SELECTED SOLUTIONS TO THE REVISION EXERCISES:

1. In the following questions the operations are as follows

$r_n(X)$	transaction n reads data item X,
$w_n(X)$	transaction n writes data item X,
c_n	transactions n commits, and
a_n	transaction n aborts.

a. Is the following execution recoverable? Give reasons for your answer. Assume that initially X, Y and Z equal 1 and each time a transaction writes the data item is incremented by 1.

   $r_2(Y)$, $w_2(X)$, $r_1(X)$, $w_1(Z)$, $r_2(Z)$, c_1, a_2.

b. Examine the following histories. Draw their serialization graph and identify which of them is serializable given reasons.
   (i) $r_1(X)$, $w_1(X)$, $r_2(Y)$, $w_2(Y)$, $r_1(Z)$, $w_1(Z)$, c_1, c_2.
   (ii) $r_1(X)$, $w_1(X)$, $w_2(X)$, $r_2(Y)$, $r_1(Y)$, $w_1(Z)$, c_1, c_2.
   (iii) $r_1(X)$, $r_2(X)$, $w_1(X)$, $w_2(X)$, $w_1(Y)$, c_2, c_1.
   (iv) $r_1(X)$, $r_2(X)$, $r_3(X)$, $r_1(Y)$, $w_2(Y)$, $r_3(Z)$, $w_3(Z)$, $r_1(Z)$, $w_1(X)$, $r_2(Z)$, c_2, c_1, c_3.

ANSWER:

a. non-recoverable because
   transaction 1 commits while transaction 2 has not; and
   data item X has been first written by transaction 2 which is later read by transaction1.

   $$w_2(X) \rightarrow r_1(X) \rightarrow c_1$$

   Hence, to be recoverable transaction 2 must be committed first.

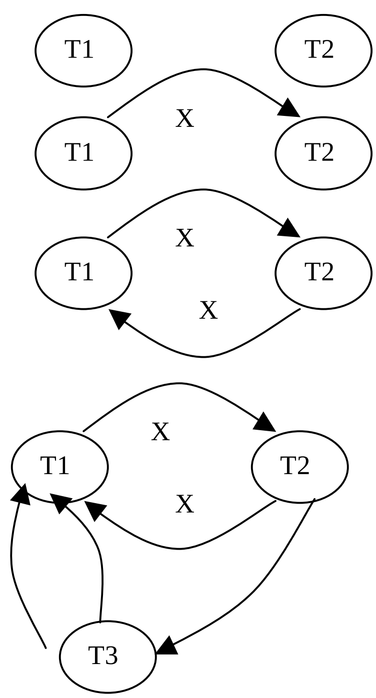
b. questions (i) and (ii) are serializable whereas questions (iii) and (iv) are non-serializable because of the existence of a cycle in each of the graph.

(i) serializable (no cycle)

(ii) serializable (no cycle)

(iii) non-serializable (cycle T_1-T_2-T_1)

(iv) non-serializable (cycle T_1-T_2-T_1)



2. Given the following database schema:

EMPLOYEE								
FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	DNO

DEPARTMENT			
DNAME	DNUMBER	MGRSSN	MGRSTARTDATE

DEPT_LOCATION	
DNUMBER	DLOCATION

PROJECT			
PNAME	PNUMBER	PLOCATION	DNUM

DEPENDENT				
ESSN	DEP-NAME	SEX	BDATE	RELATIONSHIP

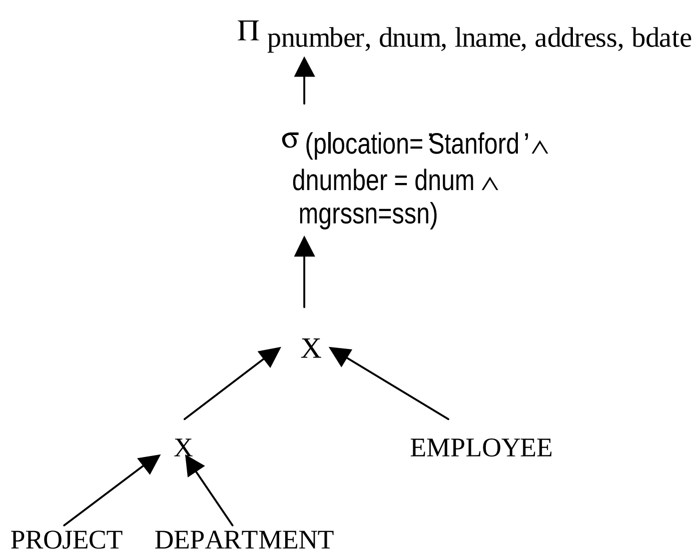
For each of the following queries, prepare the initial (canonical) query tree, then show how the query tree is optimized by the use of heuristic optimization.

a. For every project located in ‘Stanford’, list the project number, the controlling department number, and the department manager’s lastname, address and birthdate.

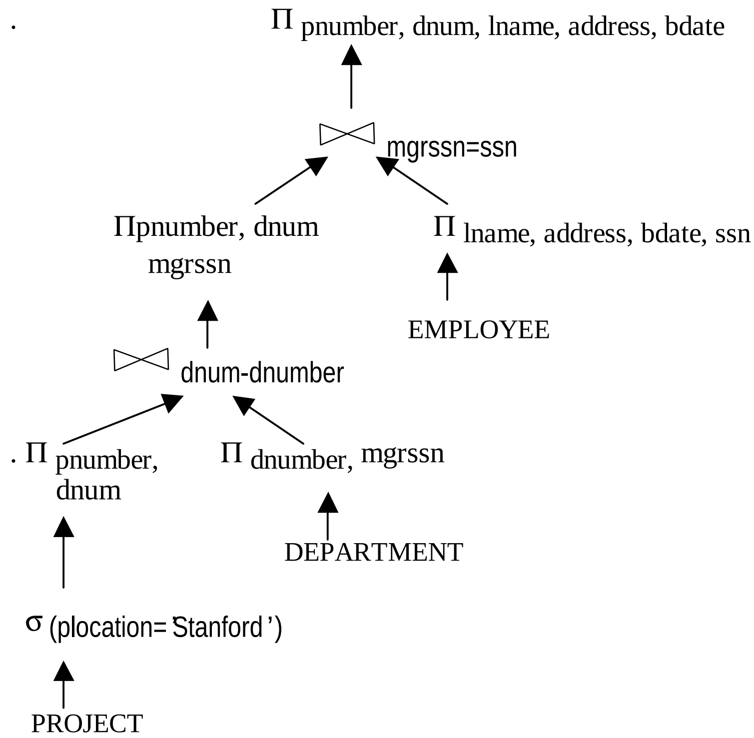
ANSWER:

a.

CANONICAL



HEURISTIC OPTIMIZATION



2c.

Consider the relational database schema represented by the Bachman diagram presented in question 2. Suppose that all the relations were created by (and hence are owned by) user X, who wants to grant the following privileges to user accounts A, B, C, D, and E:

a) Account A can retrieve or modify any relation except DEPENDENT and can grant any of these privileges to other users.

b) Account B can retrieve all the attributes of EMPLOYEE and DEPARTMENT except for SALARY, MGRSSN, and MGRSTARTDATE.

c) Account C can retrieve or modify WORKS_ON but can only retrieve the FNAME, MINIT, LNAME, SSN attributes of EMPLOYEE and PNAME, PNUMBER attributes of PROJECT.

d) Account D can retrieve any attribute of EMPLOYEE or DEPENDENT and can modify DEPENDENT.

e) Account E can retrieve any attribute of EMPLOYEE but only for EMPLOYEE tuples that have DNO = 3.

Write SQL statements to grant these privileges. Use views where appropriate.

NB - UPDATE and INSERT can specify particular attributes
SELECT, DELETE cannot be attribute specific, these must be created via views

a) GRANT SELECT, UPDATE
ON EMPLOYEE, DEPARTMENT, DEPT_LOCATIONS, PROJECT, WORKS_ON
TO ACCOUNTA
WITH GRANT OPTION;

b) CREATE VIEW EMPS AS
SELECT FNAME, MINIT, LNAME, SSN, BDATE, ADDRESS, SEX SUPERSSN, DNO
FROM EMPLOYEE;

GRANT SELECT ON EMPS
TO ACCOUNTB;

CREATE VIEW DEPTS AS
SELECT DNAME, DNUMBER
FROM DEPARTMENT;

GRANT SELECT ON DEPTS
TO ACCOUNTB;

c) GRANT SELECT, UPDATE
ON WORKS_ON
TO ACCOUNTC;

CREATE VIEW EMP1 AS
SELECT FNAME, MINIT, LNAME, SSN
FROM EMPLOYEE;

GRANT SELECT ON EMP1
TO ACCOUNTC;

CREATE VIEW PROJ1 AS
SELECT PNAME, PNUMBER
FROM PROJECT;

GRANT SELECT ON PROJ1
TO ACCOUNTC;

d) GRANT SELECT ON EMPLOYEE, DEPENDENT
TO ACCOUNTD;
GRANT UPDATE ON DEPENDENT
TO ACCOUNTD;

e) CREATE VIEW DNO3_EMPLOYEES AS
SELECT * FROM EMPLOYEE
WHERE DNO = 3;
GRANT SELECT ON DNO3_EMPLOYEES
TO ACCOUNTE;

3. Suppose a query written in SQL is as follows:

```
Select E.Ename
      From J, G, E
      Where G.Eno = E.Eno
      And G.Jno = J.Jno
      And E.Ename != "Fred"
      And J.Name = "CS"
      And (G.Duration = 12 or G.Duration = 24)
```

Draw the **most optimized** query tree for the above query.

ANSWER:

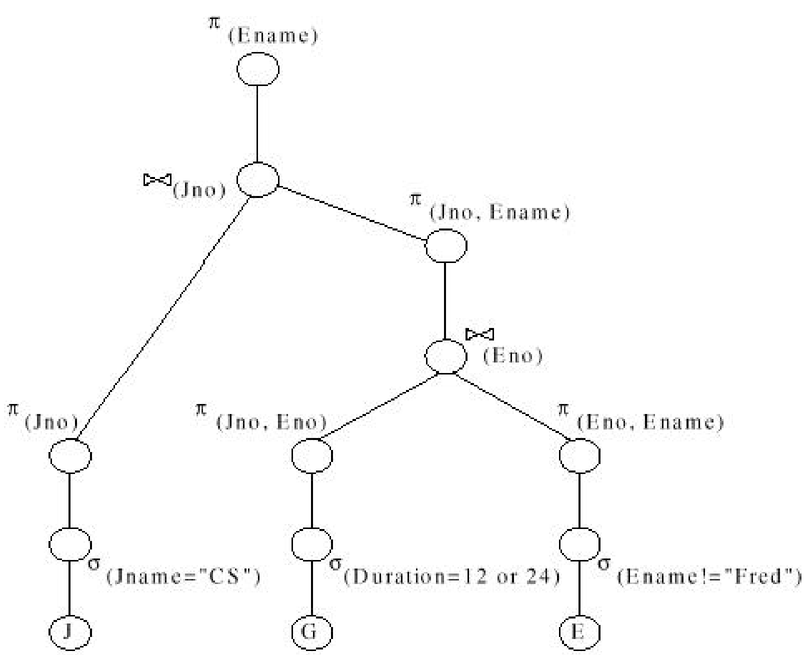
The three **selection** operations (`E.Ename != "Fred", J.Name = "CS", and (G.Duration = 12 or G.Duration = 24)` must be done first.

Before tables G and E are joined based on Eno, both tables G and E perform a **projection** operation. Table G needs attributes Eno (for the join with E) and Jno (later for the join with J) only. On the other hand, table E needs attributes Eno (for the join with G) and Ename (to be finally projected by the query).

Table J can also perform a **projection** operation to produce attribute Jno (later for the join operation with G).

Once all selection and projections are completed, the join operations can start. The **first join** operation is to join table G and E based on attribute Eno. Since attribute Eno is no longer needed after the first join, the result of the first join operation is then **projected** to obtain attributes Jno (for the next join) and Ename (for the final projection).

The **second join** operation is to join the result of the first join operation with table J based on attribute Jno. The final result is to **project** attribute Ename.
The query tree is shown as follows.



4. The following list represents the sequence of events in an interleaved execution of a set of transaction T1, T2, …, T12. (Note: A, B, …, H are intended to be items in the database).

Time	***Transaction***	***Activity***
Time t0	…..	…..
Time t1	T1	Read A
Time t2	T2	Read B
…	T1	Read C
…	T4	Read D
…	T5	Read A
…	T2	Read E
	T2	Update E
	T3	Read F
	T2	Read F
	T5	Update A
	T1	Commit
	T6	Read A
	T5	Rollback
	T6	Read C
	T6	Update C
	T7	Read G
	T8	Read H
	T9	Read G
	T9	Update G
	T8	Read E
	T7	Commit
	T9	Read H
	T3	Read G
	T10	Read A
	T9	Update H
	T6	Commit
	T11	Read C
	T12	Read D
	T12	Read C
	T2	Update F
	T11	Update C
	T12	Read A
…	T10	Update A
…	T12	Update D
…	T4	Read G
Time t*n* …… ……		

Assume that READ R acquires an S-lock on R, and UPDATE R promotes that lock to X-level. Assume also that all locks are held until the next synchpoint. Are there any deadlocks at Time t*n*? Explain your answer. Although you may use a table to arrive at your answer, your final result should be expressed as a *Wait-For Graph*.

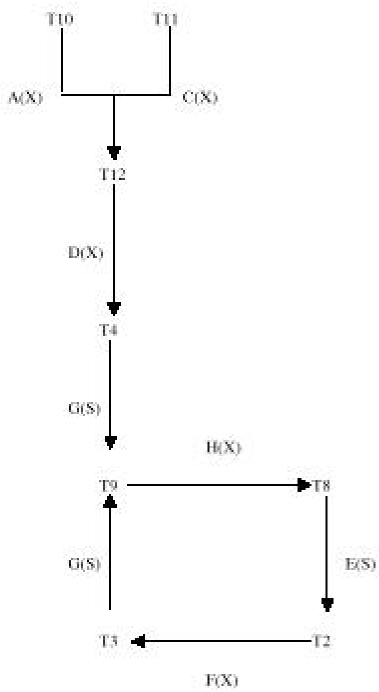
Using the table below, it can be determined all of the transactions which are in the waiting state.

	A	B	C	D	E	F	G	H
T2		R						
T4				R				
T2					R			
T2					U(X)			
T3						R		
T2						R		
T8								R
T9							R	
T9							U(X)	
T8					Wait			
T9								R
T3							Wait	
T10	R							
T9								Wait
T11			R					
T12				R				
T12			R					
T2						Wait		
T11			Wait					
T12	R							
T10	Wait							
T12				Wait				
T4							Wait	

Item A:	T10	waiting on	T12
Item B:	none		
Item C:	T11	waiting on	T12
Item D:	T12	waiting on	T4
Item E:	T8	waiting on	T2
Item F:	T2	waiting on	T3
Item G:	T3	waiting on	T9
	T4	waiting on	T9
Item H:	T9	waiting on	T8

At time t*n* no transactions are doing any useful work at all! There is one deadlock, involving transactions T2, T3, T9, and T8; in addition, T4 is waiting for T9, T12 is waiting for T4, and T10 and T11 are both waiting for T12.

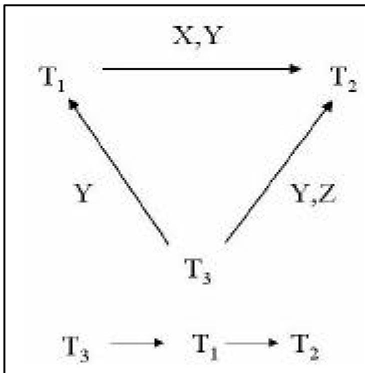
We can represent the above situations by means of a graph (the *WaitFor Graph*), in which the nodes represent transactions and a directed edge from node T*i* to node T*j* indicates that T*i* is waiting for T*j*. Edges are labelled with name of the database item and level of lock they are waiting for.



5.

a.

T1	T2	T3
		r(y)
		r(z)
r(x)		
w(x)		
		w(y)
		w(z)
	r(z)	
r(y)		
w(y)		
	r(y)	
	w(y)	
	r(x)	
	w(x)	



Serializable - equivalent as depicted to T3 then T1 then T2

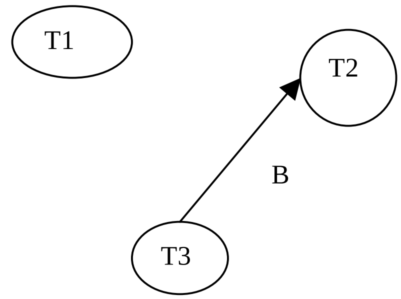
6.

Using wait-for-graphs determine if the following transaction sequences are in deadlock:

a.

T1	r(a)
T1	w(a)
T2	r(b)
T1	commit
T3	r(b)
T2	w(b)

	A	B
T1	R	
T1	W	
T2		R
T1	COMMIT	
T3		Wait (R)
T2		W

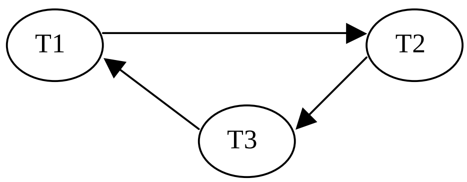


8. Exercises 17.12 and 17.13 fom Begg and Connoly pg.601

17.12 *(a) Explain what is meant by the constrained write rule, and explain how to test whether a schedule is serializable under the constrained write rule. Using the above method, determine whether the following schedule is serializable:*

S = [$R_1(Z)$, $R_2(Y)$, $W_2(Y)$, $R_3(Y)$, $R_1(X)$, $W_1(X)$, $W_1(Z)$, $W_3(Y)$, $R_2(X)$, $R_1(Y)$, $W_1(Y)$, $W_2(X)$, $R_3(W)$, $W_3(W)$]
where $R_i(Z)/W_i(Z)$ indicates a read/write by transaction i on data item Z.

Constrained write rule: transaction updates a data item based on its old value, which is first read by the transaction. A precedence graph can be produced to test for serializability.

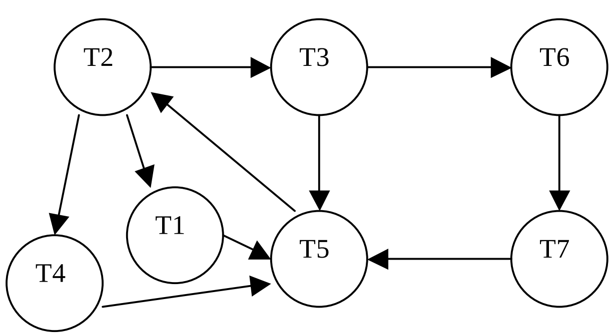


Cycle in precedence graph, which implies that schedule is not serializable.

(b) Would it be sensible to produce a concurrency control algorithm based on serializability? Give justification for your answer. How is serializability used in standard concurrency control algorithms?

No - interleaving of operations from concurrent transactions is typically determined by operating system scheduler. Hence, it is practically impossible to determine how the operations will be interleaved beforehand to ensure serializability. If transactions are executed and then you test for serializability, you would have to cancel the effect of a schedule if it turns out not to be serializable. This would be impractical!

17.13 *Produce a wait-for-graph for the following transaction scenario and determine whether deadlock exists.*



9. Example 17.1-17.9 (pg.561-580)

10. Exercise 18.15-18.17 (pg.640)

18.15 *Calculate the cost of the three strategies cited in Example 18.1 if the Staff relation has 10,000 tuples, Branch has 500 tuples, there are 500 Managers (one for each Branch), and there are 10 London branches.*

Costs: (1) 10,010,500
(2) 30,500
(3) 11,520.

18.16 *Using the Hotel schema given in the Exercises of Chapter 13, determine whether the following queries are semantically correct:*

(a) *SELECT r.type, r.price*
FROM room r, hotel h
WHERE r.hotel_number = h.hotel_number AND
h.hotel_name = 'Grosvenor Hotel' AND
r.type > 100;
Not semantically correct: hotel_number and hotel_name not in schema; type is character string and so cannot be compared with an integer value (100).

(b) *SELECT g.guest_no, g.name*
FROM hotel h, booking b, guest g
WHERE h.hotel_no = b.hotel_no AND h.hotel_name = 'Grosvenor Hotel';
Not semantically correct: hotel_name not in schema; Guest table not connected to remainder of query.

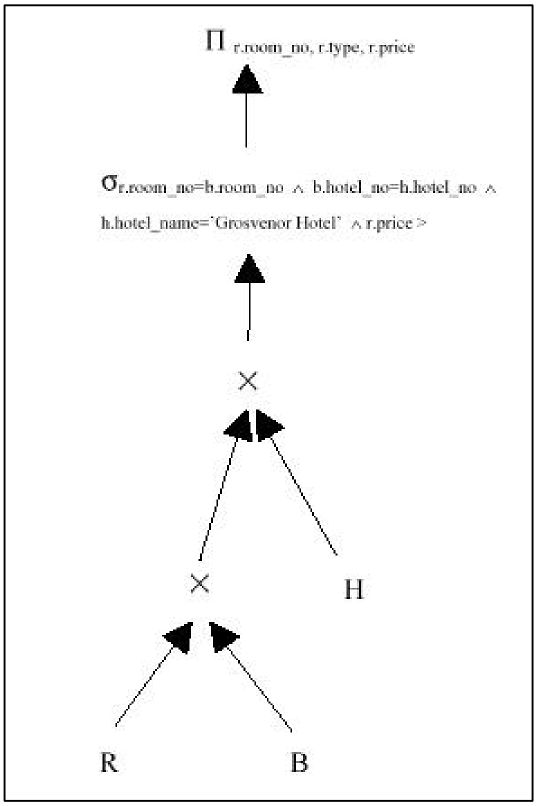
(c) *SELECT r.room_no, h.hotel_no*
FROM hotel h, booking b, room r
WHERE h.hotel_no = b.hotel_no AND h.hotel_no = 'H21' AND
b.room_no = r.room_no AND type = 'S' AND b.hotel_no = 'H22';
Not semantically correct: hotel_no cannot be both H21 in Hotel and H22 in Booking.

18.17 *Again, using the Hotel schema given in the Exercises of Chapter 13, draw a relational algebra tree for each of the following queries and use the heuristic rules given in Section 18.3.2 to transform the queries into a more efficient form:*

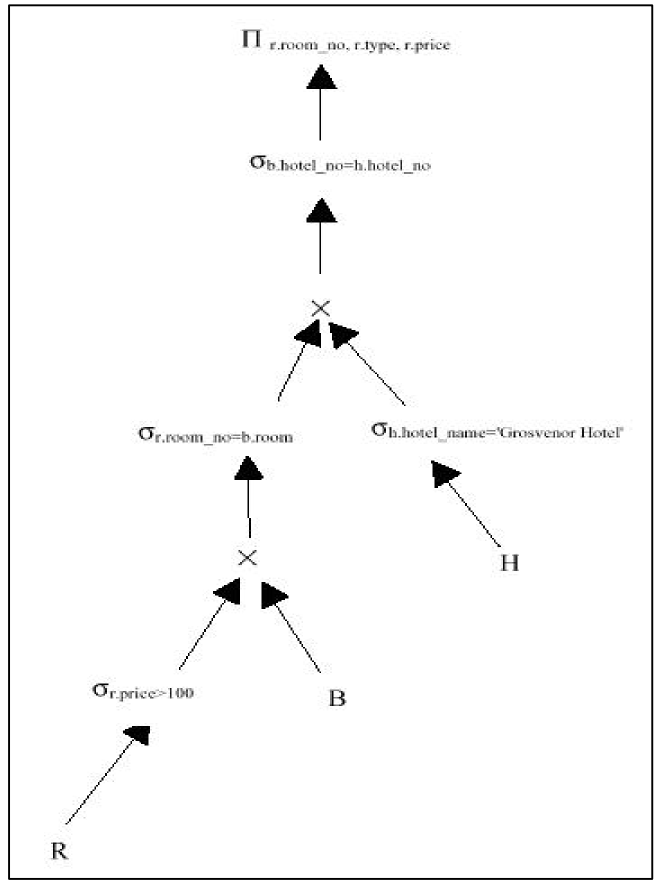
(a) *SELECT r.rno, r.type, r.price*
FROM room r, booking b, hotel h
WHERE r.room_no = b.room_no AND b.hotel_no = h.hotel_no AND
h.hotel_name = 'Grosvenor Hotel' AND r.price > 100;

(b) *SELECT g.guest_no, g.name*
FROM room r, hotel h, booking b, guest g
WHERE h.hotel_no = b.hotel_no AND g.guest_no = b.guest_no AND
h.hotel_no = r.hotel_no AND h.hotel_name = 'Grosvenor Hotel' AND
date_from >= '1-Jan-98' AND date_to <= '31-Dec-98';

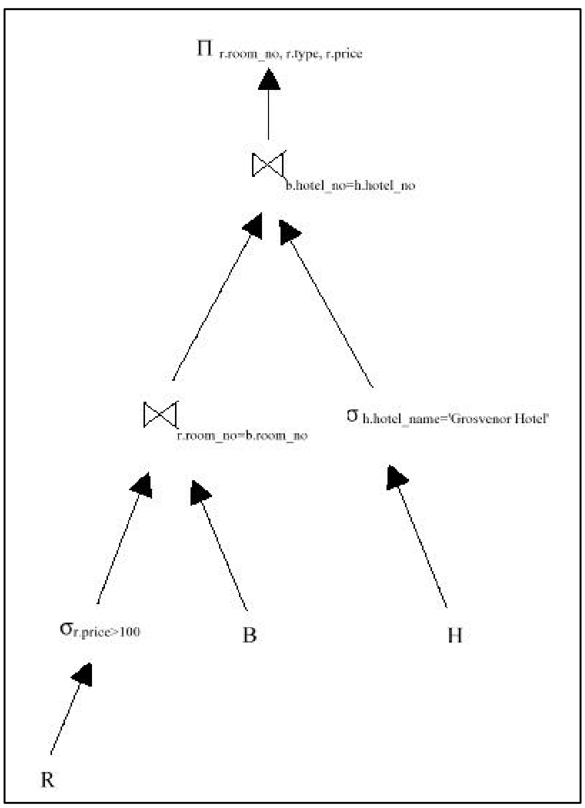
Discuss each step and state any transformation rules used in the process.



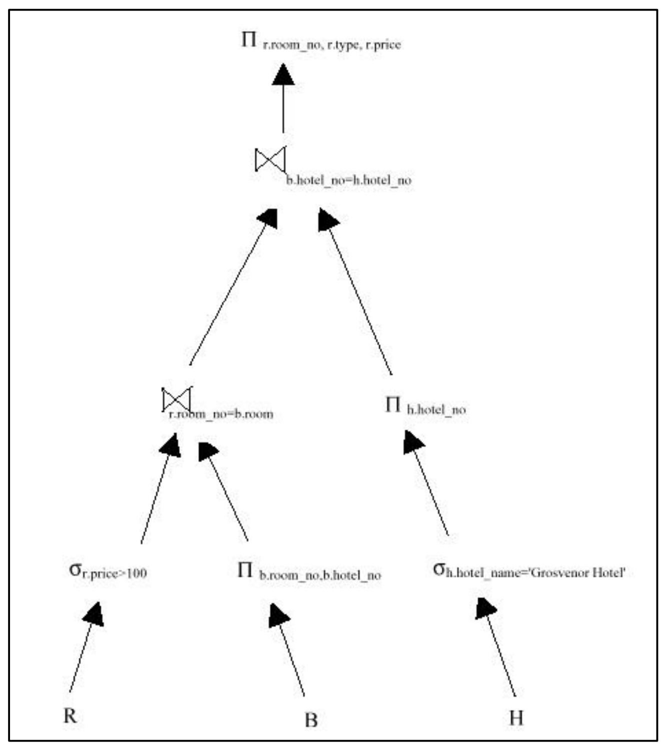
Π r.room_no, r.type, r.price
σr.room_no=b.room_no ∧ b.hotel_no=h.hotel_no ∧
h.hotel_name='Grosvenor Hotel' ∧ r.price >
×
×
H
R
B



Π r.room_no, r.type, r.price
σb.hotel_no=h.hotel_no
×
σr.room_no=b.room
σh.hotel_name='Grosvenor Hotel'
×
H
σr.price>100
B
R

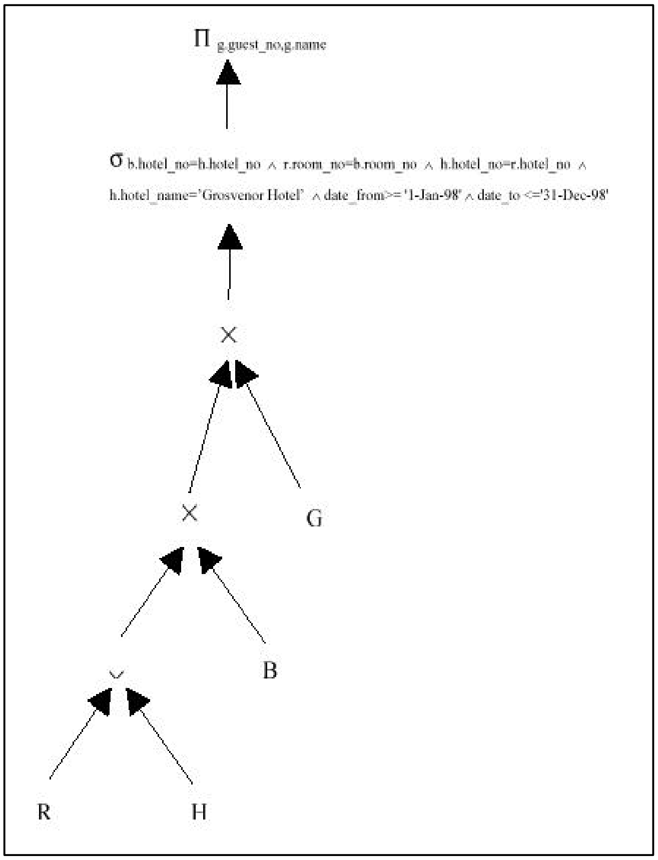


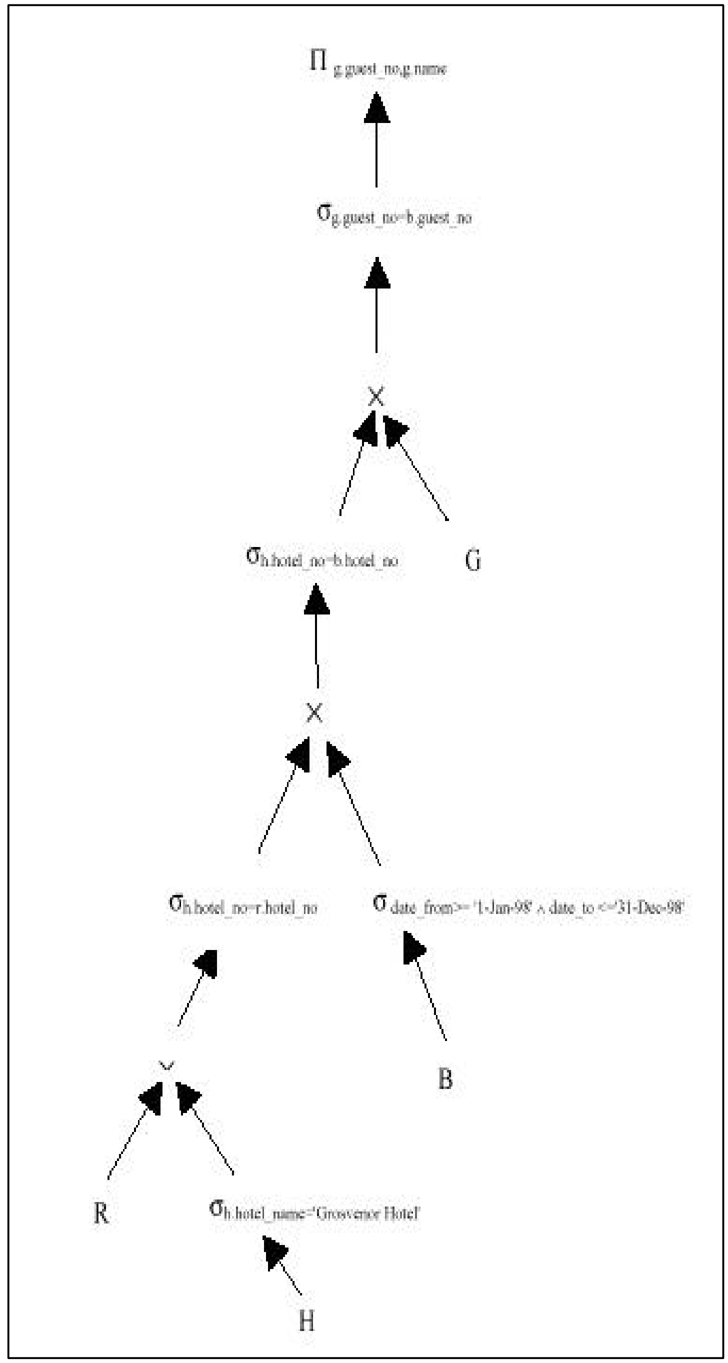
Π r.room_no, r.type, r.price
⋈ b.hotel_no=h.hotel_no
⋈ r.room_no=b.room_no
σ h.hotel_name='Grosvenor Hotel'
σr.price>100
B
H
R

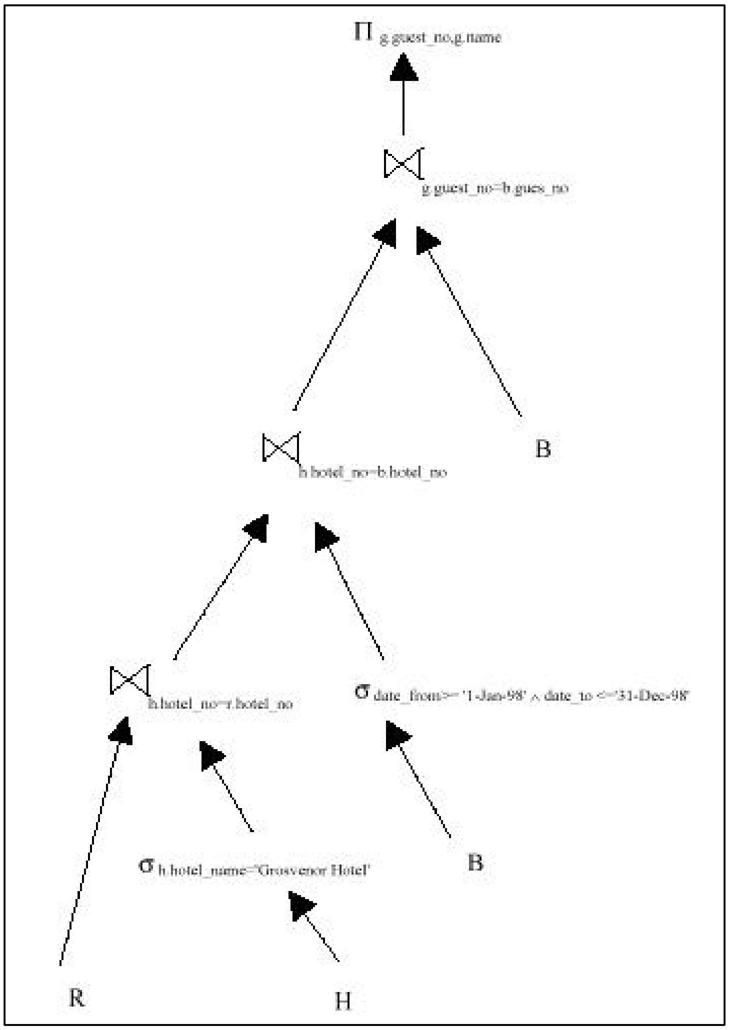


Π r.room_no, r.type, r.price
⋈ b.hotel_no=h.hotel_no
⋈ r.room_no=b.room
Π h.hotel_no
σr.price>100
Π b.room_no,b.hotel_no
σh.hotel_name='Grosvenor Hotel'
R
B
H

(b)


Π g.guest_no,g.name
σ b.hotel_no=h.hotel_no ∧ r.room_no=b.room_no ∧ h.hotel_no=r.hotel_no ∧
h.hotel_name='Grosvenor Hotel' ∧ date_from>='1-Jan-98' ∧ date_to <='31-Dec-98'
×
×
G
B
R
H


Π g.guest_no,g.name
σ g.guest_no=b.guest_no
×
σ h.hotel_no=b.hotel_no
G
×
σ h.hotel_no=r.hotel_no
σ date_from>='1-Jan-98' ∧ date_to <='31-Dec-98'
B
R
σ h.hotel_name='Grosvenor Hotel'
H



11. Example 18.1- 18.5 (pg. 604-632)

12. Define the followings:

- Properties of Transaction (pg. 560)
- Define concurreny control (pg.561), dirty read/unrepeatable read (p564)
- precedence graph
- Two phase locking with example
- Deadlock with example
- Wait for graph with example
- Hierarchy of granularity (p.582)
- conjunctive normal form, disjunctive normal form (p.609)
- Transformation rules for RA operations (p.613)
- Heuristic processing strategies. (p.616)
- objectives of query processing

13. Fill in the blanks.