UNION (U)

- R U S includes all tuples that are either in relation R <u>or</u> in S.

$$R \cup S = \{t \mid t \in R \text{ or } t \in S\}$$

- Duplicate tuples are eliminated.

- Get the SSNs of all employees who either work in department 5 <u>or</u> whose salary is greater than $30,000.

$$DEP5\_EMPS \leftarrow \pi_{SSN} (\sigma_{DNO=5} (EMPLOYEE))$$

$$HIGH\_SAL\_EMPS \leftarrow \pi_{SSN} (\sigma_{SALARY > 30000} (EMPLOYEE))$$

$$RESULT \leftarrow DEP5\_EMPS \cup HIGH\_SAL\_EMPS$$

UNION (∪)

- R ∪ S includes all tuples that are either in relation R or in S.

$$R \cup S = \{t \mid t \in R \text{ or } t \in S\}$$

- Duplicate tuples are eliminated.

- Get the SSNs of all employees who either work in department 5 or whose salary is greater than $30,000.

$$DEP5\_EMPS \leftarrow \pi_{SSN} (\sigma_{DNO=5} (EMPLOYEE))$$

$$HIGH\_SAL\_EMPS \leftarrow \pi_{SSN} (\sigma_{SALARY > 30000} (EMPLOYEE))$$

$$RESULT \leftarrow DEP5\_EMPS \cup HIGH\_SAL\_EMPS$$

UNION

R :

A	B
3	1
3	2
7	1

S :

A	B
3	2
7	3

R U S :

A	B
3	1
3	2
7	1
7	3

INTERSECTION (∩)

- $R \cap S$ includes all tuples that are in <u>both</u> R <u>and</u> S.

$$R \cap S = \{t \mid t \in R \text{ and } t \in S\}$$

- Duplicate tuples are eliminated.

- Get the SSNs of all employees who work in department 5 <u>and</u> whose salary is greater than $30,000.

$$DEP5\_EMPS \leftarrow \pi_{SSN} (\sigma_{DNO = 5} (EMPLOYEE))$$

$$HIGH\_SAL\_EMPS \leftarrow \pi_{SSN} (\sigma_{SALARY > 30000} (EMPLOYEE))$$

$$RESULT \leftarrow DEP5\_EMPS \cap HIGH\_SAL\_EMPS$$

INTERSECTION

R :

A	B
3	1
3	2
7	1

S :

A	B
3	2
7	3

R ∩ S :

A	B
3	2

DIFFERENCE (–)

- R – S includes all tuples that are in R, but not in S.

$$R - S = \{t \mid t \in R \text{ and } t \notin S\}$$

- Duplicate tuples are eliminated.

- Get the SSNs of all employees who work in department 5, but whose salary is not greater than $30,000.

$DEP5\_EMPS \leftarrow \pi_{SSN}(\sigma_{DNO=5}(EMPLOYEE))$

$HIGH\_SAL\_EMPS \leftarrow \pi_{SSN}(\sigma_{SALARY > 30000}(EMPLOYEE))$

$RESULT \leftarrow DEP5\_EMPS - HIGH\_SAL\_EMPS$

DIFFERENCE

R :

A	B
3	1
3	2
7	1

S :

A	B
3	2
7	3

R – S :

A	B
3	1
7	1

Type Compatibility:

- ∪, ∩, – 의 3 연산자는 다음 조건을 만족해야 사용가능.

- 즉, relation R과 S는 다음 두 조건을 만족해야 함.

  1) R과 S의 attribute의 개수가 서로 같음.

  2) 같은 위치에 서로 대응하는 attribute의 domain이 같음.

  : domain(Ai) = domain(Bi) for $1 \le i \le n$

- 참고 : attribute의 이름은 서로 달라도 상관 없음.

- 이 경우 (관례상) 결과 relation은 첫 번째 relation R의 attribute 들의 이름들을 가짐.

연산 규칙

- Union과 Intersection : 교환(commutative) 법칙 성립

$$R \cup S = S \cup R$$

$$R \cap S = S \cap R$$

- Union과 Intersection : 결합(associative) 법칙 성립

$$R \cup (S \cup T) = (R \cup S) \cup T$$

$$(R \cap S) \cap T = R \cap (S \cap T)$$

- Difference : 교환(commutative) 법칙 성립 안 함.

$$R - S \neq S - R$$

CARTESIN PRODUCT (x)

- R x S combines both tuples from relation R and S.

$$R \times S = \{rs \mid r \in R \text{ and } s \in S\}$$

- 즉, R $(A_1, A_2, \ldots, A_m)$ x S $(B_1, B_2, \ldots, B_n)$ 의 결과는 Q $(A_1, A_2, \ldots, A_m, B_1, B_2, \ldots, B_n)$로 정의됨. 단,

  (1) Q는 (m + n) 개의 attribute들을 가짐.

  (2) Q는 R과 S에 속한 각 tuple들을 모두 연결하여 합침

- If R has m tuples (|R| = m 로 표기) and S has n tuples, then, |R x S| = |R| * |S| = m * n .

CARTESIN PRODUCT

R :

A	B
a1	b1
a2	b2

S :

C	D	E
c1	d1	e1
c2	d1	e2
c2	d2	e2
c3	d3	e3

R x S :

A	B	C	D	E
a1	b1	c1	d1	e1
a1	b1	c2	d1	e2
a1	b1	c2	d2	e2
a1	b1	c3	d3	e3
a2	b2	c1	d1	e1
a2	b2	c2	d1	e2
a2	b2	c2	d2	e2
a2	b2	c3	d3	e3

CARTESIN PRODUCT

- Get all combined tuples of both departments and employees.

DEPT

DNO	dname	mgr-ssn
d1	DB	22222
d2	security	33333
d3	network	55555

EMP

ssn	ename	age
11111	abe	28
22222	bob	46
33333	ann	31
44444	jim	50
55555	eve	38

DEPT x EMP

(Total 15 tuples)

DNO	dname	mgr-ssn	ssn	ename	age
d1	DB	22222	11111	abe	28
d1	DB	22222	22222	bob	46
d1	DB	22222	33333	ann	31
d1	DB	22222	44444	jim	50
d1	DB	22222	55555	eve	38
d2	security	33333	11111	abe	28
. . . .	. . . .	. . . . . .	. . . . . .	. . . . . .	
d3	network	55555	55555	eve	38

JOIN ($\bowtie$)

- $R(A_1, A_2, ..., A_n)$과 $S(B_1, B_2, ..., B_m)$의 **JOIN**은 다음과 같이 표기.

$$R \underset{condition}{\bowtie} S$$

- R과 S의 JOIN 결과인 $Q(A_1, A_2, ..., A_n, B_1, B_2, ..., B_m)$는
  (1) Q는 (m + n) 개의 attribute들을 가짐.
  (2) Q는 R과 S에 속한 각 tuple들 중 join condition을 만족하는 것들만 연결하여 합침.

- **(Join) Condition**은 다음과 같이 표현.

$$\mathbf{A_i}\ op\ \mathbf{B_j} \quad (\text{단, } op \in \{=, \neq, >, \geq, <, \leq\})$$

- A_i 와 B_j 는 **join attribute**라 하며, 각각 R과 S에 속하고, A_i 와 B_j 는 서로 같은 domain(= data type)을 가짐.

JOIN ($\bowtie$)

R :

A	B
a1	b1
a2	b3

S :

C	D	E
b1	d1	e1
b2	d1	e2
b3	d2	e2
b3	d3	e3

$R \underset{B = C}{\bowtie} S$

A	B	C	D	E
a1	b1	b1	d1	e1
a2	b3	b3	d2	e2
a2	b3	b3	d3	e3

JOIN (⋈)

R :

A	B
a1	b1
a2	b3

S :

C	D	E
b1	d1	e1
b2	d1	e2
b3	d2	e2
b3	d3	e3

$R \bowtie_{B \neq C} S$

A	B	C	D	E
a1	b1	b2	d1	e2
a1	b1	b3	d2	e2
a1	b1	b3	d3	e3
a2	b3	b1	d1	e1
a2	b3	b2	d1	e2

Types of Join

- If JOIN involves join condition with equality comparison(=) only, such a join is called an **Equi-Join**.

- Equi- Join is the most widely used operator; For example, for implementing a relationship between relations (PK vs FK).

- Otherwise, if the other comparisons $\{\neq, >, \geq, <, \leq\}$ are used in join condition, such a join is called a **Condition(= Theta) Join**.

- Tuples whose join attributes are **null**, or join condition is evaluated to **false** do not appear in the join result.

Types of Join

- Notice that in the result of equi-join, we always have pairs of join attributes that have **identical** values in every tuple.

- Because one of the pair of join attributes identical values is redundant, we may need to remove it.

- In this case, **Natural Join** (denoted by *) is created to project out (remove) the second join attribute in equi-join result.

- Natural join generally requires that two join attributes have the same name in both relations; If not, renaming is applied first.

Equi-Join

- Get manager information for each department.

DEPT

DNO	dname	mgr-ssn
d1	DB	33333
d2	security	2222
d3	network	55555

EMP

ssn	ename	age
11111	abe	28
22222	bob	46
33333	ann	31
44444	jim	50
55555	eve	38

$$\text{DEPT} \bowtie_{\text{mgr-ssn} = \text{ssn}} \text{EMP}$$

DNO	dname	mgr-ssn	ssn	ename	age
d1	DB	33333	33333	ann	31
d2	security	22222	22222	bob	46
d3	network	55555	55555	eve	38

Condition Join

- Get flights information connecting from Seoul to Guam.

SEO-JPN

flt-no	dept	arr
100	08:00	10:30
200	10:00	12:30
300	11:00	13:30

JPN-GUAM

flight	depart	arrive
400	10:00	16:00
500	12:00	18:00
600	13:00	19:00

$$\text{SEO-JPN} \bowtie_{\text{arr} < \text{depart}} \text{JPN-GUAM}$$

flt-no	dept	arr	flight	depart	arrive
100	08:00	10:30	500	12:00	18:00
100	08:00	10:30	600	13:00	19:00
200	10:00	12:30	600	13:00	19:00

Natural Join

- Get department's information for each working employee.

EMP

ssn	ename	DNO
11111	abe	d2
22222	bob	d1
33333	ann	d2
44444	jim	d3

DEPT

DNO	dname	phone
d1	DB	290-10
d2	security	390-11
d3	network	590-55

EMP * DEPT
DNO = DNO

ssn	ename	DNO	dname	phone
11111	abe	d2	security	390-11
22222	bob	d1	DB	290-10
33333	ann	d2	security	390-11
44444	jim	d3	network	590-55

- Note : The second join attribute is removed in join result.

Equi-Join : Self Join

- Get the names and ages of direct supervisors of employees with age = 52.

EMP

SSN	name	age	Super-SSN
11111	bob	52	22222
22222	cal	38	33333
33333	tom	52	44444
44444	abe	60	null
55555	sam	54	44444

$$\text{Supervisor} \leftarrow \pi_{\text{Super-SSN}} (\sigma_{\text{age} = 52} (\text{EMPLOYEE}))$$

$$\text{RESULT} \leftarrow \pi_{\text{name, age}} (\text{Supervisor} \bowtie_{\text{Super-SSN} = \text{SSN}} \text{EMP})$$

Join : Exercise

COURSE

CID	cname
CS200	OS
CS250	DB
CS300	PL

ENROLL

CID	SID	credit
CS200	12345	3
CS200	23456	3
CS300	23456	2
CS250	23456	3
CS250	45678	3

STUDENT

SID	sname	age
12345	bob	22
23456	ann	18
34567	jim	30
45678	eve	27

- Get SIDs of students who enroll courses.

$$\pi_{SID}(\text{ENROLL})$$

- Get names of students who enroll courses.

$$\pi_{sname}(\text{ENROLL} \bowtie_{SID = SID} \text{STUDENT})$$

- Get names of students who enroll "DB" course.

$$\pi_{sname}((\sigma_{cname=DB}(\text{COURSE})) \bowtie_{CID = CID} \text{ENROLL} \bowtie_{SID = SID} \text{STUDENT}))$$

Join : Exercise

COURSE

CID	cname
CS200	OS
CS250	DB
CS300	PL

ENROLL

CID	SID	credit
CS200	12345	3
CS200	23456	3
CS300	23456	2
CS250	23456	3
CS250	45678	3

STUDENT

SID	sname	age
12345	bob	22
23456	ann	18
34567	jim	30
45678	eve	27

- Get names of students who do not enroll "DB" course.
- Get names of courses enrolled by student with ID "23456.
- Get names of courses enrolled by student with name "ann.

DIVISION (÷)

- Division : $R(Z) \div S(X) = T(Y)$ where $X \subseteq Z$;
  - Let $Y = Z - X$; Thus, $Z = X \cup Y$;
    (즉, Y는 relation S에는 없고, R에만 속한 attribute(s))

- 위 DIVISION 연산의 결과는 relation T(Y) 임. 단,
  - T(Y) 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 T(Y), the values in t must appear in R in combination with **every** tuple in S.

- DIVISION is useful for special kind of query that satisfy "***all*** .... condition"

DIVISION

R

A	B
a1	b1
a1	b2
a2	b1
a2	b2
a2	b3
a3	b2
a3	b3
a4	b1
a4	b2
a4	b3
a5	b2

S

B
b1
b2
b3

R ÷ S

A
a2
a4

R

A	B	C	D	E
a1	b1	c1	d1	e1
a1	b1	c3	d2	e3
a1	b2	c2	d1	e1
a1	b2	c2	d2	e3
a2	b2	c3	d1	e1
a2	b2	c3	d2	e2
a3	b2	c2	d1	e1
a3	b2	c2	d2	e3

S

D	E
d1	e1
d2	e3

R ÷ S

A	B	C
a1	b2	c2
a3	b2	c2

DIVISION

- Get SSNs of employees who work on **all** the projects:

WORK-ON

SSN	pno	hours
11111	p1	15
11111	p2	20
22222	p1	18
22222	p2	25
22222	p3	10
22222	p4	30
3333	p2	20
33333	p3	40
33333	p4	30

PROJECT

pno	pname	budget
p1	laptop	500M
p2	printer	700M
p3	mp3	400M
p4	memory	800M

RESULT

SSN
22222

$\text{TEMP1} \leftarrow \pi_{\text{pno}} (\text{PROJECT})$

$\text{TEMP2} \leftarrow \pi_{\text{ssn, pno}} (\text{WORK\_ON})$

$\text{RESULT} \leftarrow \text{TEMP2} \div \text{TEMP1}$

DIVISION : Exercise

COURSE

CID	cname
CS200	OS
CS250	DB
CS300	PL

ENROLL

CID	SID	credit
CS200	12345	3
CS200	23456	3
CS300	23456	2
CS250	23456	3
CS250	45678	3

STUDENT

SID	sname	age
12345	bob	22
23456	ann	18
34567	jim	30
45678	eve	27

- Get SID of students who enroll **all** the courses;

  $$\text{RESULT1} \leftarrow \pi_{\text{CID, SID}} (\text{ENROLL}) \div \pi_{\text{CID}} (\text{COURSE})$$

- Get names of students who enroll **all** the courses;

  $$\text{RESULT2} \leftarrow \pi_{\text{sname}} (\text{RESULT1} \bowtie_{\text{SID = SID}} \text{STUDENT})$$

DIVISION : Exercise

COURSE

<u>CNO</u>	cname
CS200	OS
CS250	DB
CS300	PL

ENROLL

CID	SID	credit
CS200	12345	3
CS200	23456	3
CS300	23456	2
CS250	23456	3
CS250	45678	3

STUDENT

<u>SNO</u>	sname	age
12345	bob	22
23456	ann	18
34567	jim	30
45678	eve	27

- Get SID of students who enroll **both** OS and DB courses;

  TEMP1 $\leftarrow$ π_{CNO} ($\sigma_{\text{came = 'OS' OR 'DB'}}$ (COURSE))

  RESULT $\leftarrow$ $\pi_{\text{CID, SID}}$ (ENROLL) $\div$ TEMP1

- Get name of courses enrolled by **all** the students with age < 27;

Summary: 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 $R_1 * R_2$
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 $Z = X \cup Y$.	$R_1(Z) \div R_2(Y)$

Complete Set

- 다음 5 개의 기본 연산을 "Complete Set" 이라 함;
  - Select, Project , Union, Difference, Cartesian Product
- 다른 연산들은 위의 연산들부터 정의(유도)할 수 있음.
  - Intersection : $R \cap S = R - (R - S)$
  - Join : $R \bowtie_{<condition>} S = \sigma_{<condition>} (R \times S)$
  - Division : $R(Z) \div S(X)$ where $Z = X \cup Y$ is derived by;

    ① $T1 = \pi_Y (R)$

    ② $T2 = \pi_Y ((T1 \times S) - R)$

    ③ $RESULT = T1 - T2$

Other Relational Algebra

- Outer Joins
- Outer Union
- Aggregate Functions
- Recursive Query

OUTER JOIN

- In (natural) join operation, **non-matching** tuples are lost from the join result ; Tuples with **null** in the join attributes are also lost.; To keep those lost tuples in the result, outer joins are introduced;

- LEFT OUTER JOIN (R $⟕$ S) keeps every tuple in R.
  - If there are no matching tuples in S, then the attributes of S in the join result are filled with null values.
- RIGHT OUTER JOIN (R $⟖$ S) keeps every tuple in S.
  - If there are no matching tuples in R, then the attributes of S in the join result are filled with null values.
- FULL OUTER JOIN (R $⟗$ S) keeps all tuples in both R and S.
  - If there are no matching tuples, the attributes of S and R in the join result are filled with null values

OUTER JOIN

Course

CID	cname	credit
C170	database	3
C230	math	2
C260	network	3

Student

sname	CID
bob	C170
joe	C230
eve	null

$\text{course} \;⟕_{CID\,=\,CID}\; \text{student}$

CID	cname	credit	sname
C170	database	3	bob
C230	math	2	joe
C260	network	3	null

$\text{course} \;⟖_{CID\,=\,CID}\; \text{student}$

CID	cname	credit	sname
C170	database	3	bob
C230	math	2	joe
null	null	null	eve

$\text{course} \;⟗_{CID\,=\,CID}\; \text{student}$

CID	cname	credit	sname
C170	database	3	bob
C230	math	2	joe
C260	network	3	null
null	null	null	eve

Figure 3.6
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

- 사원들 중에서 각 부서의 메니저들의 이름과 그 메니저가 관리하는 부서명을 구하라. (단, 모든 사원들의 이름도 함께 결과에 나와야 함)

$$TEMP \leftarrow (EMPLOYEE \mathbin{⟕}_{SSN = MgrSSN} DEPARTMENT)$$

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

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.

OUTER UNION (⊎)

- OUTER UNION is used to union two relations if the relations are not compatible.
- It takes the union of tuples in R(X, Y) and S(X, Z) that are partially compatible, meaning that only some of their attributes, say X, are union compatible.
- Attributes that are not compatible from either relation, say, Y and Z, are also kept in the result relation T(X, Y, Z).
- No matching tuples in either relation are added with null values for attributes Y or Z.

OUTER UNION

- Example: We want union the following relations;

  STUDENT(SSN, Name, Dept, Advisor)

  INSTRUCTOR(SSN, Name, Dept, Rank).

  - Tuples from the two relations are matched based on having the same combination of values of the shared attributes, {Name, SSN, Department}.
  - If a student is also an instructor, both {Advisor, Rank} will have a value; Otherwise, one of these two attributes will be null.
  - The result relation STUDENT_OR_INSTRUCTOR will have the following attributes:

  STUD_OR_INSTR (SSN, Name, Dept, Advisor, Rank)

OUTER UNION

STUDENT

SSN	name	dept	gpa
11111	bob	math	3.55
33333	jim	physics	4.20
44444	jim	math	2.79
77777	abe	law	3.45
88888	sam	physics	2.80

INSTRUCTOR

SSN	name	dept	rank
33333	jim	physics	TA
77777	abe	law	TA
99999	eva	math	prof

STUDENT ∪ INSTRUCTOR

SSN	name	dept	gpa	rank
11111	bob	math	3.55	null
33333	jim	physics	4.20	TA
44444	jim	math	2.79	null
77777	abe	law	3.45	TA
88888	sam	physics	2.80	null
99999	eva	math	null	prof

Rename Operator(ρ)

- In some cases, we may want to rename the attributes of a relation, or the relation name, or both.

- RENAME operator ρ can be expressed by the following forms;
  - $\rho_{S\ (B1,\ B2,\ \ldots,\ Bn\ )}(R)$ changes both:
    - ✓ the relation name to S,
    - ✓ the attribute names to $B_1, B_2, \ldots, B_n$
  - $\rho_S(R)$ changes:
    - ✓ the relation *name* only to S
  - $\rho_{(B1,\ B2,\ \ldots,\ Bn\ )}(R)$ changes:
    - ✓ the attribute names only to $B_1, B_1, \ldots, B_n$

Aggregate Functions / Grouping

- Aggregate Functions
  - COUNT : count the number of values
  - SUM : total sum of values
  - AVG : average of values
  - MAX : maximum of values
  - MIN : minimum of values

- Grouping
  - GROUPING : Group tuples based on some attribute value

- 표기 방법 : $_{\text{<grouping attributes>}}\mathcal{F}_{\text{<function list>}}(R)$

where <function list> is a list of (<function><attribute>)

Aggregate Functions

EMPLOYEE

SSN	name	DNO	salary
11111	bob	d5	30000
22222	jim	d5	40000
33333	jim	d4	25000
44444	abe	d4	43000
55555	sam	d5	38000
66666	tom	d5	25000
77777	eva	d4	25000
88888	tim	d1	55000

$\mathcal{F}$ $\text{COUNT}_{\text{SSN}}, \text{AVERAGE}_{\text{salary}}$ (EMPLOYEE)

Count_SSN	Average_sal
8	35125

Aggregate Functions

EMPLOYEE

SSN	name	DNO	salary
11111	bob	d5	30000
22222	jim	d5	40000
33333	jim	d4	25000
44444	abe	d4	43000
55555	sam	d5	38000
66666	tom	d5	25000
77777	eva	d4	25000
88888	tim	d1	55000

$\text{DNO}\ \mathcal{F}\ \text{COUNT}_{\text{SSN}},\ \text{AVERAGE}_{\text{salary}}\ (\text{EMPLOYEE})$

DNO	count-SSN	Average-sal
d5	4	33250
d4	3	31000
d1	1	55000

Aggregate Functions

EMPLOYEE

SSN	name	DNO	salary
11111	bob	d5	30000
22222	jim	d5	40000
33333	jim	d4	25000
44444	abe	d4	43000
55555	sam	d5	38000
66666	tom	d5	25000
77777	eva	d4	25000
88888	tim	d1	55000

$$\rho_{R(\text{DNO, no-emps, avg-sal})}\ {}_{\text{DNO}}\mathcal{F}\ \text{COUNT}_{\text{SSN}},\ \text{AVERAGE}_{\text{salary}}(\text{EMPLOYEE})$$

R :

DNO	no-emps	avg-sal
d5	4	33250
d4	3	31000
d1	1	55000

Recursive Query

- This query is used for **recursive** relationship between the tuples of **the same** relation.

- 예 :

  Retrieve all employees supervised by some employee e at all levels,

  - all employees e′ directly supervised by e;
  - all employees e′′ directly supervised by each employee e′;
  - all employees e′′′ directly supervised by each employee e′′;
  - so on .

- The SQL3 standard includes syntax for recursive closure.

Recursive Query

- Get SSNs of all employees (by up to level 2) supervised by 'Tim'.

(level 1)

$Tim\_SSN \leftarrow \pi_{SSN} (\sigma_{name= 'tim'} (EMPLOYEE))$

$SUPERVISE (SSN1, SSN2) \leftarrow \pi_{SSN, SUPERSSN} (EMPLOYEE)$

$RESULT1 (SSN) \leftarrow \pi_{SSN1} (SUPERVISE \bowtie_{SSN2=SSN} Tim\_SSN)$

(level 2)

$RESULT2(SSN) \leftarrow \pi_{SSN1} (SUPERVISE \bowtie_{SSN2=SSN} RESULT1)$

(level 1 plus 2)

$RESULT \leftarrow RESULT1 \cup RESULT2$

Recursive Query

EMPLOYEE

SSN	name	age	super-SSN
11111	bob	60	22222
22222	jim	40	88888
33333	jim	45	44444
44444	abe	45	88888
55555	sam	60	22222
66666	tom	25	55555
77777	eva	50	33333
88888	tim	64	null

SUPERVISE

SSN1	SSN2
11111	22222
22222	88888
33333	44444
44444	88888
55555	22222
66666	55555
77777	33333
88888	null

RESULT1

SSN
22222
44444

at level 1

∪

RESULT2

SSN
11111
55555
33333

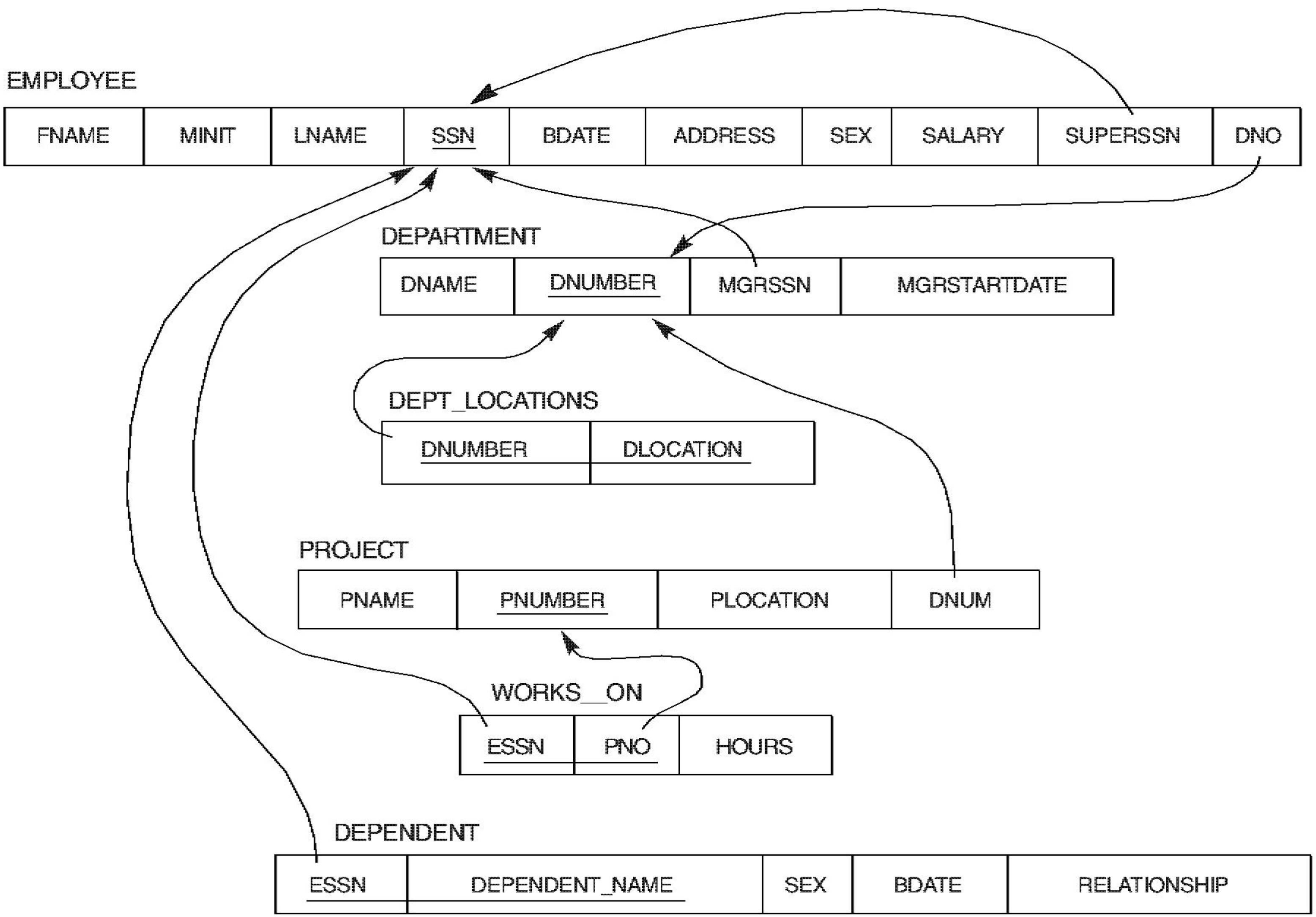
at level 2

=

RESULT

SSN
22222
44444
11111
55555
33333

Figure 7.7 Referential integrity constraints displayed on the COMPANY relational database schema diagram.



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

Exercise : Queries

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

$$\text{RESEAR\_DEPT} \leftarrow \sigma_{\text{Dname} = \text{'Research'}}(\text{DEPARTMENT})$$

$$\text{RESEAR\_EMPS} \leftarrow (\text{RESEAR\_DEPT} \bowtie_{\text{Dnumber} = \text{DNO}} \text{EMPLOYEE})$$

$$\text{RESULT} \leftarrow \pi_{\text{Fname, Lname, Address}}(\text{RESEAR\_EMPS})$$

- Q2 : Retrieve the names of employees who have no dependents.

$$\text{ALL\_EMPS} \leftarrow \pi_{\text{SSN}}(\text{EMPLOYEE})$$

$$\text{EMPS\_WITH\_DEPS(SSN)} \leftarrow \pi_{\text{ESSN}}(\text{DEPENDENT})$$

$$\text{EMPS\_WITHOUT\_DEPS} \leftarrow (\text{ALL\_EMPS} - \text{EMPS\_WITH\_DEPS})$$

$$\text{RESULT} \leftarrow \pi_{\text{Fname, Lname}}(\text{EMPS\_WITHOUT\_DEPS} * \text{EMPLOYEE})$$

- Q3 : Retrieve the name of employees who work on **all** the projects controlled by department 5.

  $\text{DEPT5\_PROJ(PNO)} \leftarrow \pi_{\text{Pnumber}} (\sigma_{\text{DNUM} = 5} (\text{PROJECT}))$

  $\text{EMP\_PROJ(SSN, PNO)} \leftarrow \pi_{\text{ESSN, PNO}} (\text{WORK\_ON})$

  $\text{RESULT\_EMP\_SSN} \leftarrow \text{EMP\_PROJ} \div \text{DEPT5\_PROJ}$

  $\text{RESULT} \leftarrow \pi_{\text{Fname Lname}} (\text{RESULT\_EMP\_SSN} * \text{EMPLOYEE})$

- Q4 : Retrieve the name of managers who have at least one dependent.

  $\text{MGR(SSN)} \leftarrow \pi_{\text{MgrSSN}} (\text{DEPARTMENT})$

  $\text{EMP\_WITH\_DEP(SSN)} \leftarrow \pi_{\text{ESSN}} (\text{DEPENDENT})$

  $\text{MGR\_WITH\_DEP} \leftarrow \text{MGR} \cap \text{EMP\_WITH\_DEP}$

  $\text{RESULT} \leftarrow \pi_{\text{Fname, Lname}} (\text{MGR\_WITH\_DEP} * \text{EMPLOYEE})$

Q5 : 위치가 Houston인 부서에서 근무하고 급여가 35,000보다 적게 받는 사원들의 부양가족들의 이름을 구하라.

$T1 \leftarrow \pi_{Dnumber} (\sigma_{Dlocation = Houston} (DEPT\text{-}LOCATIONS))$

$T2 \leftarrow \pi_{SSN} (\sigma_{Salary < 35000} (T1 \bowtie_{Dnumber = DNO} EMPLOYEE))$

$RESULT \leftarrow \pi_{DepName} (T2 \bowtie_{SSN = ESSN} DEPENDENT)$

Q6 : 남자 사원들의 직속 상사이면서, 각 부서의 manager가 수행하는 프로젝트의 이름과 위치를 구하라.

$T1(SSN) \leftarrow \pi_{SuperSSN} (\sigma_{Sex = Male} (EMPLOYEE))$

$T2(SSN) \leftarrow \pi_{MgrSSN} (DEPARTMENT)$

$T3 \leftarrow T1 \cap T2$

$T4 \leftarrow \pi_{PNO} (T3 \bowtie_{SSN = ESSN} WORK\text{-}ON)$

$RESULT \leftarrow \pi_{Pname, Ploaction} (T4 \bowtie_{PNO = Pnumber} PROJECT)$

Q7 : Alice의 부모 (단, Alice와 성별이 다름)의 직속 상사가 근무하는 부서의 manager의 이름을 구하라.

$T1 \leftarrow \sigma_{\text{DeptName = Alice}} (\text{DEPENDENT})$

$T2 \leftarrow \pi_{\text{SuperSSN}} (T1 \bowtie_{\text{(ESSN = SSN) AND (Sex <> Sex)}} \text{EMPLOYEE})$

$T3 \leftarrow \pi_{\text{DNO}} (T2 \bowtie_{\text{SuperSSN = SSN}} \text{EMPLOYEE})$

$T4 \leftarrow \pi_{\text{MgrSSN}} (T3 \bowtie_{\text{DNO = Dnumber}} \text{DEPARTMENT})$

$\text{RESULT} \leftarrow \pi_{\text{Fname, Lname}} (T4 \bowtie_{\text{MgrSSN = SSN}} \text{EMPLOYEE})$

Exercise : Queries

Q8 : 아들과 딸을 각각 모두 자녀로 갖고 있는 결혼한 사원들의 이름을 구하라

Q9 : 본인의 직속 상사와 급여가 같은 사원들의 SSN을 구하라.

Q10 : Research 부서에서 근무하는 여자 사원들이 모두 담당해서 수행하는 프로젝트들의 이름을 구하라.

Q11 : 모든 사원들의 SSN, 이름, 급여, 주소를 구하라. 단, 이들 중 manager가 있다면 이들의 SSN, 이름, 주소, 급여, 관리 부서명도 함께 구해야 함.