M. Tamer Özsu and Patrick Valduriez

Principles of Distributed Database Systems (Third Edition)

Solutions to Exercises

February 25, 2011

Springer

Contents

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

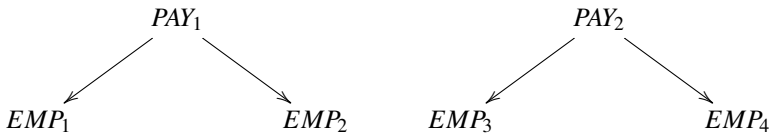
ENO	PNO	RESP	DUR
E_4	P_2	Programmer	18

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

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

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:

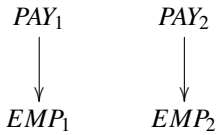


This is not simple, it is partitioned.

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

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

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 $\geq$ 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:

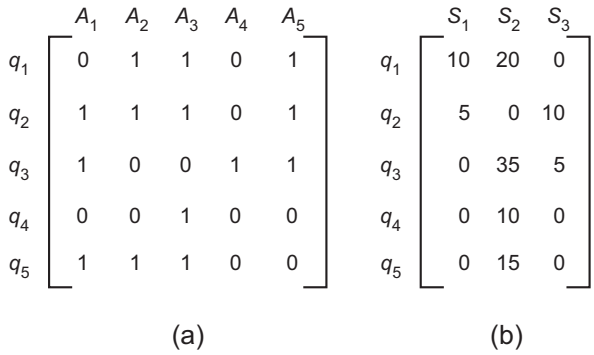


Fig. 3.21 Attribute Usage Values and Application Access Frequencies in Exercise 3.6

	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 4
Database Integration

Problem 4.1. Distributed database systems and distributed multidatabase systems represent two different approaches to systems design. Find three real-life applications for which each of these approaches would be more appropriate. Discuss the features of these applications that make them more favorable for one approach or the other.

Solution 4.1. Solution is not provided.

Problem 4.2. Some architectural models favor the definition of a global conceptual schema, whereas others do not. What do you think? Justify your selection with detailed technical arguments.

Solution 4.2. Solution is not provided.

Problem 4.3 (*). Give an algorithm to convert a relational schema to an entity-relationship one.

Solution 4.3. Solution is not provided.

Problem 4.4 (**). Consider the two databases given in Figures 4.13 and 4.14 and described below. Design a global conceptual schema as a union of the two databases by first translating them into the E-R model.

DIRECTOR(NAME, PHONE_NO, ADDRESS)
LICENSES(LIC_NO, CITY, DATE, ISSUES, COST, DEPT, CONTACT)
RACER(NAME, ADDRESS, MEM_NUM)
SPONSOR(SP_NAME, CONTACT)
RACE(R_NO, LIC_NO, DIR, MAL_WIN, FRM_WIN, SP_NAME)

Fig. 4.13 Road Race Database

Figure 4.13 describes a relational race database used by organizers of road races and Figure 4.14 describes an entity-relationship database used by a shoe manufacturer.

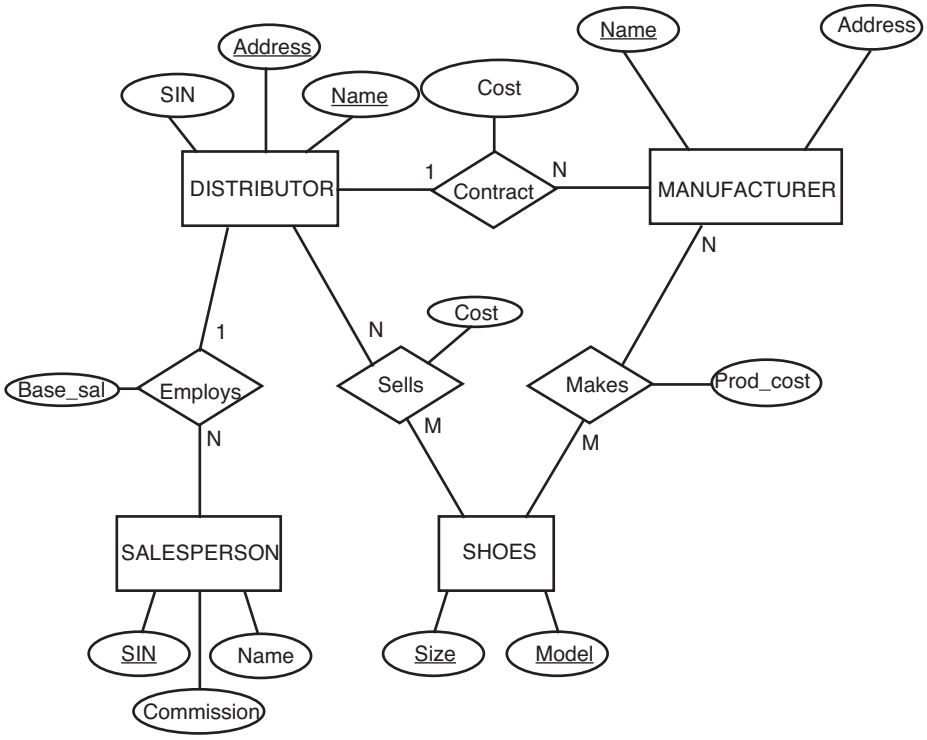


Fig. 4.14 Sponsor Database

The semantics of each of these database schemas is discussed below. Figure 4.13 describes a relational road race database with the following semantics:

DIRECTOR is a relation that defines race directors who organize races; we assume that each race director has a unique name (to be used as the key), a phone number, and an address.

LICENSES is required because all races require a governmental license, which is issued by a CONTACT in a department who is the ISSUER, possibly contained within another government department DEPT; each license has a unique LIC_NO (the key), which is issued for use in a specific CITY on a specific DATE with a certain COST.

RACER is a relation that describes people who participate in a race. Each person is identified by NAME, which is not sufficient to identify them uniquely, so a compound key formed with the ADDRESS is required. Finally, each racer may have a MEM_NUM to identify him or her as a member of the racing fraternity, but not all competitors have membership numbers.

SPONSOR indicates which sponsor is funding a given race. Typically, one sponsor funds a number of races through a specific person (CONTACT), and a number of races may have different sponsors.

RACE uniquely identifies a single race which has a license number (LIC_NO) and race number (R_NO) (to be used as a key, since a race may be planned without acquiring a license yet); each race has a winner in the male and female groups (MAL_WIN and FEM_WIN) and a race director (DIR).

Figure 4.14 illustrates an entity-relationship schema used by the sponsor's database system with the following semantics:

SHOES are produced by sponsors of a certain MODEL and SIZE, which forms the key to the entity.

MANUFACTURER is identified uniquely by NAME and resides at a certain ADDRESS.

DISTRIBUTOR is a person that has a NAME and ADDRESS (which are necessary to form the key) and a SIN number for tax purposes.

SALESPERSON is a person (entity) who has a NAME, earns a COMMISSION, and is uniquely identified by his or her SIN number (the key).

Makes is a relationship that has a certain fixed production cost (PROD_COST). It indicates that a number of different shoes are made by a manufacturer, and that different manufacturers produce the same shoe.

Sells is a relationship that indicates the wholesale COST to a distributor of shoes. It indicates that each distributor sells more than one type of shoe, and that each type of shoe is sold by more than one distributor.

Contract is a relationship whereby a distributor purchases, for a COST, exclusive rights to represent a manufacturer. Note that this does not preclude the distributor from selling different manufacturers' shoes.

Employs indicates that each distributor hires a number of salespeople to sell the shoes; each earns a BASE_SALARY.

Solution 4.4. Solution is not provided.

Problem 4.5 (*). Consider three sources:

- Database 1 has one relation Area(Id, Field) providing areas of specialization of employees; the Id field identifies an employee.
- Database 2 has two relations, Teach(Professor, Course) and In(Course, Field); Teach indicates the courses that each professor teaches and In that specifies possible fields that a course can blong to.
- Database 3 has two relations, Grant(Researcher, GrantNo) for grants given to researchers, and For(GrantNo, Field) indicating which fields the grants are for.

The objective is to build a GCS with two relations: Works(Id, Project) stating that an employee works for a particular project, and Area(Project, Field) associating projects with one or more fields.

(a) Provide a LAV mapping between Database 1 and the GCS.
(b) Provide a GLAV mapping between the GCS and the local schemas.
(c) Suppose one extra relation, Funds(GrantNo, Project), is added to Database 3. Provide a GAV mapping in this case.

Solution 4.5.

(a) S1.Area(Id, Field) $\mapsto$ Works(Id, Project), Area(Project, Field) where S1 is the schema of database 1.

(b) Works(Id, Project), Area(Project, Field) $\mapsto$ V(Id, Field)
V(Id, Field) $\mapsto$ Area(Id, Field)
V(Id, Field) $\mapsto$ Teach(Id, Course), In(Course, Field)
V(Id, Field) $\mapsto$ Grant(Id, GrantNo), For(GrantNo, Field)

(c) Works(Id, Project) $\mapsto$ Grant(Id, GrantNo), Funds(GrantNo, Project)
Area(Project, Field) $\mapsto$ Funds(GrantNo, Project), For(GrantNo, Field)

Problem 4.6. Consider a GCS with the following relation: Person(Name, Age, Gender). This relation is defined as a view over three LCSs as follows:

```
CREATE VIEW Person AS
SELECT Name, Age, "male" AS Gender
FROM   SoccerPlayer
UNION
SELECT Name, NULL AS Age, Gender
FROM   Actor
UNION
SELECT Name, Age, Gender
FROM   Politician
WHERE  Age > 30
```

For each of the following queries, discuss which of the three local schemas (SoccerPlayer, Actor, and Politician) contribute to the global query result.

(a)
```
SELECT Name FROM person
```

(b)
```
SELECT Name FROM Person
WHERE Gender = "female"
```

(c)
```
SELECT Name FROM Person WHERE Age > 25
```

(d)
```
SELECT Name FROM Person WHERE Age < 25
```

(e)
```
SELECT Name FROM Person
WHERE Gender = "male" AND Age = 40
```

Solution 4.6. It is easiest to show the solution with the following table where $[i, j] = +$ means that local schema j contributes to query i and $[i, j] = -$ means it does not.

Query	SoccerPlayer	Actor	Politician
a	+	+	+
b	−	+	+
c	+	−	+
d	+	−	−
e	+	−	+

Problem 4.7. A GCS with the relation Country(Name, Continent, Population, HasCoast) describes countries of the world. The attribute HasCoast indicates if the country has direct access to the sea. Three LCSs are connected to the global schema using the LAV approach as follows:

```
CREATE VIEW EuropeanCountry AS
SELECT Name, Continent, Population, HasCoast
FROM   Country
WHERE  Continent = "Europe"

CREATE VIEW BigCountry AS
SELECT Name, Continent, Population, HasCoast
FROM   Country
WHERE  Population >= 30000000

CREATE VIEW MidsizeOceanCountry AS
SELECT Name, Continent, Population, HasCoast
FROM   Country
WHERE  HasCoast = true AND Population > 10000000
```

(a) For each of the following queries, discuss the results with respect to their completeness, i.e., verify if the (combination of the) local sources cover all relevant results.

1. `SELECT Name FROM Country`
2. `SELECT Name FROM Country WHERE Population > 40`
3. `SELECT Name FROM Country WHERE Population > 20`

(b) For each of the following queries, discuss which of the three LCSs are necessary for the global query result.

1. `SELECT Name FROM Country`
2. `SELECT Name FROM Country WHERE Population > 30 AND Continent = "Europe"`
3. `SELECT Name FROM Country WHERE Population < 30`
4. `SELECT Name FROM Country WHERE Population > 30 AND HasCoast = true`

Solution 4.7.

(a)
1. This is not complete, since small non-European countries without access to the ocean are not covered.
2. This is complete – it is covered by BigCountry
3. This is not complete, since countries without access to the ocean and whose populations are ≥ 20 but < 30 are not covered.

(b)
1. All three are necessary.
2. EuropeanCountry or BigCountry are necessary.

3. EuropeanCountry and MidsizeOceanCountry are required.
4. BigCountry or MidSizeOceanCountry are required.

Problem 4.8. Consider the following two relations PRODUCT and ARTICLE that are specified in a simplified SQL notation. The perfect schema matching correspondences are denoted by arrows.

PRODUCT	$\longrightarrow$	ARTICLE
Id: int PRIMARY KEY	$\longrightarrow$	Key: varchar(255) PRIMARY KEY
Name: varchar(255)	$\longrightarrow$	Title: varchar(255)
DeliveryPrice: float	$\longrightarrow$	Price: real
Description: varchar(8000)	$\longrightarrow$	Information: varchar(5000)

(a) For each of the five correspondences, indicate which of the following match approaches will probably identify the correspondence:

1. Syntactic comparison of element names, e.g., using edit distance string similarity
2. Comparison of element names using a synonym lookup table
3. Comparison of data types
4. Analysis of instance data values

(b) Is it possible for the listed matching approaches to determine false correspondences for these match tasks? If so, give an example.

Solution 4.8. The solution is provided in the following table where "+" means the particular matcher would identify the correspondence properly, "−" means it would not, "?" means it *may* identify but with reduced probability of success depending on specific (unspecified) circumstances (for example, the success of using a synonym table obviously depends on whether this table holds the asked synonyms or variations thereof), and "NA" means that matcher is not applicable in this circumstance.

Matcher	**Product-Article**	**Id-Key**	**Name-Title**	**DeliveryPrice-Price**	**Description-Information**
1	−	−	−	+	−
2	+	+	+/?	+	+/?
3	NA	−	+	+/?	+/?
4	NA	−	?	+	?

Problem 4.9. Consider two relations $S(a,b,c)$ and $T(d,e,f)$. A match approach determines the following similarities between the elements of S and T:

	$T.d$	$T.e$	$T.f$
$S.a$	0.8	0.3	0.1
$S.b$	0.5	0.2	0.9
$S.c$	0.4	0.7	0.8

Based on the given matcher's result, derive an overall schema match result with the following characteristics:

- Each element participates in exactly one correspondence.
- There is no correspondence where both elements match an element of the opposite schema with a higher similarity than its corresponding counterpart.

Solution 4.9. (a,d), (b,f), (c,e)

Problem 4.10 (*). Figure 4.16 illustrates the schema of three different data sources:

- MyGroup contains publications authored by members of a working group;
- MyConference contains publications of a conference series and associated workshops;
- MyPublisher contains articles that are published in journals.

The arrows show the foreign key-to-primary key relationships.

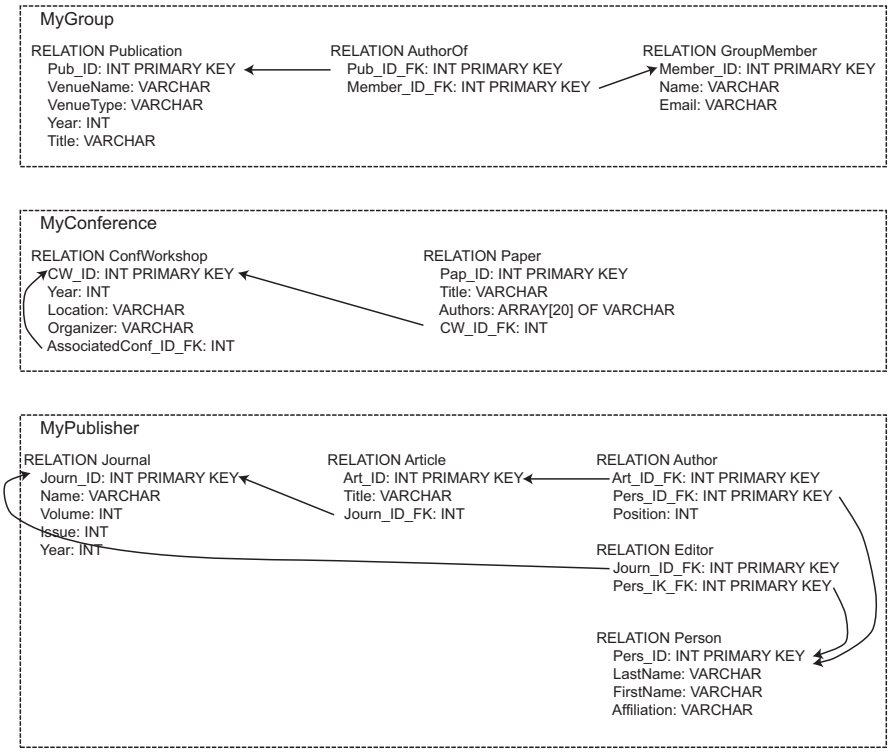


Fig. 4.15 Figure for Problem 4.10

The sources are defined as follows:
MyGroup

- Publication
  - Pub_ID: unique publication ID
  - VenueName: name of the journal, conference or workshop
  - VenueType: “journal”, “conference”, or “workshop”
  - Year: year of publication
  - Title: publication’s title
- AuthorOf
  - many-to-many relationship representing “group member is author of publication”
- GroupMember
  - Member_ID: unique member ID
  - Name: name of the group member
  - Email: email address of the group member

MyConference

- ConfWorkshop
  - CW_ID: unique ID for the conference/workshop
  - Name: name of the conference or workshop
  - Year: year when the event takes place
  - Location: event’s location
  - Organizer: name of the organizing person
  - AssociatedConf_ID_FK: value is NULL if it is a conference, ID of the associated conference if the event is a workshop (this is assuming that workshops are organized in conjunction with a conference)
- Paper
  - Pap_ID: unique paper ID
  - Title: paper’s title
  - Author: array of author names
  - CW_ID_FK: conference/workshop where the paper is published

MyPublisher

- Journal
  - Journ_ID: unique journal ID
  - Name: journal’s name
  - Year: year when the event takes place

  - Volume: journal volume
  - Issue: journal issue
- Article
  - Art_ID: unique article ID
  - Title: title of the article
  - Journ_ID_FK: journal where the article is published
- Person
  - Pers_ID: unique person ID
  - LastName: last name of the person
  - FirstName: first name of the person
  - Affiliation: person's affiliation (e.g., the name of a university)
- Author
  - represents the many-to-many relationship for "person is author of article"
  - Position: author's position in the author list (e.g., first author has Position 1)
- Editor
  - represents the many-to-many relationship for "person is editor of journal issue"

(a) Identify all schema matching correspondences between the schema elements of the sources. Use the names and data types of the schema elements as well as the given description.

(b) Classify your correspondences along the following dimensions:

1. Type of schema elements (e.g., attribute-attribute or attribute-relation)
2. Cardinality (e.g., 1:1 or 1:N)

(c) Give a consolidated global schema that covers all information of the source schemas.

Solution 4.10.

(a) This one is easy – just follow the figure and the semantics.

(b)

1. "attribute-attribute": MyGroup.Publication.Title - MyPublisher.Article.Title
   "attribute-relation": MyConference.Paper.Authors - MyPublisher.Author
   and MyGroup.Publication.VenueType - MyPublisher.Journal

2. "one-to-one": MyGroup.Publication.Year - MyConferenceConfWorkshop.Year
"one-to-many": MyGroup.GroupMember.Name - {MyPublisher.Person.LastName, MyPublisher.Person.FirstName}

(c) One possible solutions is the following where "FK" indicates foreign key.

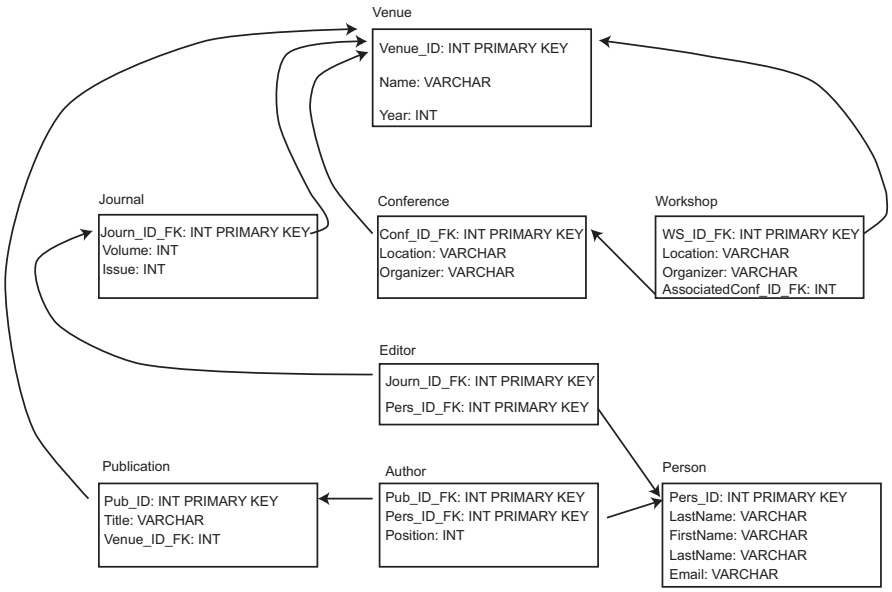


Fig. 4.16 Solution of Problem 4.10c

Problem 4.11 (*). Figure 4.17 illustrates (using a simplified SQL syntax) two sources S_1 and S_2. S_1 has two relations, Course and Tutor, and S_2 has only one relation, Lecture. The solid arrows denote schema matching correspondences. The dashed arrow represents a foreign key relationship between the two relations in S_1.

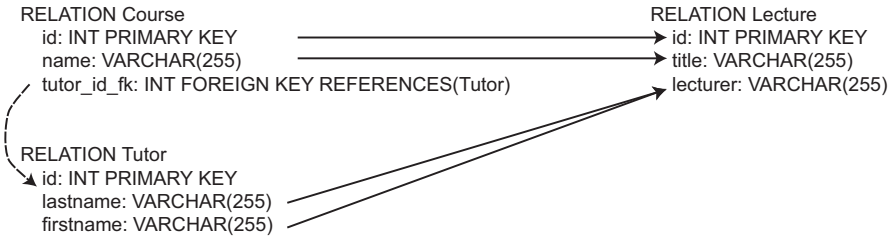


Fig. 4.17 Figure for Exercise 11

The following are three schema mappings (represented as SQL queries) to transform S_1's data into S_2.

1.
```
SELECT C.id, C.name as Title, CONCAT(T.lastname,
       T.firstname) AS Lecturer)
FROM   Course AS C
JOIN   Tutor AS T ON (C.tutor_id_fk = T.id)
```

2.
```
SELECT C.id, C.name AS Title, NULL AS Lecturer)
FROM   Course AS C
UNION
SELECT T.id AS ID, NULL AS Title, T,
       lastname AS Lecturer)
FROM   Course AS C
FULL OUTER JOIN Tutor AS T ON(C.tutor_id_fk=T.id)
```

3.
```
SELECT C.id, C.name as Title, CONCAT(T.lastname,
       T.firstname) AS Lecturer)
FROM   Course AS C
FULL OUTER JOIN Tutor AS T ON(C.tutor_id_fk=T.id)
```

Discuss each of these schema mappings with respect to the following questions:

(a) Is the mapping meaningful?
(b) Is the mapping complete (i.e., are all data instances of S_1 transformed)?
(c) Does the mapping potentially violate key constraints?

Solution 4.11.

(a) The mapping is meaningful. It is not complete, since tutors without courses are not considered. It does not violate primary key constraints.
(b) The mapping is not meaningful since the tutor and course are not linked. It is not complete, since tutor.firstname is not considered. It does violate primary key constraints since you need tutor.id = course.id.
(c) The mapping is meaningful. It is complete. It does violate primary key constraints since course.id is NULL.

Problem 4.12 (*). Consider three data sources:

- Database 1 has one relation AREA(ID, FIELD) providing areas of specialization of employees where ID identifies an employee.
- Database 2 has two relations: TEACH(PROFESSOR, COURSE) and IN(COURSE, FIELD) specifying possible fields a course can belong to.
- Database 3 has two relations: GRANT(RESEARCHER, GRANT#) for grants given to researchers, and FOR(GRANT#, FIELD) indicating the fields that the grants are in.

Design a global schema with two relations: WORKS(ID, PROJECT) that records which projects employees work in, and AREA(PROJECT, FIELD) that associates projects with one or more fields for the following cases:

(a) There should be a LAV mapping between Database 1 and the global schema.
(b) There should be a GLAV mapping between the global schema and the local schemas.
(c) There should be a GAV mapping when one extra relation FUNDS(GRANT#, PROJECT) is added to Database 3.

Solution 4.12. Solution is note provided.

Problem 4.13 (**). Logic (first-order logic, to be precise) has been suggested as a uniform formalism for schema translation and integration. Discuss how logic can be useful for this purpose.

Solution 4.13. This is the solution

Chapter 5
Semantic Data Control

Problem 5.1. Define in SQL-like syntax a view of the engineering database V(ENO, ENAME, PNO, RESP), where the duration is 24. Is view V updatable? Assume that relations EMP and ASG are horizontally fragmented based on access frequencies as follows:

Site 1	Site 2	Site 3
EMP_1	EMP_2	
	ASG_1	ASG_2

where

$EMP_1 = \sigma_{TITLE \neq \text{“Engineer”}}(EMP)$
$EMP_2 = \sigma_{TITLE = \text{“Engineer”}}(EMP)$
$ASG_1 = \sigma_{0<DUR<36}(ASG)$
$ASG_2 = \sigma_{DUR \geq 36}(ASG)$

At which site(s) should the definition of V be stored without being fully replicated, to increase locality of reference?

Solution 5.1. This is the solution

Problem 5.2. Express the following query: names of employees in view V who work on the CAD project.

Solution 5.2. This is the solution

Problem 5.3 (*). Assume that relation PROJ is horizontally fragmented as

$PROJ_1 = \sigma_{PNAME = \text{“CAD”}}(PROJ)$
$PROJ_2 = \sigma_{PNAME \neq \text{“CAD”}}(PROJ)$

Modify the query obtained in Exercise 5.2 to a query expressed on the fragments.

Solution 5.3. This is the solution

Problem 5.4 (**). Propose a distributed algorithm to efficiently refresh a snapshot at one site derived by projection from a relation horizontally fragmented at two other sites. Give an example query on the view and base relations which produces an inconsistent result.

Solution 5.4. This is the solution

Problem 5.5 (*). Consider the view EG of Example 5.5 which uses relations EMP and ASG as base data and assume its state is derived from that of Example 3.1, so that EG has 9 tuples (see Figure 5.4). Assume that tuple ⟨E3, P3, Consultant, 10⟩ from ASG is updated to ⟨E3, P3, Engineer, 10⟩. Apply the basic counting algorithm for refreshing the view EG. What projected attributes should be added to view EG to make it self-maintainable?

Solution 5.5. This is the solution

Problem 5.6. Propose a relation schema for storing the access rights associated with user groups in a distributed database catalog, and give a fragmentation scheme for that relation, assuming that all members of a group are at the same site.

Solution 5.6. This is the solution

Problem 5.7 (**). Give an algorithm for executing the REVOKE statement in a distributed DBMS, assuming that the GRANT privilege can be granted only to a group of users where all its members are at the same site.

Solution 5.7. This is the solution

Problem 5.8 (**). Consider the multilevel relation PROJ** in Figure 5.8. Assuming that there are only two classification levels for attributes (S and C), propose an allocation of PROJ** on two sites using fragmentation and replication that avoids covert channels on read queries. Discuss the constraints on updates for this allocation to work.

Solution 5.8. This is the solution

Problem 5.9. Using the integrity constraint specification language of this chapter, express an integrity constraint which states that the duration spent in a project cannot exceed 48 months.

Solution 5.9. This is the solution

Problem 5.10 (*). Define the pretests associated with integrity constraints covered in Examples 5.11 to 5.14.

Solution 5.10. This is the solution

Problem 5.11. Assume the following vertical fragmentation of relations EMP, ASG and PROJ:

Site 1	Site 2	Site 3	Site 4
EMP_1	EMP_2		
	$PROJ_1$	$PROJ_2$	
		ASG_1	ASG_2

where

$EMP_1 = \Pi_{ENO, ENAME}(EMP)$
$EMP_2 = \Pi_{ENO, TITLE}(EMP)$
$PROJ_1 = \Pi_{PNO, PNAME}(PROJ)$
$PROJ_2 = \Pi_{PNO, BUDGET}(PROJ)$
$ASG_1 = \Pi_{ENO, PNO, RESP}(ASG)$
$ASG_2 = \Pi_{ENO, PNO, DUR}(ASG)$

Where should the pretests obtained in Exercise 5.9 be stored?

Solution 5.11. This is the solution

Problem 5.12 (**). Consider the following set-oriented constraint:

```
CHECK ON e:EMP, a:ASG
    (e.ENO = a.ENO and (e.TITLE = "Programmer")
    IF a.RESP = "Programmer")
```

What does it mean? Assuming that EMP and ASG are allocated as in the previous exercice, define the corresponding pretests and theri storage. Apply algorithm ENFORCE for an update of type INSERT in ASG.

Solution 5.12. This is the solution

Problem 5.13 (**). Assume a distributed multidatabase system with no global transaction support. Assume also that there are two sites, each with a (different) EMP relation and a integrity manager that communicates with the component DBMS. Suppose that we want to have a global unique key constraint on EMP. Propose a simple strategy using differential relations to check this constraint. Discuss the possible actions when a constraint is violated.

Solution 5.13. This is the solution

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

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

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.

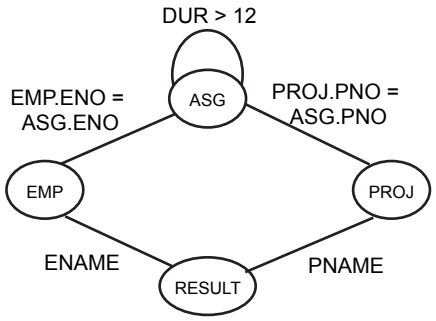


Fig. 7.11 Query Graph for Problem 7.2

The corresponding operator tree is given in Figure 7.12.

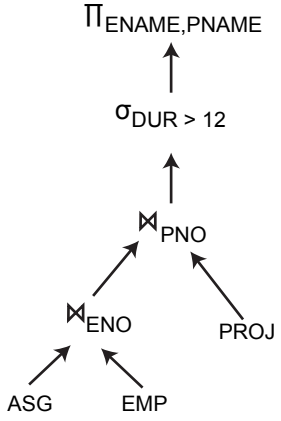


Fig. 7.12 Operator Tree for Problem 7.2

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.

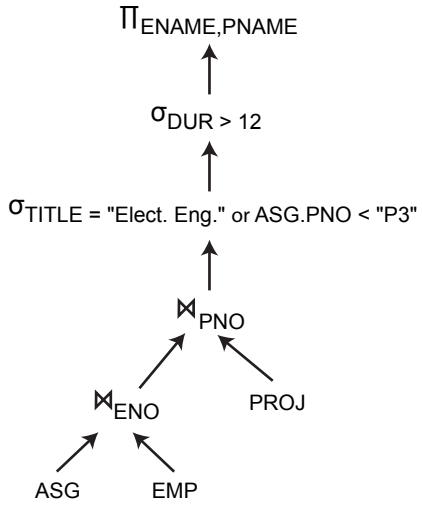


Fig. 7.13 Generic Operator Tree for Problem 7.3

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.

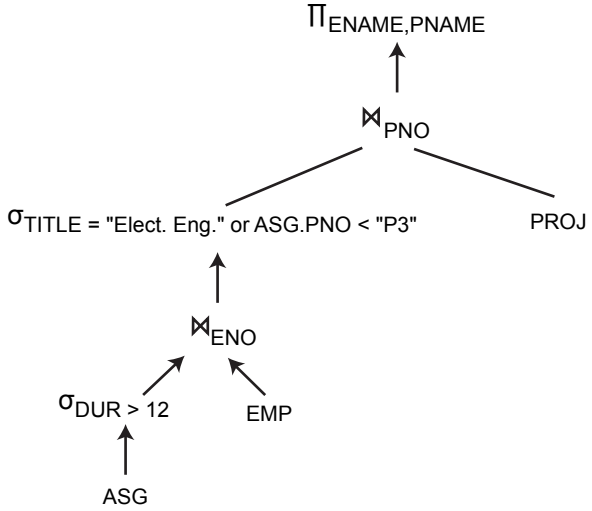


Fig. 7.14 Final Operator Tree for Problem 7.3

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.

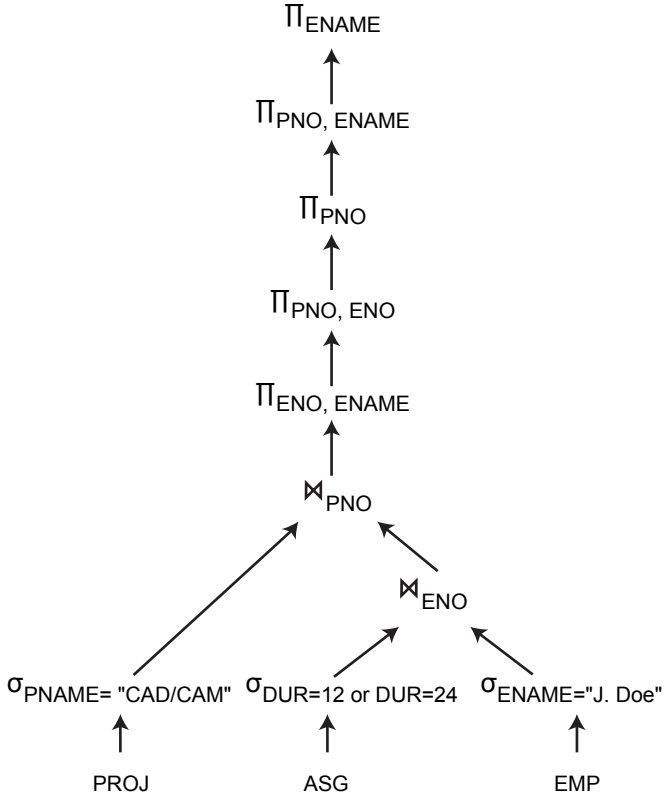


Fig. 7.15 Problem 7.4 - Projections after Joins

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:

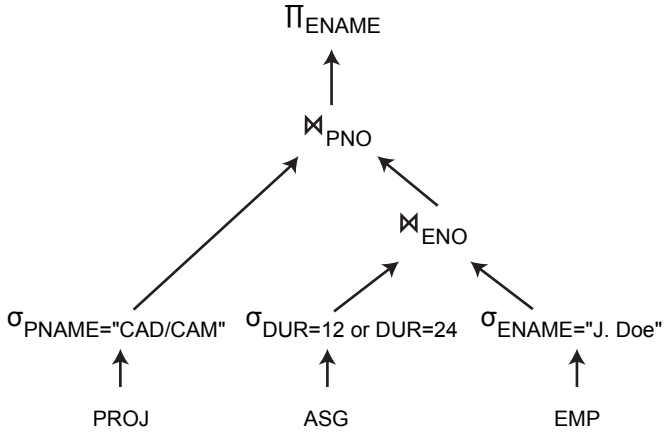


Fig. 7.16 Problem 7.4 - Eliminate Projections

```
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 $\text{DUR} > 24$, 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:

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

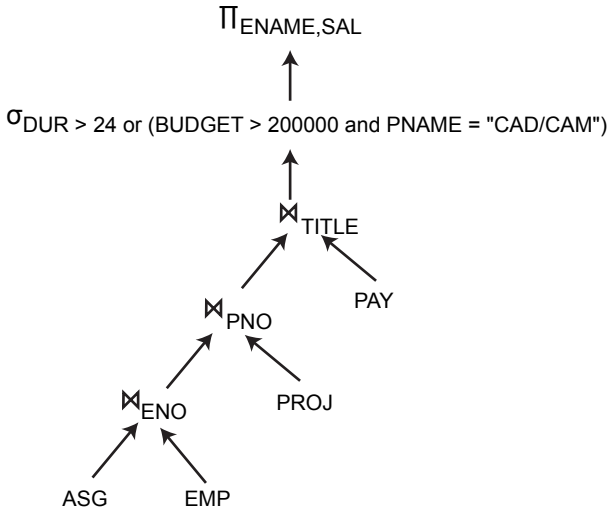


Fig. 7.17 Generic Operator Tree for Problem 7.5

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.

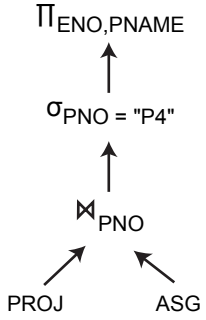


Fig. 7.18 Generic Operator Tree for Problem 7.6

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.

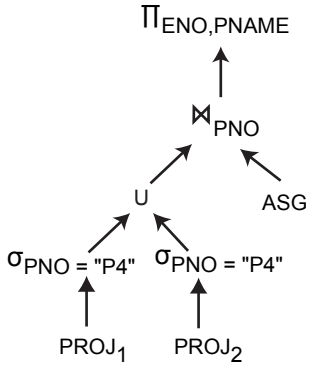


Fig. 7.19 Localized Operator Tree for Problem 7.6

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

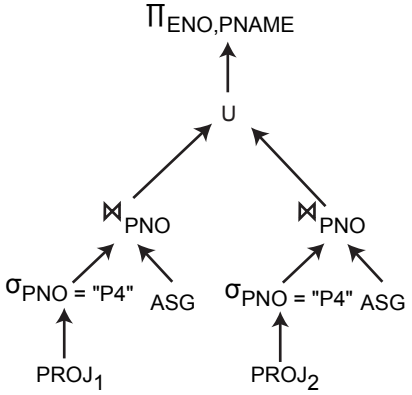


Fig. 7.20 New Localized Operator Tree for Problem 7.6

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

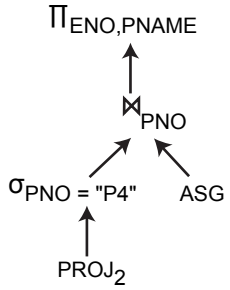


Fig. 7.21 Final Localized Operator Tree for Problem 7.6

$$\text{ASG}_1 = \sigma_{\text{PNO} \leq \text{"P2"}}(\text{ASG})$$
$$\text{ASG}_2 = \sigma_{\text{"P2"} < \text{PNO} \leq \text{"P3"}}(\text{ASG})$$
$$\text{ASG}_3 = \sigma_{\text{PNO} > \text{"P3"}}(\text{ASG})$$

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.

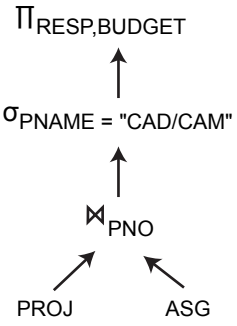


Fig. 7.22 Generic Operator Tree for Problem 7.7

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. $\text{PROJ}_1 \bowtie \text{ASG}_1$

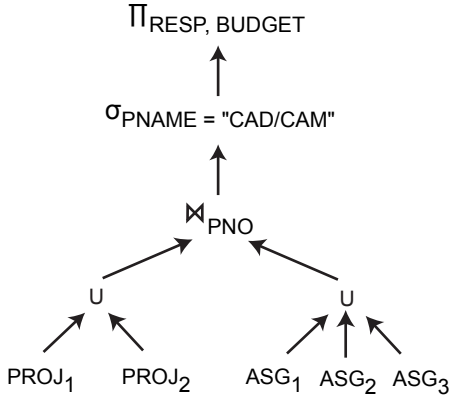


Fig. 7.23 Localized Operator Tree for Problem 7.7

2. $\text{PROJ}_1 \bowtie \text{ASG}_2$
3. $\text{PROJ}_1 \bowtie \text{ASG}_3$
4. $\text{PROJ}_2 \bowtie \text{ASG}_1$
5. $\text{PROJ}_2 \bowtie \text{ASG}_2$
6. $\text{PROJ}_2 \bowtie \text{ASG}_3$

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.

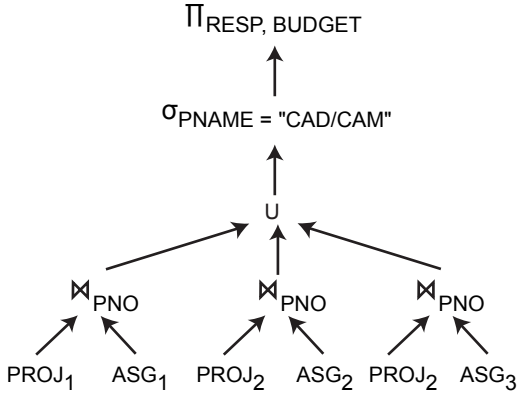


Fig. 7.24 Final Operator Tree for Problem 7.7

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.

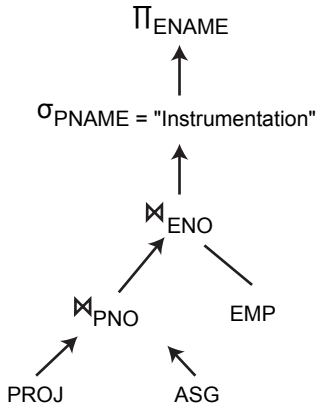


Fig. 7.25 Generic Operator Tree for Problem 7.8

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:

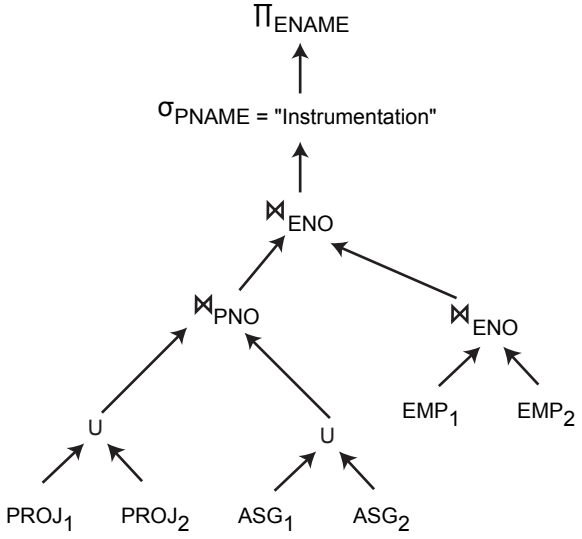


Fig. 7.26 Localized Operator Tree for Problem 7.8

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

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.

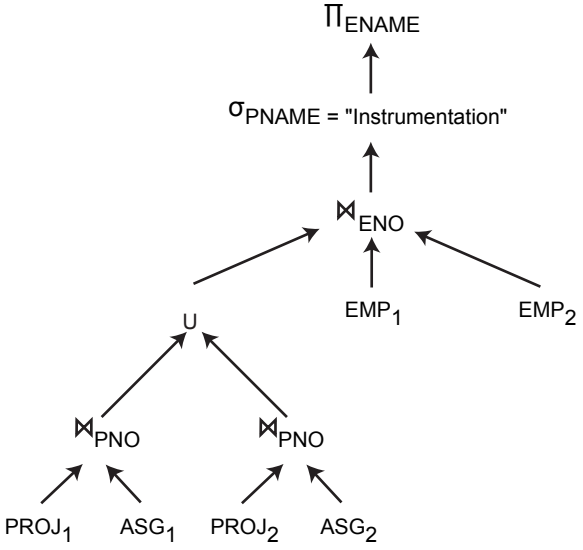


Fig. 7.27 Reduced Operator Tree for Problem 7.8

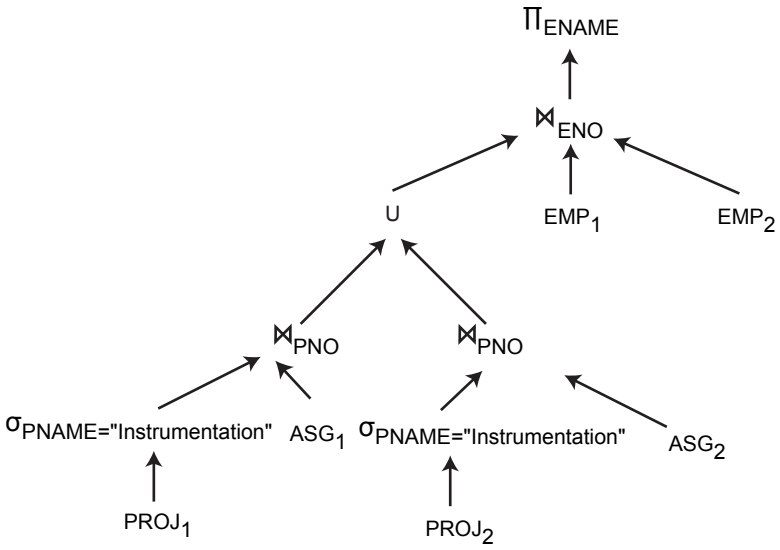


Fig. 7.28 Problem 7.8 - Selection Pushed Down

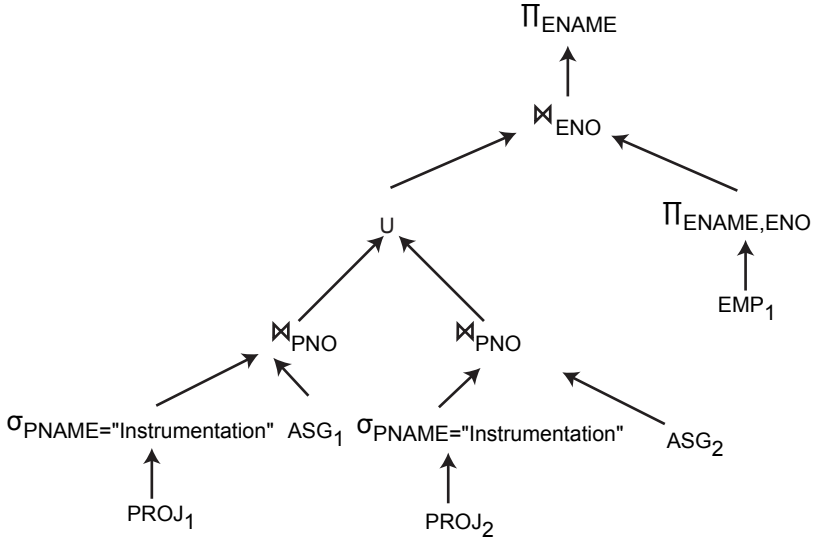


Fig. 7.29 Problem 7.8 - Projection Pushed Down

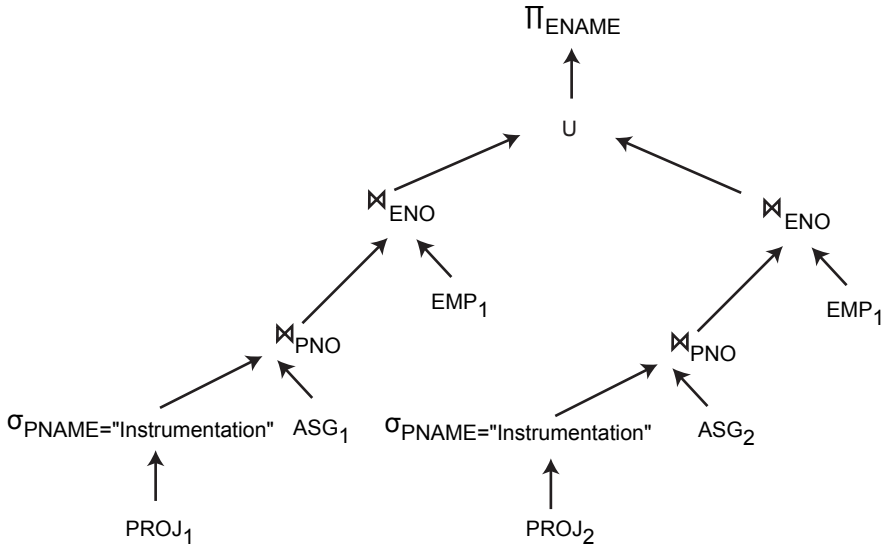


Fig. 7.30 Final Operator Tree for Problem 7.8

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.

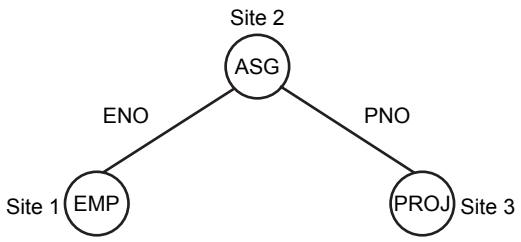


Fig. 8.12 Relation Statistics

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. $\Pi_{\text{ENO}}(\text{ASG}) \rightarrow$ 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 $T_{TR} = 1$.

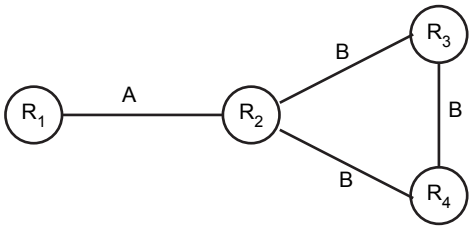


Fig. 8.19 Join Graph

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. $R_1 = \sigma_{\text{BUDGET} \text{¿} 2000 \vee \text{DUR} \text{¿} 24}(\text{ASG} \bowtie \text{PROJ})$
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 9
Multidatabase Queries

Problem 9.1 (**). Can any type of global optimization be performed on global queries in a multidatabase system? Discuss and formally specify the conditions under which such optimization would be possible.

Solution 9.1. This is the solution

Problem 9.2 (*). Consider a marketing application with a ROLAP server at site s_1 which needs to integrate information from two customer databases, each at site s_2 within the corporate network. Assume also that the application needs to combine customer information with information extracted from Web data sources about cities in 10 different countries. For security reasons, a web server at site s_3 is dedicated to Web access outside the corporate network. Propose a multidatabase system architecture with mediator and wrappers to support this application. Discuss and justify design choices.

Solution 9.2. This is the solution

Problem 9.3 (**). Consider the global relations EMP(ENAME, TITLE, CITY) and ASG(ENAME, PNAME, CITY, DUR). City in ASG is the location of the project of name PNAME (i.e., PNAME functionnally determines CITY). Consider the local relations EMP1(ENAME, TITLE, CITY), EMP2(ENAME, TITLE, CITY), PROJ1(PNAME, CITY), PROJ2(PNAME, CITY) and ASG1(ENAME, PNAME, DUR). Consider query Q which selects the names of the employees assigned to a project in Rio de Janeiro for more than 6 months and the duration of their assignment.

(a) Assuming the GAV approach, perform query rewriting.
(b) Assuming the LAV approach, perform query rewriting using the bucket algorithm.
(c) Same as (b) using the MinCon algorithm.

Solution 9.3. This is the solution

Problem 9.4 (*). Consider relations EMP and ASG of Example 9.7. We denote by $|R|$ the number of pages to store R on disk. Consider the following statistics about the data:

$|\text{EMP}| = 1\,000$
$|\text{EMP}| = 100$
$|\text{ASG}| = 10\,000$
$|\text{ASG}| = 2\,000$
$selectivity(\text{ASG.DUR} > 36) = 1\%$

The mediator generic cost model is:

$cost(\sigma_{A=v}(R)) = |R|$
$cost(\sigma(X)) = cost(X)$ where X contains at least one operator.
$cost(R \bowtie_A^{ind} S) = cost(R) + |R| * cost(\sigma_{A=v}(S))$ using an indexed join algorithm.
$cost(R \bowtie_A^{nl} S) = cost(R) + |R| * cost(S)$ using a nested loop join algorithm.

Consider the MDBMS input query Q:

```
SELECT *
FROM   EMP, ASG
WHERE  EMP.ENO=ASG.ENO
AND    ASG.DUR>36
```

Consider four plans to process Q:

$P_1 = EMP \bowtie_{ENO}^{ind} \sigma_{DUR>36}(ASG)$
$P_2 = EMP \bowtie_{ENO}^{nl} \sigma_{DUR>36}(ASG)$
$P_3 = \sigma_{DUR>36}(ASG) \bowtie_{ENO}^{ind} EMP$
$P_4 = \sigma_{DUR>36}(ASG) \bowtie_{ENO}^{nl} EMP$

(a) What is the cost of plans P_1 to P_4?
(b) Which plan has the minimal cost?

Solution 9.4. It is easy to compute the following:

$$\begin{aligned} cost(P_1) &= |EMP| + ||EMP|| * |ASG| \\ &= 2\,000\,100 \\ cost(P_2) &= |EMP| + |EMP| * |ASG| \\ &= 200\,100 \\ cost(P_3) &= |ASG| + 1\% * ||ASG|| * |EMP| \\ &= 12\,000 \\ cost(P_4) &= |ASG| + |\sigma_{DUR>36}(ASG)| * |EMP| \\ &= 4\,000 \end{aligned}$$

Thus P_4 has the minimal cost.

Problem 9.5 (*). Consider relations EMP and ASG of the previous exercice. Suppose now that the mediator cost model is completed with the following cost information issued from the component DBMSs.

The cost of accessing EMP tuples at db_1 is:

$$cost(\sigma_{A=v}(R)) = |\sigma_{A=v}(R)|$$

The specific cost of selecting ASG tuples that have a given ENO at D_2 is:

$$cost(\sigma_{\text{ENO}=v}(\text{ASG})) = |\sigma_{\text{ENO}=v}(\text{ASG})|$$

(a) What is the cost of plans P_1 to P_4?
(b) Which plan has the minimal cost?

Solution 9.5. It is easy to compute the following:

$$\begin{aligned} cost(P_1) &= |EMP| + ||EMP|| * ||\sigma_{ENO=v}(ASG)|| \\ &= 10\,100 \\ cost(P_2) &= |EMP| + |EMP| * |ASG| \\ &= 200\,100 \\ cost(P_3) &= |ASG| + 1\% * ||ASG|| * ||\sigma_{ENO=v}(EMP)|| \\ &= 2\,100 \\ cost(P_4) &= |ASG| + |\sigma_{DUR>36}(ASG)| * |EMP| \\ &= 4\,000 \end{aligned}$$

Thus P_3 has the minimal cost.

Problem 9.6 (**). What are the respective advantages and limitations of the query-based and operator-based approaches to heterogeneous query optimization from the points of view of query expressiveness, query performance, development cost of wrappers, system (mediator and wrappers) maintenance and evolution?

Solution 9.6. This is the solution

Problem 9.7 (**). Consider Example 9.8 by adding, at a new site, component database db_4 which stores relations EMP(ENO, ENAME, CITY) and ASG(ENO, PNAME, DUR). db_4 exports through its wrapper w_3 join and scan capabilities. Let us assume that there can be employees in db_1 with corresponding assignments in db_4 and employees in db_4 with corresponding assignments in db_2.

(a) Define the planning functions of wrapper w_3.
(b) Give the new definition of global view EMPASG(ENAME, CITY, PNAME, DUR).
(c) Give a QEP for the same query as in Example 9.8.

Solution 9.7. This is the solution

Problem 9.8 (**). Consider three relations $R(A,B)$, $S(B,C)$ and $T(C,D)$ and query Q $(\sigma_p^1(R) \bowtie_1 S \bowtie_2 T)$, where $\bowtie_1$ and $\bowtie_2$ are natural joins. Assume that S has an index on attribute B and T has an index on attribute C. Furthermore, σ_p^1 is an expensive

predicate (i.e., a predicate over the results of running a program over values of $R.A$). Using the Eddy approach for adaptive query processing, answer the following questions:

(a) Propose the set C of constraints on Q to produce an Eddy-based QEP.
(b) Give a query graph G for Q.
(c) Using C and G, propose an Eddy-based QEP.
(d) Propose a second QEP that uses State Modules. Discuss the advantages obtained by using state modules in this QEP.

Solution 9.8. This is the solution

Problem 9.9 (**). Propose a data structure to store tuples in the Eddy buffer pool to help choosing quickly the next tuple to be evaluated according to user specified preference, for instance, produce first results earlier.

Solution 9.9. This is the solution

Problem 9.10 (**). Propose a predicate scheduling algorithm based on the Cherry picking approach introduced in Section 9.4.3.3.

Solution 9.10. This is the solution

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:

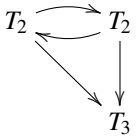
Conflicting ops	H_1	H_2	H_3	H_4
$W_2(x), W_1(x)$	$\prec$	$\succ$	$\prec$	$\prec$
$W_2(x), R_3(x)$	$\prec$	$\succ$	$\prec$	$\prec$
$W_2(x), R_1(x)$	$\prec$	$\prec$	$\prec$	$\prec$
$W_1(x), R_3(x)$	$\prec$	$\prec$	$\succ$	$\prec$
$W_2(y), R_3(y)$	$\prec$	$\succ$	$\prec$	$\prec$
$R_2(x), W_1(x)$	$\succ$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

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.

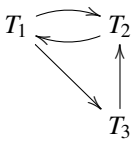
Problem 11.2. Which of the above histories $H_1 - H_4$ are serializable?

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.

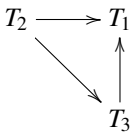
- H_1 is not serializable since it has the following serialization graph, which contains a cycle:



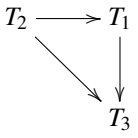
- H_2 is **not** serializable since it has the following serialization graph, which contains a cycle:



- 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

Chapter 13
Data Replication

Problem 13.1. For each of the four replication protocols (eager centralized, eager distributed, lazy centralized, lazy distributed), give a scenario/application where the approach is more suitable than the other approaches. Explain why.

Solution 13.1. This is the solution

Problem 13.2. A company has several geographically distributed warehouses storing and selling products. Consider the following partial database schema:

ITEM(ID, ItemName, Price, ...)

STOCK(ID, Warehouse, Quantity, ...)

CUSTOMER(ID, CustName, Address, CreditAmt, ...)

CLIENT-ORDER(ID, Warehouse, Balance, ...)

ORDER(ID, Warehouse, CustID, Date)

ORDER-LINE(ID, ItemID, Amount, ...)

The database contains relations with product information (`ITEM` contains the general product information, `STOCK` contains, for each product and for each warehouse, the number of pieces currently on stock). Furthermore, the database stores information about the clients/customers, e.g., general information about the clients is stored in the `CUSTOMER` table. The main activities regarding the clients are the rdering of products, the payment of bills and general information requests. There exist several tables to register the orders of a customer. Each order is regustered in the `ORDER` and `ORDER-LINE` tables. For each order/purchase, one entry exists in the order table, having an ID, indicating the customer-id, the warehouse at which the order was submitted, the date of the order, etc. A client can have several orders pending at a warehouse. Within each order, several products can be ordered. `ORDER-LINE` contains an entry for each product of the order, which may include one or more products. `CLIENT-ORDER` is a summary table that lists, for each client and for each warehouse, the sum of all existing orders.

(a) The company has a customer service group consisting of several employees that receive customers' orders and payments, query the data of local customers

to write bills or register paychecks, etc. Furthermore, they answer any type of requests which the customers might have. For instance, ordering products changes (update/insert) the `CLIENT-ORDER`, `ORDER`, `ORDER-LINE`, and `STOCK` tables. To be flexible, each employee must be able to work with any of the clients. The workload is estimated to be 80% queries and 20% updates. Since the workload is query oriented, the management has decided to build a cluster of PCs each equipped with its own database to accelerate queries through fast local access. How would you replicate the data for this purpose? Which replica control protocol(s) wold you use to keep the data consistent?

(b) The company's management has to decide each fiscal quarter on their product offerings and sales strategies. For this purpose, they must continually observe and analyze the sales of the different products at the different warehouses as well as observe consumer behavior. How would you replicate the data for this purpose? Which replica control protocol(s) would you use to keep the data consistent?

Solution 13.2. This is the solution

Problem 13.3 (*). An alternative to ensuring that the refresh transactions can be applied at all of the slaves in the same order in lazy single master protocols with limited transparency is the use of a replication graph as discussed in Section 13.3.3. Develop a method for distributed management of the replication graph.

Solution 13.3. This is the solution

Problem 13.4. Consider data items x and y replicated across the sites as follows:

Site 1	Site 2	Site 3	Site 4
x	x		x
	y	y	y

(a) Assign votes to each site and give the read and write quorum.

(b) Determine the possible ways that the network can partition and for each specify in which group of sites a transaction that updates (reads and writes) x can be terminated and what the termination condition would be.

(c) Repeat (**b**) for y.

Solution 13.4. (a) There are many possible vote assignments. We will discuss a very simple case where we assign one vote to each of the sites. Then for both x and y $V = 3$, and we can set, for example,

$V_r = V_w = 2$

$V_r + V_w = 4 > 3$

(b) Consider the following table for both x and y

Partitions	(b) x	(c) y
$\{1, 2, 3, 4\}$	abort or commit	abort or commit
$\{1\}, \{2, 3, 4\}$	neither; abort or commit	neither; abort or commit
$\{2\}, \{1, 3, 4\}$	neither; abort or commit	neither; abort or commit
$\{3\}, \{1, 2, 4\}$	neither; abort or commit	neither; abort or commit
$\{4\}, \{1, 2, 3\}$	neither; abort or commit	neither; abort or commit
$\{1, 2\}, \{3, 4\}$	abort or commit; neither	neither; abort or commit
$\{1, 3\}, \{2, 4\}$	neither; abort or commit	neither; abort or commit
$\{1, 4\}, \{2, 3\}$	abort or commit; neither	neither; abort or commit
$\{1\}, \{2\}, \{3, 4\}$	neither; neither; neither	neither; neither; abort or commit
$\{1\}, \{3\}, \{2, 4\}$	neither; neither; abort or commit	neither; neither; abort or commit
$\{1\}, \{4\}, \{2, 3\}$	neither; neither; neither	neither; neither; abort or commit
$\{3\}, \{4\}, \{1, 2\}$	neither; neither; abort or commit	neither; neither; neither
$\{2\}, \{4\}, \{1, 3\}$	neither; neither; neither	neither; neither; neither
$\{2\}, \{3\}, \{1, 4\}$	neither; neither; abort or commit	neither; neither; neither
$\{1\}, \{2\}, \{3\}, \{4\}$	neither; neither; neither; neither	neither; neither; neither; neither

(c) The table above specifies what happens for y as well.

Note: It is important to note that when votes are being assigned to sites, they are actually being assigned to replicas. Assigning it directly to sites would give, in this case, $V = 4$.. Although this works in this particular case it does not work in general. For example, consider the case where there are 100 sites and 10 replicas. Setting $V = 100$, and $V_r = 50$ $V_w = 51$ is wrong because there is no way to obtain 50 or 51 votes from 10 replicas (assuming no additional weight to some replicas).

Also, it is dangerous (in this example) to set $V_r = V_w = 3$. Again, this would work in this case, but it would severely restrict the cases in which one can terminate transactions. To demonstrate this, prepare a table like the above for the case where $V_r = V_w = 3$, and compare it with what is given above. It will be clear that there are far fewer cases when it is possible to terminate transactions. The objective in setting V_r and V_w is to maximize the number of cases where you can terminate transactions.

Problem 13.5 (**). In the NODO protocol, we have seen that each conflict class group has a master. However, this is not inherent to the protocol. Design a multi-master variation of NODO in which a transaction might be executed by any replica. What condition should be enforced to guarantee that each updated transaction is processed only by one replica?

Solution 13.5. This is the solution

Problem 13.6 (**). In the NODO protocol, if the DBMS could provide additional introspection functionality, it would be possible to execute in certain circumstances transactions of the same conflict class in parallel. Determine which functionality would be needed from the DBMS. Also characterize formally under which circumstances concurrent execution of transactions in the same conflict class could be allowed to be executed in parallel whilst respecting 1-copy consistency. Extend the NODO protocol with this enhancement.

Solution 13.6. This is the solution

Chapter 14
Parallel Database Systems

Problem 14.1 (*). Consider the centralized server organization with several application servers accessing one database server. Also assume that each application server stores a subset of the data directory that is fully stored on the database server. Assume also that the local data directories at different application servers are not necessarily disjoint. What are the implications on data directory management and query processing for the database server if the local data directories can be updated by the application servers rather than the database server?

Solution 14.1. This is the solution

Problem 14.2 (**). Propose an architecture for a parallel shared-memory database server and provide a qualitative comparison with shared-nothing architecture on the basis of expected performance, software complexity (in particular, data placement and query processing), extensibility, and availability.

Solution 14.2. This is the solution

Problem 14.3. Specify the parallel hash join algorithm for the parallel shared-memory database server architecture proposed in Exercise 14.2.

Solution 14.3. This is the solution

Problem 14.4 (*). Explain the problems associated with clustering and full partitioning in a shared-nothing parallel database system. Propose several solutions and compare them.

Solution 14.4. This is the solution

Problem 14.5. Propose a parallel semijoin algorithm for a shared-nothing parallel database system. How should the parallel join algorithms be extended to exploit this semijoin algorithm?

Solution 14.5. This is the solution

Problem 14.6. Consider the following SQL query:

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

Give four possible operator trees: right-deep, left-deep, zigzag and bushy. For each one, discuss the opportunities for parallelism.

Solution 14.6. This is the solution

Problem 14.7. Consider a nine way join (ten relations are to be joined) calculate the number of possible right-deep, left-deep and bushy trees, assuming that each relation can be joined with anyone else. What do you conclude about parallel optimization?

Solution 14.7. This is the solution

Problem 14.8 (**). Propose a data placement strategy for a cluster architecture that maximizes *intra-node* parallelism (intra-operator parallelism within a shared-memory node).

Solution 14.8. This is the solution

Problem 14.9 (**). How should the DP execution model presented in Section 14.4.4 be changed to deal with inter-query parallelism?

Solution 14.9. This is the solution

Problem 14.10 (**). Consider a multi-user centralized database system. Describe the main change to allow inter-query parallelism from the database system developer and administrator's points of view. What are the implications for the end-user in terms of interface and performance?

Solution 14.10. This is the solution

Problem 14.11 (**). Same question for intra-query parallelism on a shared-memory architecture or for a shared-nothing architecture.

Solution 14.11. This is the solution

Problem 14.12 (*). Consider the database cluster architecture in Figure 14.21. Assuming that each cluster node can accept incoming transactions, make precise the DBcluster middleware box by describing the different software layers, and their components and relationships in terms of data and control flow. What kind of information need be shared between the cluster nodes? how?

Solution 14.12. This is the solution

Problem 14.13 (**). Discuss the issues of fault-tolerance for the preventive replication protocol (see Section 14.5.2).

Solution 14.13. This is the solution

Problem 14.14 (**). Compare the preventive replication protocol with the NODO replication protocol (see Chapter 13) in the context of a cluster system in terms of: replication configurations supported, network requirements, consistency, performance, fault-tolerance.

Solution 14.14. This is the solution

Problem 14.15 (*). Let us consider a database cluster for an online store application. The database is concurrently accessed by short update transactions (e.g., product orders) and long read-only decision support queries (e.g., stock analysis). Discuss how database replication with freshness control can be useful in improving the response time of the decision support queries. What can be the impact on transaction load?

Solution 14.15. This is the solution

Problem 14.16 (**). Consider two relations R(A,B,C,D,E) and S(A,F,G,H). Assume there is a clustered index on attribute A for each relation. Assuming a database cluster with full replication, for each of the following queries, determine whether Virtual Partioning can be used to obtain intra-query parallelism and, if so, write the corresponding subquery and the final result composition query.

(a)
```
SELECT   B, COUNT(C)
FROM     R
GROUP BY B
```

(b)
```
SELECT   C, SUM(D), AVG(E)
FROM     R
WHERE    B=:v1
GROUP BY C
```

(c)
```
SELECT   B, SUM(E)
FROM     R, S
WHERE    R.A=S.A
GROUP BY B
HAVING   COUNT(*) > 50
```

(d)
```
SELECT   B, MAX(D)
FROM     R, S
WHERE    C = (SELECT SUM(G) FROM S WHERE S.A=R.A)
GROUP BY B
```

(e)
```
SELECT   B, MIN(E)
FROM     R
WHERE    D > (SELECT MAX(H) FROM S WHERE G >= :v1)
GROUP BY B
```

Solution 14.16. This is the solution

Chapter 15
Distributed Object Database Management Systems

Problem 15.1. Explain the mechanisms used to support encapsulation in distributed object DBMSs. In particular:

(a) Describe how the encapsulation is hidden from the end users when both the objects and the methods are distributed.
(b) How does a distributed object DBMS present a single global schema to end users? How is this different from supporting fragmentation transparency in relational database systems?

Solution 15.1. This is the solution

Problem 15.2. List the new data distribution problems that arise in object DBMSs, that are not present in relational DBMSs, with respect to fragmentation, migration and replication.

Solution 15.2. This is the solution

Problem 15.3 (**). Partitioning of object databases has the premise of reducing the irrelevant data access for user applications. Develop a cost model to execute queries on unpartitioned object databases, and horizontally or vertically partitioned object databases. Use your cost model to illustrate the scenarios under which partitioning does in fact reduce the irrelevant data access.

Solution 15.3. This is the solution

Problem 15.4. Show the relationship between clustering and partitioning. Illustrate how clustering can deteriorate/improve the performance of queries on a partitioned object database system.

Solution 15.4. This is the solution

Problem 15.5. Why do client-server object DBMSs primarily employ data shipping architecture while relational DBMSs emply function shipping?

Solution 15.5. This is the solution

Problem 15.6. Discuss the strengths and weaknesses of page and object servers with respect to data transfer, buffer management, cache consistency, and pointer swizzling mechanims.

Solution 15.6. This is the solution

Problem 15.7. What is the difference between caching information at the clients and data replication?

Solution 15.7. This is the solution

Problem 15.8 (*). A new class of applications that object DBMSs support are interactive and deal with large objects (e.g., interactive multimedia systems). Which one of the cache consistency algorithms presented in this chapter are suitable for this class of applications operating across wide area networks?

Solution 15.8. This is the solution

Problem 15.9 (**). Hardware and software pointer swizzling mechanisms have complementary strengths and weaknesses. Propose a hybrid pointer swizzling mechanism that incorporates the strengths of both.

Solution 15.9. This is the solution

Problem 15.10 (**). Explain how derived horizontal fragmentation can be exploited to facilitate efficient path queries in distributed object DBMSs. Give examples.

Solution 15.10. This is the solution

Problem 15.11 (**). Give some heuristics that an object DBMS query optimizer that accepts OQL queries may use to determine how to decompose a query so that parts can be function shipped and other parts have to be executed at the originating client by data shipping.

Solution 15.11. This is the solution

Problem 15.12 (**). Three alternative ways of performing *distributed* complex object assembly are discussed in this chapter. Give an algorithm for the alternative where complex operations, such as joins and complete assembly of remote objects, are performed at remote sites and the partial results are shipped to the central site for final assembly.

Solution 15.12. This is the solution

Problem 15.13 (*). Consider the airline reservation example of Chapter 10. Define a Reservation class (type) and give the forward and backward commutativity matrixes for it.

Solution 15.13. This is the solution

Chapter 16
Peer-to-Peer Data Management

Problem 16.1. What is the fundamental difference between P2P and client-server architectures? Is a P2P system with a centralized index equivalent to a client-server system? List the main advantages and drawbacks of P2P file sharing systems from different points of view:

- end-users;
- file owners;
- network administrators.

Solution 16.1. This is the solution

Problem 16.2 (**). A P2P overlay network is built as a layer on top of a physical network, typically the Internet. Thus, they have different topologies and two nodes that are neighbors in the P2P network may be far apart in the physical network. What are the advantages and drawbacks of this layering? What is the impact of this layering on the design of the three main types of P2P networks (unstructured, structured and superpeer)?

Solution 16.2. This is the solution

Problem 16.3 (*). Consider the unstructured P2P network in Figure 16.4 and the bottom-left peer that sends a request for resource. Illustrate and discuss the two following search strategies in terms of result completeness:

- flooding with TTL=3;
- gossiping with each peer has a partial view of at most 3 neighbours.

Solution 16.3. This is the solution

Problem 16.4 (*). Consider Figure 16.7, focusing on structured networks. Refine the comparison using the scale 1-5 (instead of low - moderate - high) by considering the three main types of DHTs: tree, hypercube and ring.

Solution 16.4. This is the solution

Problem 16.5 (**). The objective is to design a P2P social network application, on top of a DHT. The application should provide basic functions of social networks: register a new user with her profile; invite or retrieve friends; create lists of friends; post a message to friends; read friends' messages; post a comment on a message. Assume a generic DHT with put and get operations, where each user is a peer in the DHT.

Solution 16.5. This is the solution

Problem 16.6 (**). Propose a P2P architecture of the social network application, with the (key, data) pairs for the different entities which need be distributed. Describe how the following operations: create or remove a user; create or remove a friendship; read messages from a list of friends. Discuss the advantages and drawbacks of the design.

Solution 16.6. This is the solution

Problem 16.7 (**). Same question, but with the additional requirement that private data (e.g., user profile) must be stored at the user peer.

Solution 16.7. This is the solution

Problem 16.8. Discuss the commonalities and differences of schema mapping in multidatabase systems and P2P systems. In particular, compare the local-as-view approach presented in Chapter 4 with the pairwise schema mapping approach in Section 16.2.1.

Solution 16.8. This is the solution

Problem 16.9 (*). The FD algorithm for top-k query processing in unstructured P2P networks (see Algorithm 16.4) relies on flooding. Propose a variation of FD where, instead of flooding, random walk or gossiping is used. What are the advantages and drawbacks?

Solution 16.9. This is the solution

Problem 16.10 (*). Apply the TPUT algorithm (Algorithm 16.2) to the three lists of the database in Figure 16.10 witk $k = 3$. For each step of the algorithm, show the intermediate results.

Solution 16.10. This is the solution

Problem 16.11 (*). Same question applied to Algorithm DHTop (see Algorithm 16.5.

Solution 16.11. This is the solution

Problem 16.12 (*). Algorithm 16.6 assumes that the input relations to be joined are placed arbitrarily in the DHT. Assuming that one of the relations is already hashed on the join attributes, propose an improvement of Algorithm 16.6.

Solution 16.12. This is the solution

Problem 16.13 (*). To improve data availability in DHTs, a common solution is to replicate $(k, data)$ pairs at several peers using several hash functions. This produces the problem illustrated in Example 16.7. An alternative solution is to use a non-replicated DHT (with a single hash function) and have the nodes replicating (k, data) pairs at some of their neighbors. What is the effect on the scenario in Example 16.7? What are the advantages and drawbacks of this approach, in terms of availability and load balancing?

Solution 16.13. This is the solution

Chapter 17
Web Data Management

Problem 17.1 (**). Consider the graph in Figure 17.28. A node P_i is said to be a *reference* for for node P_j iff there exists an edge from P_j to P_i ($P_j \rightarrow P_i$) and there exist a node P_k such that $P_i \rightarrow P_k$ and $P_j \rightarrow P_k$.

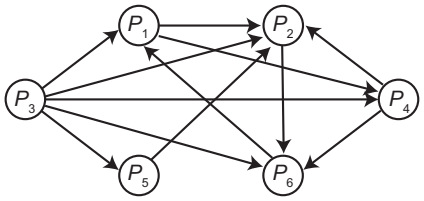


Fig. 17.28 Figure for Problem 17.1

(a) Indicate the reference nodes for each node in the graph.
(b) Find the cost of compressing each node using the formula given in [Adler and Mitzenmacher, 2001] for each of its reference nodes.
(c) Assuming that (i) for each node we only choose one reference node, and (ii) there must not be cyclic references in the final result, find the optimal set of references that maximizes compression. (Hint: note that this can be systematically done by creating a root node r, and letting all the nodes in the graph point to r, and then finding the minimum spanning tree starting from $r(cost(P_x, r) = \lceil log\ n \rceil * out\_deg(P_x))$.)

Solution 17.1. This is the solution

Problem 17.2. How does web search differ from web querying?

Solution 17.2. This is the solution

Problem 17.3 (**). Consider the generic search engine architecture in Figure 17.4. Propose an architecture for a web site with a shared-nothing cluster that implements

all the components in this figure as well as web servers in an environment that will support very large sets of web documents and very large indexes, and very high numbers of web users. Define how web pages in the page directory and indexes should be partitioned and replicated. Discuss the main advantages of your architecture with respect to scalability, fault-tolerance and performance.

Solution 17.3. This is the solution

Problem 17.4 (**). Consider your solution in Problem 17.3. Now consider a keyword search query from a web client to the web search engine. Propose a parallel execution strategy for the query that ranks the result web pages, with a summary of each web page.

Solution 17.4. This is the solution

Problem 17.5 (*). To increase locality of access and performance in different geographical regions, propose an extension of the web site architecture in Problem 17.4 with multiple sites, with web pages being replicated at all sites. Define how web pages are replicated. Define also how a user query is routed to a web site. Discuss the advantages of your architecture with respect to scalability, availability and performance.

Solution 17.5. This is the solution

Problem 17.6 (*). Consider your solution in Problem 17.5. Now consider a keyword search query from a web client to the web search engine. Propose a parallel execution strategy for the query that ranks the result web pages, with a summary of each web page.

Solution 17.6. This is the solution

Problem 17.7 (**). Given an XML document modeled as tree, write an algorithm that matches simple XPath expression that only contains child axes and no branch predicates, For example, /A/B/C should return all C elements who are children of some B elements who are in turn the children of the root element A. Note that A may contain child element other than B, and such is true for B as well.

Solution 17.7. This is the solution

Problem 17.8 (**). Consider two web data sources that we model as relations EMP1(Name, City, Phone) and EMP2(Firstname, Lastname, City). After schema integration, assume the view EMP(Firstname, Name, City, Phone) defined over EMP1 and EMP2, where each attribute in EMP comes from an attribute of EMP1 or EMP2, with EMP2.Lastname being renamed as Name. Discuss the limitations of such integration. Now consider that the two web data sources are XML. Give a corresponding definition of the XML schemas of EMP1 and EMP2. Propose an XML schema that integrates EMP1 and EMP2, and avoids the problems identified with EMP.

Solution 17.8. This is the solution

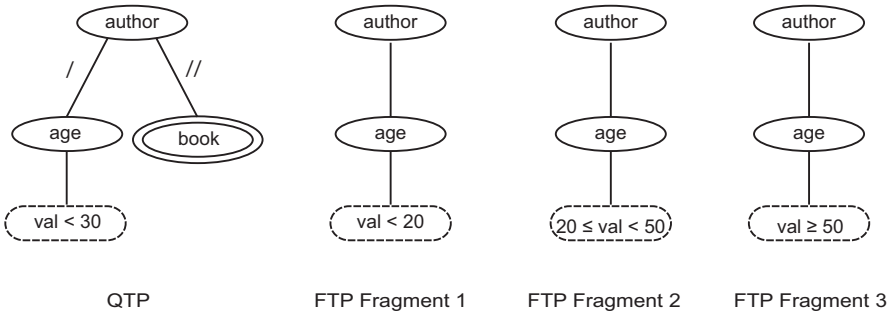


Fig. 17.29 Figure for Problem 17.9

Problem 17.9. Consider the QTP and the set of FTPs shown in Figure 17.29 and the vertical fragmentation schema in Figure **??**. Determine the fragment(s) that can be excluded from the distributed query plan for this QTP.

Solution 17.9. This is the solution

Problem 17.10 (**). Consider the QTP and the FTP shown in Figure 17.30. Can we exclude the fragment defined by this FTP from a query plan for the QTP? Explain your answer

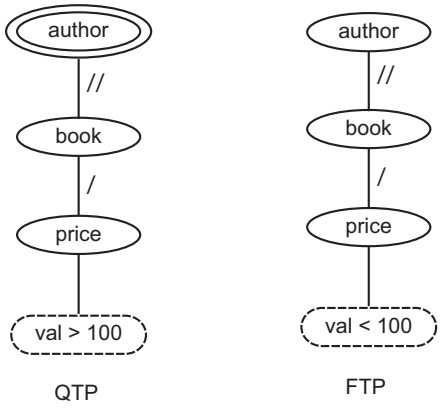


Fig. 17.30 Figure for Problem 17.10

Solution 17.10. This is the solution

Problem 17.11 (*). Localize the QTP shown in Figure 17.31 for distributed evaluation based on the vertical fragmentation schema shown in Figure **??**.

Solution 17.11. This is the solution

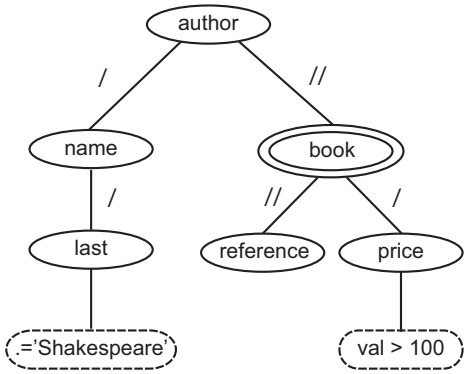


Fig. 17.31 Figure for Problem 17.11

Problem 17.12 (**). When evaluating the query from Problem 17.11, can any of the fragments be skipped using the method based on the Dewey decimal system? Explain your answer.

Solution 17.12. This is the solution