc226 and CSc227.
- c. Retrieve the Sid values of students who have enrolled in CSc226 or CSc227.
- d. Retrieve the names of students who have not enrolled in any class.

