**CS 411 Database Systems Homework 5**

**Spring 2023**

**Submitted by: Payal Mantri (payaljm2)**

* 1. **Solution**

**Assumption:** We also consider patients who have never been to a hospital . i.e Patients who do not have an entry in appointment table.   
Doctors work department is obtained only from DepartmentID in Doctor Table. We **do not** take into the consideration the departments whose courses are taken by doctor as Doctors work department

(π Patient.PatientID Patient -

(π Appointment.PatientID ( σ (Doctor.City = 'Chicago') (Appointment ⨝ Doctor))))

∪

(π Appointment.PatientID ( σ (Doctor.City = 'Chicago' ∧ Department.Dept\_Name = 'Cardiology') (Appointment ⨝ (Doctor ⨝ Department) )))

* 1. **Solution**  
       
     Patient IDs who have been treated by doctors from other department other than Orthopedic  
       
     πAppointment.PatientID(σ Department.Dept\_Name ≠ 'Orthopedic' (Appointment⨝Doctor⨝Department))  
       
     Patients who have always been treated by doctor from Orthopedic Department is obtained by subtracting above results from PatientId in appointments table .

We do this subtraction because one patient can be treated by doctor from Orthopedic and doctor from cardiology   
  
π Appointment.PatientID Appointment -

π Appointment.PatientID(σ Department.Dept\_Name ≠ 'Orthopedic' (Appointment⨝Doctor⨝Department))  
  
  
Patients who have never seen a doctor before age of 50.   
**Assumption**: We also consider Patients who have never been to hospital (Patients whose IDs do not exist in Appointment Table , in this part).   
Hence, we do a left outer join .

π Patient.PatientID σ Appointment.Date - Patient.dob > 50(Patient⟕Appointment)

Final Solution – we get intersection of bothe above cases

(π Appointment.PatientID Appointment -

π Appointment.PatientID (σ (Department.Dept\_Name ≠ 'Orthopedic') (Appointment⨝Doctor⨝Department)))

∩

(π Patient.PatientID σ (Appointment.Date - Patient.dob > 50) (Patient⟕Appointment))

* 1. IDs of Doctors who have treated at least one patient with same name as doctors name-  
     We join using additional condition Patient.Name=Doctor.Name to   
       
     π Appointment.DoctorID ( Patient⨝ (Patient.PatientID = Appointment.PatientID ∧ Doctor.Name = Patient.Name) (Appointment⨝Doctor) )

Doctors who have taken course outside their department  
π Doctor.DoctorID (Doctor ⨝ (Doctor.DoctorID = Course.DoctorID ∧Doctor.DepartmentID ≠Course.DepartmentID) Course)

Doctors who have taken course only in their department is obtained by subtracting above result from Coursr.DoctorID

**Assumption: we only consider doctor who have taken at least one course. Hence we subtract from Course.DoctorID**

π Course.DoctorID Course -

( π Course.DoctorID (Doctor ⨝ Doctor.DoctorID = Course.DoctorID ∧Doctor.DepartmentID ≠Course.DepartmentID Course))  
  
Finally solution is OR(union) of two queries  
  
(π Appointment.DoctorID ( Patient ⨝ (Patient.PatientID = Appointment.PatientID ∧ Doctor.Name = Patient.Name ) (Appointment⨝Doctor) ))

∪

(π Course.DoctorID Course -

( π Course.DoctorID (Doctor ⨝ Doctor.DoctorID = Course.DoctorID ∧Doctor.DepartmentID ≠Course.DepartmentID Course)))

**E1:** (ᴨ DepartmentID(ᴨDoctorID, DepartmentIDDoctor – ᴨDoctorID, DepartmentID(Department ⋈ Doctor))) U (ᴨDepartmentID((ᴨDepartmentID Department ᴨDepartmentIDDoctor) ⋈ Doctor))

**E2:** ᴨDepartmentID(Doctor ⋈ Department) U (ᴨDepartmentIDDoctor – ᴨDepartmentID(Doctor ⋈ Department))

E1 and E2 are equivalent

Explanation:

E1:

(ᴨ DepartmentID(ᴨDoctorID, DepartmentIDDoctor – ᴨDoctorID, DepartmentID(Department ⋈ Doctor))) U (ᴨDepartmentID((ᴨDepartmentID Department ᴨDepartmentIDDoctor) ⋈ Doctor))

Apply Commutative Rule R U S = S U R

(ᴨDepartmentID((ᴨDepartmentID Department ᴨDepartmentIDDoctor) ⋈ Doctor)) U

(ᴨ DepartmentID(ᴨDoctorID, DepartmentIDDoctor – ᴨDoctorID, DepartmentID(Department ⋈ Doctor)))

* 1. The relational algebra expression E1 and E2 are not equivalent

**E1:** (ᴨPatientID(𝞼Patient.City == ‘Chicago’Patient) ⋈ Appointment) U (Appointment ⋈ ᴨDoctorID(𝞼Doctor.City = ‘Nashville’Doctor))

**E2:** ᴨAppointments.PatientID, Appointments.DoctorID,Appointments.Date((𝞼Patient.City == ‘Chicago’Patient ⋈ Appointment) ⋈Appointment.DoctorID = Doctor.DoctorID 𝞼Doctor.City == ‘Nashville’Doctor) (10pt)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Patient Table | | | | | |
| PatientID | Name | Gender | Phone | City | dob |
| 11 | Cook | M | 1234567809 | Chicago | 09-19-1993 |
| 12 | Riya | F | 8765432190 | Champaign | 08-12-1980 |
| 13 | John | M | 8758929898 | Urbana | 09-23-2001 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Doctor Table | | | | | |
| DoctorID | Name | DepartmentID | Phone | City | salary |
| 1 | Shawn | 11 | 8765321902 | Nashville | 90000 |
| 3 | Nancy | 13 | 7892134786 | Chicago | 120000 |

|  |  |  |
| --- | --- | --- |
| Apopointments Table | | |
| PatientID | DoctorID | Date |
| 11 | 1 | 10-09-2022 |
| 12 | 3 | 03-02-2023 |
| 13 | 1 | 04-03-2023 |
| 11 | 3 | 10-08-2022 |

Lets see the execution steps E1:

For E1: 𝞼Patient.City == ‘Chicago’Patient

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Patient Table | | | | | |
| PatientID | Name | Gender | Phone | City | dob |
| 11 | Cook | M | 1234567809 | Chicago | 09-19-1993 |

(ᴨPatientID(𝞼Patient.City == ‘Chicago’Patient) =>

|  |
| --- |
| PatientID |
| 11 |

(ᴨPatientID(𝞼Patient.City == ‘Chicago’Patient) ⋈ Appointment) [result 1]

[ignoring the common column PatientId to be written twice]

|  |  |  |
| --- | --- | --- |
| PatientID | DoctorID | Date |
| 11 | 1 | 10-09-2022 |
| 11 | 3 | 10-08-2022 |

𝞼Doctor.City = ‘Nashville’Doctor

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Doctor Table | | | | | |
| DoctorID | Name | DepartmentID | Phone | City | salary |
| 1 | Shawn | 11 | 8765321902 | Nashville | 90000 |

ᴨDoctorID(𝞼Doctor.City = ‘Nashville’Doctor) =>

|  |
| --- |
| DoctorID |
| 1 |

(Appointment ⋈ ᴨDoctorID(𝞼Doctor.City = ‘Nashville’Doctor)) [result2]

|  |  |  |
| --- | --- | --- |
| PatientID | DoctorID | Date |
| 11 | 1 | 10-09-2022 |
| 13 | 1 | 04-03-2023 |

**E1 final result** **:** (ᴨPatientID(𝞼Patient.City == ‘Chicago’Patient) ⋈ Appointment) U (Appointment ⋈ ᴨDoctorID(𝞼Doctor.City = ‘Nashville’Doctor))

=> [result1] [result2]

i.e

|  |  |  |
| --- | --- | --- |
| PatientID | DoctorID | Date |
| 11 | 1 | 10-09-2022 |
| 13 | 1 | 04-03-2023 |
| 11 | 3 | 10-08-2022 |

**Similarly for E2**

𝞼Patient.City == ‘Chicago’Patient

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Patient Table | | | | | |
| PatientID | Name | Gender | Phone | City | dob |
| 11 | Cook | M | 1234567809 | Chicago | 09-19-1993 |

(𝞼Patient.City == ‘Chicago’Patient ⋈ Appointment)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Patient .PatientID | Patient .Name | Patient .Gender | Patient .Phone | Patient .City | Patient .dob | Appointments. PatientID | Appointments. DoctorID | Appointments. Date |
| 11 | Cook | M | 1234567809 | Chicago | 09-19-1993 | 11 | 1 | 10-09-2022 |
| 11 | Cook | M | 1234567809 | Chicago | 09-19-1993 | 11 | 3 | 10-08-2022 |

𝞼Doctor.City == ‘Nashville’Doctor

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Doctor Table | | | | | |
| DoctorID | Name | DepartmentID | Phone | City | salary |
| 1 | Shawn | 11 | 8765321902 | Nashville | 90000 |

((𝞼Patient.City == ‘Chicago’Patient ⋈ Appointment) ⋈Appointment.DoctorID = Doctor.DoctorID 𝞼Doctor.City == ‘Nashville’Doctor)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Patient .PatientID | Patient .Name | Patient .Gender | Patient .Phone | Patient .City | Patient .dob | Appointments. PatientID | Appointments. DoctorID | Appointments. Date |
| 11 | Cook | M | 1234567809 | Chicago | 09-19-1993 | 11 | 1 | 10-09-2022 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Doctor.DoctorID | Doctor.Name | Doctor.DepartmentID | Doctor.Phone | Doctor.City | Doctor.salary |
| 1 | Shawn | 11 | 8765321902 | Nashville | 90000 |

**E2 final solution :** ᴨAppointments.PatientID, Appointments.DoctorID,Appointments.Date((𝞼Patient.City == ‘Chicago’Patient ⋈ Appointment) ⋈Appointment.DoctorID = Doctor.DoctorID 𝞼Doctor.City == ‘Nashville’Doctor)

|  |  |  |
| --- | --- | --- |
| Appointments. PatientID | Appointments. DoctorID | Appointments. Date |
| 11 | 1 | 10-09-2022 |

**E1 results in 3 rows whereas E2 outputs only 1 row. Hence E1 and E2 are not equivalent**

* 1. Two-way multi pass sort

**Solution**

1. Divide the relation R into smaller chunks that can fit in memory. Since each block holds 3 values and there are 12 blocks, we can divide the relation into 4 chunks of 3 blocks each:  
     
   Chunk 1: [54,13,81], [64,82,78], [48,32,74], [66,12,24]

Chunk 2: [69,53,58], [18,39,71] [95,47,10], [53,87,75]

Chunk 3: [74,43,8], [55,90,62], [88,88,43], [5,5,18]

1. Read Each chunk one by one and write it back  
   Chunk 1: [54,13,81], [64,82,78], [48,32,74], [66,12,24]   
   [12,13,24], [32,48, 54],[64,66,74], [78,81,82]

Chunk 2: [69,53,58], [18,39,71] [95,47,10], [53,87,75]

[10,18,39], [47,53,53], [58 ,69, 71], [75,87,95]

Chunk 3: [74,43,8], [55,90,62], [88,88,43], [5,5,18]

[5,5,8], [18,43,43], [55,62,74], [88,88,90]

1. Merge M-1 runs (3 chunk) at time to construct the output

[**12**,13,24], [**10**,18,39], [**5**,5,8], [ \_, \_, \_ ] output [5,5,8]

Load new block from Chunk 3

[**12**,13,24], [**10**,18,39], [**18**,43,43] , [ \_, \_, \_ ] output [10, 12, 13]

[12,13,**24**], [10,**18**,39], [**18**,43,43] , [ \_, \_, \_ ]output [18, 18, 24]

Load block from Chunk 1

[**32**,48, 54], [10,18,**39**], [18,**43**,43] , [ \_, \_, \_ ] [32, 39, \_]

Load block from Chunk 2

[**32**,48, 54], [**47**,53,53] [18,**43**,43] , [ \_, \_, \_ ]output [32, 39, 43]

[32,**48**, 54], [**47**,53,53] [18,43,**43**] , [ \_, \_, \_ ][43, \_, \_]

Load new block from Chunk 3

[32,**48**, 54], [**47**,53,53] [**55**,62,74] , [ \_, \_, \_ ][43, 47, 48]

[32,48, **54**], [47,**53**,53] [**55**,62,74] , [ \_, \_, \_ ][53, 53,\_]

Load new block from Chunk 2

[32,48, **54**], [**58** ,69, 71] [**55**,62,74] , [ \_, \_, \_ ][53, 53, 54]

Load new block from Chunk 1

[**64**,66,74], [**58** ,69, 71] [**55**,62,74] , [ \_, \_, \_ ][55, 58, 62]

[**64**,66,74], [58 ,**69**, 71] [55,62,**74**] , [ \_, \_, \_ ][64, 66, 69]

[64,66,**74**], [58 ,69, **71**] [55,62,**74**] , [ \_, \_, \_ ][71, \_, \_]

Load new block from Chunk 2  
[64,66,**74**], [**75**,87,95] [55,62,**74**] , [ \_, \_, \_ ][71, 74, \_]

Load new block from Chunk 1

[**78**,81,82], [**75**,87,95], [55,62,**74**] , [ \_, \_, \_ ] [71, 74, 74]

Load new block from Chunk 3

[**78**,81,82], [**75**,87,95], [**88**,88,90], [ \_, \_, \_ ] [75, 78, 81]

[78,81,**82**], [75,**87**,95], [**88**,88,90], [ \_, \_, \_ ] [82, 87, 88]

[75,87,**95**], [88,**88**,90], [ \_, \_, \_ ] [88, 90, 95]

The final merged solution  
[5,5,8], [10, 12, 13], [18, 18, 24], [32, 39, 43], [43, 47, 48], [53, 53, 54], [55, 58, 62], [64, 66, 69], [71, 74, 74], [75, 78, 81], [88, 90, 95]

1. * 1. No the one pass join is not feasible on “Join A” – **Collections ⋈ Museums**.   
          
        number of memory blocks available = M = 42

Minimum Memory requirement for one pass join

= min(B(Collections), B(Museums))+2

= min( 50, 320 ) + 2

= 52

But there are only 42 memory blocks available and hence one pass join is not possible

* + 1. For block based Nested loop join using relations R ⋈S

Total cost: B(R) + B(S)B(R)/(M-2)  
We take smaller relation as outer relation .  
For our case R = Museums B(R) = 50

S= Collections B(S) = 320   
M= 42

Total cost = 50+ 320\* ceil = 50 + 320 \* 2 = 690

Total cost = 690

* + 1. **What is the total number of blocks for the result after "Join A"?**

Number of tuples in resulting table = 7200

Each block can hold = 80 tuples   
Blocks required for resulting table = 7200/80 = 90

* + 1. **Is one-pass join feasible for “Join B”? Justify your answer.**Let Resultant table after “Join A” be called ResultJoinA

Join B = **Paintings ⋈ ResultJoinA**

B(Paintings) = 36

B(ResultJoinA) = 90

Minimum memory requirement for Join B using one pass

= min(B(Paintings), B(ResultJoinA))+2

= min(36, 90) + 2

= 38

Since we have M=42 blocks in memory and need only 38 , one pass join is feasible

* + 1. **If YES, calculate the cost using one-pass join**.   
         
       Cost = B(R)+B(S) = B(Paintings)+ B(ResultJoinA) = 36+90 = 126

**Total cost using one pass join = 126**



Table

Description automatically generated  
**Solution:**

|  |  |  |  |
| --- | --- | --- | --- |
| Subquery | Size | Cost | Plan |
| A join B | 320000 | 0 | AB |
| A join C | 6000 | 0 | AC |
| A join D | 100 | 0 | AD |
| B join C | 24000 | 0 | CB |
| B join D | 40000 | 0 | DB |
| C join D | 750 | 0 | DC |
| ABC | 24000 | 6000 | (AC)B |
| ABD | 800 | 100 | (AD)B |
| ACD | 15 | 100 | (AD)C |
| BCD | 6000 | 750 | (CD)B |
| ABCD | 120 | 115 | ((AD)C)B |

* A join B = 2000\*8000/ (50) = 320K
* A join C = 2000\*6000/ (50\*40) = 6k
* A join D = 2000\* 5000/(50\*50\*40) = 100
* B join C = 8000\* 6000/(40\*50) = 24k
* B join D = 8000\*5000/(40\* 25) = 40k
* C join D = 6000\*5000/(40\*50\*20) = 750
* ABC will have plan (AC)B

|  |  |  |
| --- | --- | --- |
| AD(w,x,y,z) | B(w,x) | (AC)B (w,x,y,x) |
| T(AC) = 6000 | T(B) =8000 | T((AC)B) = 24k |
| V(AC, w) = 30 | V(B, w) = 40 | V((AC)B, w) = 30 |
| V(AC, x) = 50 | V(B, x) = 25 | V((AC)B, x) = 25 |
| V(AC, y) = 40 |  | V((AC)B, y) = 40 |
| V(AC, z) =10 |  | V((AC)B, z) = 10 |

Size (AC)B = 6000\*8000/(40\*50) = 24000

Cost((AC)B) = Cost(A C) + Cost(B) + size(AC)  
 = 0+0+6000 = 6k

* ABD will have plan (AD)B

|  |  |  |
| --- | --- | --- |
| AD(w,x,y,z) | B(w,x) | (AC)B (w,x,y,x) |
| T(AD) = 100 | T(B) =8000 | T((AC)B) = 800 |
| V(AD, w) = 40 | V(B, w) = 40 | V((AC)B, w) = 40 |
| V(AD, x) = 20 | V(B, x) = 25 | V((AC)B, x) = 20 |
| V(AD, y) = 40 |  | V((AC)B, y) = 40 |
| V(AD, z) =20 |  | V((AC)B, z) = 20 |

Size (AC)B =100\*8000/(40\*25) = 800  
Cost((AD)B) = Cost(AD) + Cost(B) + size(AD)  
 = 0+0+100 = 24k

* ACD will have plan (AD)C

|  |  |  |
| --- | --- | --- |
| * AD(w,x,y,z) | C(w,x,z) | (AD)C (w,x,y,x) |
| T(AD) = 100 | T(C) =6000 | T((AD)C)= 15 |
| V(AD, w) = 40 | V(C, w) = 30 | V((AD)C, w) = 30 |
| V(AD, x) = 20 | V(C, x) = 50 | V((AD)C, x) = 20 |
| V(AD, y) = 40 |  | V((AD)C, y) = 40 |
| V(AD, z) =20 | V(C, z)= 10 | V((AD)C, z) = 10 |

Size (AD)C =100\*6000/(40\*50\*20) = 15

Cost((AD)C) = Cost(AD) + Cost(C) + size(AD)

= 0+0+100 = 100

* BCD will have plan (CD)B

|  |  |  |
| --- | --- | --- |
| * CD(w,x,y,z) | B(w,x) | (CD)B (w,x,y,x) |
| T(CD) = 750 | T(B) =8000 | T((CD)B)= 6000 |
| V(CD, w) = 30 | V(B, w) = 40 | V((CD)B, w) = 30 |
| V(CD, x) = 20 | V(B, x) = 25 | V((CD)B, x) = 20 |
| V(CD, y) = 50 |  | V((CD)B, y) = 50 |
| V(CD, z) =10 |  | V((CD)B, z) = 10 |

Size (CD)B =750\*8000/(40\*25)=6000

Cost((CD)B) = Cost(CD) + Cost(B) + size(CD)  
 = 0+0+750 = 750

* For ABCD, to determine plan, we compare Cost of ABC, ABD, ACD, BCD.

Cost of ABD and ACD is 100 and is least

Among ABD and ACD, ACD has a size 15 and hence the plan will be

(ACD)B. **Final plan ((AD)C)B**

|  |  |  |
| --- | --- | --- |
| * (AD)C(w,x,y,z) | B(w, x) | (CD)B (w,x,y,x) |
| T((AD)C) = 15 | T(B) =8000 | T((CD)B)= 6000 |
| V((AD)C, w) = 30 | V(B, w) = 40 | V((CD)B, w) = 30 |
| V((AD)C, x) = 20 | V(B, x) = 25 | V((CD)B, x) = 20 |
| V((AD)C, y) = 40 |  | V((CD)B, y) = 50 |
| V((AD)C, z) =10 |  | V((CD)B, z) = 10 |

Size of ((AD)C)B = 15\*8000/(40\*25)

= 120

Cost ((AD)C)B = cost((AD)C) + cost(B) + Size(B) + size((AD)C)

= 100 + 0 +0 + 15

=115

**Hence most efficient join for ABCD is given by plan ((AD)C)B**

**Extra Credit**

1. Cost of scanning the table using table scan is B(R)

Hence Cost = B(Actors) = 100  
  
Size of selection Age>=40   
  
We have   
T(σP(R)) = T(R)\*selP(R)   
 = T(R) \* (number of distinct values satisfying the condition/ number of distinct values)

T(σAge>=40(Actors)) = T(Actors) \* Number of distinct values in Actors >= 40 /Number of disctinct value for age

Size of selection = T(σAge>=40(Actors)) = 6000\* 60/82 = 4390.24 4391

1. What is the cost and size of the selection Playwright= “William Shakespeare” Assume we have an unclustered index on Playwright   
     
   For unclustered index, table scan has T(R) \* 1/V(R,a)   
     
   Cost = T(Plays)\* 1/V(Plays, Plawright)  
    = 200/25   
   Cost = 8  
     
   Size = T(σP(R)) = T(R)\*selP(R)  
    = T(R)\* 1/V(R, a)   
   Size = T(Plays)\*1/ V(Plays, Plawright)  
    = 200/25  
   Size = 8   
   i.e T(**σ** Playwright=’WilliamShakespeare’ Plays) = 8
2. What is the cost of executing Join A if we use an index-based nested loop join

For join R⋈ S, Cost for index based nested loop join for unclustered index = B(R) + T(R)T(S)/V(S,a)  
  
Given use selection result as inner table S= **σ** Playwright=’WilliamShakespeare’ Plays

And R= ActsIn   
 B(R) = B(ActsIn) = 1250  
 T(R) = T(ActsIn) =25000  
 T(S) = T(**σ** Playwright=’WilliamShakespeare’ Plays) = 8

V(S, a) = V(**σ** Playwright=’WilliamShakespeare’ Plays, Playwright) = 1

Cost = B(R) + T(R)T(S)/V(S,a)

= 1250+ 25000 \* 8 /1

= 201250

1. Suppose we plan to use a hash-based algorithm for JOIN A and the two-pass hash-based join. What is the cost of executing JOIN A? What is the **precise** range for the possible memory size M? **Ceil** the lower boundary. (3 points)

Join A = (**σ** Playwright=’WilliamShakespeare’ Plays) ⋈ ActsIn

For join R⋈ S using two pass hashing based join , Cost = 3(B(R)+B(S))

So we have Cost (JoinA) = 3(B(**σ** Playwright=’WilliamShakespeare’ Plays) + B(ActsIn))

= 3(1+1250)  
 Cost( JoinA) = 3753

For determing range of M ,   
 we know **min(B(R), B(S)) <= (M-1)(M-2)** min( B(σ Playwright=’WilliamShakespeare’ Plays) , B(ActsIn)) <= (M-1)(M-2)  
 min(1, 1250) <= (M-1)(M-2)  
 1 <= (M-1)(M-2)

Solving this inequality we get   
 i.e. & i.e.

Upper bound: smaller relation block size + 2 = 1 + 2 = 3  
 Ceiling the lower boundary   
 we get final range of M  **and**

1. Assuming that
   * The number of tuples after Projection B is 4000
   * The block size for the result tables after projection B/C is 200 tuples
   * The memory capacity is 25 blocks

Condition for one-pass algorithm:

min(B(R), B(S)) <= M-2  
  
min(B(Projection B), B(Projection C)) <= M-2

B(Projection B) = 4000/200 = 20

B(Projection C) = T(C) /200   
 **From a . we have** T(C) = Size of selection = T(σAge>=40(Actors)) = 6000\* 60/82 = 4390.24 4391B(Projection C) = T(C) /200 = 4391/200 = 21.95

min(B(Projection B), B(Projection C)) <= M-2

min(20,21.95)<= M-2  
M-2>= 20

M>=22  
We have memory of M=25 >=22   
Hence one pass algorithm can be carried for Union operation.  
  
Cost of Union = B(R)+B(S)  
Cost = B(Projection B)+ B(Projection C)

= 20 + 21.95  
 = 41.95

42