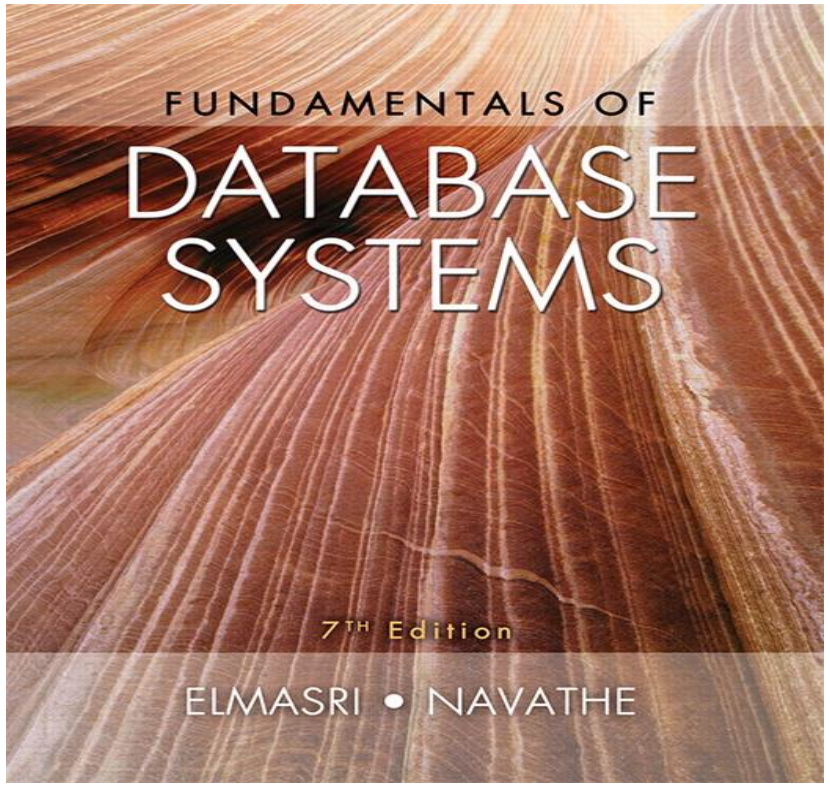
Comp-3150: Database Management Systems

- Ramez Elmasri , Shamkant B. Navathe(2016) Fundamentals of Database Systems (7th Edition), Pearson, isbn 10: 0-13-397077-9; isbn-13:978-0-13-397077-7.

Chapter 8:
The Relational Algebra and

The Relational Calculus



Chapter 8: The Relational Algebra and The Relational Calculus: Outline

- 1. Relational Algebra
  - 1.1 Unary Relational Operations
    - (SELECT (symbol: σ (sigma))
    - PROJECT (symbol: π (pi))
    - RENAME (symbol: ρ (rho))
  - 1.2 Binary Relational Operations
    - JOIN (several variations of JOIN exist)($\bowtie_{<joincondition>}$)
    - DIVISION ($\div$)
  - 1.3 Relational Algebra Operations From Set Theory
    - UNION ($\cup$), INTERSECTION ($\cap$), DIFFERENCE (or MINUS, **–**)
    - CARTESIAN PRODUCT (**x**)
  - 1.4 Additional Relational Operations (not fully discussed)
  - 1.5 Examples of Queries in Relational Algebra
- 2. Relational Calculus
  - 2.1 Tuple Relational Calculus

1. Relational Algebra

- The formal languages for the relational model are:
  - the relational algebra and relational calculus.
- A data model must have a set of operations for manipulating its data structure and constraints.
- The basic set of operations for the relational model is:
  - the relational algebra which expresses the data retrieval requests as relational algebra expressions.
- A sequence of relational algebra operations is a relational algebra expression,
  - which produces a relation result that is result of a database query.

1. Relational Algebra

- Thus relational algebra provides:
  - (1) a formal foundation for relational model operations.
  - (2) It is used for query processing and optimization.
  - (3) Some of its concepts are implemented in the RDBMSs.
- The relational calculus provides a declarative (rather than procedural) language for specifying relational queries
  - as it tells what the query result should be and not how or sequence of steps for retrieving it.
- The relational algebra has two groups of operations

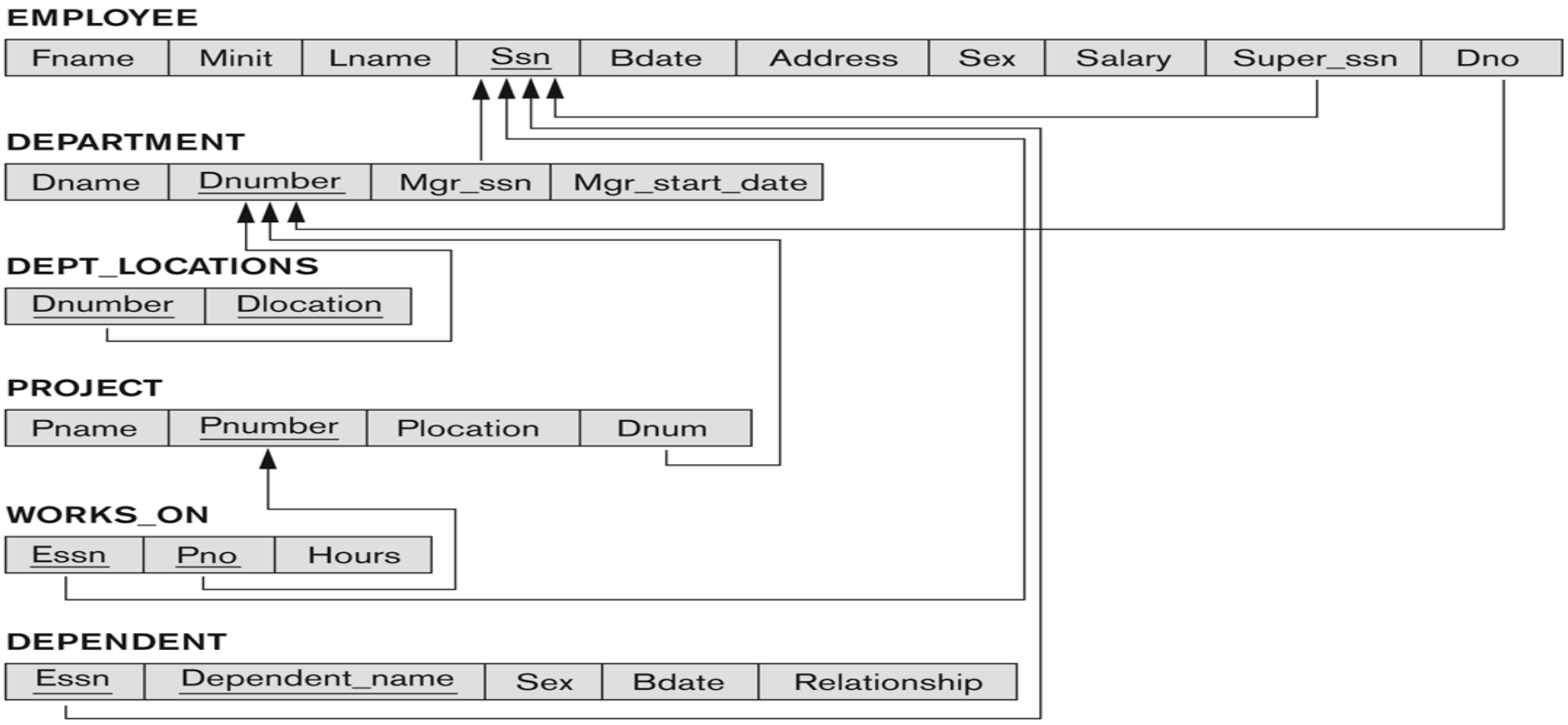
1. Relational Algebra

- 1.1. Unary Relational Operations
  - SELECT (symbol: σ (sigma))
  - PROJECT (symbol: π (pi))
  - RENAME (symbol: ρ (rho))
- 1.2. Binary Relational Operations
  - JOIN (several variations of JOIN exist) ($\bowtie_{<joincondition>}$)
  - DIVISION ($\div$)
- 1.3. Relational Algebra Operations From Set Theory
  - UNION ($\cup$), INTERSECTION ($\cap$), DIFFERENCE (or MINUS, **–**)
  - CARTESIAN PRODUCT (**x**)
- 1.4. Additional Relational Operations
  - OUTER JOINS, OUTER UNION
  - AGGREGATE FUNCTIONS (These compute summary of information: for example, SUM, COUNT, AVG, MIN, MAX)

Database State for COMPANY

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

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



1.1 Unary Relational Operations: SELECT

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

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

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

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

1.1 Unary Relational Operations: SELECT

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

1.1 Unary Relational Operations: SELECT

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

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:
- $\pi_{LNAME, FNAME, SALARY}(EMPLOYEE)$
- This can be renamed as:
  R(Last_name, First_name, Salary) ← $\pi_{LNAME, FNAME, SALARY}(EMPLOYEE)$

Unary Relational Operations: PROJECT (also Rename)

- We can define a formal RENAME operation (ρ) to rename either the relation or attribute names or both.
- The general RENAME operation applied to a relation $R(A_1, A_2, \ldots, A_n)$ of degree n is of the form:
- (i) $\rho_{S(B1, B2, \ldots, Bn)}(R)$ for renaming both table R to S and its attributes from $A_1, A_2, \ldots, A_n$ to $B_1, B_2, \ldots, B_n$

or

- (ii) $\rho_s(R)$ for renaming only table R to **S.**
- (iii) $\rho_{(B1, B2, \ldots, Bn)}(R)$ for renaming only attributes of table R from $A_1, A_2, \ldots, A_n$ to $B_1, B_2, \ldots, B_n$
- Where rho (ρ) denotes RENAME operator, S is the new relation name and $(B_1, B_2, \ldots, B_n)$ are the new attribute names.
- E.g. if attributes of R are $A_1, A_2, \ldots, A_n$, with $P_{S(B1, B2, \ldots, Bn)}(R)$, the relation is renamed S with new attributes $B_1, B_2, \ldots, B_n$ for $A_1, A_2, \ldots, A_n$

Unary Relational Operations: PROJECT

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

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

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

Unary Relational Operations: PROJECT

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

Examples of applying SELECT and PROJECT operations

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

(a)

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

(b)

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

(c)

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

1.3 Relational Algebra Operations from Set Theory: UNION

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

1.3 Relational Algebra Operations from Set Theory: UNION

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

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

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

Figure 8.3 Result of the UNION operation RESULT ← RESULT1 ∪ RESULT2

RESULT1

Ssn
123456789
333445555
666884444
453453453

RESULT2

Ssn
333445555
888665555

RESULT

Ssn
123456789
333445555
666884444
453453453
888665555

Relational Algebra Operations from Set Theory

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

Relational Algebra Operations from Set Theory: INTERSECTION

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

Relational Algebra Operations from Set Theory: SET DIFFERENCE

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

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

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

(a) STUDENT

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

INSTRUCTOR

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

(b)

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

(c)

Fn	Ln
Susan	Yao
Ramesh	Shah

(d)

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

(e)

Fname	Lname
John	Smith
Ricardo	Browne
Francis	Johnson

Some properties of UNION, INTERSECT, and DIFFERENCE

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

Relational Algebra Operations from Set Theory: CARTESIAN PRODUCT

- CARTESIAN (or CROSS) PRODUCT Operation
  - This operation is used to combine tuples from two relations in a combinatorial fashion.
  - Denoted by R(A1, A2, . . ., An) x S(B1, B2, . . ., Bm)
  - Result is a relation Q with degree n + m attributes:
    - Q(A1, A2, . . ., An, B1, B2, . . ., Bm), in that order.
  - The resulting relation state has one tuple for each combination of tuples—one from R and one from S.
  - Hence, if R has n_R tuples (denoted as $|R| = n_R$), and S has n_S tuples, then R x S will have $n_R * n_S$ tuples.
  - The two operands do NOT have to be "type compatible"

Relational Algebra Operations from Set Theory: CARTESIAN PRODUCT

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

Relational Algebra Operations from Set Theory: CARTESIAN PRODUCT

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

Figure 8.5 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	3321Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291Berry, Bellaire, TX	F	43000	888665555	4
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5

EMPNAMES

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

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	. . .

Figure 8.5 (continued) The CARTESIAN PRODUCT (CROSS PRODUCT) operation.

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

- JOIN Operation (denoted by $\bowtie_{<joincondition>}$)
  - 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(A1, A2, . . ., An) and S(B1, B2, . . ., Bm) is:

$$R \bowtie_{<joincondition>} S$$

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

Binary Relational Operations: JOIN (cont.)

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

Figure 8.6 Result of the JOIN operation DEPT_MGR ← 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:
  - $R(A1, A2, \ldots, An) \bowtie_{R.Ai=S.Bj} S(B1, B2, \ldots, Bm)$
  - Result is a relation Q with degree n + m attributes:
    - Q(A1, A2, . . ., An, B1, B2, . . ., Bm), in that order.
  - The resulting relation state has one tuple for each combination of tuples—r from R and s from S, but *only if they satisfy the join condition* r[Ai]=s[Bj] or R.Ai = S.Bj
  - Hence, if R has n_R tuples, and S has n_S tuples, then the join result will generally have *less than* $n_R * n_S$ tuples.
  - Only related tuples (based on the join condition) will appear in the result

Complete Set of Relational Operations

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

Examples of Queries in Relational Algebra – Single expressions

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

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

Division Operation (÷)

- DIVISION Operation
  - The DIVISION operation is useful for a special kind of query as: Retrieve the names of employees who work on all the projects that 'John Smith' works on. That is, if John Smith works on the set of projects with Pno = {1, 2}, any employee selected must have worked on all the Pnos in this set.
  - The division operation is applied to two relations
  - R(Z) ÷ S(X) are the two input relation operands of the division operator and the resulting relation is T(Y). For example, Works_on(Essn,Pno) ÷ D(Pno) will give result containing all Essns who have worked on all Pno's in D(Pno) set.
  - For R(Z) ÷ S(X), denominator relation has its set of attributes X (eg. S(a: varchar2(2))) as a subset of the numerator relation's set of attributes, Z (a: varchar2(2), b: varchar2(2)). The resulting relation, T(Y) has the set of attributes Y = Z - X (eg. b: varchar2(2)) which is the set of attributes of R that are not attributes of S.
  - The result of this DIVISION is a relation T(Y) that includes a tuple t that must appear in the result T if tuples t_R appear in the numerator relation R in combination with every tuple in the denominator relation S.

Division Operation (÷) and (natural join operator *)

- We can answer this query as:
- (a) Get Denomenator (Pnos worked by John Smith) as:
  Smith_Pnos ← π_{Pno} ($\sigma_{Lname='Smith' \text{ and } Fname='John'}$ (
  WORKS_ON $\underset{Essn=Ssn}{\bowtie}$ EMPLOYEE))
  (b) Get Numerator (Essn with Pnos worked by all employees) as:
  Ssn_Pnos ← $\pi_{Essn,\ Pno}$ (WORKS_ON)
  (c) Get all employees who worked on all project worked on by Smith as:
  SSNs(Ssn) ← Ssn_Pnos ÷ Smith_Pnos
- See the result of these operations in Fig. 8.8 on page 256 of book.
- Note that the natural join (*) which is a join of two tables on join foreign/primary key attributes (e.g., Ssn) with the same name can be used for example to get the names of these employees working on all projects worked on by Smith as: $\pi_{Fname,\ Lname}$(SSNs * EMPLOYEE)

Fig 8.8: Division Operation (÷)

Figure 8.8 The DIVISION operation. (a) Dividing SSN_PNOS by SMITH_PNOS. (b) T ← R ÷ S.

(a)

SSN_PNOS

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

SMITH_PNOS

Pno
1
2

SSNS

Ssn
123456789
453453453

(b)

R

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

S

A
a1
a2
a3

T

B
b1
b4

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, MINIMUM and COUNT.
- The COUNT function is used for counting tuples or values.

Aggregate Function Operation

- Use of the Aggregate Function operation $\Im$ (called script)
- The general format that includes grouping attributes is:
- $<\text{grouping attributes}> \Im_{<\text{function list}>} (R)$
  where <grouping attributes> is a list of attributes of relation R, <function list> is a list of (<function><attribute>) pairs. Function is one of SUM, AVERAGE, MAXIMUM, MINIMUM, COUNT. The result has the grouping attributes plus one attribute for each element in the function list.
- $\Im_{\text{MAX Salary}}$ (EMPLOYEE) retrieves the maximum salary value from the EMPLOYEE relation
  - $\Im_{\text{MIN Salary}}$ (EMPLOYEE) retrieves the minimum Salary value from the EMPLOYEE relation
  - $\Im_{\text{SUM Salary}}$ (EMPLOYEE) retrieves the sum of the Salary from the EMPLOYEE relation
  - $\Im_{\text{COUNT SSN, AVERAGE Salary}}$ (EMPLOYEE) computes the count (number) of employees and their average salary
    - Note: count just counts the number of rows, without removing duplicates

Using Grouping with Aggregation

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

Figure 8.10 The aggregate function operation.

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

b. $\text{Dno } \Im \text{ COUNT Ssn, AVERAGE Salary}(\text{EMPLOYEE})$.

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

- a is renamed with ρ, b has no renaming and c has no grouping.

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

2. Relational Calculus

- Relational Calculus is another formal query language for the relational model.
- Two variations of it are:
  - tuple relational calculus and
  - domain relational calculus.
- In both variations one declarative expression is written to specify a retrieval query.
- The expression has no description of how, or in what order, to evaluate a query.
- A calculus expression specifies what is to be retrieved rather than how to retrieve it.
- Relational calculus is a nonprocedural language as opposed to the relational algebra that is procedural.
- A calculus expression may be written in different ways that do not determine how the query is evaluated.

2. Relational Calculus

- Any retrieval that can be specified in the relational algebra can also be specified in relational calculus, and vice versa.
- A relational query language L is relationally complete if we can express in L any query that can be expressed in relational calculus.
- This relational completeness property is used as a basis for comparing the expressive power of high-level query languages.
- Most languages such as SQL are relationally complete but have more expressive power than relational algebra or relational calculus:
  - as they have additional operations like aggregate functions, grouping and ordering.

Tuple Relational Calculus

- The tuple relational calculus (TRC) is based on specifying a number of tuple variables.
- Each tuple variable usually ranges over a particular database relation, meaning that the variable may take as its value any individual tuple from that relation.
- A simple tuple relational calculus query is of the form **{t | COND(t)}**
- **Expressed in general also as: {$t_1.A_j$, $t_2.A_k$, …, $t_n.A_m$ | COND(t_1, t_2, …, t_{n+m})}**
- where t is a tuple variable and COND (t) is a conditional expression involving t. Also, the $\mathbf{t_i}$'s are tuple variables and $\mathbf{A_j}$'s are attributes of the relation on which $\mathbf{t_i}$ ranges.
- COND is a condition (atom or formula) of TRC of the form:
- (i) Relation ($\mathbf{t_i}$), eg. Employee(t)
- (ii) **t_1.A op t_1.B, eg. E.ssn = d.Essn**
- **(iii) t_1.A op c or c op t_1.A, eg. Salary > 30000**
- **(iv) A formula (F) is made up of atoms and atoms are connected with logical ops and quantifiers (∃, ∀), eg. F_1 AND F_2, F_1 OR F_2 , NOT(F_1,** (∃ t)(F) and (∀ t)(F).
- The result of such a query is the set of all tuples t that satisfy COND (t).

Tuple Relational Calculus

- Example: To find the first and last names of all employees whose salary is above $50,000, we can write the following tuple calculus expression:

  {t.FNAME, t.LNAME | EMPLOYEE(t) AND t.SALARY>50000}

- The condition EMPLOYEE(t) specifies that the **range relation** of tuple variable t is EMPLOYEE.
- The first and last name (PROJECTION in relational algebra ($\pi_{FNAME, LNAME}$)) of each EMPLOYEE tuple t that satisfies the condition t.SALARY>50000 (SELECTION in relational algebra ($\sigma_{SALARY>50000}$)) will be retrieved.

The Existential and Universal Quantifiers

- Two special symbols called quantifiers can appear in formulas; these are the universal quantifier ($\forall$) and the existential quantifier ($\exists$).
- Informally, a tuple variable t is bound if it is quantified, meaning that it appears in an ($\forall$ t) or ($\exists$ t) clause; otherwise, it is free.
- If F is a formula, then so are ($\exists$ t)(F) and ($\forall$ t)(F), where t is a tuple variable.
  - The formula ($\exists$ t)(F) is true if the formula F evaluates to true for some (at least one) tuple assigned to free occurrences of t in F; otherwise ($\exists$ t)(F) is false.
  - The formula ($\forall$ t)(F) is true if the formula F evaluates to true for every tuple (in the universe) assigned to free occurrences of t in F; otherwise ($\forall$ t)(F) is false.

Example Query Using Existential Quantifier

- Retrieve the name and address of all employees who work for the 'Research' department. The query can be expressed as :

{t.FNAME, t.LNAME, t.ADDRESS | EMPLOYEE(t) and (∃ d) (DEPARTMENT(d) and d.DNAME='Research' and d.DNUMBER=t.DNO) }

- The only *free tuple variables* in a relational calculus expression should be those that appear to the left of the bar (|).
  - In above query, t is the only free variable; it is then *bound successively* to each tuple.
- If a tuple *satisfies the conditions* specified in the query, the attributes FNAME, LNAME, and ADDRESS are retrieved for each such tuple.
  - The conditions EMPLOYEE (t) and DEPARTMENT(d) specify the range relations for t and d.
  - The condition d.DNAME = 'Research' is a selection condition and corresponds to a SELECT operation in the relational algebra, whereas the condition d.DNUMBER = t.DNO is a JOIN condition.