M. Tamer Özsu and Patrick Valduriez

Principles of Distributed Database Systems (Third Edition)

Solutions to Exercises

February 25, 2011

Springer

Chapter 3
Distributed Database Design

Problem 3.1 (*). Given relation EMP as in Figure 3.3 [which is duplicated below], let p_1: TITLE $<$ "Programmer" and p_2: TITLE $>$ "Programmer" be two simple predicates. Assume that character strings have an order among them, based on the alphabetical order.

EMP

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng
E2	M. Smith	Syst. Anal.
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E5	B. Casey	Syst. Anal.
E6	L. Chu	Elect. Eng.
E7	R. Davis	Mech. Eng.
E8	J. Jones	Syst. Anal.

ASG

ENO	PNO	RESP	DUR
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P3	Consultant	10
E3	P4	Engineer	48
E4	P2	Programmer	18
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36
E8	P3	Manager	40

PROJ

PNO	PNAME	BUDGET	LOC
P1	Instrumentation	150000	Montreal
P2	Database Develop.	135000	New York
P3	CAD/CAM	250000	New York
P4	Maintenance	310000	Paris

PAY

TITLE	SAL
Elect. Eng.	40000
Syst. Anal.	34000
Mech. Eng.	27000
Programmer	24000

Fig. 3.3 Modified Example Database

(a) Perform a horizontal fragmentation of relation EMP with respect to $\{p_1, p_2\}$.

(b) Explain why the resulting fragmentation (EMP_1, EMP_2) does not fulfill the correctness rules of fragmentation.

(c) Modify the predicates p_1 and p_2 so that they partition EMP obeying the correctness rules of fragmentaion. To do this, modify the predicates, compose all minterm predicates and deduce the corresponding implications, and then perform a horizontal fragmentation of EMP based on these minterm predicates. Finally, show that the result has completeness, reconstruction and disjointness properties.

Solution 3.1.

(a) Since there are two predicates, there will be the following two partitions:

EMP_1: TITLE $<$ "Programmer"

ENO	ENAME	TITLE
E_1	J. Doe	Elect. Eng.
E_3	A. Lee	Mech. Eng.
E_6	L. Chu	Elect. Eng.
E_7	R. Davis	Mech. Eng.

EMP_2: TITLE $>$ "Programmer"

ENO	ENAME	TITLE
E_2	M Smith	Syst. Anal.
E_5	B. Casey	Syst. Anal.
E_6	J. Jones	Syst. Anal.

(b) The resulting fragmentation is incomplete since E_4, who is a Programmer cannot be found in either fragment.

(c) This can be accomplished by changing p_1 as TITLE $\leq$ "Programmer" or changing p_2 as TITLE $\geq$ "Programmer". If p_1 is changed, E_4 will be added to EMP_1; if p_2 is modified, E_4 will be added to EMP_2.

Problem 3.2 (*). Consider relation ASG in Figure 3.3. Suppose there are two applications that access ASG. The first is issued at five sites and attempts to find the duration of assignment of employees given their numbers. Assume that managers, consultants, engineers, and programmers are located at four different sites. The second application is issued at two sites where the employees with an assignment duration of less than 20 months are managed at one site, whereas those with longer duration are managed at a second site. Derive the primary horizontal fragmentation of ASG using the foregoing information.

Solution 3.2. For the first application, we have the following simple predicates:

p_1: RESP = "Manager"
p_2: RESP = "Consultant"
p_3: RESP = "Engineer"
p_4: RESP = "Programmer"

For the second application we have

p_5: DUR ≤ 20
p_6: DUR > 20

Accordingly, we can form the following minterms:

m_1: RESP = "Manager" $\wedge$ DUR ≤ 20
m_2: RESP = "Manager" DUR > 20
m_3: RESP = "Consultant" $\wedge$ DUR ≤ 20
m_4: RESP = "Consultant" DUR > 20
m_5: RESP = "Engineer" $\wedge$ DUR ≤ 20
m_6: RESP = "Engineer" DUR > 20
m_7: RESP = "Programmer" $\wedge$ DUR ≤ 20
m_8: RESP = "Programmer" DUR > 20

Note that m_4, m_5, and m_8 are empty, so in the end we get the following fragments:

ASG_1

ENO	PNO	RESP	DUR
E_1	P_1	Manager	12

ASG_2

ENO	PNO	RESP	DUR
E_5	P_2	Manager	24
E_6	P_3	Manager	48
E_8	P_4	Manager	40

ASG_2

ENO	PNO	RESP	DUR
E_5	P_2	Manager	24
E_6	P_3	Manager	48
E_8	P_4	Manager	40

ASG_3

ENO	PNO	RESP	DUR
E_3	P_3	Consultant	10

ASG_6

ENO	PNO	RESP	DUR
E_3	P_4	Engineer	48
E_7	P_3	Engineer	36

ASG_7

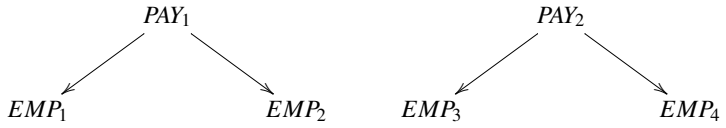
ENO	PNO	RESP	DUR
E_4	P_2	Programmer	18

Problem 3.3. Consider relations EMP and PAY in Figure 3.3. EMP and PAY are horizontally fragmented as follows:

$$\begin{aligned}
\text{EMP}_1 &= \sigma_{\text{TITLE}=\text{“Elect.Eng.”}}(\text{EMP}) \\
\text{EMP}_2 &= \sigma_{\text{TITLE}=\text{“Syst.Anal.”}}(\text{EMP}) \\
\text{EMP}_3 &= \sigma_{\text{TITLE}=\text{“Mech.Eng.”}}(\text{EMP}) \\
\text{EMP}_4 &= \sigma_{\text{TITLE}=\text{“Programmer”}}(\text{EMP}) \\
\text{PAY}_1 &= \sigma_{\text{SAL}\geq 30000}(\text{PAY}) \\
\text{PAY}_2 &= \sigma_{\text{SAL}<30000}(\text{PAY})
\end{aligned}$$

Draw the join graph of EMP $\ltimes_{\text{TITLE}}$ PAY. Is the graph simple or partitioned? If it is partitioned, modify the fragmentation of either EMP or PAY so that the join graph of EMP$\ltimes_{\text{TITLE}}$ PAY is simple.

Solution 3.3. The join graph is the following:

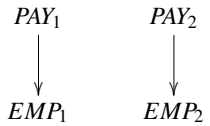


This is not simple, it is partitioned.

To make it simple, we can change either EMP or PAY fragmentation. Let us change EMP fragmentation as follows:

$$\begin{aligned}
&\text{EMP}_1\text{: } \sigma_{\text{TITLE}=\text{“Elect. Eng.”}\wedge\text{TITLE}=\text{“Syst. Anal.”}}\ (\text{EMP}) \\
&\text{EMP}_1\text{: } \sigma_{\text{TITLE}=\text{“Mech. Eng.”}\wedge\text{TITLE}=\text{“Programmer”}}\ (\text{EMP})
\end{aligned}$$

This results in



Problem 3.4. Give an example of a *CA* matrix where the split point is not unique and the partition is in the middle of the matrix. Show the number of shift operations required to obtain a single, unique split point.

Solution 3.4. Not worked out.

Problem 3.5 (**). Given relation PAY as in Figure 3.3, let p_1: SAL < 30000 and p_2: SAL ≥ 3000 be two simple predicates. Perform a horizontal fragmentation of PAY with respect to these predicates to obtain PAY_1, and PAY_2. Using the fragmentation of PAY, perform further derived horizontal fragmentation for EMP. Show completeness, reconstruction, and disjointness of the fragmentation of EMP.

Solution 3.5. First follow COM_MIN algorithm to get the set of simple predicates Pr'.

$p_1 = \text{PAY} < 30000$
$p_2 = \text{PAY} \geq 3000$

Initial input: $Pr = \{p_1, p_2\}$

Start with p_1, so $Pr' = \{p_1\}$ and $Pr = \{p_2\}$. Fragmenting accordingly will give

$f_1 = \sigma_{\text{SAL}<30000}\ \text{PAY} \Longrightarrow F = \{f_1\}$.

Now consider the remaining simple predicate p_2. So, $Pr' = \{p_1, p_2\}$ and $Pr = \emptyset$. Fragmenting according to p_2 will give

$f_2 = \sigma_{\text{SAL}\geq 3000}\ \text{PAY} \Longrightarrow F = \{f_1, f_2\}$.

This is the result that COM_MIN returns; Pr is complete and minimal.

Now we apply PHORIZONTAL algorithm. The following minterm predicates can be formed from Pr returned by COM_MIN:

$m_1 = p_1 \wedge p_2$
$m_2 = p_1 \wedge \neg p_2$
$m_3 = \neg p_1 \wedge p_2$
$m_4 = \neg p_1 \wedge \neg p2$

Among these minterm predicates we note the following implications:

$\neg p_1 \Rightarrow p_2$
$\neg p_2 \Rightarrow p_1$

Therefore, m_4 is not needed. Consequently, we get three fragments:

$\text{PAY}_1 = \sigma_{p_1 \wedge p_2}\ \text{PAY}$
$\text{PAY}_2 = \sigma_{p_1 \wedge \neg p_2}\ \text{PAY}$
$\text{PAY}_3 = \sigma_{\neg p_1 \wedge p_2}\ \text{PAY}$

These fragments will have the following content:

$\text{PAY}_1 = \{\langle \text{Mech. Eng., 27000} \rangle, \langle \text{Programmer, 24000} \rangle\}$
$\text{PAY}_2 = \emptyset$
$\text{PAY}_1 = \{\langle \text{Syst. Anal., 34000} \rangle, \langle \text{Elect. Eng., 40000} \rangle\}$

Problem 3.6 (**). Let $Q = \{q_1, q_2, q_3, q_4, q_5\}$ be a set of queries, $A = \{A_1, A_2, A_3, A_4, A_5\}$ be a set of attributes, and $S = \{S_1, S_2, S_3\}$ be a set of sites. The matrix of Figure 3.21a describes the attribute usage values and the matrix of Figure 3.21b gives the application access frequencies. Assume that $ref_i(q_k) = 1$ for all q_k and S_i and that A_1 is the key attribute. Use the bond energy and vertical partitioning algorithms to obtain a vertical fragmentation of the set of attributes in A.

Solution 3.6. First drive the Attribute Affinity (AA) matrix:

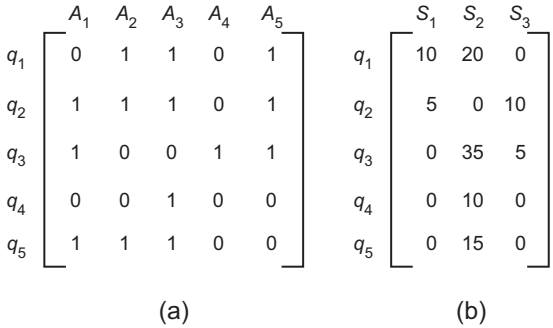


Fig. 3.21 Attribute Usage Values and Application Access Frequencies in Exercise 3.6

	A_1	A_2	A_3	A_4	A_5
A_1	70	30	30	40	55
A_2	30	60	60	0	45
A_3	30	60	70	0	45
A_4	40	0	0	40	40
A_5	55	45	45	40	85

Note: Although, as noted in the book, it doesn't make sense to compute the diagonal values in the AA matrix, we show them here since they are used in the following calculations.

Now we start applying the BEA algorithm. We first fix the first two columns and the Clustered Affinity (CA) Matrix looks like the following:

	A_1	A_2
A_1	70	30
A_2	30	60
A_3	30	60
A_4	40	0
A_5	55	45

Next we consider placing A_3 – there are three places where it can be placed: (a) to the left of A_1, which has a contribution of 16950; (b) in between A_1 and A_2, which has a contribution of 22050, and (c) to the right of A_2, which has a contribution of 21450. Thus, the bets ordering is A_1, A_3, A_2 resulting in the following CA matrix:

	A_1	A_3	A_2
A_1	70	30	30
A_2	30	60	60
A_3	30	70	60
A_4	40	0	0
A_5	55	45	45

When A_4 and A_5 are placed similarly, and the rows are ordered in the same was as columns, we get the following CA matrix:

$$\begin{array}{c} \\ A_4 \\ A_1 \\ A_5 \\ A_3 \\ A_2 \end{array} \begin{array}{c} \begin{array}{ccccc} A_4 & A_1 & A_5 & A_3 & A_2 \end{array} \\ \begin{bmatrix} 40 & 40 & 40 & 0 & 0 \\ 40 & 70 & 55 & 30 & 30 \\ 40 & 55 & 85 & 45 & 45 \\ 0 & 30 & 45 & 70 & 60 \\ 0 & 30 & 45 & 60 & 60 \end{bmatrix} \end{array}$$

Now comes partitioning. When you work through it, you find that the best partitioning is $TA = \{A_4\}$ and $B_A = \{A_1, A_5, A_3, A_2\}$. Since A_1 is the key, it needs to be in both partitions. Thus we get the fragments $F_1 = \{A_1, A_4\}$ and $F_2 = \{A_1, A_5, A_3, A_2\}$.

Problem 3.7 (**). Write an algorithm for derived horizontal fragmentation.

Solution 3.7. No solution provided.

Problem 3.8 (**). Assume the following view definition

```
CREATE  VIEW    EMPVIEW(ENO, ENAME, PNO, RESP) =
AS      SELECT  E.ENO, E. ENAME, ASG.PNO, ASG.RESP
        FROM    EMP, ASG
        WHERE   EMP.ENO=ASG.ENO
        AND     DUR=24
```

is accessed by application q_1, located at sites 1 and 2, with frequencies 10 and 20, respectively. Let us further assume that there is another query q_2 defined as

```
SELECT  ENO, DUR
FROM    ASG
```

which is run at sites 2 and 3 with frequencies 20 and 10, respectively. Based on the above information, construct the $use(q_i, A_j)$ matrix for the attributes of both relations EMP and ASG. Also construct the affinity matrix containing all attributes of EMP and ASG. Finally, transform the affinity matrix so that it could be used to split the relation into two vertical fragments using heuristics or BEA.

Solution 3.8. No solution provided.

Problem 3.9 (**).
Formally define the three correctness criteria for derived horizontal fragmentation.

Solution 3.9. No solution provided.

Problem 3.10 (*). Given a relation $R(K, A, B, C)$ (where K is the key) and the following query

```
SELECT  *
FROM    R
WHERE   R.A = 10 AND R.B=15
```

(a) What will be the outcome of running PHF on this query?
(b) Does the COM_MIN algorithm produce in this case a complete and minimal predicate set? Justify your answer.

Solution 3.10. No solution provided.

Problem 3.11 (*). Show that the bond energy algorithm generates the same results using either row or column operation.

Solution 3.11. This is the solution

Problem 3.12 (**). Modify algorithm PARTITION to allow n-way partitioning, and compute the complexity of the resulting algorithm.

Solution 3.12. No solution provided.

Problem 3.13 (**). Formally define the three correctness criteria for hybrid fragmentation.

Solution 3.13. No solution provided.

Problem 3.14. Discuss how the order in which the two basic fragmentation schemas are applied in hybrid fragmentation affects the final fragmentation.

Solution 3.14. No solution provided.

Problem 3.15 (**). Describe how the following can be properly modeled in the database allocation problem.

(a) Relationships among fragments
(b) Query processing
(c) Integrity enforcement
(d) Concurrency control mechanisms

Solution 3.15. No solution provided.

Problem 3.16 (**). Consider the various heuristic algorithms for the database allocation problem.

(a) What are some of the reasonable criteria for comparing these heuristics? Discuss.
(b) Compare the heuristic algorithms with respect to these criteria.

Solution 3.16. No solution provided.

Problem 3.17 (*). Pick one of the heuristic algorithms used to solve the DAP, and write a program for it.

Solution 3.17. No solution provided.

Problem 3.18 (**). Assume the environment of Exercise 3.8. Also assume that 60% of the accesses of query q_1 are updates to PNO and RESP of view EMPVIEW and that ASG.DUR is not updated through EMPVIEW. In addition, assume that the data transfer rate between site 1 and site 2 is half of that between site 2 and site 3. Based on the above information, find a reasonable fragmentation of ASG and EMP and an optimal replication and placement for the fragments, assuming that storage costs do not matter here, but copies are kept consistent.

Hint: Consider horizontal fragmentation for ASG based on DUR=24 predicate and the corresponding derived horizontal fragmentation for EMP. Also look at the affinity matrix obtained in Example 3.8 for EMP and ASG together, and consider whether it would make sense to perform a vertical fragmentation for ASG.

Solution 3.18. No solution provided.

Chapter 7
Query Decomposition and Data Localization

Problem 7.1. Simplify the following query, expressed in SQL, on our example database using idempotency rules:

```
SELECT ENO
FROM   ASG
WHERE  RESP = "Analyst"
AND    NOT(PNO="P2" OR DUR=12)
AND    PNO ≠ "P2"
AND    DUR=12
```

Solution 7.1. First let us write the query as an algebraic expression:

$$\Pi_{\text{ENO}}\sigma_{\text{RESP=“Analyst”}\wedge\neg(\text{PNO=“P2”}\vee\text{DUR=12})\wedge\text{PNO}\neq\text{“P2”})\wedge\text{DUR=12}}\text{ASG}$$

Consider only the selection predicate and note that this is already in conjunctive normal form. First push the negation inside the second conjunct term to get

RESP=“Analyst” $\wedge\neg$PNO = “P2” $\wedge\neg$DUR = 12 $\wedge$ PNO $\neq$ “P2” $\wedge$ DUR = 12

Notice that

$$\text{DUR} = 12 \wedge \neg\text{DUR} = 12 \Leftrightarrow false$$

Also note that

$$\neg\text{PNO} = \text{“P2”} \Leftrightarrow \text{PNO} \neq \text{“P2”}$$

which results in

$$\text{PNO} \neq \text{“P2”} \wedge \text{PNO} \neq \text{“P2”} \Leftrightarrow \text{PNO} \neq \text{“P2”}$$

Thus, the predicate becomes

$$\text{RESP=“Analyst”} \wedge \text{PNO} \neq \text{“P2”} \wedge false$$

This reduces to $false$ and, therefore, the query result would be empty.

Problem 7.2. Give the query graph of the following query, expressed in SQL, on our example database:

```
SELECT ENAME, PNAME
FROM   EMP, ASG, PROJ
WHERE  DUR > 12
AND    EMP.ENO = ASG.ENO
AND    PROJ.PNO = ASG.PNO
```

and map it into an operator tree.

Solution 7.2. The query graph for this query is given in Figure 7.11.

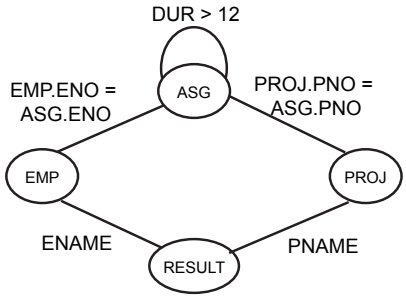


Fig. 7.11 Query Graph for Problem 7.2

The corresponding operator tree is given in Figure 7.12.

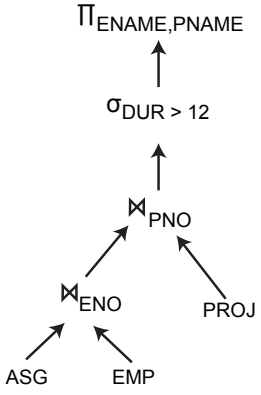


Fig. 7.12 Operator Tree for Problem 7.2

Problem 7.3 (*). Simplify the following query:

```
SELECT ENAME, PNAME
FROM   EMP, ASG, PROJ
WHERE  (DUR > 12 OR RESP = "Analyst")
AND    EMP.ENO = ASG.ENO
AND    (TITLE = "Elect. Eng."
OR     ASG.PNO < "P3")
AND    (DUR > 12 OR RESP NOT= "Analyst")
AND    ASG.PNO = PROJ.PNO
```

and transform it into an optimized operator tree using the restructuring algorithm (Section 7.1.4) where select and project operations are applied as soon as possible to reduce the size of intermediate relations.

Solution 7.3. For simplification, consider only the compound selection predicate in the WHERE clause:

$$(\text{DUR} > 12 \vee \text{RESP} = \text{“Analyst”}) \wedge (\text{TITLE} = \text{“Elect.Eng.”} \vee \text{ASG.PNO} < \text{“P2”}) \\ \wedge (\text{DUR} > 12 \vee \text{RESP NOT} = \text{“Analyst”})$$

Let p_1 be $\langle \text{DUR} > 12 \rangle$, p_2 be $\langle \text{RESP = “Analyst”} \rangle$, p_3 be $\langle \text{TITLE= “Elect. Eng.”} \rangle$, and p_4 be $\langle \text{ASG.PNO} < \text{“P2”} \rangle$, The query qualification is

$$(p_1 \vee p_2) \wedge (p_3 \vee p_4) \wedge (p_1 \vee \neg p_2)$$

First apply rules 1 and 3 of Section 7.1.1 which yields:

$$((p_1 \vee p_2) \wedge (p_1 \vee \neg p_2)) \wedge (p_3 \vee p_4)$$

Then apply rule 5 of Section 7.1.1 which yields:

$$(p_1 \wedge (p_2 \vee \neg p_2)) \wedge (p_3 \vee p_4)$$

Then apply rules 8 and 3 of Section 7.1.3 which yields:

$$p_1 \wedge (p_3 \vee p_4)$$

or

$$(\text{DUR} > 12) \wedge (\text{TITLE} = \text{“Elect.Eng.”} \vee \text{ASG.PNO} < \text{“P3”})$$

which cannot be further simplified.

The first (generic) operator tree is given in Figure 7.13.

Next commute selections over joins. $\sigma_{\text{DUR}>12}$ is commuted with the two joins and is applied to ASG since that is the only relation that has the DUR attribute.

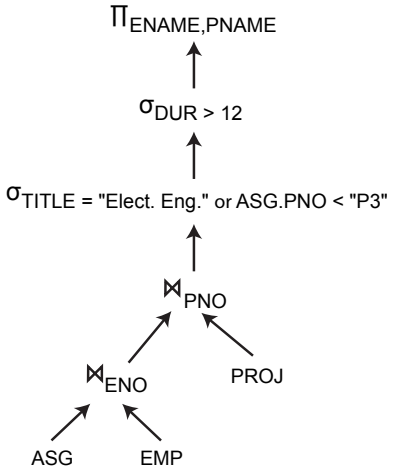


Fig. 7.13 Generic Operator Tree for Problem 7.3

Also, $\sigma_{\text{TITLE}=\text{“Elect.Eng.”} \vee \text{ASG.PNO}<\text{“P3”}}$ is commuted over the first join. It cannot be commuted any further since the predicate is a disjunct and has to be applied to the result of the join $\bowtie_{\text{ENO}}$. This results in the operator tree in Figure 7.14, which is the final one.

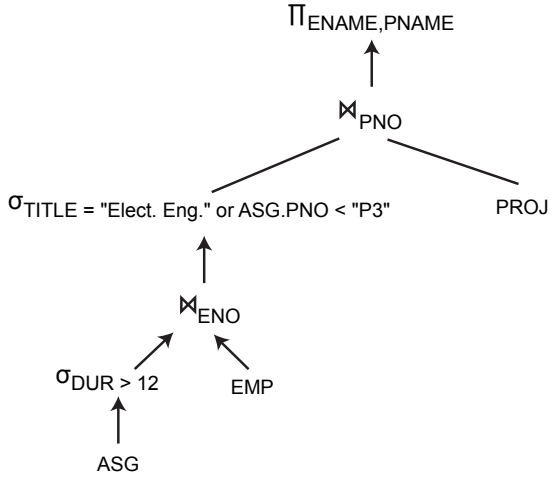


Fig. 7.14 Final Operator Tree for Problem 7.3

Problem 7.4. Transform the operator tree of Figure 7.5 back to the tree of Figure 7.3 using the restructuring algorithm. Describe each intermediate tree and show which rule the transformation is based on.

Solution 7.4. There are multiple ways of proceeding. The following is one way:

1. Using the commutativity of projection with join, move all projections outside of the joins (i.e., do them after the joins) to get the operator tree in Figure 7.15.

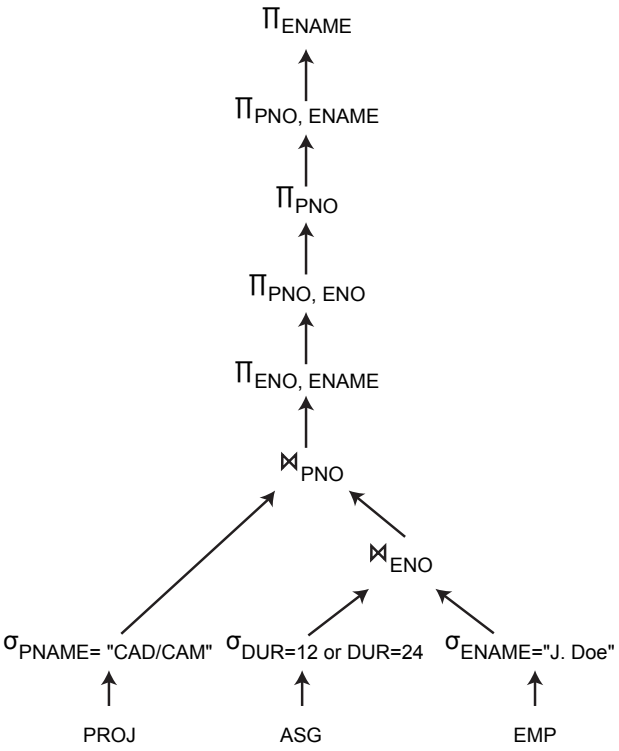


Fig. 7.15 Problem 7.4 - Projections after Joins

2. Now use the idempotency of projection. Since {ENAME} $\subset$ {PNO, ENO, ENAME}, get rid of all the projections other than Π_{ENAME}. This gives us operator tree of Figure 7.16.
3. Finally, use the commutativity of selection with joins to pull selections higher up in the tree to get Figure 7.3.

Problem 7.5 (*). Consider the following query on our Engineering database:

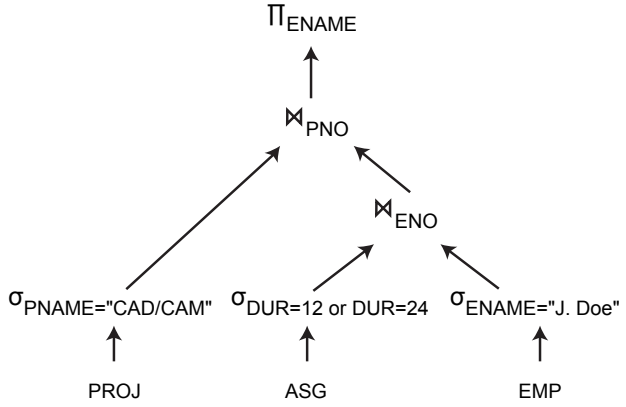


Fig. 7.16 Problem 7.4 - Eliminate Projections

```
SELECT ENAME,SAL
FROM   EMP,PROJ,ASG,PAY
WHERE  EMP.ENO = PROJ.ENO
AND    EMP.TITLE = PAY.TITLE
AND    (BUDGET>200000 OR DUR>24)
AND    ASG.JNO = PROJ.JNO
AND    (DUR>24 OR PNAME="CAD/CAM")
```

Compose the selection predicate corresponding to the WHERE clause and transform it, using the idempotency rules, into the simplest equivalent form. Furthermore, compose an operator tree corresponding to the query and transform it, using relational algebra transformation rules, to three equivalent forms.

Solution 7.5. The selection predicate of this query is the following:

$$(\text{BUDGET} > 200000 \vee \text{DUR} > 24) \wedge (\text{DUR} > 24 \vee \text{PNAME} = \text{“CAD/CAM”})$$

Note that this is in conjunctive normal form. If it is converted to disjunctive normal form and then simplify for the two DUR > 24, it becomes

$$\text{DUR} > 24 \vee (\text{BUDGET} > 200000 \wedge \text{PNAME} = \text{“CAD/CAM”})$$

The generic operator tree corresponding to the simplified query is given in Figure 7.17

Problem 7.6. Assume that relation PROJ of the sample database is horizontally fragmented as follows:

$$\text{PROJ}_1 = \sigma_{\text{PNO} \leq \text{“P2”}}\ (\text{PROJ})$$
$$\text{PROJ}_2 = \sigma_{\text{PNO} > \text{“P2”}}(\text{PROJ})$$

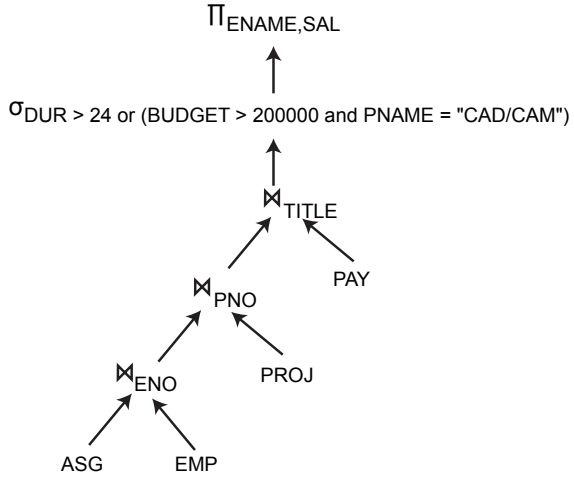


Fig. 7.17 Generic Operator Tree for Problem 7.5

Transform the following query into a reduced query on fragments:

```
SELECT BUDGET
FROM   PROJ,ASG
WHERE  PROJ.PNO = ASG.PNO
AND    PNO = "P4"
```

Solution 7.6. The generic operator tree is of the form given in Figure 7.18.

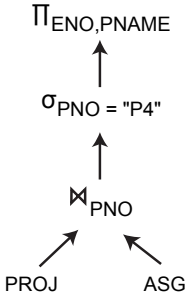


Fig. 7.18 Generic Operator Tree for Problem 7.6

Since PROJ is horizontally fragmented, its localization program is

$$\text{PROJ} = \text{PROJ}_1 \cup \text{PROJ}_2$$

Replacing PROJ with its localization program and pushing down the select predicate on PROJ gives the operator tree in Figure 7.19.

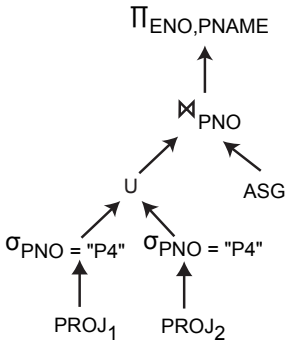


Fig. 7.19 Localized Operator Tree for Problem 7.6

The join can be distributed over union to get the query of Figure 7.20.

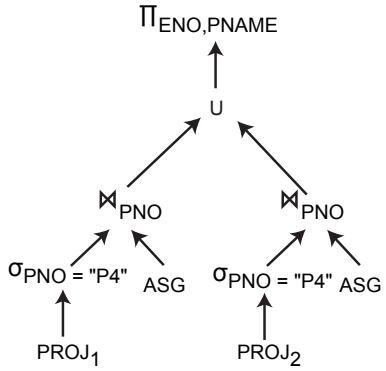


Fig. 7.20 New Localized Operator Tree for Problem 7.6

We can then eliminate the left subtree of the union since PROJ_1 cannot have have tuples with PNO = “P4”. The final operator tree is in Figure 7.21.

Problem 7.7. Assume that relation PROJ is horizontally fragmented as in Problem 7.6, and that relation ASG is horizontally fragmented as

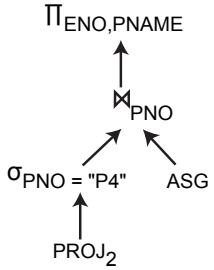


Fig. 7.21 Final Localized Operator Tree for Problem 7.6

$$\begin{aligned} ASG_1 &= \sigma_{PNO \leq \text{"P2"}}(ASG) \\ ASG_2 &= \sigma_{\text{"P2"} < PNO \leq \text{"P3"}}(ASG) \\ ASG_3 &= \sigma_{PNO > \text{"P3"}}(ASG) \end{aligned}$$

Transform the following query into a reduced query on fragments, and determine whether it is better than the generic query:

```
SELECT RESP, BUDGET
FROM   ASG, PROJ
WHERE  ASG.PNO = PROJ.PNO
AND    PNAME = "CAD/CAM"
```

Solution 7.7. The generic query corresponds to the operator tree of Figure 7.22.

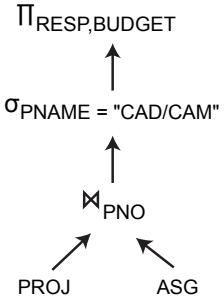


Fig. 7.22 Generic Operator Tree for Problem 7.7

Both relations are horizontally fragmented. So, replacing them with their localization programs yields Figure 7.23.

Distributing join over union results in six joins, whose results are unioned:

1. $PROJ_1 \bowtie ASG_1$

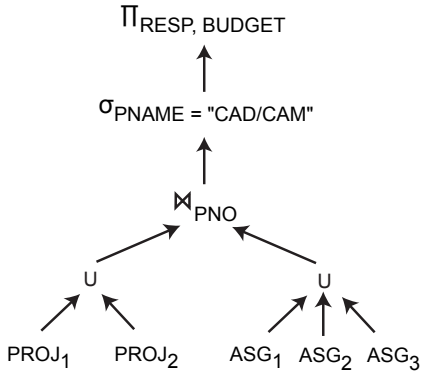


Fig. 7.23 Localized Operator Tree for Problem 7.7

2. $PROJ_1 \bowtie ASG_2$
3. $PROJ_1 \bowtie ASG_3$
4. $PROJ_2 \bowtie ASG_1$
5. $PROJ_2 \bowtie ASG_2$
6. $PROJ_2 \bowtie ASG_3$

From these, note that the fragmentation predicates of (2), (3), and (4) conflict, so they can be eliminated. The reduced query is depicted in Figure 7.24, which is better than the generic query because more parallelism is introduced and a number of unnecessary joins are eliminated, thus reducing the cost of the join operator.

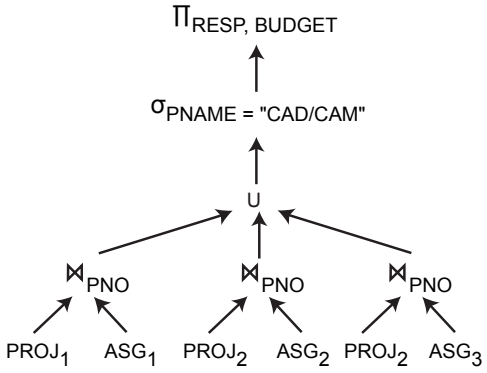


Fig. 7.24 Final Operator Tree for Problem 7.7

Problem 7.8. Assume that relation PROJ is fragmented as in Problem 7.6. Furthermore, relation ASG is indirectly fragmented as

$$\text{ASG}_1 = \text{ASG} \ltimes_{\text{PNO}} \text{PROJ}_1$$
$$\text{ASG}_2 = \text{ASG} \ltimes_{\text{PNO}} \text{PROJ}_2$$

and relation EMP is vertically fragmented as

$$\text{EMP}_1 = \Pi_{\text{ENO,ENAME}}(\text{EMP})$$
$$\text{EMP}_2 = \Pi_{\text{ENO,TITLE}}(\text{EMP})$$

Transform the following query into a reduced query on fragments:

```
SELECT ENAME
FROM   EMP,ASG,PROJ
WHERE  PROJ.PNO = ASG.PNO
AND    PNAME = "Instrumentation"
AND    EMP.ENO = ASG.ENO
```

Solution 7.8. This is similar to Problem 7.7. The generic operator tree is depicted in Figure 7.25.

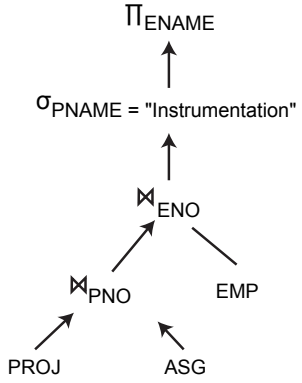


Fig. 7.25 Generic Operator Tree for Problem 7.8

The localization programs for the base relations are as follows:

- $\text{PROJ} = \text{PROJ}_1 \cup \text{PROJ}_2$
- $\text{ASG} = \text{ASG}_1 \cup \text{ASG}_2$
- $\text{EMP} = \text{EMP}_1 \bowtie \text{EMP}_2$

which results in the operator tree given in Figure 7.26.

Considering the subtree under $\bowtie_{\text{PNO}}$. When the join is distributed over union, the following four joins result, which are then unioned:

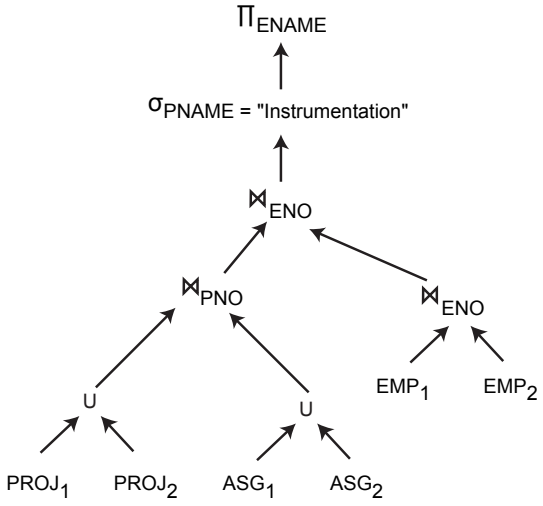


Fig. 7.26 Localized Operator Tree for Problem 7.8

- $\text{PROJ}_1 \bowtie \text{ASG}_1$
- $\text{PROJ}_1 \bowtie \text{ASG}_2$
- $\text{PROJ}_2 \bowtie \text{ASG}_1$
- $\text{PROJ}_2 \bowtie \text{ASG}_2$

Since ASG_1 (ASG_2) is derived from PROJ_1 (PROJ_2), only two joins will produce results: $\text{PROJ}_1 \bowtie \text{ASG}_1$ and $\text{PROJ}_2 \bowtie \text{ASG}_2$, so the other two can be eliminated.

Note that there are two $\bowtie_{ENO}$ feeding into one another. These can reduce to a single join, which results in the operator tree of Figure 7.27.

At this point, selection can be commuted with joins and union (i.e., pushed down the tree). Since the selection only applies to the PROJ relation, it is pushed down one path of the joins (Figure 7.28).

At this point, the idempotency of projection can be used to introduce another projection ($\Pi_{\text{ENAME,ENO}}$). When this new projection is commuted with $\bowtie_{\text{ENO}}$, EMP_2 is eliminated since it does not have the ENAME attribute. To do this, first apply the rule:

$$\Pi_A(R \bowtie_B S) = \Pi_A(R \bowtie_B (\Pi_{A,B} S))$$

where A is only an attribute of S. This results in Figure 7.29.

At this point, notice that $\Pi_{\text{ENAME,ENO}}$ over EMP_1 is useless as it is only defined over the two projection attributes anyway. So, the projection can be eliminated. Finally, we can distribute $\bowtie_{\text{ENO}}$ over $\cup$ to get Figure 7.30.

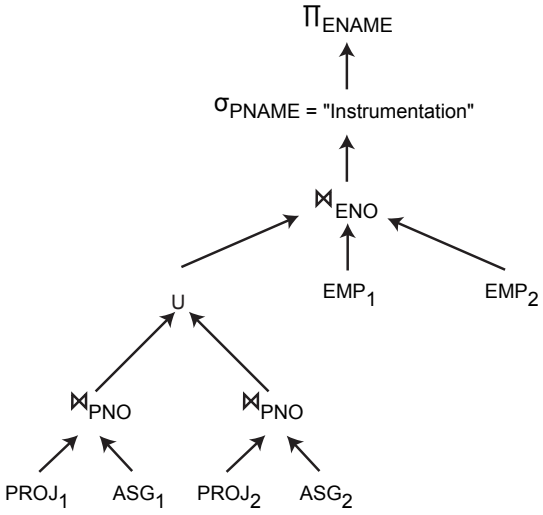


Fig. 7.27 Reduced Operator Tree for Problem 7.8

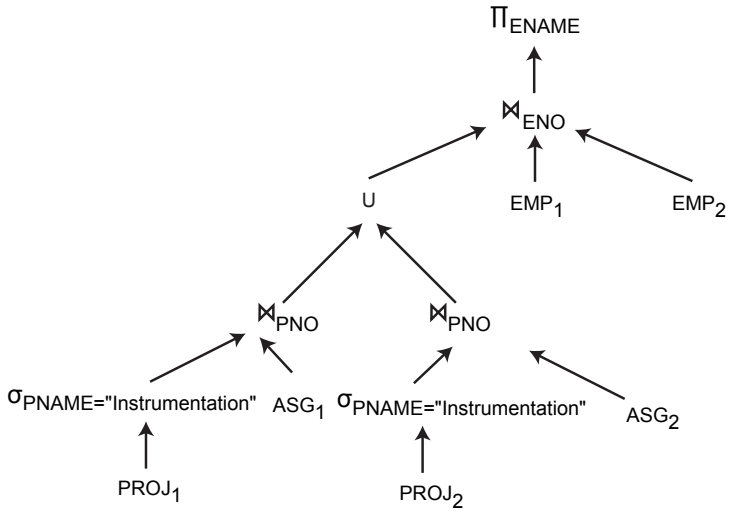


Fig. 7.28 Problem 7.8 - Selection Pushed Down

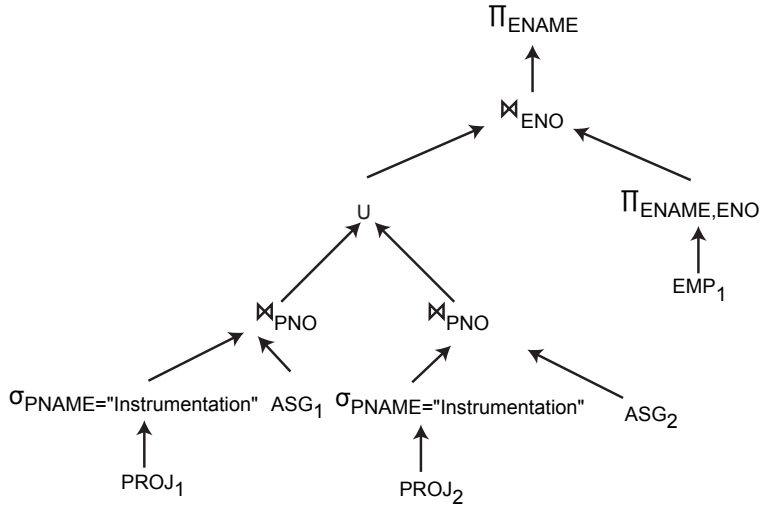


Fig. 7.29 Problem 7.8 - Projection Pushed Down

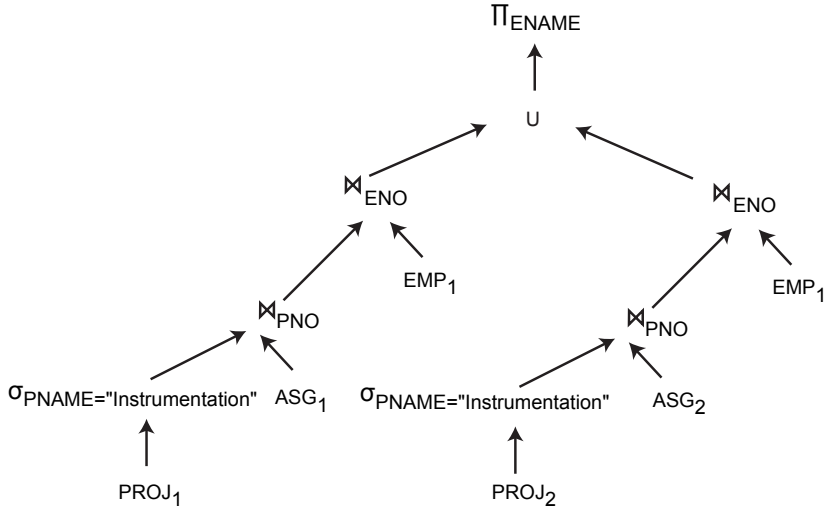


Fig. 7.30 Final Operator Tree for Problem 7.8

Chapter 8
Optimization of Distributed Queries

Problem 8.1. Apply the dynamic query optimization algorithm in Section 8.2.1 to the query of Exercise 8.3, and illustrate the successive detachments and substitutions by giving the monorelation subqueries generated.

Solution 8.1. The input query is :

```
SELECT ENAME, PNAME
FROM   EMP, ASG, PROJ
WHERE  DUR > 12
AND    EMP.ENO = ASG.ENO
AND    (TITLE="Elect. Eng."
OR     ASG.PNO < "P3")
AND    ASG.PNO = PROJ.PNO
```

After detachment of the selection on ASG, this query is replaced by q_1 followed by q_2, where GVAR is an intermediate relation.

q_1:
```
SELECT ENO, PNO INTO GVAR
FROM   ASG
WHERE  DUR > 12
```

q_2:
```
SELECT ENAME, PNAME
FROM   EMP, GVAR, PROJ
WHERE  EMP.ENO=GVAR.ENO
AND    (TITLE = "Elect. Eng."
OR     ASG.PNO < "P3")
AND    PROJ.PNO=GVAR.PNO
```

Query q_1 is monorelation and can be preformed by the OVQP. Query q_2 can be further detached into q_{21} followed by q_{22}, where EGVAR is an intermediate relation.

q_{21}:
```
SELECT ENAME, PNO INTO EGVAR
FROM   EMP, GVAR
WHERE  EMP.ENO=GVAR.ENO
AND    (TITLE = "Elect. Eng."
OR     ASG.PNO < "P3")
```

q_{22}:
```
SELECT ENAME, PNAME
FROM   EGVAR, PROJ
WHERE  EGVAR.PNO=PROJ.PNO
```

Queries q_{21} and q_{22} are irreducible and must be processed by tuple substitution.

Problem 8.2. Consider the join graph of Figure 8.12 [duplicated below] and the following information: *size*(EMP) = 100, *size*(ASG) = 200, *size*(PROJ) = 300, *size*(EMP ⋈ ASG) = 300, and *size*(ASG ⋈ PROJ) = 200. Describe an optimal join program based on the objective function of total transmission time.

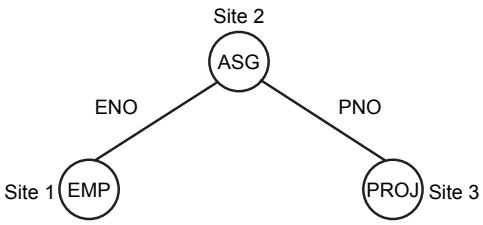


Fig. 8.12 Relation Statistics

Solution 8.2. The optimal join program is :

1. ASG → site 3
2. EMP → site 3
3. Site 3 computes EMP ⋈ ASG ⋈ PROJ

The total transmission time will be that of transferring ASG and EMP to site 3.

Problem 8.3. Consider the join graph of Figure 8.12 and make the same assumptions as in Exercise 8.2. Describe an optimal join program that minimizes response time (consider only communication).

Solution 8.3. The optimal join program is the same as for Exercise 8.2 and the response time will be that of the larger relation (ASG).

Problem 8.4. Consider the join graph of Figure 8.12, and give a program (possibly not optimal) that reduces each relation fully by semijoins.

Solution 8.4. First, we reduce EMP by semijoin as follows :

1. Π_{ENO}(ASG) → site 1
2. Site 1 computes EMP' = EMP $\ltimes_{\text{ENO}}$ ASG

Second, we reduce PROJ by semijoin :

1. Π_{PNO} (ASG) → site 3

2. Site 3 computes PROJ' = PROJ $\ltimes_{\text{PNO}}$ ASG

Third, we reduce ASG by semijoin :

1. Π_{ENO} (EMP) $\rightarrow$ site 2
2. Π_{PNO} (ASG) $\rightarrow$ site 2
3. Site 2 computes ASG' = (ASG $\ltimes_{\text{ENO}}$ EMP) $\ltimes_{\text{PNO}}$ PROJ

Finally, we send semijoined relations EMP' and PROJ' to site 2 which joins all semijoined relations.

1. EMP' $\rightarrow$ site 2
2. PROJ' $\rightarrow$ site 2
3. Site 2 computes EMP' $\bowtie_{\text{ENO}}$ ASG' $\bowtie_{\text{PNO}}$ PROJ'

Problem 8.5. Consider the join graph of Figure 8.12 and the fragmentation depicted in Figure 8.18. Also assume that *size*(EMP $\bowtie$ ASG) = 2000 and *size*(ASG $\bowtie$ PROJ) = 1000. Apply the dynamic distributed query optimization algorithm in Section 8.4.1 in two cases, general network and broadcast network, so that communication time is minimized.

Rel.	Site 1	Site 2	Site 3
EMP	1000	1000	1000
ASG		2000	
PROJ	1000		

Fig. 8.18 Fragmentation

Solution 8.5. First, we apply detachment to the query to obtain q_1 followed by q_2, which we express in relational algebra as :

$$q_1\text{: ASG' = ASG} \bowtie \text{PROJ}$$
$$q_2\text{: Result = EMP} \bowtie \text{ASG'}$$

q_1 and q_2 are irreducible and need be processed by tuple substitution at the processing site.

Let us now apply the dynamic algorithm for the case of a point-to-point network. For q_1, the arrangement of the sites by decreasing order of amounts of useful data

is site 2, site 1. Thus, site 2 is chosen as processing site and PROJ is sent to site 2 which computes q_1.

For q_2, the arrangement of the sites is site 2, site 1, site 3. Thus, site 2 is chosen as processing site and each fragment of EMP (at sites 1 and 3) is sent to site 2 which then computes q_2.

Let us now consider the case of a broadcast network. For q_1, we have the same strategy as in the case of a point-to-point network. For q_2, it is better to replicate ASG' at sites 1 and 3 (at a communication cost of 1000 kbytes) and to compute the result there. This increases parallelism, but the result is fragmented at sites 1 and 3.

Problem 8.6. Consider the join graph of Figure 8.19 and the statistics given in Figure 8.20. Apply the semijoin-based distributed query optimization algorithm in Section 8.4.3 with $T_{MSG} = 20$ and $T_{TR} = 1$.

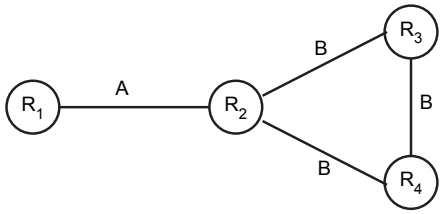


Fig. 8.19 Join Graph

relation	size
R_1	1000
R_2	1000
R_3	2000
R_3	1000

(a)

attribute	size	SF_{SJ}
$R_1.A$	200	0.5
$R_2.A$	100	0.1
$R_2.A$	100	0.2
$R_3.B$	300	0.9
$R_4.B$	150	0.4

(b)

Fig. 8.20 Relation Statistics

Solution 8.6. The initial set of beneficial semijoins contains:

SJ_1: $R_1 \ltimes R_2$, whose benefit is $900 = (1-0.1)*1000$ and cost is 120
SJ_2: $R_3 \ltimes R_2$, whose benefit is $1600 = (1-0.2)*2000$ and cost is 120
SJ_3: $R_3 \ltimes R_4$, whose benefit is $1200 = (1-0.4)*2000$ and cost is 170
SJ_4: $R_2 \ltimes R_4$, whose benefit is $600 = (1-0.4)*1000$ and cost is 170
SJ_5: $R_2 \ltimes R_1$, whose benefit is $500 = (1-0.5)*1000$ and cost is 220
SJ_6: $R_4 \ltimes R_2$, whose benefit is $800 = (1-0.2)*1000$ and cost is 120

Furthermore there are two nonbeneficial semijoins:

SJ_7: $R_2 \ltimes R_3$, whose benefit is $100 = (1-0.9)*1000$ and cost is 320
SJ_8: $R_4 \ltimes R_3$, whose benefit is $100 = (1-0.9)*1000$ and cost is 320

At the first iteration of the selection of beneficial semijoins, SJ_2 is appended to the execution strategy ES. The effect on the statistics of R'_3 (the result of SJ_2) is :

- $size(R'_3) = 2000 * 0.2 = 400$;
- $SF_{SJ}(R'_3.B) = 0.9*0.2 = 0.18$;
- $size(R'_3.B) = 300 * 0.2 = 60$.

At the second iteration, there are two new (or different) beneficial semijoins:

SJ_3: $R'_3 \ltimes R_4$, whose benefit is $328 = (1-0.18)*400$ and cost is 170
SJ_8: $R_4 \ltimes R'_3$, whose benefit is $820 = (1-0.18)*1000$ and cost is 80

The most beneficial semijoin (among all beneficial semijoins) is now SJ_8 which is appended to ES. The effect on the statistics of R'_4 (the result of SJ_8) is :

- $size(R'_4) = 1000 * 0.18 = 180$;
- $SF_{SJ}(R'_4.B) = 0.4*0.18 = 0.072$;
- $size(R'_4.B) = 150 * 0.18 = 27$.

At the third iteration, there are three different beneficial semijoins:

SJ_3: $R'_3 \ltimes R'_4$, whose benefit is $371 = (1-0.072)*400$ and cost is 47
SJ_4: $R_2 \ltimes R'_4$, whose benefit is $928 = (1-0.072)*1000$ and cost is 47
SJ_6: $R'_4 \ltimes R_2$, whose benefit is $144 = (1-0.2)*180$ and cost is 120

SJ_4 is the most beneficial semijoin and is appended to ES. The effect on the statistics of R'_2 (the result of SJ_4) is :

- $size(R'_2) = 1000 * 0.072 = 72$;
- $SF_{SJ}(R'_2.B) = 0.2*0.072 = 0.014$;
- $size(R'_2.B) = 100 * 0.072 = 7$.

At the fourth iteration, the most beneficial semijoin is:

SJ_1: $R_1 \ltimes R'_2$, whose benefit is $986 = (1-0.014)*1000$ and cost is 27

SJ_1 is appended to ES. The effect on the statistics of R'_1 (the result of SJ_1) is :

- $size(R'_1) = 1000 * 0.014 = 14$;
- $SF_{SJ}(R'_1.B) = 0.5 * 0.014 = 0.007$;
- $size(R'_1.B) = 200 * 0.014 = 2$.

At the fifth iteration, there are two beneficial semijoins:

SJ_3: $R'_3 \ltimes R'_4$, whose benefit is 371 and cost is 47
SJ_5: $R'_2 \ltimes R'_1$, whose benefit is $71.5 = (1 - 0.007) * 72$ and cost is 34

The most beneficial semijoin is SJ_3 which is appended to ES. Finally, at the sixth iteration, SJ_5 is appended to ES.

Let us assume that each relation R_i is stored at site i. After reduction by semijoins, the site with the largest amount of data is site 3 (with 400 kbytes). Therefore, site 3 is chosen as assembly site.

The post-optimization phase removes SJ_3 from ES since SJ_3 reduces R_3 at site 3. Let $R'_2 = R_2 \ltimes (R_4 \ltimes (R_3 \ltimes (R_2))$, the final strategy is to send $R'_2 \ltimes (R_1) \ltimes R'_2)$, $R_1 \ltimes (R_2 \ltimes (R_4 \ltimes (R_3 \ltimes (R_2))))$, and $R_4 \ltimes (R_3) \ltimes R_2)$ to site 3, where the final result is computed.

Problem 8.7. Consider the query in Problem 8.5. Assume that relations EMP, ASG, PROJ and PAY have been stored at sites 1, 2, and 3 according to the table in Figure 8.21. Assume also that the transfer rate between any two sites is equal and that data transfer is 100 times slower than data processing performed by any site. Finally, assume that $size(R \bowtie S) = max(size(R), size(S))$ for any two relations R and S, and the selectivity factor of the disjunctive selection of the query in Exercise 8.5 is 0.5. Compose a distributed program which computes the answer to the query and minimizes total time.

Rel.	Site 1	Site 2	Site 3
EMP	2000		
ASG		3000	
PROJ			1000
PAY			500

Fig. 8.21 Fragmentation Statistics

Solution 8.7. The input query is :

```
SELECT ENAME,SAL
FROM   EMP,PROJ,ASG,PAY
WHERE  EMP.ENO = PROJ.ENO
AND    EMP.TITLE = PAY.TITLE
AND    (BUDGET>200000 OR DUR>24)
AND    ASG.PNO = PROJ.PNO
```

Let us apply the static technique described in Section 8.4.2 which is based on join ordering. Since joins do not have good selectivity,we assume that ship-whole is selected for transferring inner joined relations. One distributed program that minimizes total time is to simply send relations PROJ, EMP and PAY one-at-a-time to site 2 which holds the largest amount of data and perform the join as they arrive. In the following program, we assume that the righ-hand side relation of a join is the inner relation to be sent using ship-whole to site 2. We also ignore projections.

1. $R_1 = \sigma_{\text{BUDGET} \not< 2000 \vee \text{DUR} \not< 24}(\text{ASG} \bowtie \text{PROJ})$
2. $R_2 = R_1 \bowtie \text{EMP}$
3. Result = $R_2 \bowtie \text{PAY}$

Problem 8.8. In Section 8.4.4, we described the hybrid algorithm SQAllocation for linear join trees. Extend this algorithm to support bushy join trees. Apply it to the bushy join tree in Figure 8.3(b) using the data placement and site loads shown in Figure 8.18.

Solution 8.8. In the SQAllocation algorithm, a query Q is represented as an ordered sequence of subqueries $Q = q_1, .., q_m$ and each subquery q_i is the maximum processing unit that accesses a single base relation and communicates with it neighboring subqueries. We can decompose the query corresponding to the bushy tree as $Q = q_1, q_2, q_3, q_4, q_5$ where q_1 is associated with R_1, q_2 with R_2 joined with the result of q_1, q_3 with R_3, q_4 with R_4 joined with the result of q_3. However, q_5 which takes as input two intermediate results cannot be associated with a base relation. A solution is then to associate it with the two intermediate relations, i.e. produced by q_2 and q_4. Thus, the SQAllocation algorithm must be extended to be able to allocate subqueries that do not access base relations. The solution is simply to consider this kind of subqueries last, after all subqueries accessing a base relation have been allocated.

The extended SQAllocation algorithm performs 5 iterations. In the first one, it selects q_4 which has the least allocation flexibility and allocates it to s_1. In the second iteration, the next subqueries to select are either q_2 or q_3 which have same allocation flexibility. Let us choose q_2 and assume it gets allocated to s_4 (it could get allocated to s_2 which has same load as s_4. In the third iteration, the next subquery selected is q_3 and it is allocated to s_1 which has the same load as s_3 but a benefit of 1 (versus 0 for s_3) as a result of the allocation of q_4. In the fourth iteration, q_1 gets allocated to s_3. In the last iteration, q_5 would be allocated to s_1 whose load is less than that of s_3.

Chapter 11
Distributed Concurrency Control

Problem 11.1. Which of the following histories are conflict equivalent?

$$H_1 = \{W_2(x), W_1(x), R_3(x), R_1(x), W_2(y), R_3(y), R_3(z), R_2(x)\}$$
$$H_2 = \{R_3(z), R_3(y), W_2(y), R_2(z), W_1(x), R_3(x), W_2(x), R_1(x)\}$$
$$H_3 = \{R_3(z), W_2(x), W_2(y), R_1(x), R_3(x), R_2(z), R_3(y), W_1(x)\}$$
$$H_4 = \{R_2(z), W_2(x), W_2(y), W_1(x), R_1(x), R_3(x), R_3(z), R_3(y)\}$$

Solution 11.1. The easiest way to reason is to create the following table that shows the conflicting operations and their ordering in each of the histories:

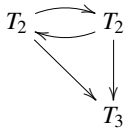
Conflicting ops	H_1	H_2	H_3	H_4
$W_2(x), W_1(x)$	$\prec$	$\succ$	$\prec$	$\prec$
$W_2(x), R_3(x)$	$\prec$	$\succ$	$\prec$	$\prec$
$W_2(x), R_1(x)$	$\prec$	$\prec$	$\prec$	$\prec$
$W_1(x), R_3(x)$	$\prec$	$\prec$	$\succ$	$\prec$
$W_2(y), R_3(y)$	$\prec$	$\succ$	$\prec$	$\prec$
$R_2(x), W_1(x)$	$\succ$	–	–	–

For any of the histories to be conflict-equivalent, they need to have identical relationships, i.e., we need to find the entries in the columns to be identical. From this table, one can see that there are no two columns that are identical. Therefore, none of these histories are conflct-equivalent.

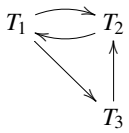
Problem 11.2. Which of the above histories $H_1 - H_4$ are serializable?

Solution 11.2. Again, it is best to refer to the table in Problem 1. From this table, you can reason about serializability by building serialization graphs for each history as follows.

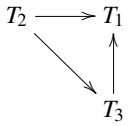
- H_1 is not serializable since it has the following serialization graph, which contains a cycle:



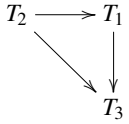
- H_2 is **not** serializable since it has the following serialization graph, which contains a cycle:



- H_3 is serializable since it it has the following serialization graph that is equivalent to the serial history $T_2 \rightarrow T_3 \rightarrow T_2$:



- H_4 is serializable since it it has the following serialization graph that is equivalent to the serial history $T_2 \rightarrow T_1 \rightarrow T_3$:



Problem 11.3. Give a history of two complete transactions which is not allowed by a strict 2PL scheduler but is accepted by the basic 2PL scheduler.

Solution 11.3. No solution is provided.

Problem 11.4 (*). One says that history H is *recoverable* if, whenever transaction T_i reads (some item x) from transaction T_j $(i \neq j)$ in H and C_i occurs in H, then $C_j \prec_S C_i$. T_i "reads x from" T_j in H if

1. $W_j(x) \prec_H R_i(x)$, and
2. $A_j$not $\prec_H R_i(x)$, and
3. if there is some $W_k(x)$ such that $W_j(x) \prec_H W_k(x) \prec_H R_i(x)$, then $A_k \prec_H R_i(x)$.

Which of the following histories are recoverable?

$H_1 = \{W_2(x), W_1(x), R_3(x), R_1(x), C_1, W_2(y), R_3(y), R_3(z), C_3, R_2(x), C_2\}$
$H_2 = \{R_3(z), R_3(y), W_2(y), R_2(z), W_1(x), R_3(x), W_2(x), R_1(x), C_1, C_2, C_3\}$
$H_3 = \{R_3(z), W_2(x), W_2(y), R_1(x), R_3(x), R_2(z), R_3(y), C_3, W_1(x), C_2, C_1\}$
$H_4 = \{R_2(z), W_2(x), W_2(y), C_2, W_1(x), R_1(x), A_1, R_3(x), R_3(z), R_3(y), C_3\}$

Solution 11.4.
H_1: T_3 reads from T_1 (it reads x)
T_3 reads from T_2 (it reads y)
$C_1 \prec C_3$ and $C_3 \prec C_2$
Therefore, H_1 is not recoverable.
H_2: T_3 reads from T_1
T_1 reads from T_2
$C_1 \prec C_3$ and $C_1 \prec C_2$
Therefore, H_2 is not recoverable.
H_3: T_1 reads from T_2
T_3 reads from T_2
$C_2 \prec C_1$ and $C_3 \prec C_2$
Therefore, H_3 is not recoverable.
H_4: T_3 reads from T_2 (it reads x)
T_3 reads from T_2 (it reads y)
$C_2 \prec C_3$
Therefore, H_4 is recoverable.

Problem 11.5 (*). Give the algorithms for the transaction managers and the lock managers for the distributed two-phase locking approach.

Solution 11.5. No solution is provided.

Problem 11.6 (**). Modify the centralized 2PL algorithm to handle phantoms. (See Chapter 10 for a definition of phantoms.)

Solution 11.6. No solution is provided.

Problem 11.7. Timestamp ordering-based concurrency control algorithms depend on either an accurate clock at each site or a global clock that all sites can access (the clock can be a counter). Assume that each site has its own clock which "ticks" every 0.1 second. If all local clocks are resynchronized every 24 hours, what is the maximum drift in seconds per 24 hours permissible at any local site to ensure that a timestamp-based mechanism will successfully synchronize transactions?

Solution 11.7. No solution is provided.

Problem 11.8 (**). Incorporate the distributed deadlock strategy described in this chapter into the distributed 2PL algorithms that you designed in Problem 11.5.

Solution 11.8. This is the solution

Problem 11.9. Explain the relationship between transaction manager storage requirement and transaction size (number of operations per transaction) for a transaction manager using an optimistic timestamp ordering for concurrency control.

Solution 11.9. Consider the optimistic timestamp ordering algorithm. The basic tenets of the algorithm are the following:

1. Timestamps are assigned onto to transactions.
2. Timestamps are assigned at the beginning of validation phase and copied to all the subtransactions.
3. For each subtransaction, local validation is performed.
4. Then a global validation is performed

The storage overhead is caused by having to store the read and write sets. This can be specified as $S =$ number of reads $+$ number of writes. Assume m is the storage requirement for storing the fact that one item is read or written. Now, assuming that no item is read or written multiple times, the storage overhead is mS. This is required for each transaction, but we only need to maintain this for a transaction T_i if there is another transaction that is currently overlapping T_i. Suppose t is the maximum number of overlapping transactions at any time. Then we need tmS space.

Problem 11.10 (*). Give the scheduler and transaction manager algorithms for the distributed optimistic concurrency controller described in this chapter.

Solution 11.10. No solution is provided, but the basic outline is as follows:
Transaction Manager (TM):

1. Divide transactons into subtrasnactions.
2. Send subtransactions to local sites for read/compute phase.
3. When all subtransactions have sent info at beginning of heir valudation phase, assign timestamp and send to Scheduler (SC).
4. Do global validation.
5. Send Commit/Abort to local SC.

Schedule (SC):

1. Receive subtransactions.
2. Execute operations in subtransactions.
3. Return to TM at the beginning of validation phase.
4. Using TS do local valudation – Return to TM.
5. If “Commit” do writes and commit.

Important points:

- Timestamp must be assigned by TM.
- TM must create subtransactions.
- SC must do validation.

Problem 11.11. Recall from the discussion in Section 11.7 that the computational model that is used in our descriptions in this chapter is a centralized one. How would the distributed 2PL transaction manager and lock manager algorithms change if a distributed execution model were to be used?

Solution 11.11. No solution is provided.

Problem 11.12. It is sometimes claimed that serializability is quite a restrictive correctness criterion. Can you give examples of distributed histories that are correct (i.e., maintain the consistency of the local databases as well as their mutual consistency) but are not serializable?

Solution 11.12. No solution is provided.

Chapter 12
Distributed DBMS Reliability

Problem 12.1. Briefly describe the various implementations of the process pairs concept. Comment on how process pairs may be useful in implementing a fault-tolerant distributed DBMS.

Solution 12.1. No solution is provided.

Problem 12.2 (*). Discuss the site failure termination protocol for 2PC using a distributed communication topology.

Solution 12.2. We don't provide a full algorithm, but here are the considerations. The important point to observe here is the uniformity of the protocol. There is no need to consider the coordinator and participant cases separately (other than the INITIAL state).

Timeouts in INITIAL state.

Handle these in the same manner as in the centralized 2PC protocol.

Timeouts in WAIT/READY state.

If a timeout occurs in this state, then it must be the case that not all the votes have been collected. Furthermore, it must be the case that no "Vote-abort" messages have been received. In this case, the site has to remain blocked until it receives all the votes.

Timeouts in ABORT/COMMIT state.

Does not happen.

Problem 12.3 (*).

Design a 3PC protocol using the linear communication topology.

Solution 12.3. No solution is provided.

Problem 12.4 (*). In our presentation of the centralized 3PC termination protocol, the first step involves sending the coordinator's state to all participants. The participants move to new states according to the coordinator's state. It is possible to design the termination protocol such that the coordinator, instead of sending its own state information to the participants, asks the participants to send their state information to the coordinator. Modify the termination protocol to function in this manner.

Solution 12.4. There could be multiple answers to this. Here is one approach. The basic idea of the change is to allow participants to move to a new state without waiting for the coordinator to send its state first. This would result in the state transitions as shown in Figures 12.1 and 12.2.

Fig. 12.1 Coordinator States

Fig. 12.2 Participant States

Problem 12.5 (**). In Section 12.7 we claimed that a scheduler which implements a strict concurrency control algorithm will always be ready to commit a transaction when it receives the coordinator's "prepare" message. Prove this claim.

Solution 12.5. We will prove by contradiction. First note that a strict concurrency control algorithm does not allow cascading aborts. This means that a subsequent transaction cannot perform conflicting operations until the preceding ones have committed or aborted.

Assume that there exists a site which is not ready to commit a transaction when it receives the "Prepare" message. Since there are no cascading aborts, this situation is not caused by conflicting transactions that have not completed. Thus, it must be due to the actions of this particular transaction. However, the coordinator has already sent its prepare message. This message is not sent until all the sites have finished processing. This is a contradiction.

Problem 12.6 (**). Assuming that the coordinator is implemented as part of the transaction manager and the participant as part of the scheduler, give the transaction manager, scheduler, and the local recovery manager algorithms for a non-replicated distributed DBMS under the following assumptions.

(a) The scheduler implements a distributed (strict) two-phase locking concurrency control algorithm.

(b) The commit protocol log records are written to a central database log by the LRM when it is called by the scheduler.
(c) The LRM may implement any of the protocols that have been discussed in Section 12.3.3. However, it is modified to support the distributed recovery procedures as we discussed in Section 12.7.

Solution 12.6. No solution is provided.

Problem 12.7 (*). Write the detailed algorithms for the no-fix/no-flush local recovery manager.

Solution 12.7. No solution is provided.

Problem 12.8 (**). Assume that

(a) The scheduler implements a centralized two-phase locking concurrency control,
(b) The LRM implements no-fix/no-flush protocol.

Give detailed algorithms for the transaction manager, scheduler, and local recovery managers.

Solution 12.8. This is the solution