Relational Algebra

Chapter 5

Contents

Contents

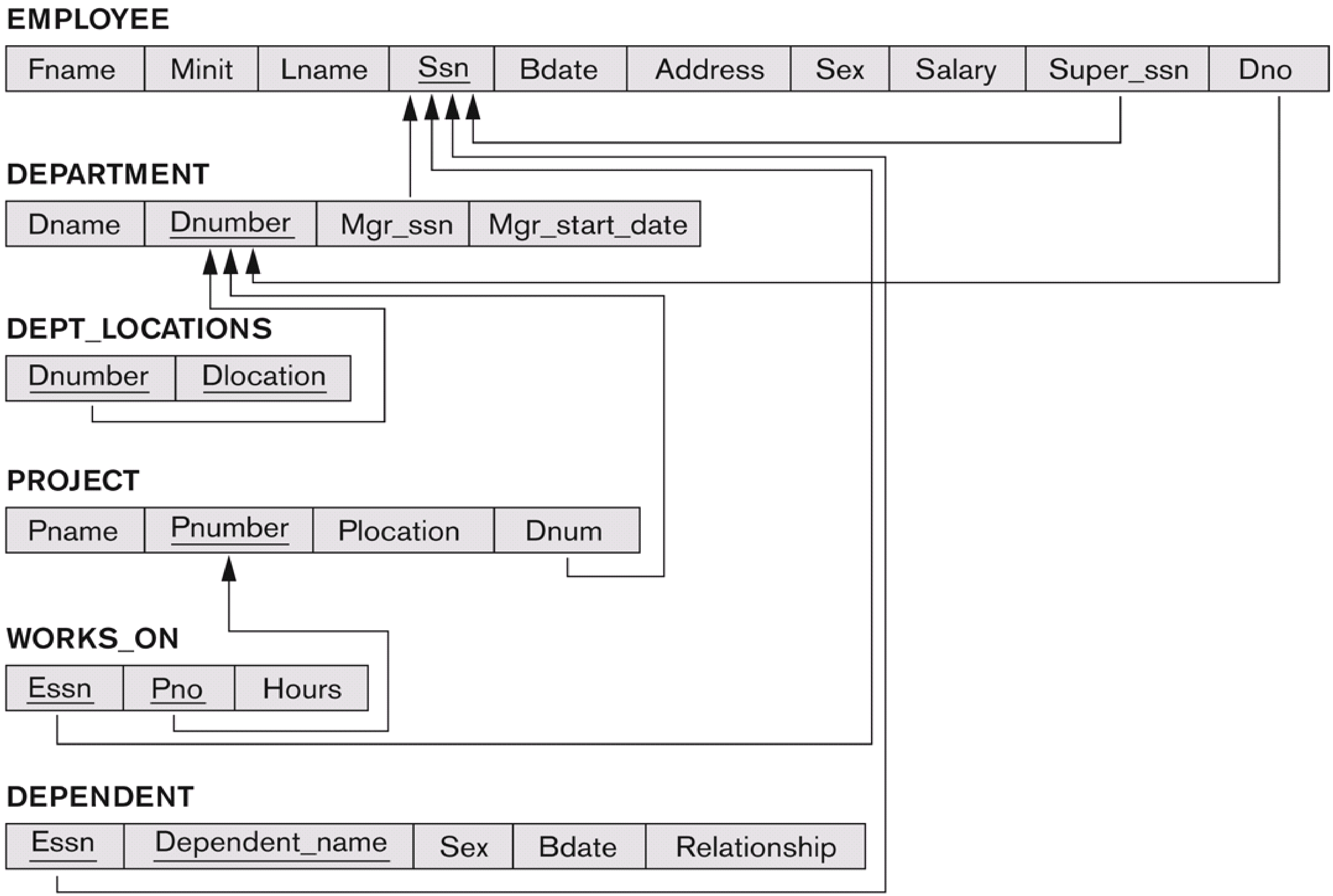
Relational Algebra Overview

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

Relational Algebra Overview

- Unary Relational Operations:
  - SELECT (symbol: **σ** (sigma))
  - PROJECT (symbol: **π** (pi))
  - RENAME (symbol: **ρ** (rho))
- Relational Algebra Operations from Set Theory:
  - UNION (∪), INTERSECTION (∩), DIFFERENCE (or MINUS, –)
  - CARTESIAN PRODUCT (**x**)
- Binary Relational Operations:
  - JOIN (several variations of JOIN exist)
  - DIVISION
- Additional Relational Operations:
  - OUTER JOINS, OUTER UNION
  - AGGREGATE FUNCTIONS (SUM, COUNT, AVG, MIN, MAX)

COMPANY Database Schema



The following query results refer to this database state

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

The following query results refer to this database state

DEPARTMENT

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

DEPT_LOCATIONS

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

WORKS_ON

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

PROJECT

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

DEPENDENT

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

Unary Relational Operations: SELECT

- The **SELECT** operation (denoted by **σ (sigma)**) is used to select a *subset* of the tuples from a relation based on a **selection condition**.
- Examples:
  - Select the EMPLOYEE tuples whose *department number is 4*:

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

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

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

Unary Relational Operations: SELECT

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

Unary Relational Operations: SELECT

- **SELECT** Operation Properties
  - The relation $S = \sigma_{<selection\ condition>}(R)$ has the same schema (same attributes) as R.
  - SELECT σ is **commutative**:

$$\sigma_{<cond1>}(\sigma_{<cond2>}(R)) = \sigma_{<cond2>}(\sigma_{<cond1>}(R))$$

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

$$\sigma_{<cond1>}(\sigma_{<cond2>}(\sigma_{<cond3>}(R)) = \sigma_{<cond2>}(\sigma_{<cond3>}(\sigma_{<cond1>}(R))) = \sigma_{<cond1>AND<cond2>AND<cond3>}(R)$$

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

Example of SELECT operation

R

A	B	C
1	2	3
4	5	6
1	2	7
8	4	5

$\sigma_{A \leq 4}$ **(R)**

A	B	C
1	2	3
4	5	6
1	2	7

Unary Relational Operations: PROJECT

- **PROJECT** Operation is denoted by π **(pi).**
- This operation keeps certain *columns* (attributes) from a relation and discards the other columns.
  - PROJECT creates a vertical partitioning: the list of specified columns (attributes) is kept in each tuple, the other attributes in each tuple are discarded.
- Example: To list each *employee's first and last name and salary*, the following is used:

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

Unary Relational Operations: PROJECT

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

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

  - <attribute list> is the desired list of attributes from relation R

- The project operation ***removes any duplicate tuples*** because the result of the **project** operation ***do not allow*** duplicate elements.

Unary Relational Operations: PROJECT

- **PROJECT** Operation Properties:
  - The number of tuples in the result of projection $\pi_{\text{<list>}}(R)$ is always ***less than 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.**
  - $\pi_{\text{<list1>}} (\pi_{\text{<list2>}} (R)) = \pi_{\text{<list1>}} (R)$?
    - *as long as <list2> contains the attributes in <list1>*

Example of PROJECT operation

R

A	B	C
1	2	3
4	5	6
1	2	7
8	4	5

$\pi_{A, B}$ **(R)**

A	B
1	2
4	5
8	4

Examples of applying SELECT and PROJECT operations

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

(a)

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

(b)

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

(c)

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

Relational Algebra Expressions

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

- In the latter case, we must give names to the relations that hold the intermediate results.

Single expression versus sequence of relational operations

- To retrieve the *first name, last name, and salary* of all employees who work in *department number 5*, we must apply a **select** and a **project** operation.

- We can write a *single relational algebra expression* as follows:

  $$\pi_{\text{FNAME, LNAME, SALARY}}(\sigma_{\text{DNO=5}}(\text{EMPLOYEE}))$$

- OR We can explicitly show the *sequence of operations*, giving a name to each intermediate relation:
  - $\text{DEP5\_EMPS} \leftarrow \sigma_{\text{DNO=5}}(\text{EMPLOYEE})$
  - $\text{RESULT} \leftarrow \pi_{\text{FNAME, LNAME, SALARY}}(\text{DEP5\_EMPS})$

Unary Relational Operations: RENAME

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

Unary Relational Operations: RENAME

- The general **RENAME** operation ρ can be expressed by any of the following forms:
  - $\rho_{S\ (B1,\ B2,\ ...,\ Bn\ )}(R)$ changes both:
    - the relation name to S, *and*
    - the column (attribute) names to B1, B1, …..Bn
  - $\rho_S(R)$ changes:
    - the *relation name* only to S
  - $\rho_{(B1,\ B2,\ ...,\ Bn\ )}(R)$ changes:
    - the *column (attribute) names* only to B1, B1, …..Bn

Contents

Relational Algebra Operations from Set Theory: UNION

- 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).

Example of the result of a UNION operation

RESULT1

Ssn
123456789
333445555
666884444
453453453

RESULT2

Ssn
333445555
888665555

RESULT

Ssn
123456789
333445555
666884444
453453453
888665555

Figure 6.3
Result of the UNION operation RESULT ← RESULT1 ∪ RESULT2.

Relational Algebra Operations from Set Theory

- **Type Compatibility** of operands is required for the binary set operation UNION $\cup$, (also for INTERSECTION $\cap$, SET DIFFERENCE –).

- The resulting relation for $R1 \cup R2$ (also for $R1 \cap R2$, or $R1 - R2$) has the ***same attribute names*** as the first operand relation R1 (by convention).

Relational Algebra Operations from Set Theory: INTERSECTION

- INTERSECTION is denoted by $\cap$.
- 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

- 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"**.

Figure 6.4
The set operations UNION, INTERSECTION, and MINUS. (a) Two union-compatible relations. (b) STUDENT ∪ INSTRUCTOR. (c) STUDENT ∩ INSTRUCTOR. (d) STUDENT – INSTRUCTOR. (e) INSTRUCTOR – STUDENT.

Example to illustrate the result of UNION, INTERSECT, 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

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 because both are ***associative*** operations:
  - $R \cup (S \cup T) = (R \cup S) \cup T$
  - $(R \cap S) \cap T = R \cap (S \cap T)$
- 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
  - Denoted by R(A1, A2, ..., An) **x** S(B1, B2, ..., Bm)
  - Result is a relation with degree **n + m** attributes:
    - Q(A1, A2, ..., An, B1, B2, ..., Bm), in that order.
  - 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”.

Example of CARTESIAN PRODUCT operation

R

A	B	C
1	2	3
4	5	6
1	2	7
8	4	5

T

A	D
1	5
3	7

R × T

R.A	B	C	T.A	D
1	2	3	1	5
1	2	3	3	7
4	5	6	1	5
4	5	6	3	7
1	2	7	1	5
1	2	7	3	7
8	4	5	1	5
8	4	5	3	7

Contents

Binary Relational Operations: JOIN

- **JOIN** Operation (denoted by $\bowtie$)
  - The sequence of **CARTESIAN PRODECT** followed by **SELECT** is used quite commonly to identify and select related tuples from two relations.
  - A special operation, called **JOIN** combines this sequence into a single operation.
  - This operation is very important for any relational database with more than a single relation, because it allows us ***combine related tuples*** from various relations.

Binary Relational Operations: JOIN

- **JOIN** Operation (denoted by $\bowtie$)
  - The general form of a join operation on two relations $R(A1, A2, \ldots, An)$ and $S(B1, B2, \ldots, Bm)$ is:

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

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

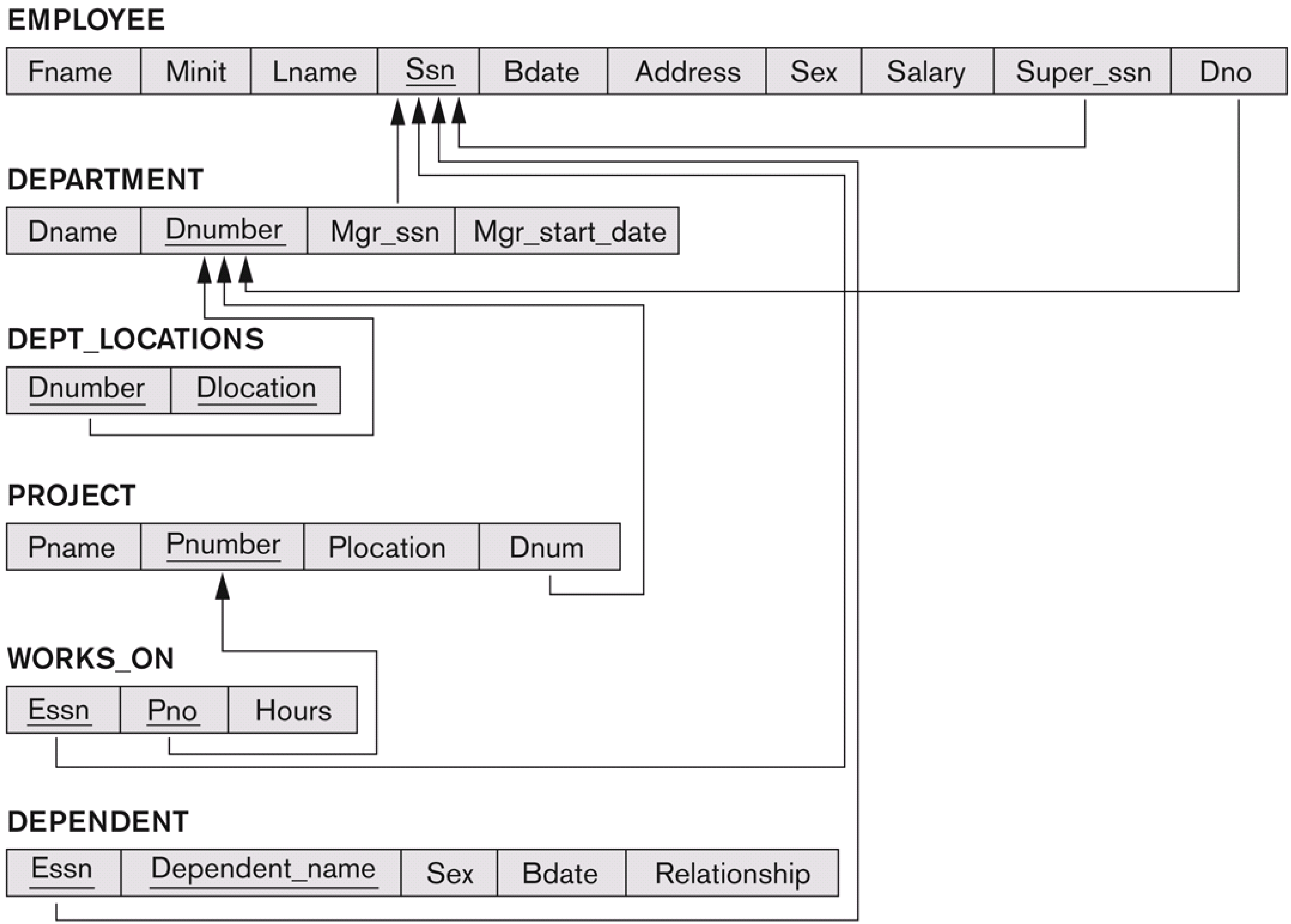
Binary Relational Operations: JOIN

- **Example:** Suppose that we want to retrieve *the name of the manager of each department*.
  - To get the manager's name, we need to *combine each DEPARTMENT tuple with the EMPLOYEE tuple* whose **SSN value matches the MGRSSN value** in the department tuple.

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

- MGRSSN = SSN is the *join condition.*
  - Combines each department record with the employee who manages the department.
  - The join condition can also be specified as: DEPARTMENT.MGRSSN= EMPLOYEE.SSN

COMPANY Database Schema



The following query results refer to this database state

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

The following query results refer to this database state

DEPARTMENT

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

DEPT_LOCATIONS

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

WORKS_ON

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

PROJECT

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

DEPENDENT

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

Example of applying the JOIN operation

DEPT_MGR

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

Result of the JOIN operation

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

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].
  - Hence, if R has n_R tuples, and S has n_S tuples, then the join result will generally have *less than* $n_R * n_S$ tuples.
  - Only related tuples (based on the join condition) will appear in the result.

Some properties of JOIN

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

- The join condition is called *theta.*

- *Theta* can be any general **boolean expression** on the attributes of R and S; for example:
  - R.Ai < S.Bj AND (R.Ak = S.Bl OR R.Ap < S.Bq)

Example of JOIN operation

R

A	B	C
1	2	3
4	5	6
1	2	7
8	4	5

$R \bowtie_{R.A > T.A} T$

R.A	B	C	T.A	D
4	5	6	1	5
4	5	6	3	7
8	4	5	1	5
8	4	5	3	7

T

A	D
1	5
3	7

Binary Relational Operations: EQUIJOIN

- 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.

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.
  - 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.
- Example: Q ← R(A,B,C,D) * S(C,D,E)
  - The implicit join condition includes *each pair* of attributes with the same name, "AND" ed together: R.C = S.C AND R.D = S.D
  - Result keeps only one attribute of each such pair:
    - Q(A,B,C,D,E)

Example of NATURAL JOIN operation

R

A	B	C
1	2	3
4	5	6
1	2	7
8	4	5

R * T

A	B	C	D
1	2	3	5
1	2	7	5

T

A	D
1	5
3	7

Example of NATURAL JOIN operation

(a)

PROJ_DEPT

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

(b)

DEPT_LOCS

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

Figure 6.7
Results of two NATURAL JOIN operations.
(a) PROJ_DEPT ← PROJECT * DEPT.
(b) DEPT_LOCS ← DEPARTMENT * DEPT_LOCATIONS.

Complete Set of Relational Operations

- The set of operations **{σ, π , $\cup$, - , X}** is called a **complete set** because any other relational algebra expressions can be expressed by a combination of these five operations.

- For example:
  - $R \cap S = (R \cup S) - ((R - S) \cup (S - R))$
  - $R \bowtie_{\text{<join condition>}} S = \sigma_{\text{<join condition>}} (R \times S)$

Binary Relational Operations: DIVISION

- **DIVISION Operation**
  - The division operation is applied to two relations $R(Z) \div S(X)$, where $Z = X \cup Y$ (Y is 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, i.e., 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.

Figure 6.8
The DIVISION operation. (a) Dividing SSN_PNOS by SMITH_PNOS. (b) $T \leftarrow R \div S$.

Example of the DIVISION operation

(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

Operations of Relational Algebra

Table 6.1 Operations of Relational Algebra

OPERATION	PURPOSE	NOTATION
SELECT	Selects all tuples that satisfy the selection condition from a relation R.	$\sigma_{\text{<selection condition>}}(R)$
PROJECT	Produces a new relation with only some of the attributes of R, and removes duplicate tuples.	$\pi_{\text{<attribute list>}}(R)$
THETA JOIN	Produces all combinations of tuples from R_1 and R_2 that satisfy the join condition.	$R_1 \bowtie_{\text{<join condition>}} R_2$
EQUIJOIN	Produces all the combinations of tuples from R_1 and R_2 that satisfy a join condition with only equality comparisons.	$R_1 \bowtie_{\text{<join condition>}} R_2$, OR $R_1 \bowtie_{(\text{<join attributes 1>}),\ (\text{<join attributes 2>})} R_2$
NATURAL JOIN	Same as EQUIJOIN except that the join attributes of R_2 are not included in the resulting relation; if the join attributes have the same names, they do not have to be specified at all.	$R_1 *_{\text{<join condition>}} R_2$, OR $R_1 *_{(\text{<join attributes 1>}),\ (\text{<join attributes 2>})} R_2$ OR $R_1 * R_2$

Operations of Relational Algebra

Table 6.1 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 $Z = X \cup Y$.	$R_1(Z) \div R_2(Y)$

Notation for Query Trees

- **Query tree**
  - Represents the input relations of query as leaf nodes of the tree.
  - Represents the relational algebra operations as internal nodes.

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

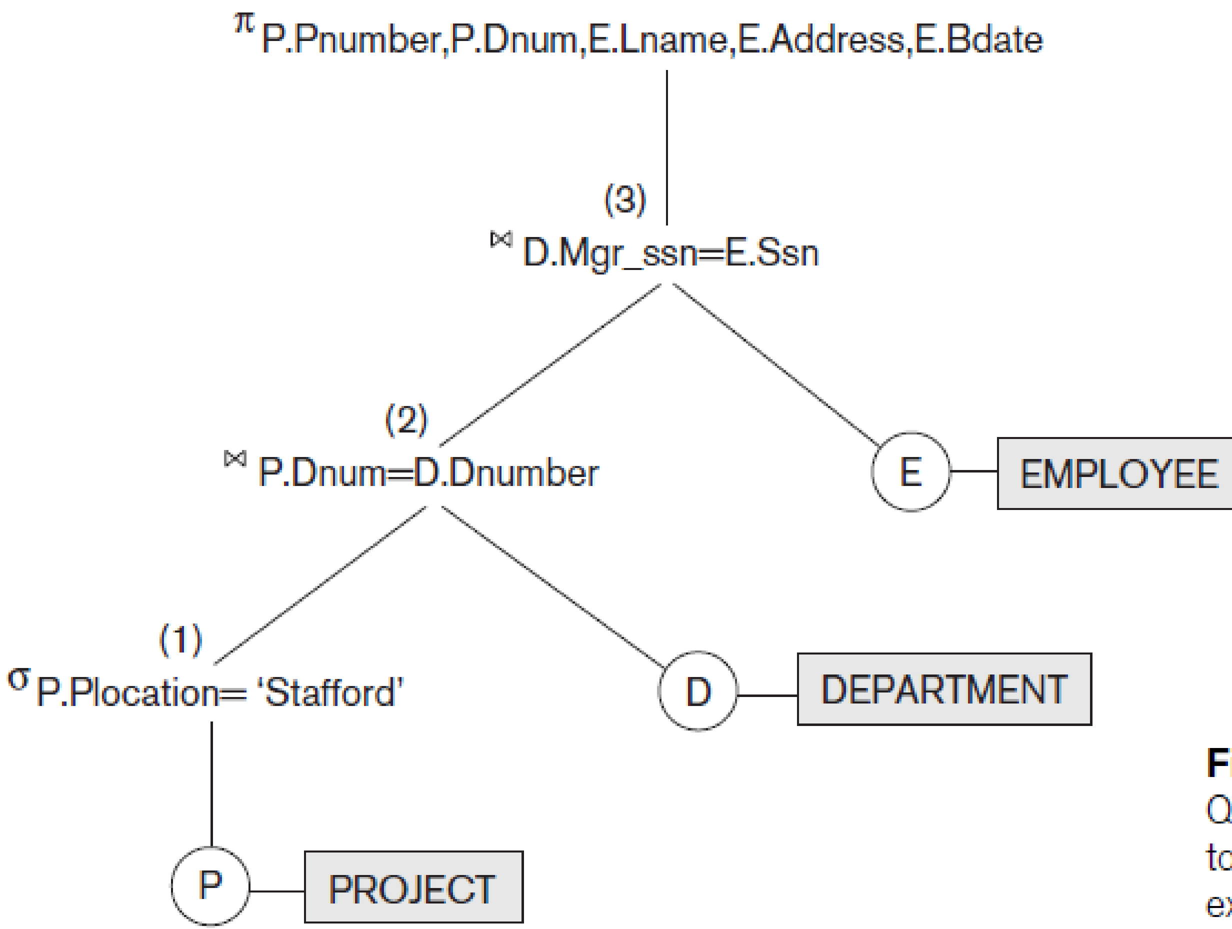


Figure 6.9
Query tree corresponding to the relational algebra expression for Q2.

Contents

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.
  - Common functions applied to collections of numeric values include **SUM, AVERAGE, MAXIMUM, and MINIMUM**. The **COUNT** function is used for counting tuples or values.

Additional Relational Operations

- **Use of the Functional operator $\mathscr{F}$**
  - $\mathscr{F}_{\text{MAX } \textit{Salary}}$ **(Employee)** retrieves the maximum salary value from the Employee relation
  - $\mathscr{F}_{\text{MIN } \textit{Salary}}$ **(Employee)** retrieves the minimum Salary value from the Employee relation
  - $\mathscr{F}_{\text{SUM } \textit{Salary}}$ **(Employee)** retrieves the sum of the Salary from the Employee relation
  - ${}_{\text{DNO}}\mathscr{F}_{\textit{COUNT SSN, AVERAGE Salary}}$ **(Employee)** groups employees by DNO (department number) and computes the count of employees and average salary per department.

  - **Note: count just counts the number of rows, without removing duplicates.**

Examples of applying aggregate functions and grouping

Figure 6.10
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. $\Im\,_{\text{COUNT Ssn, AVERAGE Salary}}(\text{EMPLOYEE})$.

R

(a)

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

(c)

Count_ssn	Average_salary
8	35125

(b)

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

Additional Relational Operations

- **Recursive Closure Operations**
  - Another type of operation that, in general, cannot be specified in the basic original relational algebra is **recursive closure.** This operation is applied to a **recursive relationship.**
  - An example of a recursive operation is to retrieve all SUPERVISEES of an EMPLOYEE e at all levels.
  - 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

- 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.
  - Outer Union operations: homework !!

Additional Relational Operations

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

The following query results refer to this database state

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

Additional Relational Operations

- Example: List all *employee names* and also the *name of the departments they manage* if they happen to manage a department (if they do not manage one, we can indicate it with a NULL value)

$$\text{TEMP} \leftarrow (\text{EMPLOYEE} \rtimes_{\text{Ssn=Mgr\_ssn}} \text{DEPARTMENT})$$

$$\text{RESULT} \leftarrow \pi_{\text{Fname, Minit, Lname, Dname}}(\text{TEMP})$$

Additional Relational Operations

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.

Example of LEFT OUTER JOIN

R

A	B	C
1	2	3
4	5	6
1	2	7
8	4	5

S

P	Q
1	5
3	7

$R \;⟕_{R.A = S.P}\; S$

A	B	C	P	Q
1	2	3	1	5
1	2	7	1	5
4	5	6	null	null
8	4	5	null	null

Example of RIGHT OUTER JOIN

R

A	B	C
1	2	3
4	5	6
1	2	7
8	4	5

S

P	Q
1	5
3	7

$R \rtimes_{R.A = S.P} S$

A	B	C	P	Q
1	2	3	1	5
1	2	7	1	5
null	null	null	3	7

Example of RIGHT OUTER JOIN

R

A	B	C
1	2	3
4	5	6
1	2	7
8	4	5

S

P	Q
1	5
3	7

$R \bowtie_{R.A = S.P} S$

A	B	C	P	Q
1	2	3	1	5
1	2	7	1	5
4	5	6	null	null
8	4	5	null	null
null	null	null	3	7

Contents

Brief Introduction to Relational Calculus

- A **relational calculus** expression creates a new relation, which is specified in terms of variables that range over rows of the stored database relations (in **tuple calculus**) or over columns of the stored relations (in **domain calculus**).
- In a calculus expression, there is *no order of operations* to specify how to retrieve the query result—a calculus expression specifies only what information the result should contain. This is the main distinguishing feature between relational algebra and relational calculus.
- Relational calculus is considered to be a **nonprocedural** language. This differs from relational algebra, where we must write a *sequence of operations* to specify a retrieval request; hence relational algebra can be considered as a **procedural** way of stating a query.

Brief Introduction to Relational Calculus

- The tuple relational calculus 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)}** where t is a tuple variable and COND (t) is a conditional expression involving t.
  - **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 ($\pi_{FNAME, LNAME}$) of each EMPLOYEE tuple t that satisfies the condition t.SALARY>50000 ($\sigma_{SALARY>50000}$) will be retrieved.

Brief Introduction to Relational Calculus

- Two special symbols called **quantifiers** can appear in formulas; these are the **universal quantifier** (∀) and the **existential quantifier** (∃).

- Informally, a tuple variable t is bound if it is quantified, meaning that it appears in an (∀ t) or (∃ t) clause; otherwise, it is **free.**

Brief Introduction to Relational Calculus

- **Example 1:** retrieve the name and address of all employees who work for the 'Research' department.

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

Brief Introduction to Relational Calculus

- **Example 2:** find the names of employees who work on *all* the projects controlled by department number 5.

  {e.LNAME, e.FNAME | EMPLOYEE(e) and ((∀ x) (not(PROJECT(x)) or not(x.DNUM=5)

  OR ((∃ w)(WORKS_ON(w) and w.ESSN=e.SSN and x.PNUMBER=w.PNO))))}

- Details: [1] Chapter 6

Brief Introduction to Relational Calculus

- Another variation of relational calculus called the domain relational calculus, or simply, **domain calculus** is equivalent to tuple calculus and to relational algebra.
- QBE (Query-By-Example): see Appendix D
- Domain calculus differs from tuple calculus in the *type of variables* used in formulas: rather than having variables range over tuples, the variables range over single values from domains of attributes. To form a relation of degree n for a query result, we must have n of these **domain variables** - one for each attribute.
- An expression of the domain calculus is of the form

  {x1, x2, . . ., xn | COND(x1, x2, . . ., xn, xn+1, xn+2, . . ., xn+m)}, where:

  - x1, x2, . . ., xn, xn+1, xn+2, . . ., xn+m are domain variables that range over domains (of attributes)
  - COND is a **condition** or **formula** of the domain relational calculus.

Brief Introduction to Relational Calculus

- Example: Retrieve the birthdate and address of the employee whose name is 'John B. Smith'.

  {uv | (∃ q) (∃ r) (∃ s) (∃ t) (∃ w) (∃ x) (∃ y) (∃ z) (EMPLOYEE(qrstuvwxyz) and q='John' and r='B' and s='Smith')}

Contents


Q
A

Exercise

1. Retrieve the *name and address of all employees who work for the 'Research' department.*
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.
3. Find the names of employees who work on *all* the projects controlled by department number 5.
4. List the names of all employees with two or more dependents.
5. Retrieve the names of employees who have no dependents.