Chapter 21

21.2 Modes of Information Integration

Exercise 21.2.1

a) INSERT INTO Computers(number, speed, memory, hd)
        SELECT id, processor, mem, disk from Systems
    INSERT INTO Monitors(screen)
        SELECT screenSize from Systems

b) INSERT INTO Systems(id, processor, mem, disk, screenSize)
        SELECT Computers.number, speed, memory, hd, screen from Computers, Monitors
(Note: Cartesian product will yield all combinations)

Exercise 21.2.2

Global Schema:
Computers_g (cID, proc, speed, memory, hd)
Monitors_g (mID, screen, maxResX, maxResY)
Systems_g (sID, mID)

Exercise 21.2.3

INSERT INTO Computers_g
        SELECT number, proc, speed, memory, hd from Computers
INSERT INTO Monitors_g
        SELECT number, screen, maxResX, maxResY from Monitors
INSERT INTO Systems_g
        SELECT id, number from Systems, Monitors
                WHERE Systems.screenSize = Monitors.Screen
(Note: We could have a one to many relationship regarding screenSize & monitor model #)

Exercise 21.2.4

SELECT MAX(hd) from Computer where speed=3

Exercise 21.2.5

PC (model_number, process_id, mem_id, hd_id)
Processor (pid, name, speed)
Memory (mid, size, speed)
Disk (did, size, speed)
Monitors (model_number, manufacturer, screenSize, maxX, maxY)
Systems (sid, pc_mod_num, monitor_mod_num)

Here is an alternative for company A & B. But normalizing it further, it can hold more information and it can easily be composed back into our global schema.

Exercise 21.2.6

INSERT INTO AutosWhse(serialNo, model, color, autoTrans, dealer)

(SELECT serialNo, model, color, ‘yes’, ‘dealer1’, FROM Cars WHERE autoTrans > 0
UNION
SELECT serialNo, model, color, ‘no’, ‘dealer1’, FROM Cars WHERE autoTrans = 0)

Exercise 21.2.7
INSERT INTO AutosWhse(serialNo, model, color, autoTrans, dealer)
(SELECT Autos.serial, model, color, ‘yes’, ‘dealer2’, FROM Autos
 WHERE EXISTS(SELECT * FROM Options WHERE serial = Autos.serial AND option=’autoTrans’)
UNION
SELECT Autos.serial, model, color, ‘no’, ‘dealer2’, FROM Autos
 WHERE NOT EXISTS(SELECT * FROM Options WHERE serial = Autos.serial AND option=’autoTrans’))

Exercise 21.2.8
a) SELECT serialNo FROM Cars WHERE autoTrans = ‘yes’
 SELECT serial FROM Autos WHERE EXISTS(SELECT * FROM Options where serial = Autos.serial AND option=’autoTrans’

b) SELECT serialNo FROM Cars WHERE autoTrans = ‘no’
 SELECT serial FROM Autos WHERE NOT EXISTS(SELECT * FROM Options where serial = Autos.serial AND option=’autoTrans’

c) SELECT serialNo FROM Cars WHERE color = ‘blue’
(Note: Since mediator breaks the query in 2 parts (dealer 1 and 2), the original query against the view of dealer=dealer1 will prevent the query going to the dealer 2 wrapper.)

Exercise 21.2.9
Global Schema = Books(ISBN10, ISBN13, name, authors, edition, year, pages, hi_price, lo_price, ave_price)

We can collect the same information from different sources and calculate the max, min, and average selling prices of this book which could be quite useful for analysis.

21.3 Wrappers In Mediator-Based Systems

Exercise 21.3.1
SELECT serialNo, model, color, autoTrans, ‘dealer1'
FROM Cars
WHERE model = ‘$m’ AND color = (SELECT localColor from GtoL where globalColor = ‘$c’)

Exercise 21.3.2
a)
SELECT ‘B’, processor, mem, disk, screenSize FROM Systems WHERE processor = ‘$sp’
SELECT ‘A’, speed, memory, hd, screen FROM Computers, Monitors WHERE speed = ‘$sp’
b)

SELECT ‘B’, processor, mem, disk, screenSize FROM Systems WHERE ScreenSize = ‘$ss’
SELECT ‘A’, speed, memory, hd, screen FROM Computers, Monitors WHERE Monitors.screen = ‘$ss’
c)
SELECT ‘B’, processor, mem, disk, screenSize FROM Systems WHERE mem = ‘$ms’ AND disk = ‘$ds’
SELECT ‘A’, speed, memory, hd, screen FROM Computers, Monitors WHERE memory = ‘$ms’ AND hd = ‘$ds’

Exercise 21.3.3
a) SELECT manf, mem, screen FROM PCMed WHERE speed = 3.1 AND disk = 120
b) SELECT MAX(disk) FROM PCMed WHERE speed = 2.8
c) SELECT * FROM PCMed WHERE mem = 512 AND screen > disk

21.4 Capability-Based Optimization

Exercise 21.4.1
a) uc[P-IV, G5, Athlon]bo[integer]u
b) uuubf
c) buuu, ubuu, uubb
d) uc’[19,22,24,30]uu
e) ubbuu, ububu, ubuub, uubbu, uubub, uuubb

Exercise 21.4.2
a) Not feasible because no adornment for just screen size
b) Feasible
c) Not feasible, same reason as a)

21.5 Optimizing Mediator Queries

Exercise 21.5.1
a) Yes,
R^{fff} [abc] -> S^{bf} [abcd] -> T^{bff} [abcde] -> T^{fbf} [abcde]
R^{fff} [abc] -> S^{bf} [abcd] -> T^{fbf} [abcde] -> T^{bff} [abcde]
R^{fff} [abc] -> T^{fbf} [abcde] -> S^{bf} [abcde] -> T^{bff} [abcde]
R^{fff} [abc] -> T^{fbf} [abcde] -> T^{bff} [abcde] -> S^{bf} [abcde]

b) No.

c) Yes,
T^{fff} [bde] -> S^{fb} [bcde] -> R^{fbf} [abcde] -> S^{bf} [abcde]
T^{fff} [bde] -> S^{fb} [bcde] -> S^{bf} [abcde] -> R^{fbf} [abcde]
T^{fff} [bde] -> R^{fbf} [abcde] -> S^{fb} [abcde] -> S^{bf} [aabcde]
T^{fff} [bde] -> R^{fbf} [abcde] -> S^{bf} [aabcde] -> S^{fb} [abcde]

Exercise 21.5.2
R^{fff} [abc] -> R^{fbb} [abc] -> S^{ff} [abcd] -> S^{bf} [abcd] -> T^{ff} [abcde] -> T^{bf} [abcde]

Exercise 21.5.3
R^{ff}, R^{fb}, R^{fu}, R^{fc}, R^{fo}, R^{bf}, R^{bb}, R^{bu}, R^{bc}, R^{co}, R^{cf}, R^{cb}, R^{cu}, R^{cc}, R^{co}

Exercise 21.5.4
If a subgoal is not chosen to be resolved at step i, it can be resolved at any step j, j > i. Each resolution step will introduce new possibilities for the next step. It is not possible to eliminate possibilities, so if there exists a solution the algorithm will find one.

21.6 Local-as-View Mediators

Exercise 21.6.1
Q_3 contains Q_1
Q_3 contains Q_2
Q_2 contains Q_4
Q_3 contains Q_4

Exercise 21.6.2
$V_1(x,a)$ and $V_2(x,b)$ and $V_1(a,b)$ and $V_2(a,c)$ and $V_1(b,c)$ and $V_2(b,y)$

Exercise 21.6.3
There are two possibilities: A(a,c) and A(c,d), or A(c,d) and A(d,b). Each will leave one predicate unsatisfied.

Exercise 21.6.4
Let $Q_1 = p_1$ and ... and p_n, and $Q_2 = p_1$ and ... and p_{n+1}. As a result, Q_2, contains Q_1.

21.7 Entity Resolution

Exercise 21.7.1
a)
All subsequences of “abcab”: “”, “a”, “b”, c”, “ab”, “ac”, “aa”, “bc”, ”ba”, “bb”, “abc”, “aba”, “abb”, “bca”, “bcb”, “bab”, “abca”, “abcb”, “bcab”, and “abcab”.

b)
All subsequences of “aabb”: “”, “a”, “b”, “aa”, “ab”, “bb”, “aab”, “abb”, “aabb”

c) 2^n

Exercise 21.7.2

Strings	Longest Common Subsequences

“she”, “hers”	“he”
“she”, “they”	“he”
“she”, “theirs”	“he”
“hers”, “they”	“he”
“hers”, “theirs”	“hers”
“they”, “theirs”	“the”

Exercise 21.7.3

a)

Strings	Shortest Common Supersequences
“she”, “hers”	“shers”
“she”, “they”	“sthey”, “tshey”
“she”, “theirs”	“stheirs”, “tsheirs”
“hers”, “they”	“thersy”, “theyrs”, “therys”
“hers”, “theirs”	“theirs”
“they”, “theirs”	“theyirs”, “theiyrs”, “theirys”, “theirsy”

b)

Shortest common supersequences of “abc” and “cb”:

“abcb”, “acbc”, “cabc”

c) $\frac{(m+n)!}{m!n!}$

Exercise 21.7.4

a)
Idempotence

Yes, the merge will satisfy the idempotent law; the longest common subsequence of any string is itself.

Commutativity

Yes, the merge will satisfy the commutative law; the longest common subsequence between string *s* and *t* is identical to the one between *t* and *s*.

Associativity

Yes, the merge will satisfy the associative law. The longest common subsequence between two strings is the intersection between the two sets of characters where each character still maintain its relative position from the strings (whether it's before of after another character in the same string). Intersection is an associative operation.

b)
Idempotence

Yes, the merge will satisfy the idempotent law; the shortest common supersequence of any string is itself.

Commutativity

Yes, the merge will satisfy the commutative law; the shortest common supersequence between string *s* and *t* is identical to the one between *t* and *s*.

Associativity

Yes, the merge will not satisfy the associative law. The shortest common supersequence between two strings is union between the two sets of characters where each character still maintain its relative position from the strings. Union is an associative operation.

Exercise 21.7.5

Idempotence

i. A record is always similar to itself since all fields are identical.
*ii.*A record merging with itself is itself since all fields have common values

Commutativity

i. This similarity function has no distinction of orders; a ≈ b = b ≈ a, where ≈ is the similarity function.

ii. This merge function has no distinction of orders. The common value between two fields is the same regardless of order.

Associativity

(a Λ b) Λ c = a Λ (b Λ c)
(b Λ a) Λ c = b Λ (a Λ c)   (commutative)

	Name	Address	Phone
a	Susan	123 Oak St.	123-123-1234
b	Susan	123 Oak St.	NULL
c	Susan	123 Oak St.	123-222-1234
a Λ b	Susan	123 Oak St.	NULL
b Λ c	Susan	123 Oak St.	

Applying the commutative law shows that between a, b, and c, there can be at most one field where values may differ. Thus, the order of the merge does not matter since the result can only differ with the remaining record by at most 1 field.

Representability

r ≈ s, r ≈ (s Λ t)

	Name	Address	Phone
a	Susan	123 Oak St.	123-123-1234
b	Susan	123 Oak Street	123-123-1234
c	Susan	NULL.	123-123-1234
d	Susan	123 Oak Street.	000-123-1234
b Λ *d*	Susan	123 Oak Street.	NULL

No, the similarity and merge functions do not have the representability property since *a* is similar to *b*, but it is not similar to the result of the merge between *b* and *d*.

Exercise 21.7.6

$a \leq b$ iff $a \wedge b = b$

A partial order is reflexive, transitive, and antisymmetric.

Reflexive: $a \leq a$

$a \wedge a = a$ (reflexivity, idempotence)

Thus, $a \leq a$.

Transitive: $a \leq b$ and $b \leq c$ implies $a \leq c$

$a \wedge b = b$ and $b \wedge c = c$

$(a \wedge b) \wedge c = c$ (substitution)
$a \wedge (b \wedge c) = c$ (associativity)
$a \wedge c = c$ (substitution)

Thus, $a \leq c$

Antisymmetric: $a \leq b$ and $b \leq a$ iff $a = b$

$a \wedge b = b$ and $b \wedge a = a$

$a = b \wedge a = a \wedge b = b$ (commutativity)

Thus, $a = b$