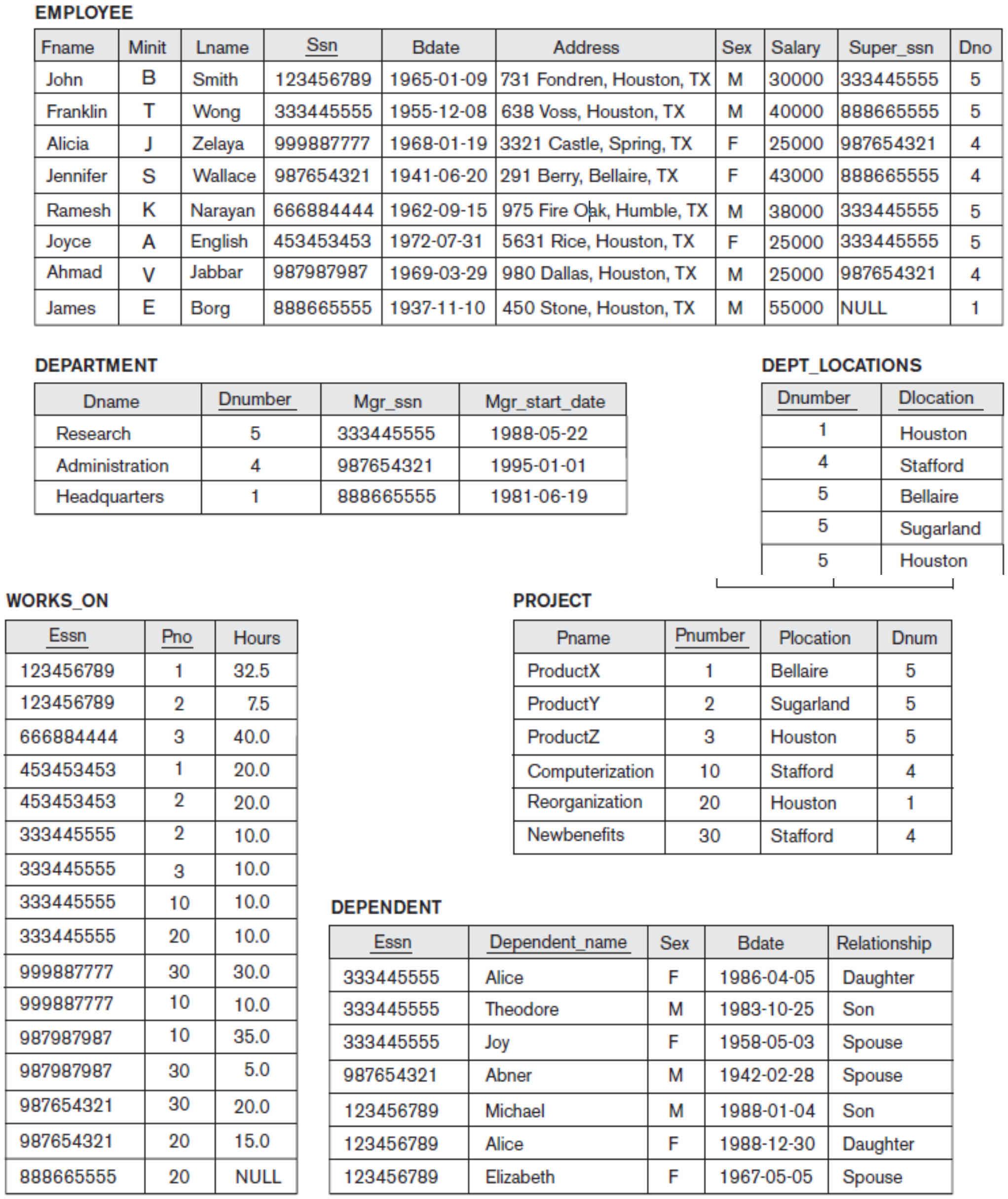
Database used



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

Summary

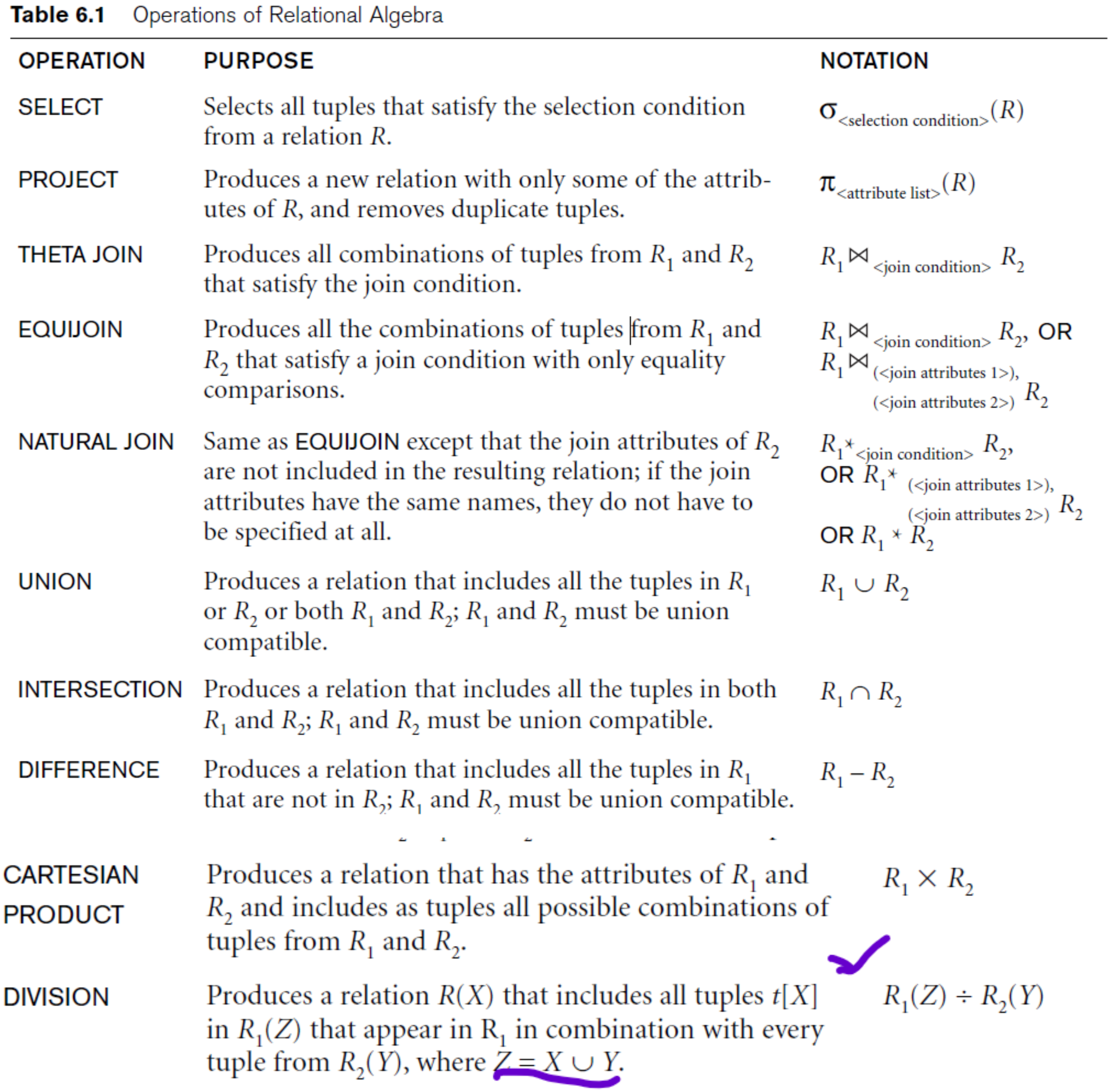


Table 6.1 Operations of Relational Algebra

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 $\underline{Z = X \cup Y}$.	✓ $R_1(Z) \div R_2(Y)$

The Relational Algebra and Relational Calculus

1. the two *formal languages* for the relational model
2. each relation is defined to be a set of tuples in the *formal* relational model

Unary Relational Operations - SELECT and PROJECT

SELECT

1. The SELECT operation is used to choose a *subset* of the tuples from a relation that satisfies a **selection condition**
2. select the EMPLOYEE tuples whose department is 4, or those whose salary is greater than $30,000,

$$\sigma_{\text{Dno}=4}(\text{EMPLOYEE})$$
$$\sigma_{\text{Salary}>30000}(\text{EMPLOYEE})$$

In general, the SELECT operation is denoted by

$$\sigma_{<\text{selection condition}>}(R)$$

where the symbol σ (sigma) is used to denote the SELECT operator, *R* is generally a *relational algebra expression* whose result is a Relation

3. Horizontal partition
4. select the tuples for all employees who either work in department 4 and make over $25,000 per year, or work in department 5 and make over $30,000

$$\sigma_{(\text{Dno}=4 \ \textbf{AND} \ \text{Salary}>25000) \ \textbf{OR} \ (\text{Dno}=5 \ \textbf{AND} \ \text{Salary}>30000)}(\text{EMPLOYEE})$$

5. All the comparison operators in the set $\{=, <, \leq, >, \geq, \neq\}$ can apply to attributes whose domains are ordered values, such as numeric or date domains.
6. If the domain of an attribute is a set of unordered values, then only the comparison operators in the set $\{=, \neq\}$ can be used. An example of an unordered domain is the domain Color = { 'red', 'blue', 'green', 'white', 'yellow',...}, where no order is specified among the various colors.
7. The **degree** of the relation resulting from a SELECT operation—its number of attributes—is the same as the degree of *R*
8. The number of tuples in the resulting relation is always *less than or equal to* the number of tuples in *R*.

$$|\sigma_c(R)| \leq |R| \text{ for any condition } C.$$

9. SELECT operation is **commutative**

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

$$\sigma_{<\text{cond1}>}(\sigma_{<\text{cond2}>}(\ldots(\sigma_{<\text{cond}n>}(R))\ \ldots)) = \sigma_{<\text{cond1}>\ \textbf{AND}<\text{cond2}>\ \textbf{AND}\ldots\textbf{AND}\ <\text{cond}n>}(R)$$

10. In SQL, the SELECT condition is typically specified in the WHERE clause of a query.

$$\sigma_{\text{Dno}=4 \ \textbf{AND} \ \text{Salary}>25000}(\text{EMPLOYEE})$$

would correspond to the following SQL query:

```
SELECT     *
FROM       EMPLOYEE
WHERE      Dno=4 AND Salary>25000;
```

The PROJECT Operation

1. The PROJECT operation, selects certain columns from the table and discards the other columns
2. Vertical partition
3. to list each employee's first and last name and salary, we can use the PROJECT operation as follows:

$$\pi_{\text{Lname, Fname, Salary}}(\text{EMPLOYEE})$$

4. The general form of the PROJECT operation is

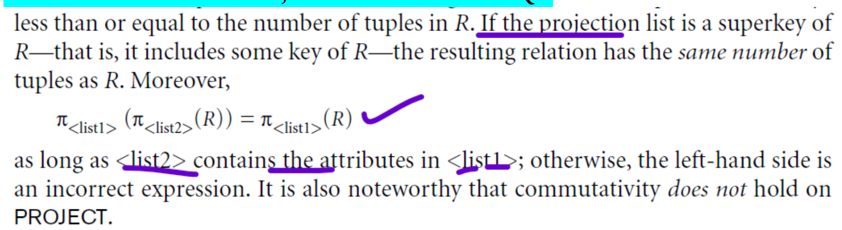
$$\pi_{<\text{attribute list}>}(R)$$

5. **The degree** is equal to the number of attributes in <attribute list>.

$$\pi_{\text{Sex, Salary}}(\text{EMPLOYEE})$$

It is 2 in the above syntax

6. The result of the PROJECT operation is a set of distinct tuples. If duplicates are not eliminated, the result would be a multiset or bag of tuples rather than a set. This was not permitted in the formal relational model, but is allowed in SQL

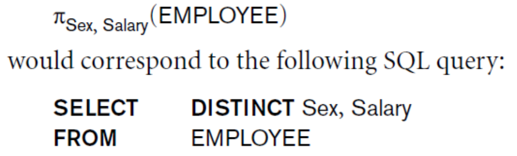


less than or equal to the number of tuples in R. If the projection list is a superkey of R—that is, it includes some key of R—the resulting relation has the *same number* of tuples as R. Moreover,

$$\pi_{<\text{list1}>}(\pi_{<\text{list2}>}(R)) = \pi_{<\text{list1}>}(R)$$

as long as <list2> contains the attributes in <list1>; otherwise, the left-hand side is an incorrect expression. It is also noteworthy that commutativity *does not* hold on PROJECT.

7. The number of tuples in a relation resulting from a PROJECT operation is always less than or equal to the number of tuples in R.
8. Commutativity does not hold on PROJECT.
9. In SQL, the PROJECT attribute list is specified in the SELECT clause of a query. For example, the following operation:



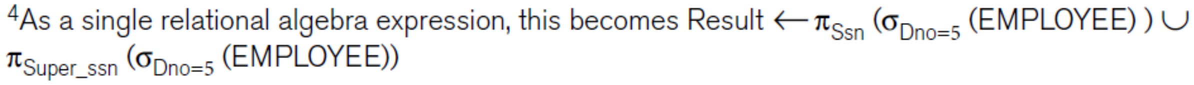
$$\pi_{\text{Sex, Salary}}(\text{EMPLOYEE})$$

would correspond to the following SQL query:

```
SELECT    DISTINCT Sex, Salary
FROM      EMPLOYEE
```

UNION, INTERSECTION, and SET DIFFERENCE (MINUS) Operations

1. This is a binary set operation. Two relations $R(A1, A2, ..., An)$ and $S(B1, B2, ..., Bn)$ are said to be **union compatible** (or **type compatible**) if they have the same degree n and if dom(Ai) = dom(Bi)
2. Duplicate tuples are eliminated in UNION
3. Retrieve the Social Security numbers of all employees who either work in department 5 or directly supervise an employee who works in department 5, we can use the UNION operation as follows-



[4]As a single relational algebra expression, this becomes Result $\leftarrow \pi_{\text{Ssn}} (\sigma_{\text{Dno}=5} (\text{EMPLOYEE})) \cup \pi_{\text{Super\_ssn}} (\sigma_{\text{Dno}=5} (\text{EMPLOYEE}))$

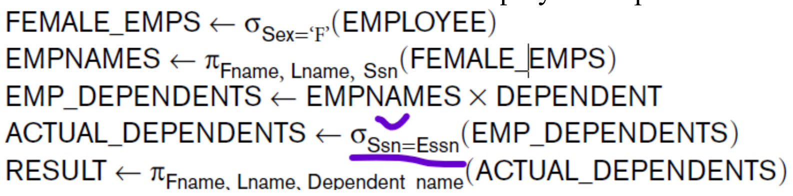
4. UNION and INTERSECTION are commutative and *associative* operations

$$R \cup S = S \cup R \quad \text{and} \quad R \cap S = S \cap R$$

5. The MINUS operation is *not commutative*
6. In SQL, there are three operations—UNION, INTERSECT, and EXCEPT
7. $R \cap S = ((R \cup S) - (R - S)) - (S - R)$

CARTESIAN PRODUCT (CROSS PRODUCT) Operation (CROSS PRODUCT or CROSS JOIN)—

1. This is also a binary set operation, but the relations on which it is applied do not have to be union compatible
2. This set operation produces a new element by combining every member (tuple) from one relation (set) with every member (tuple) from the other relation (set).
3. Retrieve a list of names of each female employee's dependents



FEMALE_EMPS $\leftarrow \sigma_{\text{Sex='F'}}(\text{EMPLOYEE})$
EMPNAMES $\leftarrow \pi_{\text{Fname, Lname, Ssn}}(\text{FEMALE\_EMPS})$
EMP_DEPENDENTS $\leftarrow$ EMPNAMES × DEPENDENT
ACTUAL_DEPENDENTS $\leftarrow \sigma_{\text{Ssn=Essn}}(\text{EMP\_DEPENDENTS})$
RESULT $\leftarrow \pi_{\text{Fname, Lname, Dependent\_name}}(\text{ACTUAL\_DEPENDENTS})$

CARTESIAN PRODUCT followed by SELECT is quite commonly used to combine related tuples from two relations, a special operation, called JOIN, was created to specify this sequence as a single operation

```
Q1A:   SELECT   Fname, Lname, Address
       FROM     (EMPLOYEE JOIN DEPARTMENT ON Dno=Dnumber)
       WHERE    Dname='Research';
```

JOIN and DIVISION

1. The JOIN operation can be specified as a CARTESIAN PRODUCT operation followed by a SELECT operation

   $$\text{DEPT\_MGR} \leftarrow \text{DEPARTMENT} \bowtie_{\text{Mgr\_ssn=Ssn}} \text{EMPLOYEE}$$
   $$\text{RESULT} \leftarrow \pi_{\text{Dname, Lname, Fname}}(\text{DEPT\_MGR})$$

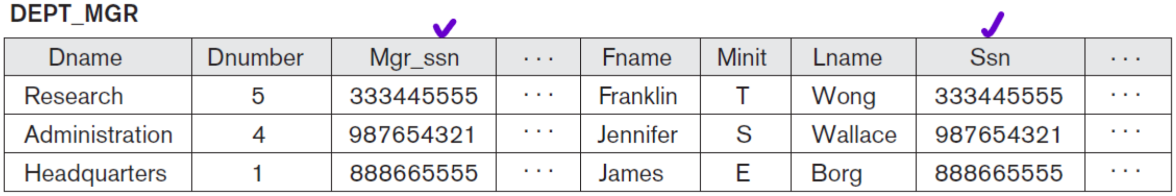
   The result of the JOIN is a relation Q with n + m attributes

   This is the main difference between CARTESIAN PRODUCT and JOIN. In JOIN, only combinations of tuples satisfying the join condition appear in the result, whereas in the CARTESIAN PRODUCT all combinations of tuples are included in the result.
2. Tuples whose join attributes are NULL or for which the join condition is FALSE do not appear in the result.
3. A general join condition is of the form <condition> AND <condition> AND...AND <condition> where each <condition> is of the form Ai θ Bj, Ai is an attribute of R, Bj is an attribute of S, Ai and Bj have the same domain, and θ (theta) is one of the comparison operators {=, <, ≤, >, ≥, ≠}. A JOIN operation with such a general join condition is called a THETA JOIN

The EQUIJOIN and NATURAL JOIN

1. The most common use of JOIN involves join conditions with equality comparisons only. Such a JOIN, where the only comparison operator used is =, is called as EQUIJOIN.
2. Result of an EQUIJOIN we always have one or more pairs of attributes that have identical values in every tuple.

DEPT_MGR



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

Figure 6.6
Result of the JOIN operation $\text{DEPT\_MGR} \leftarrow \text{DEPARTMENT} \bowtie_{\text{Mgr\_ssn=Ssn}} \text{EMPLOYEE}$. ✓

1. To remove the second (superfluous) attribute in an EQUIJOIN condition, NATURAL JOIN (*) is used.
2. Natural join requires that the two join attributes must have the same name in both relations. If this is else renaming operation is applied first.
   Example-

   $$\text{PROJ\_DEPT} \leftarrow \text{PROJECT} * \rho_{(\text{Dname, Dnum, Mgr\_ssn, Mgr\_start\_date})}(\text{DEPARTMENT})$$

   $$Q \leftarrow R *_{(<\text{list1}>),(<\text{list2}>)} S$$

Complete Set of Relational Algebra Operations

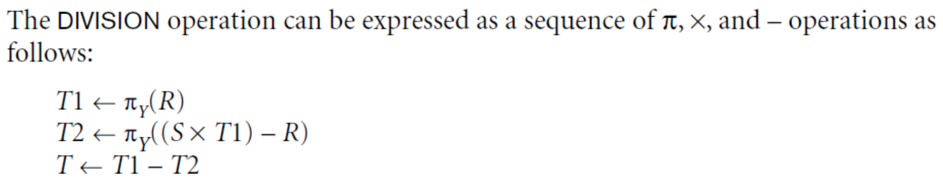
$$\{\sigma, \pi, \cup, \rho, -, \times\}$$

1. A set of relational algebra operations is a complete set; that is, any of the other original relational algebra operations can be expressed as a sequence of operations from this set.
2. a JOIN operation can be specified as a CARTESIAN PRODUCT followed by a SELECT operation
3. a NATURAL JOIN can be specified as a CARTESIAN PRODUCT preceded by RENAME and followed by SELECT and PROJECT operations

The DIVISION Operation

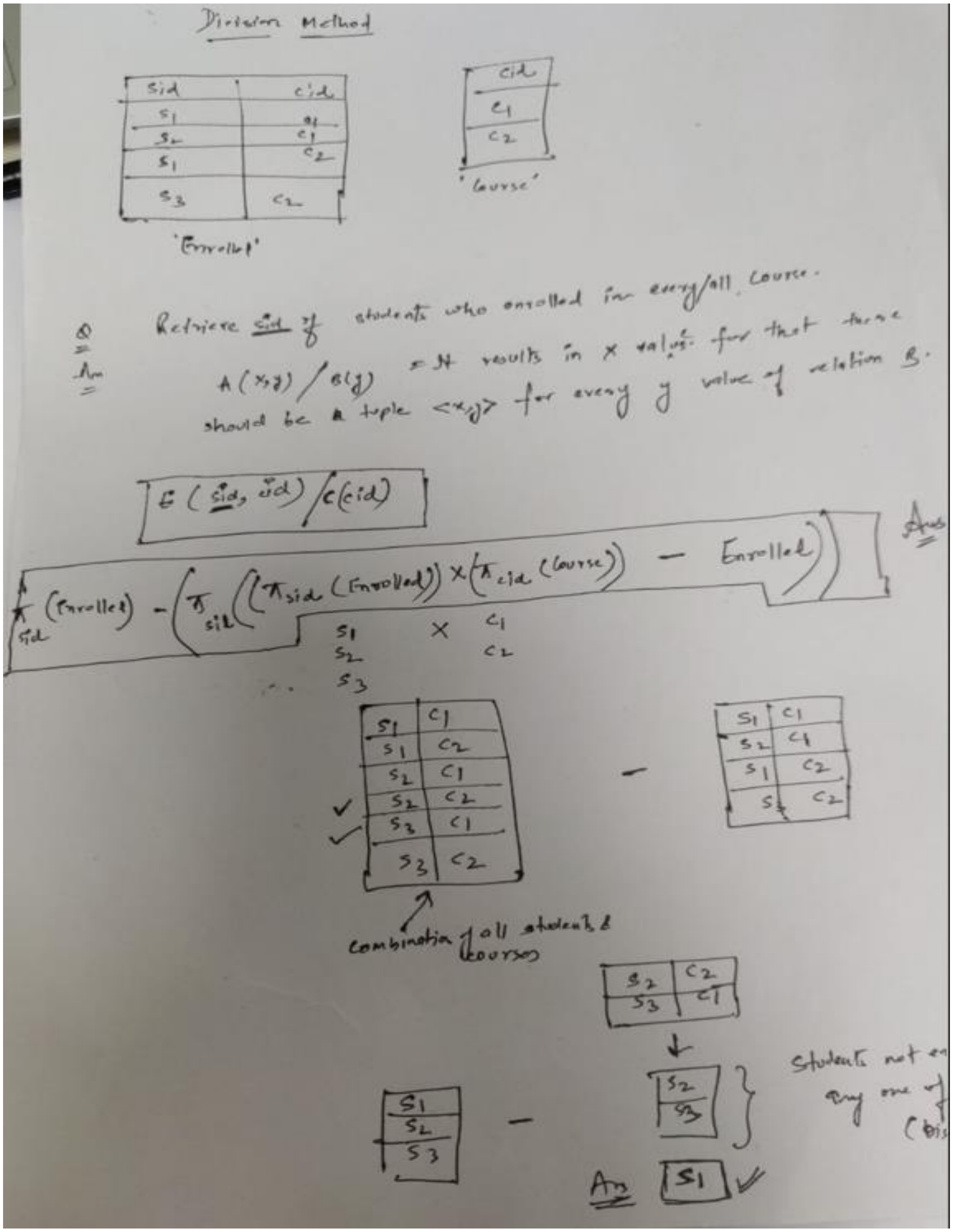
https://www.youtube.com/watch?v=lbvCaETL2FI

The DIVISION operation is defined for convenience for dealing with queries that involve universal quantification or the all condition.



The DIVISION operation can be expressed as a sequence of π, $\times$, and $-$ operations as follows:

$$T1 \leftarrow \pi_Y(R)$$
$$T2 \leftarrow \pi_Y((S \times T1) - R)$$
$$T \leftarrow T1 - T2$$



Division Method

sid	cid
s_1	c_1
s_2	c_1
s_1	c_2
s_3	c_2

'Enrolled'

cid
c_1
c_2

'Course'

Q Retrieve sid of students who enrolled in every/all course.

Ans $A(x,y) / B(y)$ = It results in x values for that there should be a tuple $<x,y>$ for every y value of relation B.

$\boxed{E(\underline{sid}, cid) / C(cid)}$

Ans

$$\pi_{sid}(\text{Enrolled}) - \left(\pi_{sid}\left(\left(\pi_{sid}(\text{Enrolled})\right) \times \left(\pi_{cid}(\text{Course})\right) - \text{Enrolled}\right)\right)$$

s_1 × c_1
s_2 c_2
s_3

s_1	c_1
s_1	c_2
s_2	c_1
✓ s_2	c_2
✓ s_3	c_1
s_3	c_2

combination of all students & courses

−

s_1	c_1
s_2	c_1
s_1	c_2
s_3	c_2

s_2	c_2
s_3	c_1

↓

s_2
s_3

Students not en any one of (Dis

s_1
s_2
s_3

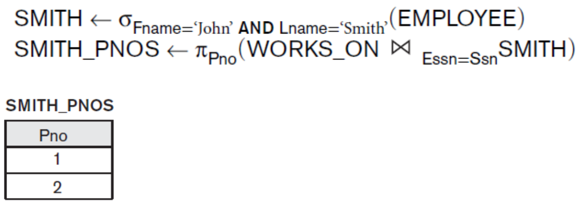
−

s_2
s_3

Ans $\boxed{s_1}$ ✓

Q. *Retrieve the names of employees who work on **all** the projects that 'John Smith' works on.*

a) First, retrieve the list of project numbers that 'John Smith' works on in the intermediate relation SMITH_PNOS:

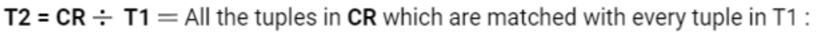


$\text{SMITH} \leftarrow \sigma_{\text{Fname='John' AND Lname='Smith'}}(\text{EMPLOYEE})$

$\text{SMITH\_PNOS} \leftarrow \pi_{\text{Pno}}(\text{WORKS\_ON} \bowtie_{\text{Essn=Ssn}} \text{SMITH})$

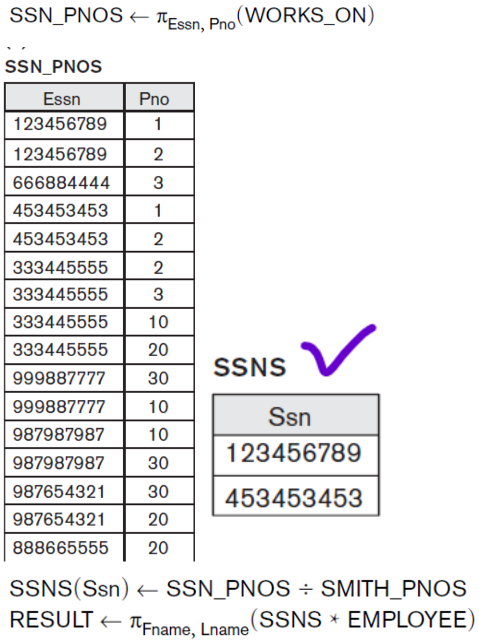
SMITH_PNOS

Pno
1
2



T2 = CR $\div$ **T1** = All the tuples in **CR** which are matched with every tuple in T1 :

b)



$\text{SSN\_PNOS} \leftarrow \pi_{\text{Essn, Pno}}(\text{WORKS\_ON})$

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

SSNS ✓

Ssn
123456789
453453453

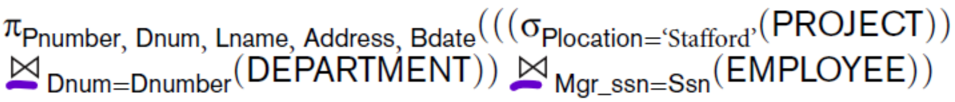
$\text{SSNS(Ssn)} \leftarrow \text{SSN\_PNOS} \div \text{SMITH\_PNOS}$

$\text{RESULT} \leftarrow \pi_{\text{Fname, Lname}}(\text{SSNS} * \text{EMPLOYEE})$

Query Tree

Q. Generate query tree for-

For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, address, and birth date.



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

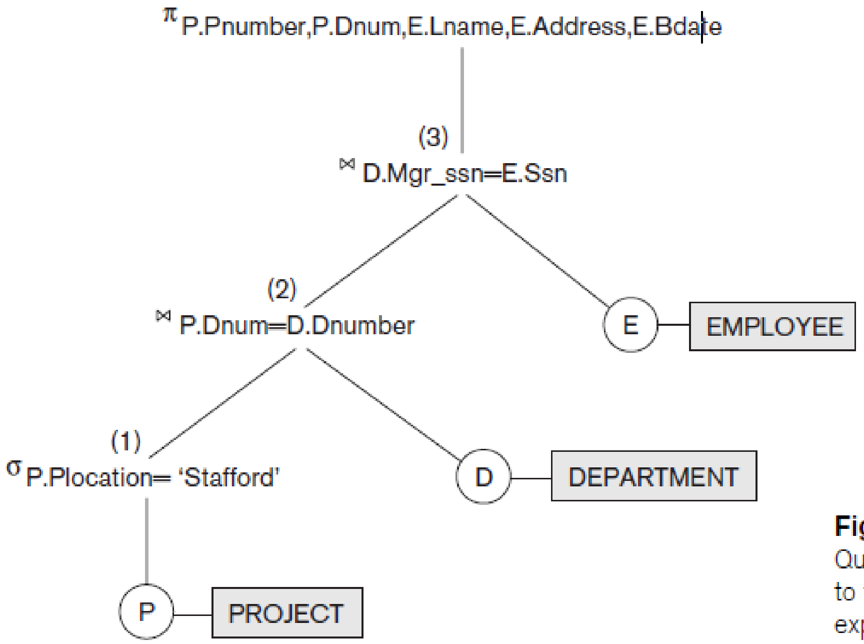


Figure 6.9
Query tree corresponding to the relational algebra expression for Q2.

Generalized Projection

The generalized projection operation extends the projection operation by allowingfunctions of attributes to be included in the projection list.

As an example, consider the relation

$$\text{EMPLOYEE}(\text{Ssn}, \text{Salary}, \text{Deduction}, \text{Years\_service})$$

A report may be required to show

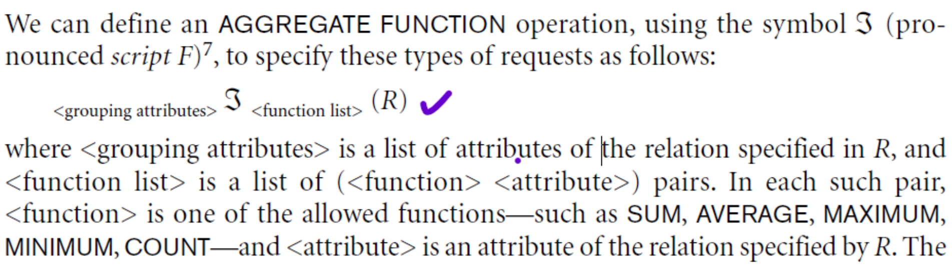
$$\text{Net Salary} = \text{Salary} - \text{Deduction},$$
$$\text{Bonus} = 2000 * \text{Years\_service}, \text{ and}$$
$$\text{Tax} = 0.25 * \text{Salary}.$$

Then a generalized projection combined with renaming may be used as follows:

✓ $$\text{REPORT} \leftarrow \rho_{(\text{Ssn}, \text{Net\_salary}, \text{Bonus}, \text{Tax})}(\pi_{\text{Ssn}, \text{Salary} - \text{Deduction}, 2000 * \text{Years\_service}, 0.25 * \text{Salary}}(\text{EMPLOYEE})).$$

Aggregate Functions and Grouping

Request that cannot be expressed in the basic relational algebra are aggregate functions such as group by function and mathematical aggregate functions, and recursive closure

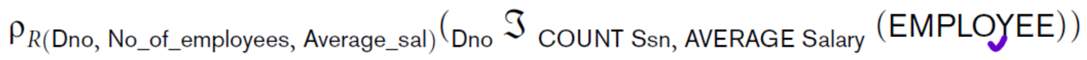


We can define an AGGREGATE FUNCTION operation, using the symbol $\Im$ (pronounced *script F*)[7], to specify these types of requests as follows:

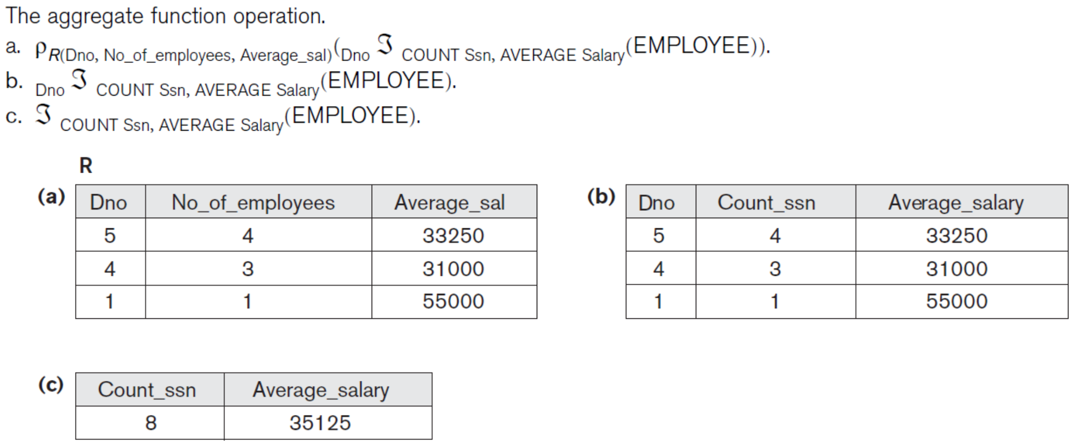
$$_{\text{<grouping attributes>}}\ \Im_{\ \text{<function list>}}\ (R)$$ ✓

where <grouping attributes> is a list of attributes of the relation specified in R, and <function list> is a list of (<function> <attribute>) pairs. In each such pair, <function> is one of the allowed functions—such as SUM, AVERAGE, MAXIMUM, MINIMUM, COUNT—and <attribute> is an attribute of the relation specified by R. The

Q. retrieve each department number, the number of employees in the department, and their average salary, while renaming the resulting attributes



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



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

(b)

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

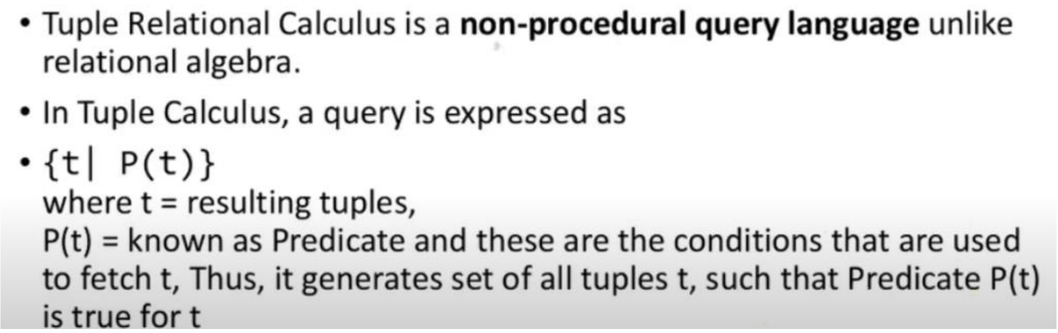
(c)

Count_ssn	Average_salary
8	35125

Tuple Relational Calculus

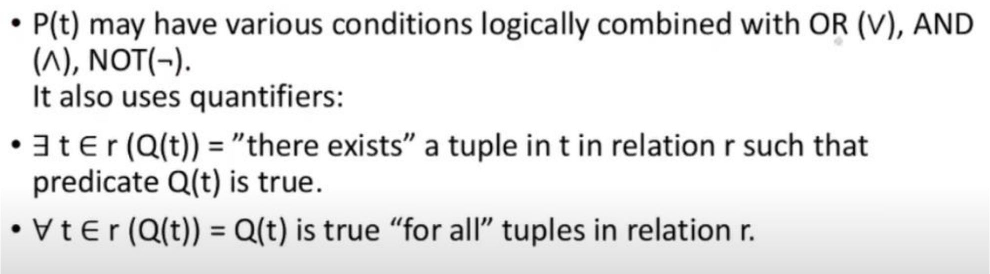
Relational Calculus -> Non-procedural- what to do?

Relational Algebra→ Procedural- How to do?



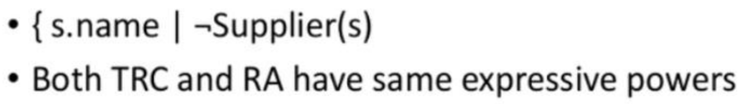
- Tuple Relational Calculus is a **non-procedural query language** unlike relational algebra.
- In Tuple Calculus, a query is expressed as
- {t| P(t)}
  where t = resulting tuples,
  P(t) = known as Predicate and these are the conditions that are used to fetch t, Thus, it generates set of all tuples t, such that Predicate P(t) is true for t

Operations



- P(t) may have various conditions logically combined with OR (∨), AND (∧), NOT(¬).
  It also uses quantifiers:
- ∃ t ∈ r (Q(t)) = "there exists" a tuple in t in relation r such that predicate Q(t) is true.
- ∀ t ∈ r (Q(t)) = Q(t) is true "for all" tuples in relation r.

Unsafe expression



- { s.name | ¬Supplier(s)
- Both TRC and RA have same expressive powers

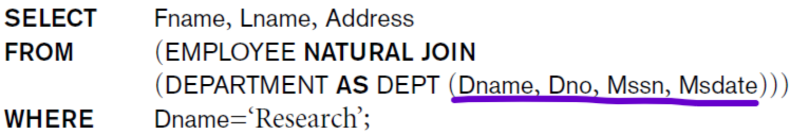
- ➔ Find all the names who does not belong to the supplier table
- ➔ Unsafe expressions are not present in relational algrebra

https://www.youtube.com/watch?v=SnsrohgiPo0

Domain Relational Calculus

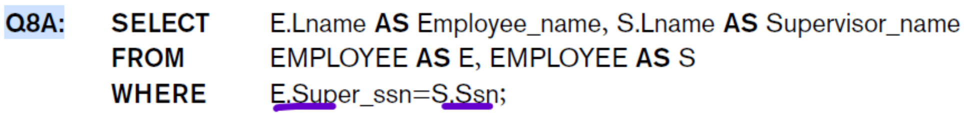
Joins in sql

1. The FROM clause in the above Q1A contains a single joined table. The attributes of such a table are all the attributes of the first table, EMPLOYEE, followed by all the attributes of the second table, DEPARTMENT.
2. In a NATURAL JOIN on two relations R and S, no join condition is specified; an implicit EQUIJOIN condition for each pair of attributes with the same name from R and S is created. Each such pair of attributes is included only once in the resulting relation.
3. If the names of the join attributes are not the same in the base relations, it is possible to rename the attributes so that they match, and then to apply NATURAL JOIN



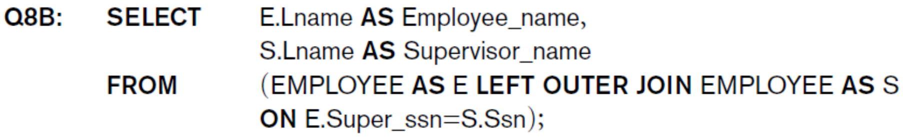
```
SELECT    Fname, Lname, Address
FROM      (EMPLOYEE NATURAL JOIN
          (DEPARTMENT AS DEPT (Dname, Dno, Mssn, Msdate)))
WHERE     Dname='Research';
```

4. The default type of join in a joined table is called an **inner join**, where a tuple is included in the result only if a matching tuple exists in the other relation.



```
Q8A:   SELECT    E.Lname AS Employee_name, S.Lname AS Supervisor_name
       FROM      EMPLOYEE AS E, EMPLOYEE AS S
       WHERE     E.Super_ssn=S.Ssn;
```

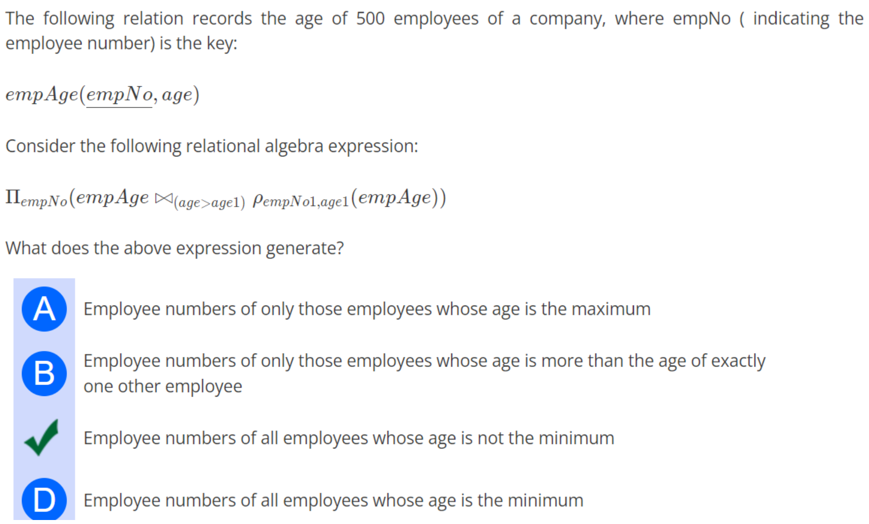
only employees who have a supervisor are included in the result; an EMPLOYEE tuple whose value for Super_ssn is NULL is excluded. If the user requires that all employees be included, an OUTER JOIN must be used explicitly.



```
Q8B:   SELECT    E.Lname AS Employee_name,
                 S.Lname AS Supervisor_name
       FROM      (EMPLOYEE AS E LEFT OUTER JOIN EMPLOYEE AS S
                 ON E.Super_ssn=S.Ssn);
```

GATE QUESTIONS

https://practicepaper.in/gate-cse/relational-algebra



The following relation records the age of 500 employees of a company, where empNo (indicating the employee number) is the key:

$empAge(\underline{empNo}, age)$

Consider the following relational algebra expression:

$\Pi_{empNo}(empAge \bowtie_{(age>age1)} \rho_{empNo1,age1}(empAge))$

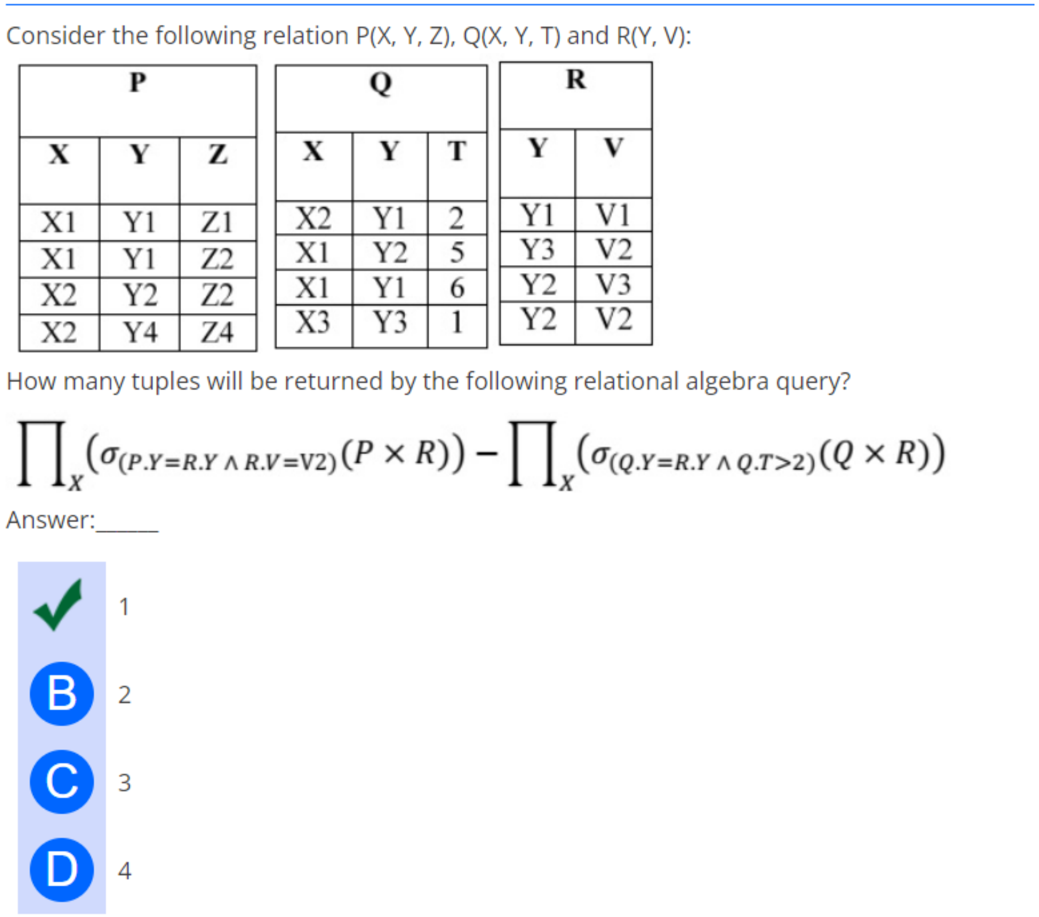
What does the above expression generate?

A Employee numbers of only those employees whose age is the maximum

B Employee numbers of only those employees whose age is more than the age of exactly one other employee

✓ Employee numbers of all employees whose age is not the minimum

D Employee numbers of all employees whose age is the minimum



Consider the following relation P(X, Y, Z), Q(X, Y, T) and R(Y, V):

P		
X	Y	Z
X1	Y1	Z1
X1	Y1	Z2
X2	Y2	Z2
X2	Y4	Z4

Q		
X	Y	T
X2	Y1	2
X1	Y2	5
X1	Y1	6
X3	Y3	1

R	
Y	V
Y1	V1
Y3	V2
Y2	V3
Y2	V2

How many tuples will be returned by the following relational algebra query?

$$\prod_X(\sigma_{(P.Y=R.Y \wedge R.V=V2)}(P \times R)) - \prod_X(\sigma_{(Q.Y=R.Y \wedge Q.T>2)}(Q \times R))$$

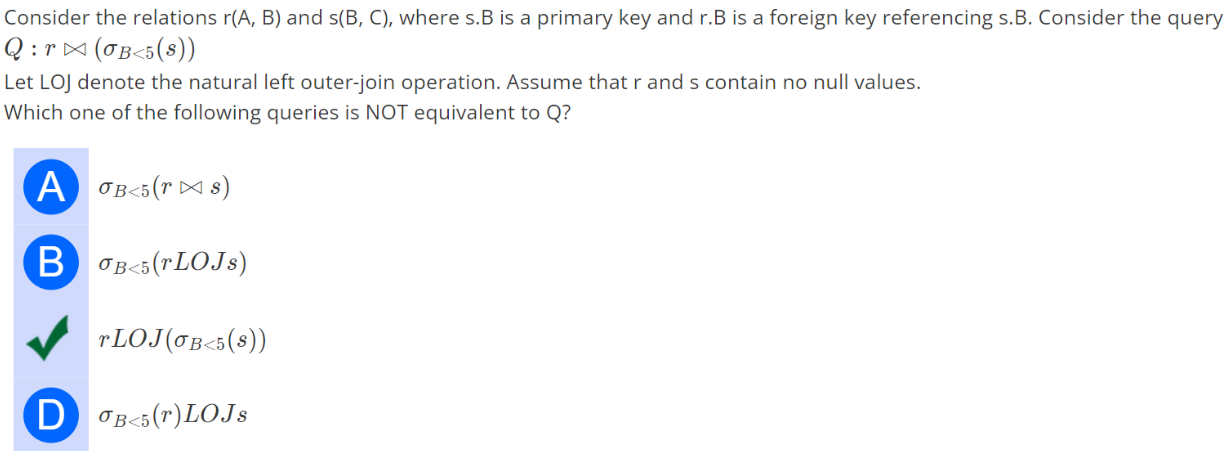
Answer:______

✓ 1

B 2

C 3

D 4



Consider the relations r(A, B) and s(B, C), where s.B is a primary key and r.B is a foreign key referencing s.B. Consider the query $Q : r \bowtie (\sigma_{B<5}(s))$

Let LOJ denote the natural left outer-join operation. Assume that r and s contain no null values.

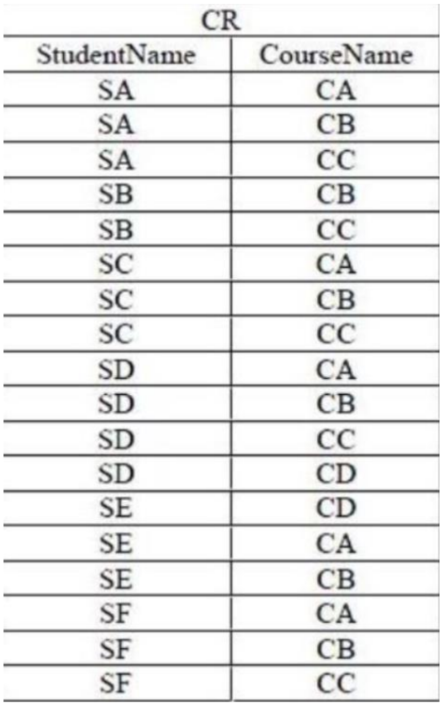
Which one of the following queries is NOT equivalent to Q?

A $\sigma_{B<5}(r \bowtie s)$

B $\sigma_{B<5}(rLOJs)$

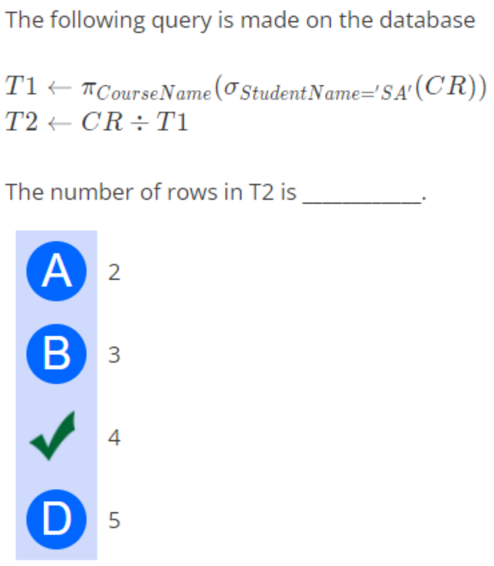
✓ $rLOJ(\sigma_{B<5}(s))$

D $\sigma_{B<5}(r)LOJs$



CR

StudentName	CourseName
SA	CA
SA	CB
SA	CC
SB	CB
SB	CC
SC	CA
SC	CB
SC	CC
SD	CA
SD	CB
SD	CC
SD	CD
SE	CD
SE	CA
SE	CB
SF	CA
SF	CB
SF	CC



The following query is made on the database

$T1 \leftarrow \pi_{CourseName}(\sigma_{StudentName='SA'}(CR))$
$T2 \leftarrow CR \div T1$

The number of rows in T2 is __________.

A 2

B 3

✓ 4

D 5

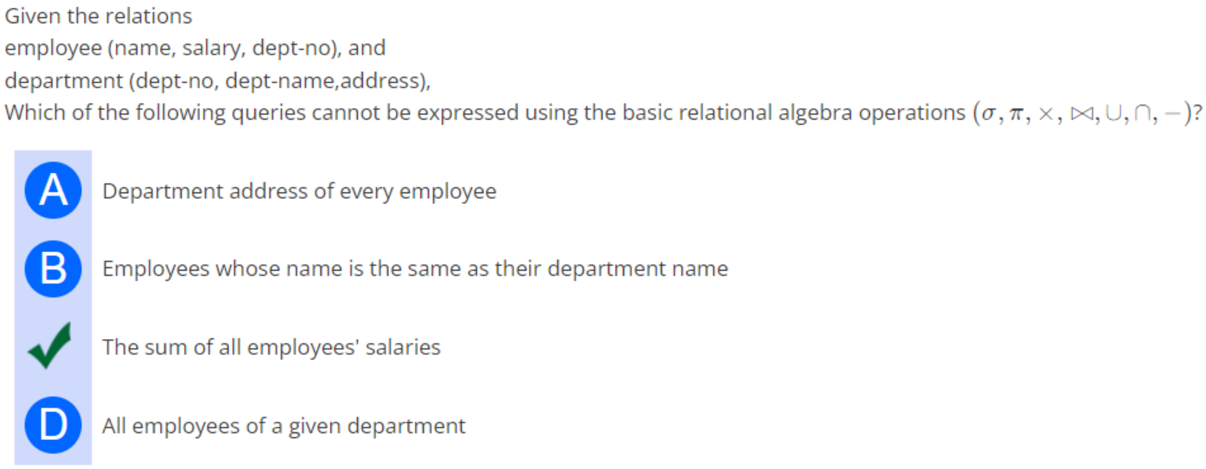
T1 WILL GIVE :-

1. CA
2. CB
3. CC

T2 = CR ÷ **T1** = All the tuples in **CR** which are matched with every tuple in T1 :

1. SA
2. SC
3. SD
4. SF

//**SB** IS NOT MATCHED WITH **CA**, **SE** IS NOT MATCHED WITH **CC**



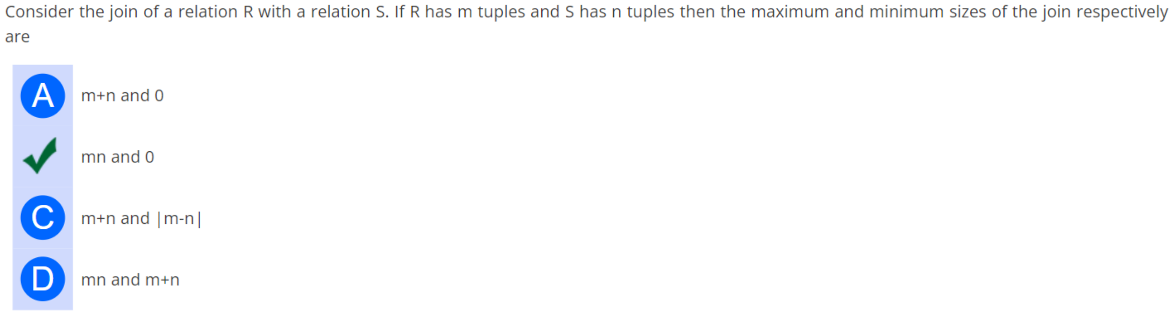
Given the relations
employee (name, salary, dept-no), and
department (dept-no, dept-name,address),
Which of the following queries cannot be expressed using the basic relational algebra operations $(\sigma, \pi, \times, \bowtie, \cup, \cap, -)$?

A Department address of every employee

B Employees whose name is the same as their department name

✓ The sum of all employees' salaries

D All employees of a given department



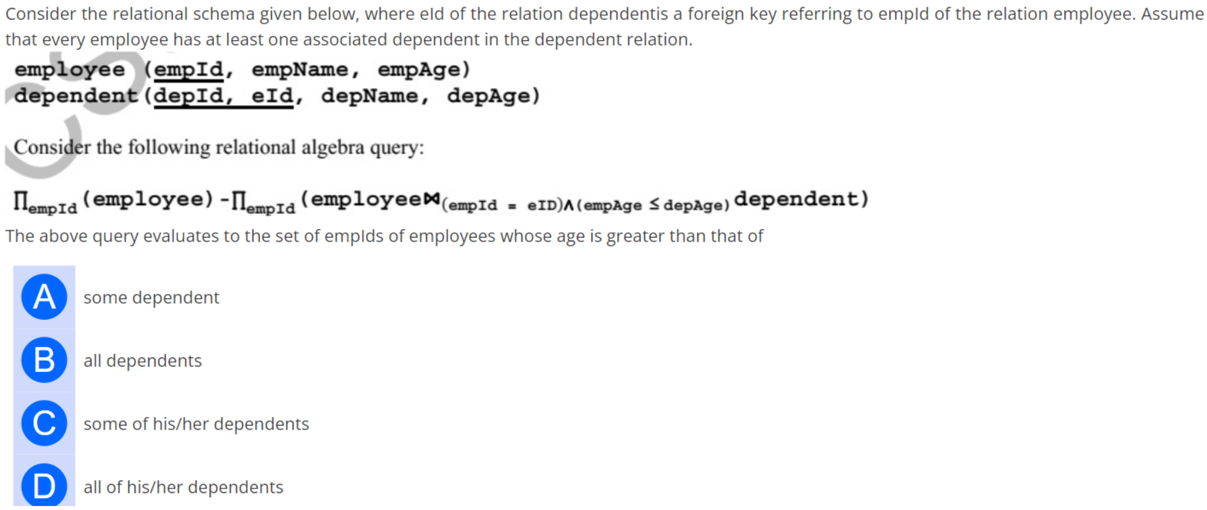
Consider the join of a relation R with a relation S. If R has m tuples and S has n tuples then the maximum and minimum sizes of the join respectively are

A m+n and 0

✓ mn and 0

C m+n and |m-n|

D mn and m+n



Consider the relational schema given below, where eId of the relation dependentis a foreign key referring to empId of the relation employee. Assume that every employee has at least one associated dependent in the dependent relation.

```
employee (empId, empName, empAge)
dependent(depId, eId, depName, depAge)
```

Consider the following relational algebra query:

$$\Pi_{empId}(employee) - \Pi_{empId}(employee \bowtie_{(empId\ =\ eID) \wedge (empAge\ \leq depAge)} dependent)$$

The above query evaluates to the set of empIds of employees whose age is greater than that of

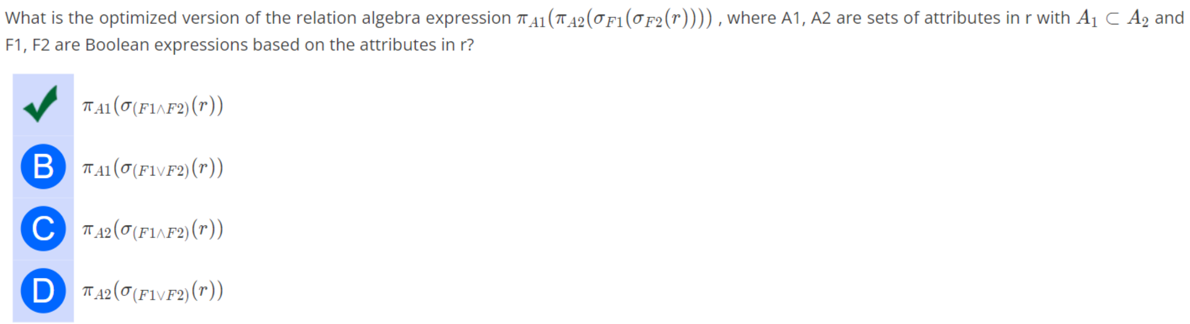
A some dependent

B all dependents

C some of his/her dependents

D all of his/her dependents

The inner query selects the employees whose age is less than or equal to at least one of his dependents. So, subtracting from the set of employees, gives employees whose age is greater than all of his dependents.



What is the optimized version of the relation algebra expression $\pi_{A1}(\pi_{A2}(\sigma_{F1}(\sigma_{F2}(r))))$, where A1, A2 are sets of attributes in r with $A_1 \subset A_2$ and F1, F2 are Boolean expressions based on the attributes in r?

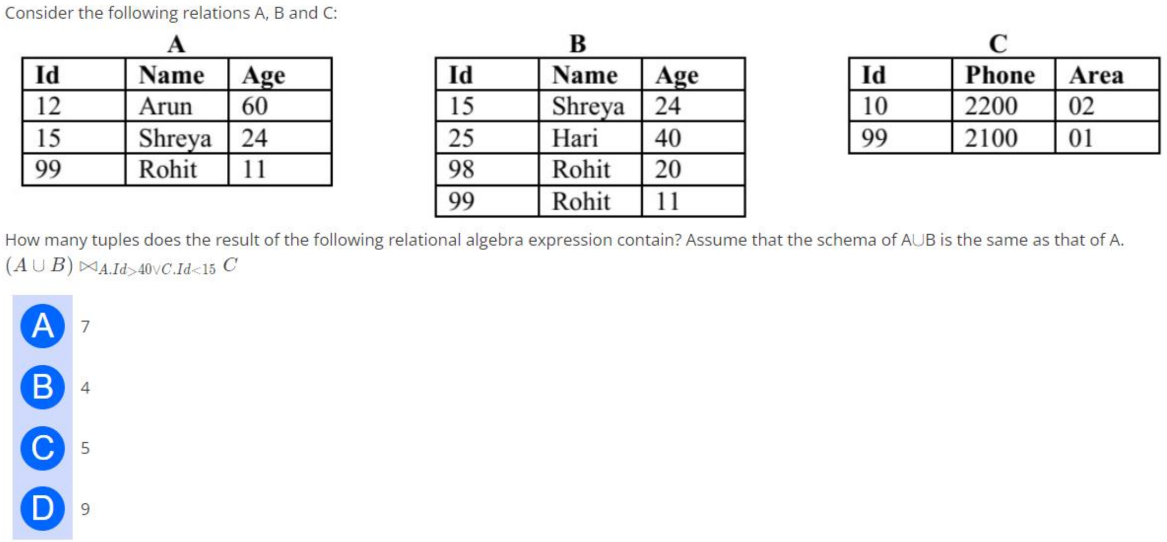
✓ $\pi_{A1}(\sigma_{(F1 \wedge F2)}(r))$

B $\pi_{A1}(\sigma_{(F1 \vee F2)}(r))$

C $\pi_{A2}(\sigma_{(F1 \wedge F2)}(r))$

D $\pi_{A2}(\sigma_{(F1 \vee F2)}(r))$

Imp



Consider the following relations A, B and C:

A

Id	Name	Age
12	Arun	60
15	Shreya	24
99	Rohit	11

B

Id	Name	Age
15	Shreya	24
25	Hari	40
98	Rohit	20
99	Rohit	11

C

Id	Phone	Area
10	2200	02
99	2100	01

How many tuples does the result of the following relational algebra expression contain? Assume that the schema of A∪B is the same as that of A.

$(A \cup B) \bowtie_{A.Id>40 \vee C.Id<15} C$

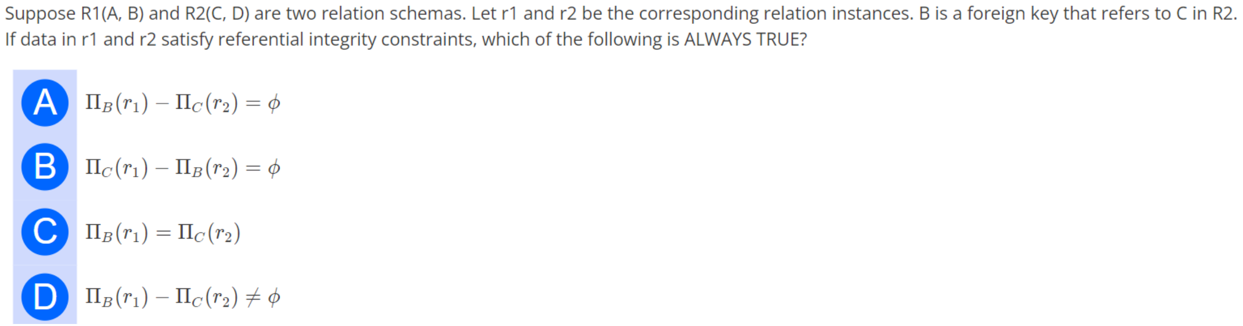
A 7

B 4

C 5

D 9

Ans is 7



Suppose R1(A, B) and R2(C, D) are two relation schemas. Let r1 and r2 be the corresponding relation instances. B is a foreign key that refers to C in R2. If data in r1 and r2 satisfy referential integrity constraints, which of the following is ALWAYS TRUE?

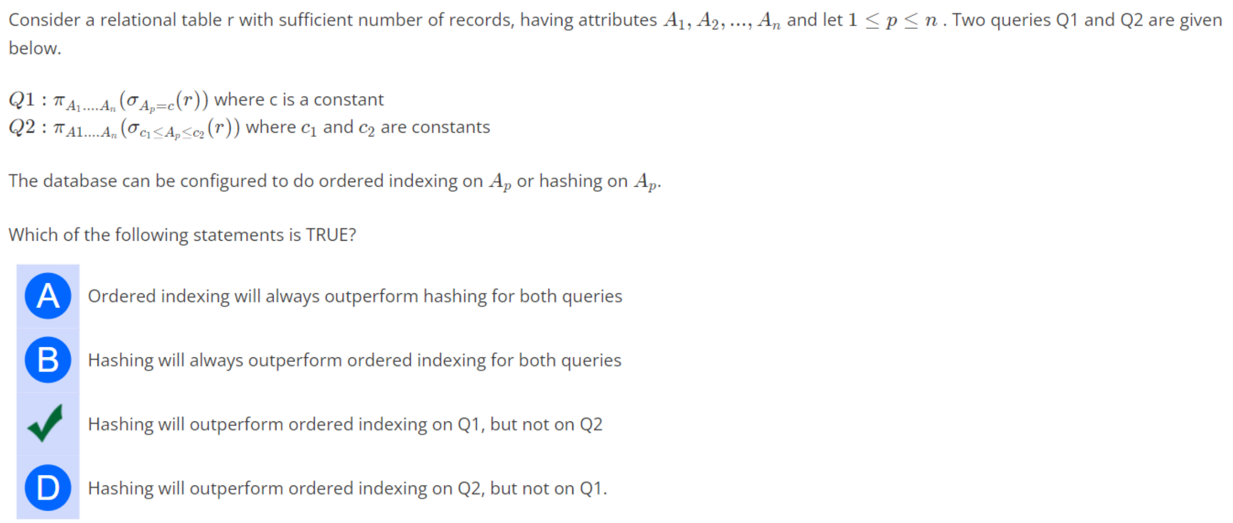
A $\Pi_B(r_1) - \Pi_C(r_2) = \phi$

B $\Pi_C(r_1) - \Pi_B(r_2) = \phi$

C $\Pi_B(r_1) = \Pi_C(r_2)$

D $\Pi_B(r_1) - \Pi_C(r_2) \neq \phi$

Ans is ‘A’



Consider a relational table r with sufficient number of records, having attributes $A_1, A_2, ..., A_n$ and let $1 \leq p \leq n$. Two queries Q1 and Q2 are given below.

$Q1 : \pi_{A_1....A_n}(\sigma_{A_p=c}(r))$ where c is a constant
$Q2 : \pi_{A1....A_n}(\sigma_{c_1 \leq A_p \leq c_2}(r))$ where c_1 and c_2 are constants

The database can be configured to do ordered indexing on A_p or hashing on A_p.

Which of the following statements is TRUE?

A Ordered indexing will always outperform hashing for both queries

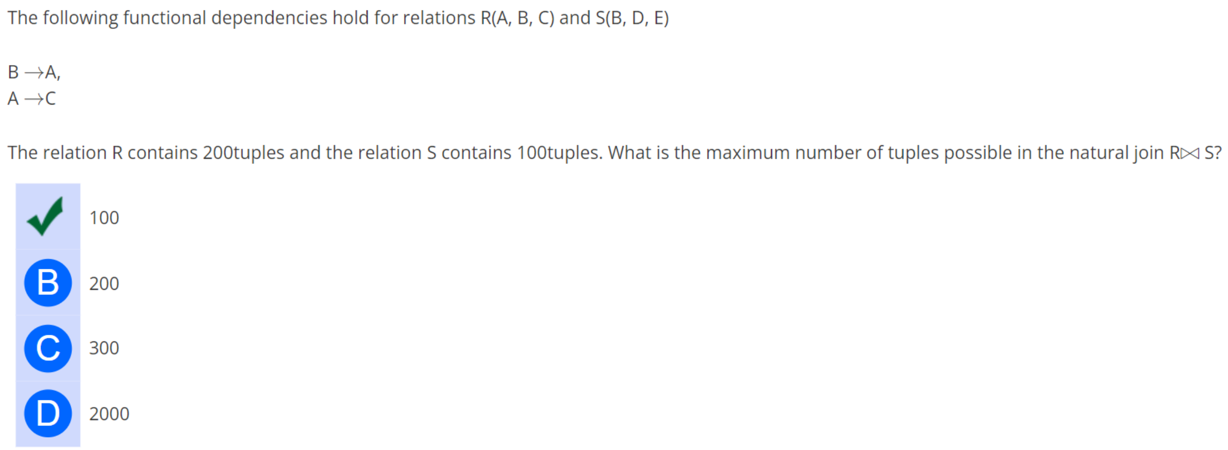
B Hashing will always outperform ordered indexing for both queries

✓ Hashing will outperform ordered indexing on Q1, but not on Q2

D Hashing will outperform ordered indexing on Q2, but not on Q1.

If record are accessed for a particular value from table, hashing will do better.

If records are accessed in a range of values, ordered indexing will perform better.



The following functional dependencies hold for relations R(A, B, C) and S(B, D, E)

B →A,
A →C

The relation R contains 200tuples and the relation S contains 100tuples. What is the maximum number of tuples possible in the natural join R⋈ S?

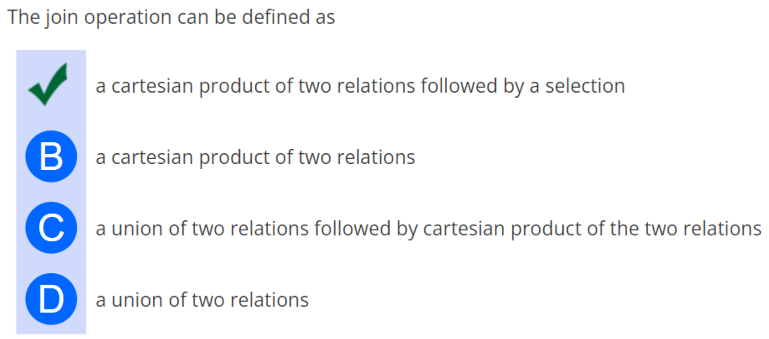
✓ 100

B 200

C 300

D 2000

Natural join will combine tuples with the same value of the common rows (if there are two common rows then both values must be equal to get into the resultant set). So by this definition, we can get at the max only common values.



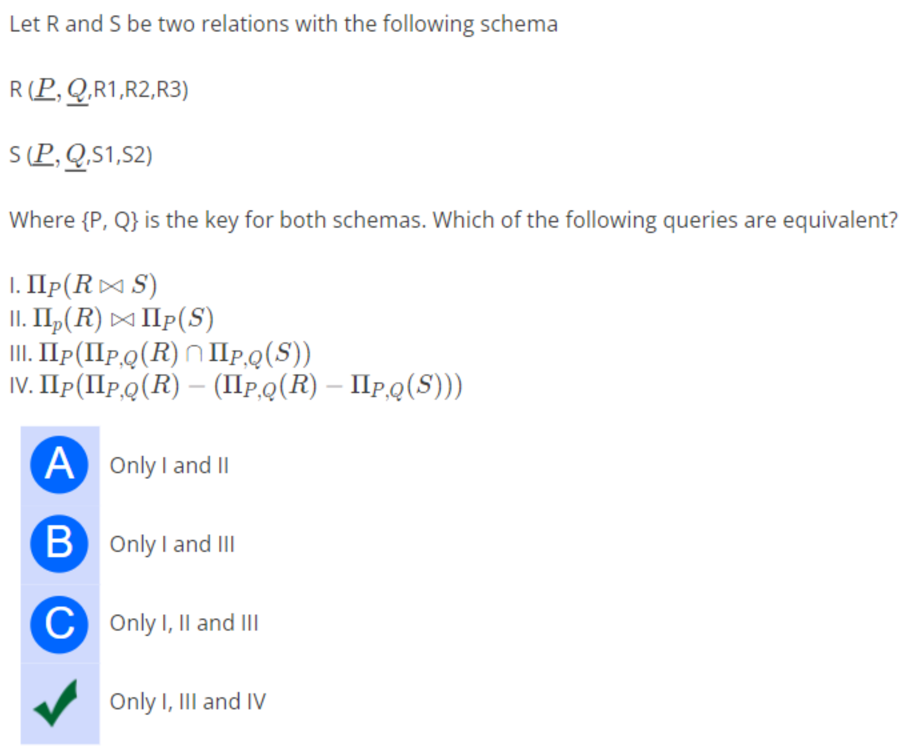
The join operation can be defined as

✓ a cartesian product of two relations followed by a selection

B a cartesian product of two relations

C a union of two relations followed by cartesian product of the two relations

D a union of two relations



Let R and S be two relations with the following schema

R ($\underline{P}$, $\underline{Q}$,R1,R2,R3)

S ($\underline{P}$, $\underline{Q}$,S1,S2)

Where {P, Q} is the key for both schemas. Which of the following queries are equivalent?

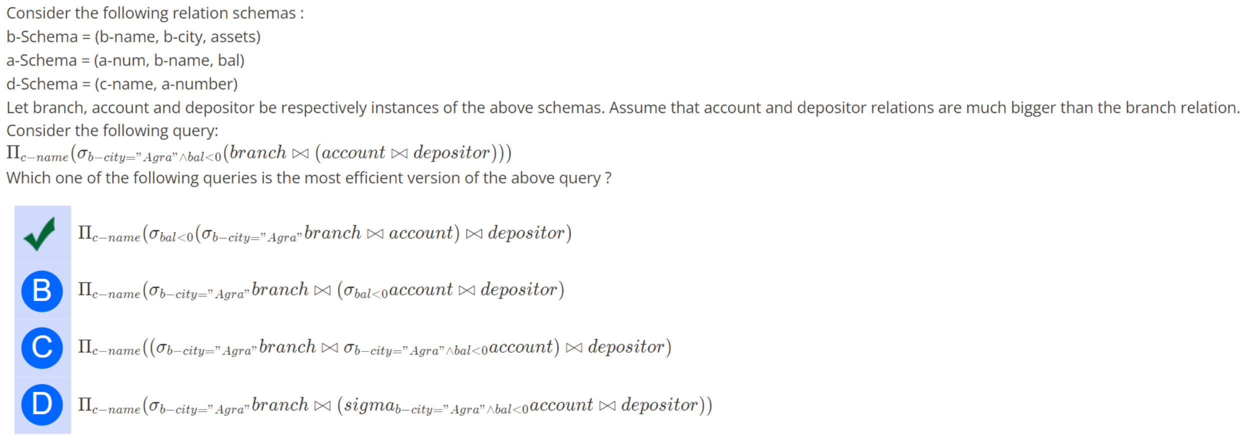
I. $\Pi_P(R \bowtie S)$
II. $\Pi_p(R) \bowtie \Pi_P(S)$
III. $\Pi_P(\Pi_{P,Q}(R) \cap \Pi_{P,Q}(S))$
IV. $\Pi_P(\Pi_{P,Q}(R) - (\Pi_{P,Q}(R) - \Pi_{P,Q}(S)))$

A Only I and II

B Only I and III

C Only I, II and III

✓ Only I, III and IV



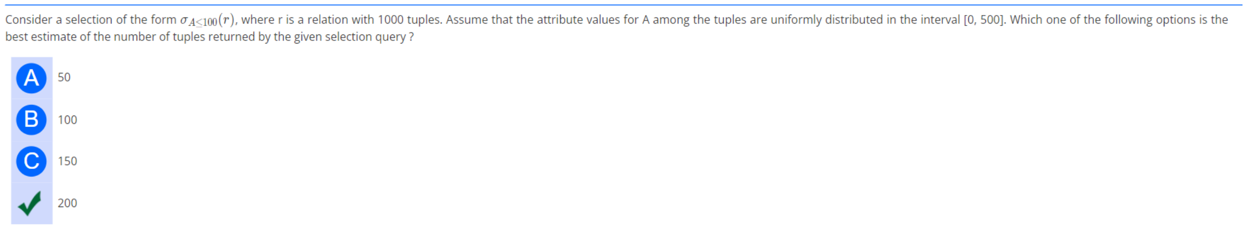
Consider the following relation schemas :
b-Schema = (b-name, b-city, assets)
a-Schema = (a-num, b-name, bal)
d-Schema = (c-name, a-number)
Let branch, account and depositor be respectively instances of the above schemas. Assume that account and depositor relations are much bigger than the branch relation.
Consider the following query:
$\Pi_{c-name}(\sigma_{b-city=\text{"}Agra\text{"} \wedge bal<0}(branch \bowtie (account \bowtie depositor)))$
Which one of the following queries is the most efficient version of the above query ?

✓ $\Pi_{c-name}(\sigma_{bal<0}(\sigma_{b-city=\text{"}Agra\text{"}}branch \bowtie account) \bowtie depositor)$

B $\Pi_{c-name}(\sigma_{b-city=\text{"}Agra\text{"}}branch \bowtie (\sigma_{bal<0}account \bowtie depositor)$

C $\Pi_{c-name}((\sigma_{b-city=\text{"}Agra\text{"}}branch \bowtie \sigma_{b-city=\text{"}Agra\text{"} \wedge bal<0}account) \bowtie depositor)$

D $\Pi_{c-name}(\sigma_{b-city=\text{"}Agra\text{"}}branch \bowtie (sigma_{b-city=\text{"}Agra\text{"} \wedge bal<0}account \bowtie depositor))$



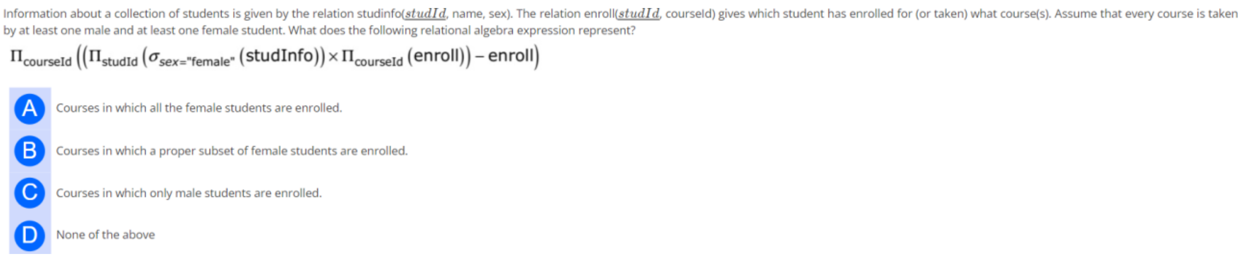
Consider a selection of the form $\sigma_{A \leq 100}(r)$, where r is a relation with 1000 tuples. Assume that the attribute values for A among the tuples are uniformly distributed in the interval [0, 500]. Which one of the following options is the best estimate of the number of tuples returned by the given selection query ?

A 50

B 100

C 150

✓ 200



Information about a collection of students is given by the relation studinfo($\underline{studId}$, name, sex). The relation enroll($\underline{studId}$, courseId) gives which student has enrolled for (or taken) what course(s). Assume that every course is taken by at least one male and at least one female student. What does the following relational algebra expression represent?

$$\Pi_{\text{courseId}}\left(\left(\Pi_{\text{studId}}\left(\sigma_{sex=\text{"female"}}(\text{studInfo})\right)\times\Pi_{\text{courseId}}(\text{enroll})\right)-\text{enroll}\right)$$

A Courses in which all the female students are enrolled.

B Courses in which a proper subset of female students are enrolled.

C Courses in which only male students are enrolled.

D None of the above

Ans is b

STUDENTINFO		
1	A	M
2	A	F
3	A	F

ENROLL	
1	$C1$
1	$C2$
2	$C1$
2	$C2$
3	$C2$

- $\pi_{courceId}(\sigma_{sex=\text{\\"}female\text{"}}(\text{studInfo}))\times\pi_{courseId}\,(\text{enroll})$

$$\Longrightarrow\begin{matrix}2\\ \\3\end{matrix}\quad * \quad\begin{matrix}C1\\ \\C2\end{matrix}\quad=\begin{matrix}2 & C1\\2 & C2\\3 & C1\\3 & C2\end{matrix}$$

- $(\pi_{studId}(\sigma_{sex=\text{\\"}female\text{"}}(\text{studInfo}))\times\pi_{courseId}(\text{enroll}))-\text{enroll})$

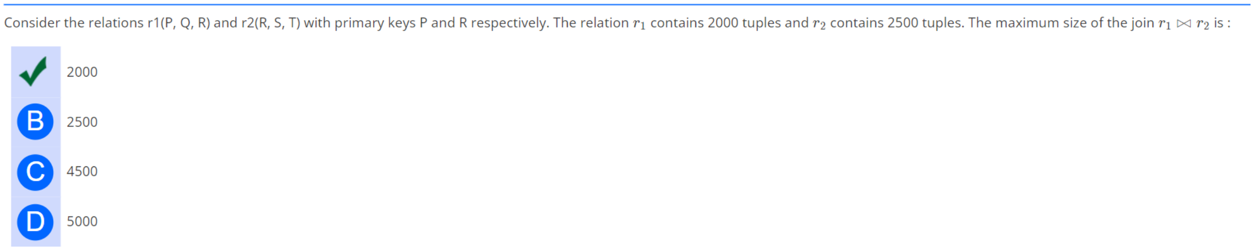
$$\Longrightarrow 3 \quad C1$$

- $\pi_{courceId}((\pi_{studId}(\sigma_{sex=\text{\\"}female\text{"}}(\text{studInfo}))\times\pi_{courseId}(\text{enroll}))-\text{enroll})$

$$\Longrightarrow C1$$

$C1$ is a course id in which not all girl students enrolled.
i.e. a proper subset of girls students appeared.

Hence **(B)** is the correct answer.



Consider the relations r1(P, Q, R) and r2(R, S, T) with primary keys P and R respectively. The relation r_1 contains 2000 tuples and r_2 contains 2500 tuples. The maximum size of the join $r_1 \bowtie r_2$ is :

✓ 2000

B 2500

C 4500

D 5000

Relational algebra queries

1. 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

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

2. Rename the attributes in a relation, we simply list the new attribute names in parentheses,

$$\text{TEMP} \leftarrow \sigma_{\text{Dno}=5}(\text{EMPLOYEE})$$
$$R(\underline{\text{First\_name, Last\_name, Salary}}) \leftarrow \pi_{\text{Fname, Lname, Salary}}(\text{TEMP})$$

Note- ρ (rho) is used to denote the RENAME operator, The first expression renames both the relation and its attributes, the second renames the relation only, and the third renames the attributes only.

$$\rho_{\underline{S}(B1,\, \underline{B2,\, \ldots,}\, Bn)}(R) \quad \text{or} \quad \rho_{\underline{S}}(R) \quad \text{or} \quad \rho_{\underline{(B1,\, B2,\, \ldots,\, Bn)}}(R)$$

3. Retrieve the Social Security numbers of all employees who either work in department 5 or directly supervise an employee who works in department 5, we can use the UNION operation as follows-

   [4]As a single relational algebra expression, this becomes Result $\leftarrow \pi_{\text{Ssn}}\ (\sigma_{\text{Dno}=5}\ (\text{EMPLOYEE})\ ) \cup \pi_{\text{Super\_ssn}}\ (\sigma_{\text{Dno}=5}\ (\text{EMPLOYEE}))$