Given the following relations:

STUDENT (SID, SNAME, SAGE, SSEX)

COURSE (CID, CNAME, CCONTEXT)

MARK (SID, CID, SCORE)

Question 1 (4 marks):

Given three relations STUDENT, COURSE and MARK, a user query is: “Find the student’s name and Course name of students whose scores are more than 85 points.”

(1) Please give the calculus query (SQL) (1 mark);

SELECT SNAME, CNAME

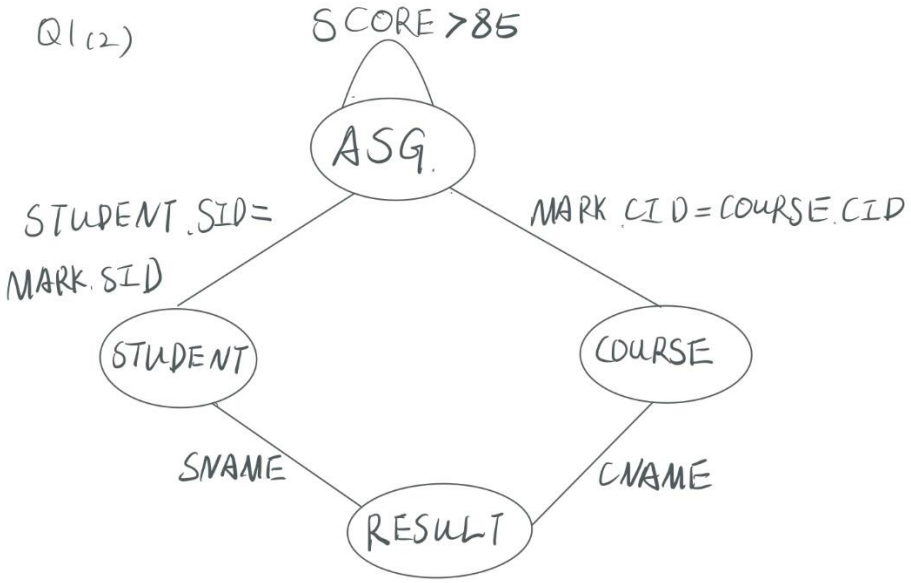
FROM STUDENT,MARK, COURSE

WHERE MARK.SCORE>85

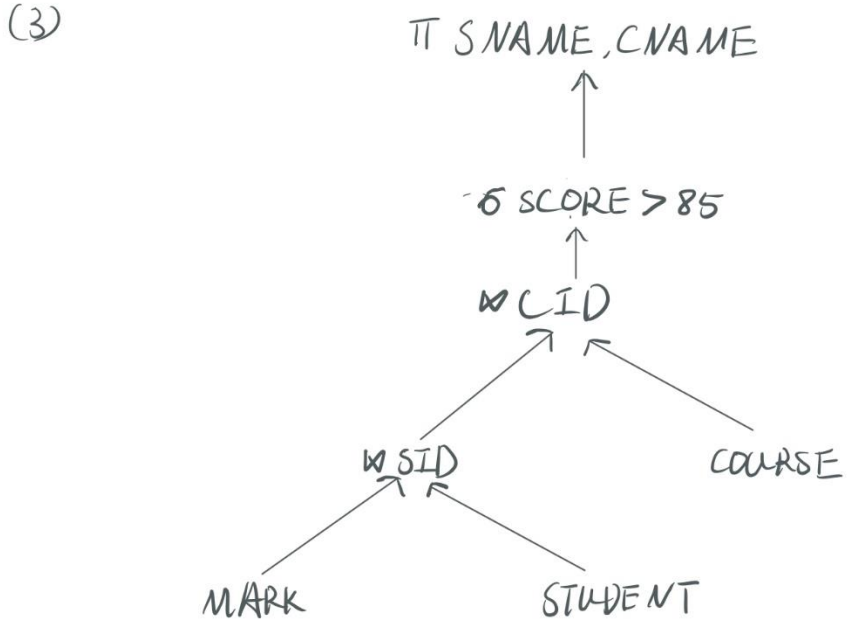
AND MARK.CID=COURSE.CID

AND STUDENT.SID=MARK.SID

(2) Please give the corresponding query graph (2 mark);



(3) Please map the SQL query into an original operator tree (Not optimized) (1 mark).



Question 2 (4 marks):

For the following query:

SELECT SNAME, CNAME

FROM STUDENT, COURSE, MARK

WHERE (SAGE＜16 OR MARK.CID＜"C2")

AND (CCONTEXT= "Science" OR SCORE> 85)

AND STUDENT.SID= MARK.SID

AND (CCONTEXT= "Science" OR SCORE≤85）

AND COURSE.CID= MARK.CID

(1) Please simplify the query (2 mark);

consider two conditions from the query

"(CCONTEXT = "Science" OR SCORE > 85)"

"(CCONTEXT = "Science" OR SCORE <= 85)"

The two portion of the above query "SCORE > 85" and "SCORE <= 85" are irrelevant, "SCORE > 85" returns score greater than "85" and "SCORE <= 85" returns score less than or equals to "85" that means all "SCORE" with any value will be returned. So both conditions are removed from the query and CCONTEXT = "Science" condition only once with in the query.

The simple query looks like this:

SELECT SNAME,CNAME

FROM STUDENT, COURSE, MARK

WHERE (SAGE<16 OR MARK.CID<"C2")

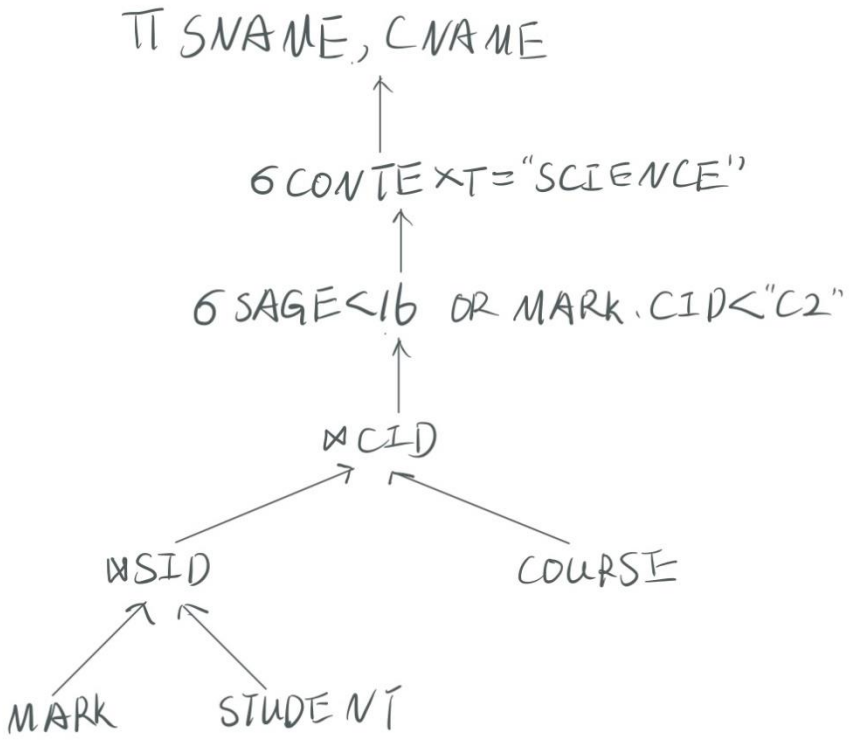
AND STUDENT.SID=MARK.SID

AND COURSE.CID=MARK.CID

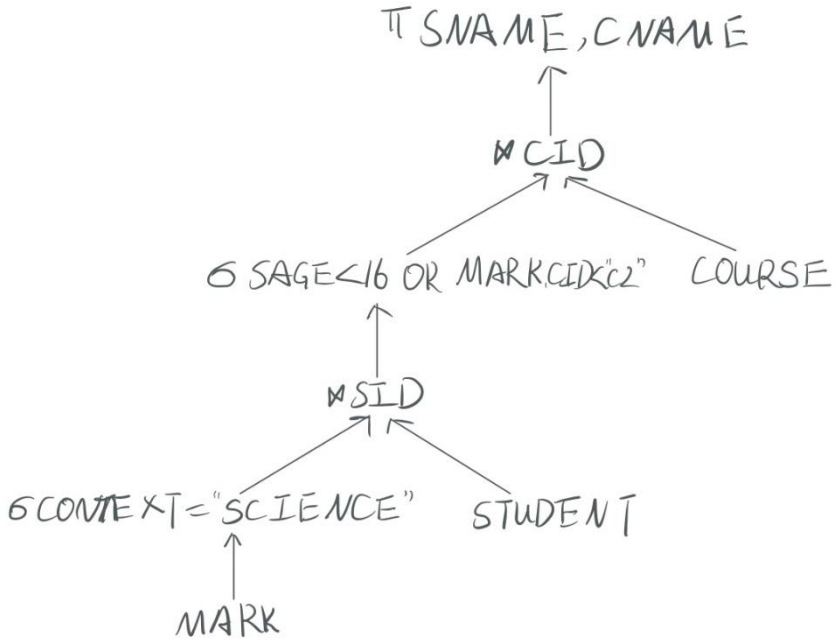
AND CONTEXT="SCIENCE"

(2) Please transform the SQL expression into an **optimized operator tree** using the restructuring algorithm where select and project operations are applied as soon as possible to reduce the size of intermediate relations (2 mark).

The genetic operator tree is:



And after the optimize the tree is:



3.Assume that relation STUDENT of the sample database is horizontally fragmented:

$STUDENT1 = \sigma SID \geq ''S3''$(STUDENT)

$STUDENT2 = \sigma\ ''S1'' \leq SID < ''S3''$(STUDENT)

$STUDENT3 = \sigma SID < ''S1''$(STUDENT)

relation MARK is indirectly fragmented as：

$MARK1 = MARK \ltimes SID(STUDENT2)$

$MARK2 = MARK \ltimes SID(STUDENT1)$

$MARK3 = MARK \ltimes SID(STUDENT3)$

and relation COURSE is vertically fragmented as：

$COURSE1 = \Pi CID,CNAME(COURSE)$

$COURSE2 = \Pi CID,CCONTEXT\ (COURSE)$

Please describe the detailed processes of how to transform the following query into a reduced query on fragments:

the query is

SELECT CNAME

FROM STUDENT, COURSE, MARK

WHERE STUDENT.SID= MARK.SID

AND COURSE.CID= MARK.CID

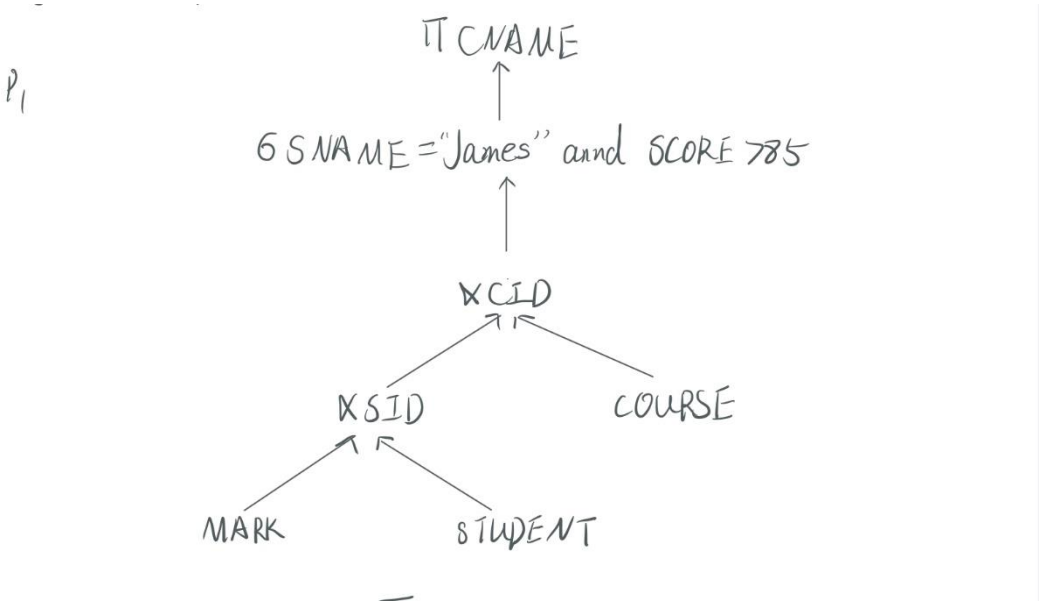
AND SNAME= "James"

AND SCORE> 85

⋈

Answer:

The Generic Operator Tree for Problem:



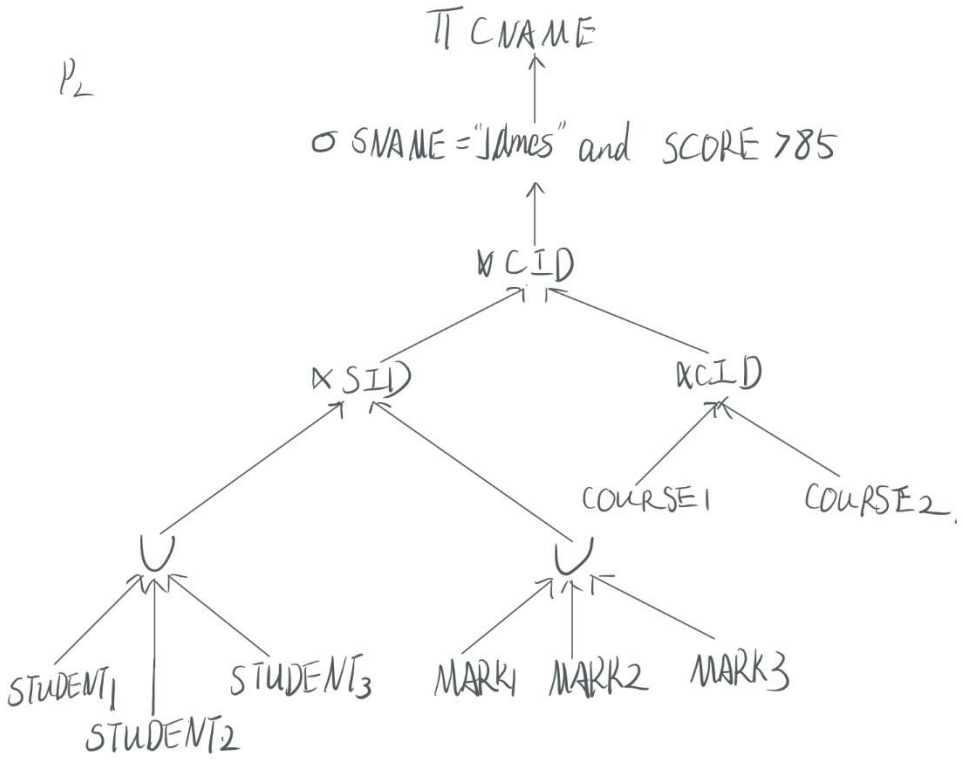
The localization programs for the base relations are as follows:

STUDENT = STUDENT1 ∪ STUDENT2 ∪ STUDENT3

MARK = MARK1 ∪ MARK2 ∪ MARK3

COURSE = COURSE1 ∝ COURSE2

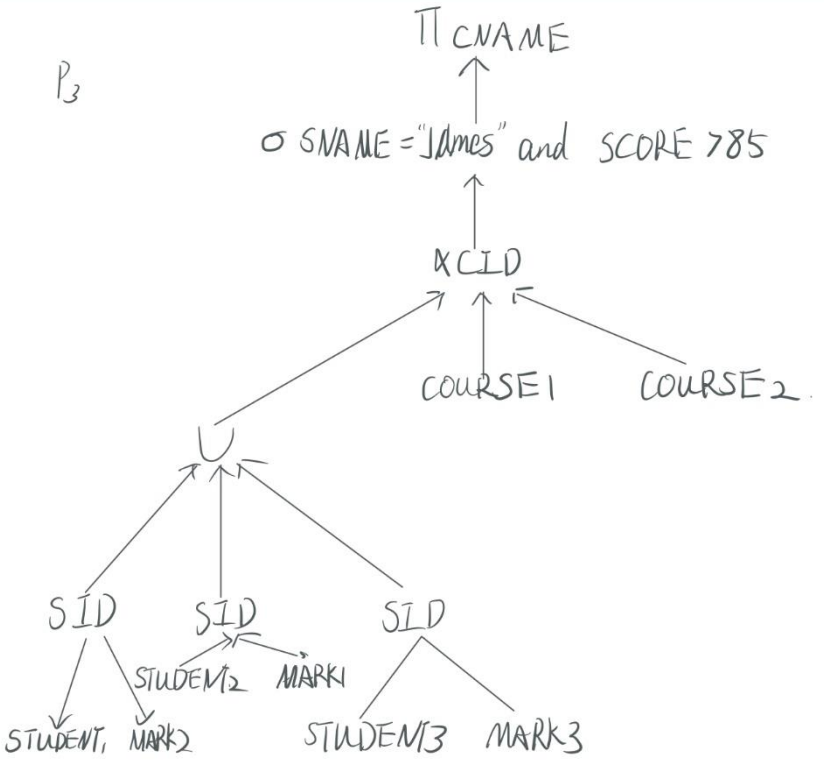
which results in the operator tree given：



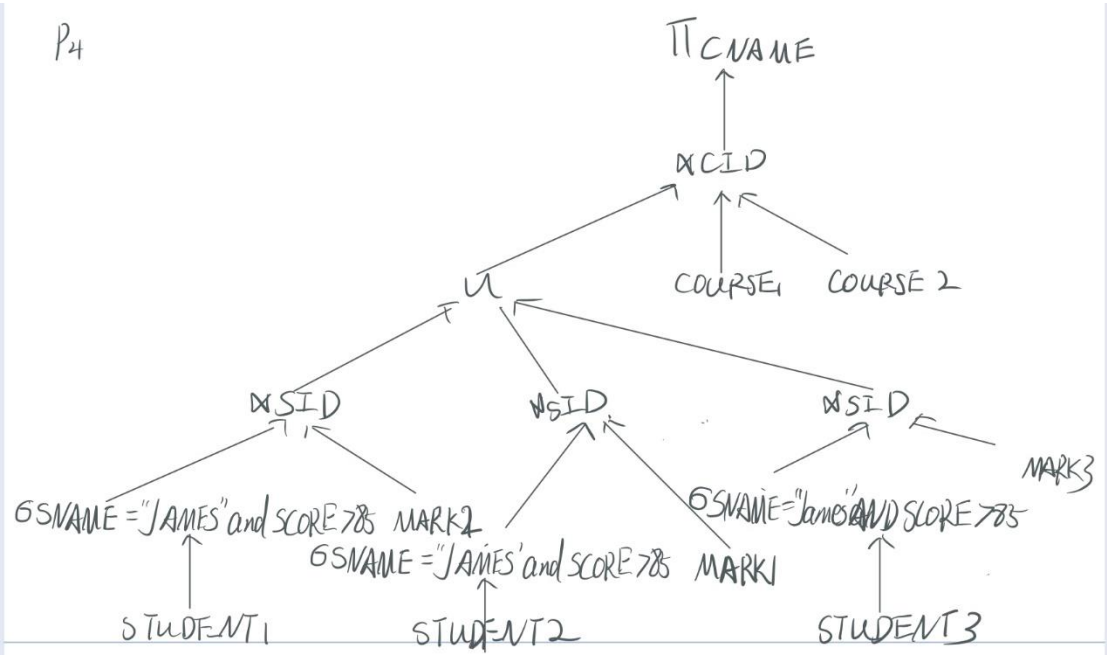
Considering the subtree under **SID**. When the join is distributed over union, the following four joins result, which are then unionid:

- STUDENT1⋈ MARK1
- STUDENT1⋈ MARK2
- STUDENT1⋈ MARK3
- STUDENT2⋈ MARK1
- STUDENT2⋈ MARK2
- STUDENT2⋈ MARK3
- STUDENT3⋈ MARK1
- STUDENT3⋈ MARK2
- STUDENT3⋈ MARK3
- Since STUDENT is derived from MARK, only three joins will produce results: STUDENT1⋈ MARK2, STUDENT2⋈ MARK1 and STUDENT3⋈ MARK3, so the other six can be eliminated.

Note that there are two ⋈ CID feeding into one another. These can reduce to a single join, which results in the operator tree below:



At this point, selection can be commuted with joins and union (i.e., pushed down the tree). Since the selection only applies to the STUDENT relation, it is pushed down one path of the joins and the tree will show below.

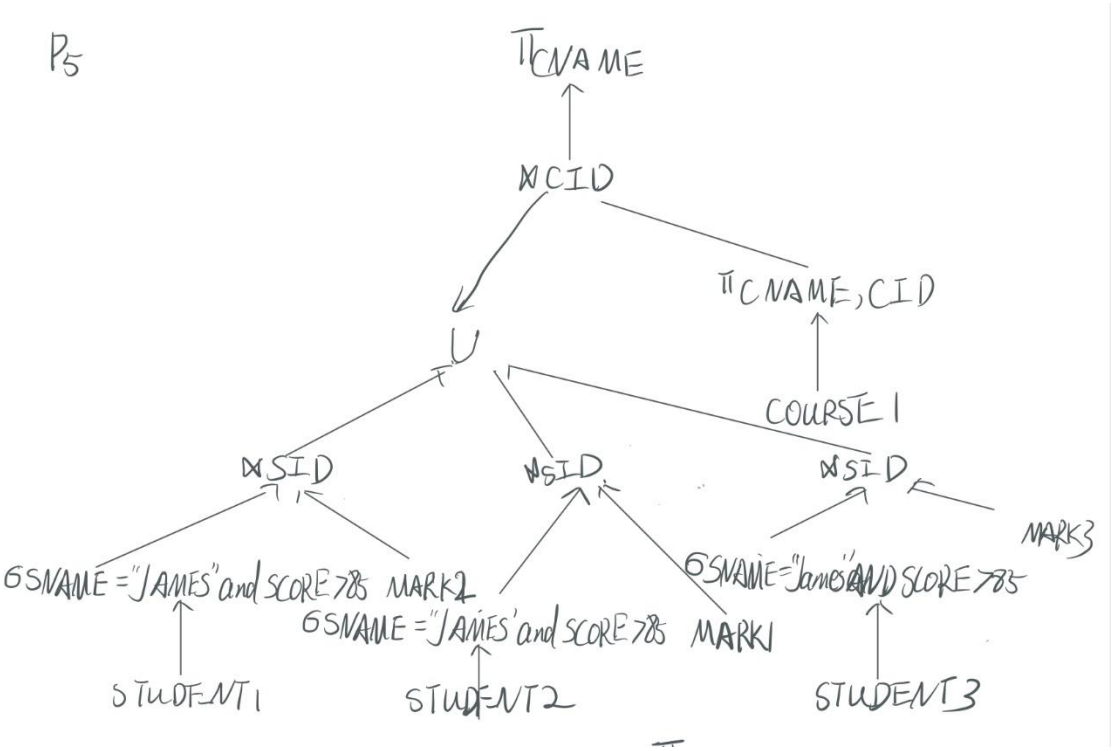


At this point, the idempotency of projection can be used to introduce another projection (CNAME; SID). When this new projection is commuted with ⋈SID, COURSE2 is eliminated since it does not have the CNAME attribute. To do this,

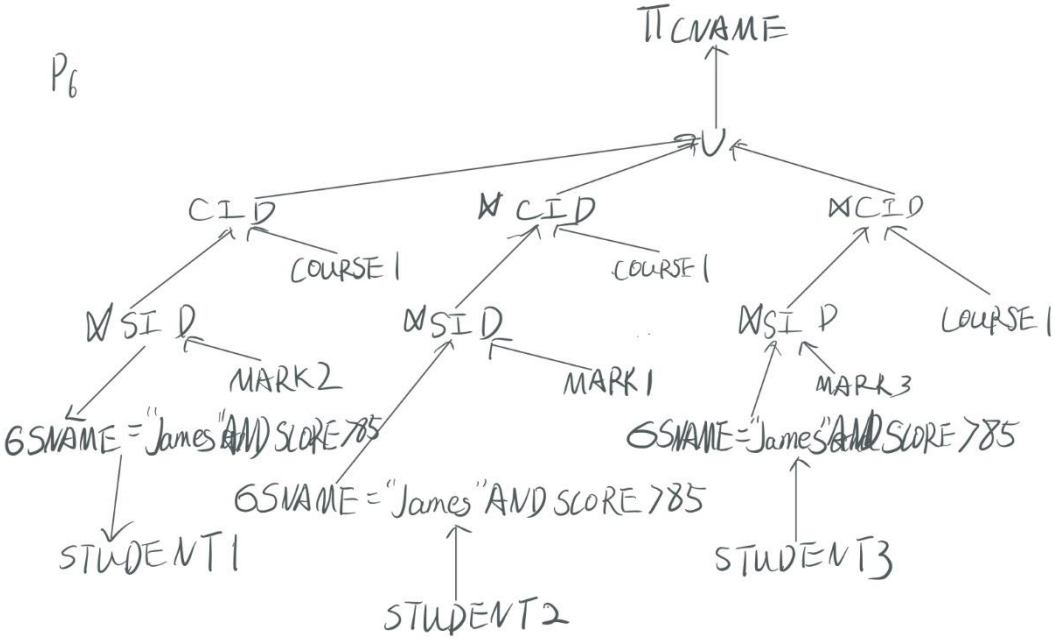
first apply the rule:

PA(R ⋈B S) = PA(R ⋈B (PA,BS))

where A is only an attribute of S. This results in the tree below:



At this point, notice that ΠCNAME;CID over COURSE1 is useless as it is only defined over the two projection attributes anyway. So, the projection can be eliminated. Finally, we can distribute CID over ∪ to get the tree.



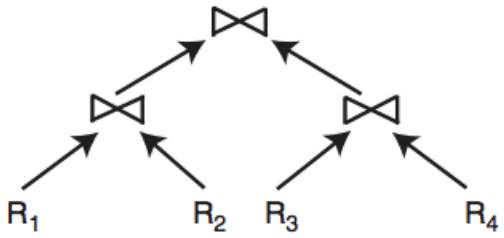
Question 4:

The first table shows the placement of the copies of the 4 relations at the 4 sites and the site loads:

sites	load	R_1	R_2	R_3	R_4
s_1	1	R_{11}		R_{31}	R_{41}
s_2	2		R_{22}		
s_3	2	R_{13}		R_{33}	
s_4	2	R_{14}	R_{24}		

Please describe the detailed processes of how to extend the **SQAllocation algorithm** to the following bushy join trees using the data placement and site loads shown in the table.

For



you need to provide **two** solutions

answer:

To use the SQAllocation algorithm, we set a Q as an ordered sequence of subqueries Q= q1,q2,q3,q4,q5 and each query is the maximum processing unit that access a single base relation and communicates with neighboring subqueries. Then we set q1 with R1, q2 with R2 and joined with the result of q1, q3 with R3 and q4 with R4 and joined with the result of q3. Q5 is the query which connect with two intermediate relations.

Solution1:

The SQAllocation algorithm have 5 iterations.

1) Select q4 because it has the least load and allocate to s1.
2) Select q2 because it has the least load(same as q3) and allocate to s4.
3) Select q3 and allocate to s1 because although it same as s3 but it has a benefit of 1 as a result of the allocation of q4.
4) Select q1 and allocate to s3.
5) Select q5 and allocate to s1 because the load is less that of s3.

Solution2:

The SQAllocation algorithm have 5 iterations.

1) Select q4 because it has the least load and allocate to s1.
2) Select q2 and allocate to s2.
3) Select q3 and allocate to s1
4) Select q1 and allocate to s4.
5) Select q5 and allocate to s1.