CITY UNIVERSITY OF HONG KONG

Course code & title	:	CS3402 Database Systems
Session	:	Semester B 2013/2014
Time allowed	:	Two hours

This paper has 5 pages (including this cover page).

1. This paper consists of SIX questions. Choose any FOUR out of SIX.
2. Please put your answers in the answer book.
3. This exam is a **closed book** one,

Candidates are allowed to use the following materials/aids:

Approved Calculator

1 Page of A4 paper with important notes on

Materials/aids other than those stated above are not permitted. Candidates will be subject to disciplinary action if any unauthorized materials or aids are found on them.

DO NOT turn this page until you are told to do so.

__**Problem one**__: **Basic Concepts** (25 points)

Answer the following questions using __not more than five (5) sentences__ for each:

(a) An ER diagram can be viewed as a graph. What do the following mean in terms of the structure of an enterprise application? **(6 points)**

- The graph is disconnected
- The graph is acyclic
- The ER diagram has the same entity set appearing several times

Answer:

- A disconnected graph implies that there are pairs of entity sets that are ***unrelated*** to each other, hence maybe better off to design separate databases (each corresponding to a connected subgraph) [2 points]
- An acyclic graph means that there is a ***unique path*** between every pair of entity sets and, thus, a ***unique relationship*** between every pair of entity sets. [2 points]
- If an ER diagram has an entity set appearing several times, it means we are missing some relationships in the model, leading to a bad design. [2 points]

(b) Will you agree that all the relationships involving a weak entity set should connect to that weak entity set using double line? If you do/don't agree, please explain briefly in either case. **(4 points)**

Answer: Not agree. [2 point]
Double line involving a weak entity set means identifying relationship. Only the entities from which this weak entity is borrowing primary keys need to connect to the weak entity using double line. [2 points]

(c) What are the types of integrity constraints that must be checked for each tuple deletion operation upon a relation? Why? **(5 points)**

Answer: We need to check for *referential integrity constraint*. [2 points]
The reason is to make sure a tuple t_i to be deleted in a relation R_1 does not have in a different relation R_2 any tuple t_j which is referring to t_i. [3 points]

(d) What is the consequence of inserting rows into a "view" V when:

- V is defined based on one physical table **(3 points)**
- V is defined based on multiple tables **(3 points)**

Answer:

- *V is defined based on just one physical table P* – the insertion will be ***legal*** and ***anomaly-free***, possibly with some null values for those "missing" attributes if V contains a subset of the attributes of table P.
- *V is defined based on multiple physical tables P1, ..., Pn* – the insertion will be ***illegal*** due to possible anomaly, because some of the tuples inserted to V may be "***lost***" when we try to find these tuples back through V's definition (SQL) upon *P1, ..., Pn*.

(e) If a functional dependency X→Y holds in two relations R and S, does the functional dependency X→Y still hold on the relation R **NJ** S (where **NJ** stands for "natural join")? Justify your answers.

(4 points)

Answer: Yes, since all the X's and Y's values will come from the common pairs appearing in both R and S (ie, a subset).

Problem two: Constraints and Functional Dependencies (25 points)

a) List all functional dependencies (FDs) satisfied by the following relation:

A	B	**C**
a1	*b1*	***c1***
a1	*b1*	***c2***
a2	*b1*	***c1***
a2	***b1***	***c3***

Note that you do not need to list any of the "trivial" FDs. **(5 points)**

Answer:

The non-trivial ones are: A→B, and

C→B (thus AC→B too, as implied).

b) Use Armstrong's axioms to prove the soundness (correctness) of the union rule, i.e., if α→β and α→γ, then α→ βγ. **(10 marks)**

Answer:

Given that α→β, we have αα→αβ (the augmentation rule), so α→αβ (union of identical sets). Since we also have α→γ, we have αβ→γβ (augmentation rule), hence α→βγ (according to transitivity rule and set union commutativity).

c) For a relational scheme *R(A, B, C, D, E)* and its associated FDs given below:

F = {*A→BC, CD→E, B→D, E→A*}

(i) Please find out all the candidate keys for *R*. **(8 marks)**

Answer:

The candidate keys are:

A
BC
CD
E

(ii) Compute the closure set for *B* (i.e., B^+). **(2 marks)**

Answer:

The closure set of *B*, ie, $B^+ = \{B, D\}$.

__Problem Three__: **File Structure and Indexing** (25 points)

1. Consider a disk with block size B = 1024 bytes. A block pointer is P = 7 bytes long. A file has *r* = 10,000 STUDENT records of fixed length and the file blocks are linked together using block pointers. Each record has the following fields: SName (13 bytes), SID (8 bytes), Department (4 bytes), Address (47 bytes), Phone (8 bytes), Birthdate (8 bytes), Sex (1 byte), CourseId (6 bytes), SemesterId (3 bytes), SectionId (3 bytes). An additional byte is used as a deletion marker.

a) Calculate record size R in bytes. **(4 points)**

13+8+4+47+8+8+1+6+3+3+1=102 bytes.

Marking scheme: close to the method of calculation get 2 points, method totally wrong will get no points.

b) How many records can one block contain? **(4 points)**

(1024-7)/102=9

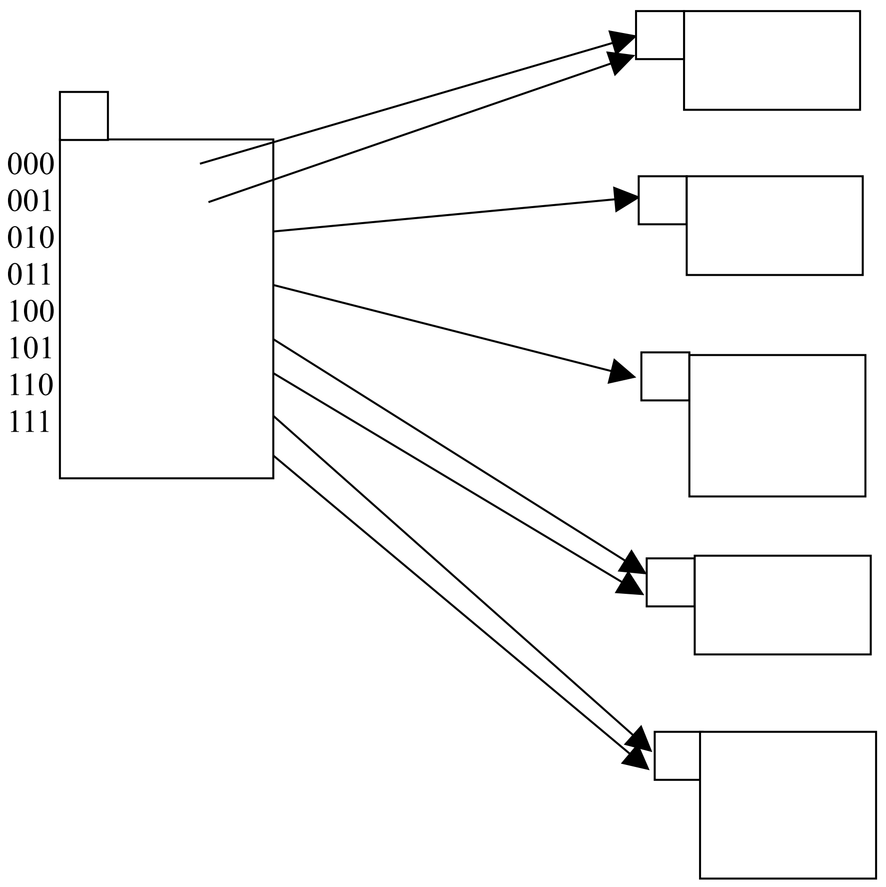
Marking scheme: Didn't consider the size of pointer will only get 2 points.

2. Suppose that we are using extendible hashing on a file containing records with the following search-key (hash key) values:

2, 3, 5, 7, 11, 17, 19, 23, 29, 31.

a) Let the hash function be ***h(x) = x mod 8***. Show the extendible hash structure for this file assuming that each bucket can hold at most three records and the records are inserted into the file in the order shown above. **(8 points)**

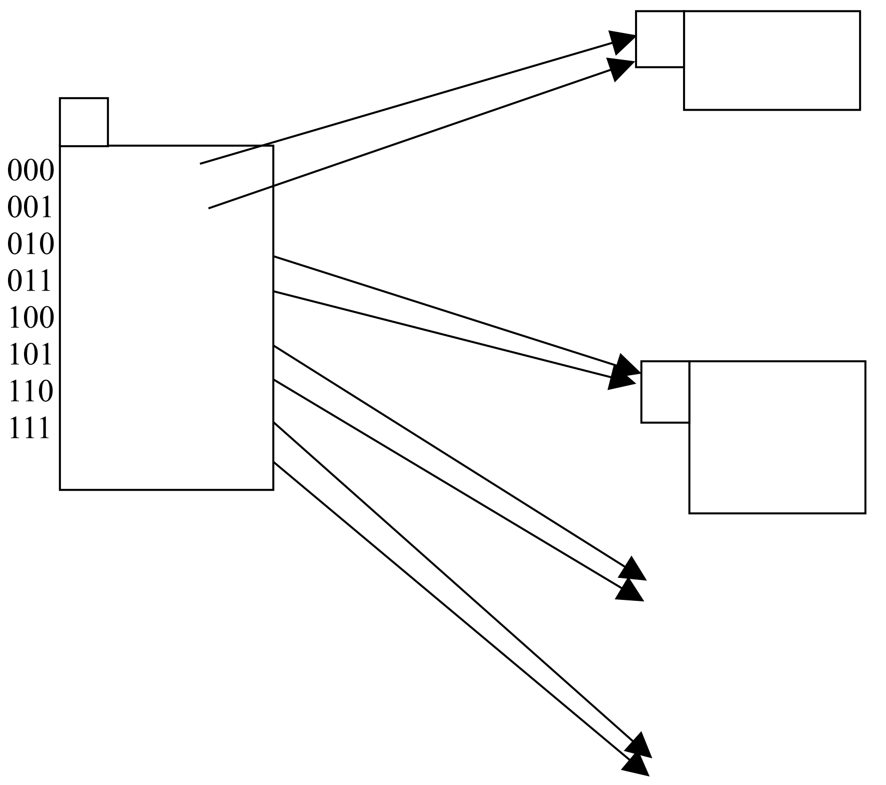
Answer:



b) Show how the above extendible hash structure changes as the result of the step below:

- Delete 3 **(4 points)**

Answer:

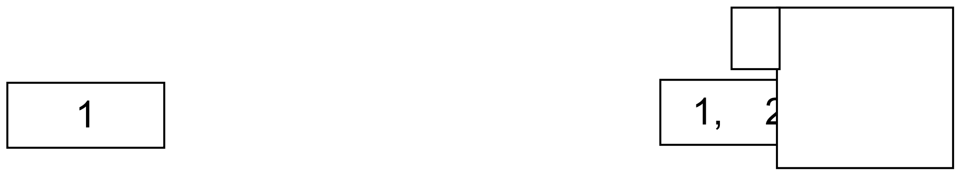


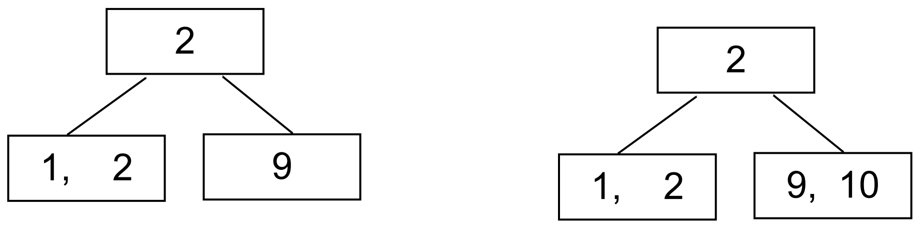
3. Insert the following numbers one by one in sequence into a B+ tree with order 3 and leaf order 2:

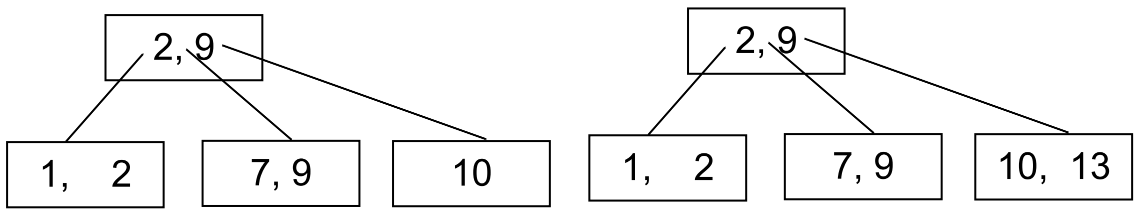
1, 2, 9, 10, 7, 13, 11.

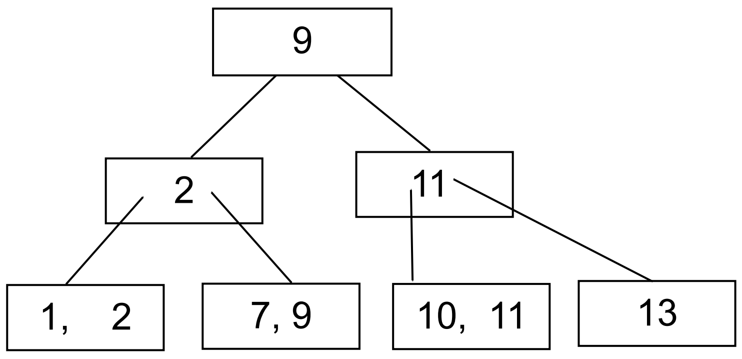
Show the picture after each insertion. **oints)**

Answer:









<u>Problem Four</u>: Conceptual Modeling and Database Design (25 points)

1)

(a) Why BCNF is called a "stronger" 3NF? (2 marks)

(b) Can a database be normalized into BCNF with or without preservation of functional dependencies? (2 marks)

(c) Justify your answers in (b) with example(s). (6 marks)

Answer:

1)

(a) A Boyce Codd Normal Form is also called a "Strong" Third Norm Form because it itself is already in the third normal form. Also, BCNF has with more constraints than 3NF by having all determinants in functional dependencies as candidate keys.

(b) The unnormalized relation can be normalized into BCNF with or without functional dependencies preservation.

(c) Case 1: An unnormalized relation can be normalized into BCNF with functional dependencies preservation.

Relation Enrolment (<u>S_ID</u>, <u>C_ID</u>, HKID)
FD: S_ID, C_ID → HKID
FD: HKID → S_ID

Relation Enrolment is not in BCNF because HKID is not a candidate key.

R1: (<u>HK_ID</u>, *S_ID)
R2: (<u>S_ID</u>, <u>C_ID</u>)

Relations R1 and R2 are in BCNF because all determinants (S_ID, C_ID) and HKID are candidate keys, and Relation Enrolment can be recovered by natural join of R1 and R2

Case 2: An unnormalized relation can be normalized into BCNF without preservation of functional dependencies.

For example, given the relational relation with its functional dependencies as shown below:

Relation Appointment (StaffNo, dentistName, patNo, patName, appDate, appTime, surgeryNo)

FD: staffNo, appDate, appTime → patNo, patName
FD: staffNo → dentistName
FD: patNo → patName, surgeryNo
FD: staffNo, appDate → surgeryNo
FD: patNo, appDate, appTime → staffNo, dentistName

Relation Appointment is not in BCNF because StaffNo is a determinant but not a candidate key, so decompose R to R1 and R2 using the 2nd functional dependency.

R1 = (staffNo, dentistName)
R2 = (staffNo, patNo, patName, appDate, appTime, surgeryNo)

R2 is not in BCNF because patNo is a determinant but not a candidate key, so decompose R2 to R3 and R4 using the 3rd functional dependency

R3 = (patNo, patName, surgeryNo)
R4 = (staffNo, patNo, appDate, appTime)

Results: BCNF: R1 = (<u>staffNo</u>, dentistName)
R3 = (<u>patNo</u>, patName, surgeryNo)
R4 = (<u>staffNo</u>, patNo, <u>appDate</u>, <u>appTime</u>)

However, R1, R3 and R4 cannot preserve FD: staffNo, appDate → surgeryNo

2)

You are required to design an Entity-Relationship Model and its relational schema for an airline reservation system with the following requirements:
The system provides seat reservation inquiry and booking of a passenger's flight. A customer can book many flights, and a flight can be booked by many customers. A customer can be a regular customer with regular price or an employee customer with a discount price. Customers will be able to reserve a seat for a flight. He/she can choose whether to take first or ordinary class flight. The customer can change or cancel a reservation. The business requirements are:

- Each customer will be able to inquire any available seat.
- Those who had bought airline tickets are not allowed to cancel or change their bookings.
- A customer can change his/her reservation provided that the date of his/her change is at least 7 days before the flight.
- A customer whose demand cannot be met will be put in a waiting list.
- To reserve a seat, the user must specify: flight number, customer name, passport number, departure date and time, customer sex, flight class and destination of the flight.

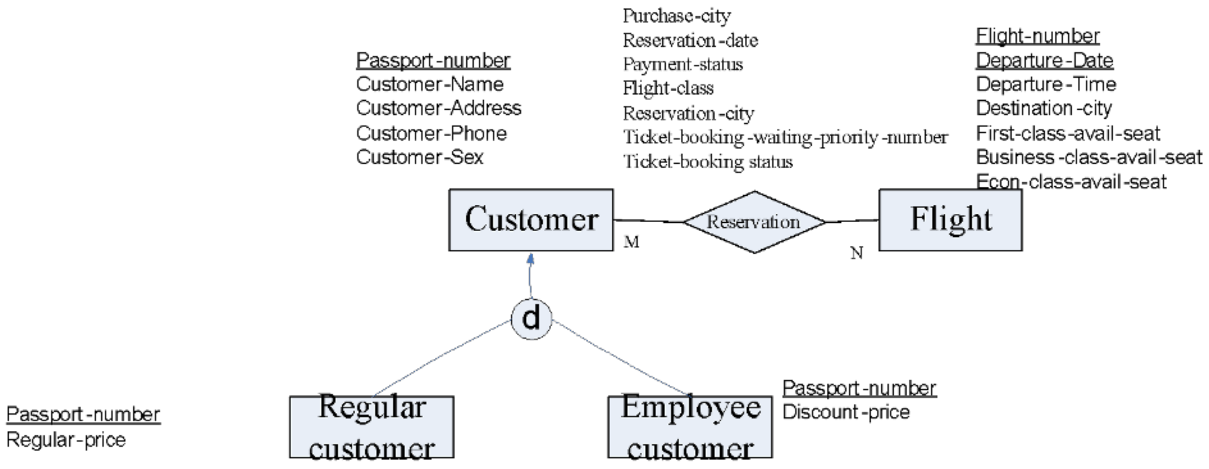
The data requirement of the airline reservation systems are:

Data	Description
Customer-Address	Customer address
Customer-Name	Customer name
Customer-Phone	Customer Phone number
Passport-number	Customer Passport number
Customer-Sex	Customer sex (M/F)
Departure-City	Departure City name
Departure-Date	Departure Date (DD/MM/YYYY)
Departure-Time	Departure Time (HH:MM)
Destination-City	Destination City name
Economic-Class-Available-Seat	Available economic seats counter
Business-Class-Available-Seat	Available business seats counter
First-Class-Available-Seat	Available first class seats counter
Flight-Number	Flight number
Flight-Class	Flight class (E=econ/B=business/F=first class)
Regular price	Non-employee price
Discount price	Employee Discount price
Payment-Status	Payment status (P=paid/U=unpaid)
Purchase-City	City where customer purchases airline ticket
Reservation-City	City where customer reserves flight seat
Reservation-Date	Date when customer reserves flight seat
Reservation-Status	Reservation status (R=reserved/U=unreserved)
Ticket-Booking-Waiting-Priority-Number	Waiting priority number for an airline ticket
Ticket-Booking-Status	Ticket booking status (C=cancel, B=booked)

(a) Extended Entity Relationship Model (10 marks)
(b) Relational Schema (5 marks)

2)

(a) The ER model for the airline reservation system is:



(10 marks)

(b) The relational schema for the airline reservation system is:

Relation Customer (<u>Passport-number</u>, Customer-name, Customer-Phone, Customer-Address, Customer-Sex)

Relation Employee-Customer (*<u>Passport-number</u>, Employee-discount-price)

Relation Regular-Customer (*<u>Passport-number</u>, Regular-price)

Relation Flight (<u>Flight-number</u>, <u>Departure-date</u>, Departure-Time, Departure-city, Destination-city, First-class-avail-seat, Business-class-avail-seat, Econ-class-avail-seat)

Relation Reservation (*<u>Passport-number</u>, *<u>Flight-number</u>, *<u>Departure-date</u>, Purchase-city, Reservation-date, Payment-status, Flight-class, Reservation-city, Ticket-booking-waiting-priority-number)

Where underlined are record keys, and "*" prefixed are foreign keys.
Employee-Customer.Passport-number ∩ Regular-Customer.Passport-number = 0 **(5 marks)**

<u>**Problem Five**</u>**: Query Optimization** (25 points)

1) Given query $\sigma_{DNO=2 \text{ AND } SALARY>50000 \text{ AND } SEX='FEMALE'}(EMPLOYEE)$, suppose that there is a clustering index on DNO and a B+ tree index on SALARY. Further assume that there are altogether 3 departments and only 20% of the employees have salary higher than 50000. Explain the best way to do the above select operation. **(4 points)**

Answer: It is a conjunctive select. We should use the condition with the smallest selectivity first and then check the remaining conditions. Hence, in this problem, we should use the B+ tree index to take all the employees who earn higher than 50000 and then for each of them check whether the other two conditions are correct or not.

2)
a) In what situation can we use the following rule when optimizing queries? Please also explain why we use this rule. **(6 points)**

$$\pi_L(R \bowtie_c S) = (\pi_{A1, \ldots, An}(R)) \bowtie_c (\pi_{B1, \ldots, Bm}(S))$$

Answer: When L={A1,…,An,B1,…Bm} and condition c only involves these attributes, we can use this rule. (3pt)
The reason we use it is to reduce the size of tables participating in the join operation, which will save the execution time. (3pt)

b) In the case when we cannot use the above rule, how should we modify it so that a similar commute rule can be used?
(4 points)

Answer:
When condition c involves attributes outside of L, then the above rule cannot be used (2pt).
In that case, we need to use the following rule:

$$\pi_L(R \bowtie_c S) = (\pi_{A1, \ldots, An, An+1, \ldots, An+r}(R)) \bowtie_c (\pi_{B1, \ldots, Bm, Bm+1, \ldots, Bm+t}(S))$$

where the new attributes are attributes in c but not in L (2pt) .

c) Explain why the following rule is not correct: **(4 points)**

$$\pi_L(R \cap S) = (\pi_L(R)) \cap (\pi_L(S))$$

Answer: For example, if R contains female employees and S contains male employees. L is ENAME. Then the left hand side contains nothing while the right hand side may contain some element if a male employee and a female employee have the same name.

3) Consider the relations:

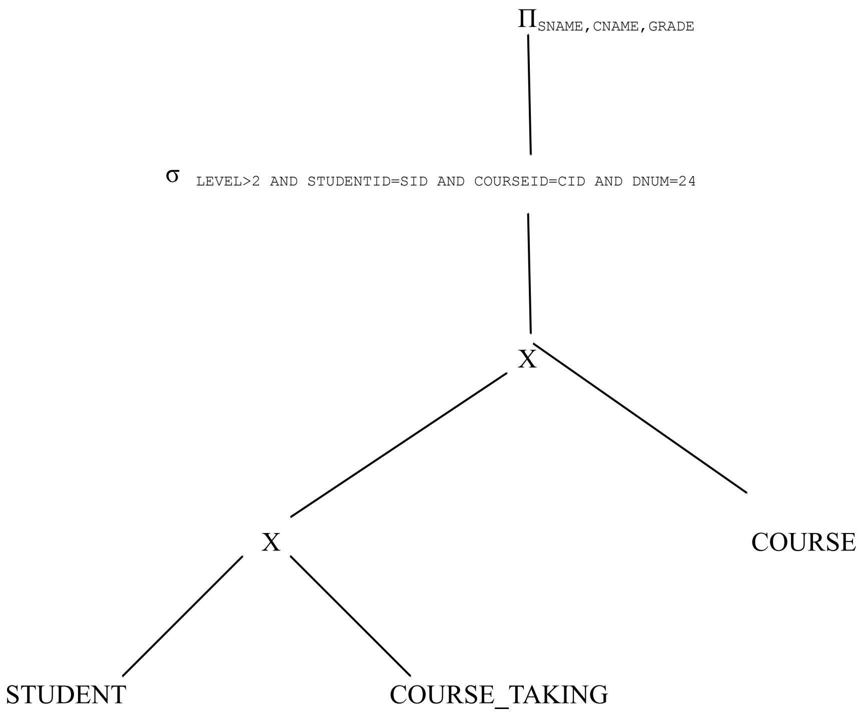
STUDENT(SNAME, SID, BDATE, ADDRESS, DNUM)
COURSE(CNAME, CID, LEVEL, LECTURER)
COURSE_TAKING(STUDENTID, COURSEID, GRADE)

as well as the following SQL query:

```
SELECT      SNAME, CNAME, GRADE
FROM STUDENT, COURSE_TAKING, COURSE
WHERE LEVEL>2 AND STUDENTID=SID
      AND COURSEID=CID AND DNUM = 24;
```

a) Draw a canonical query tree for the above SQL query. **(2 points)**

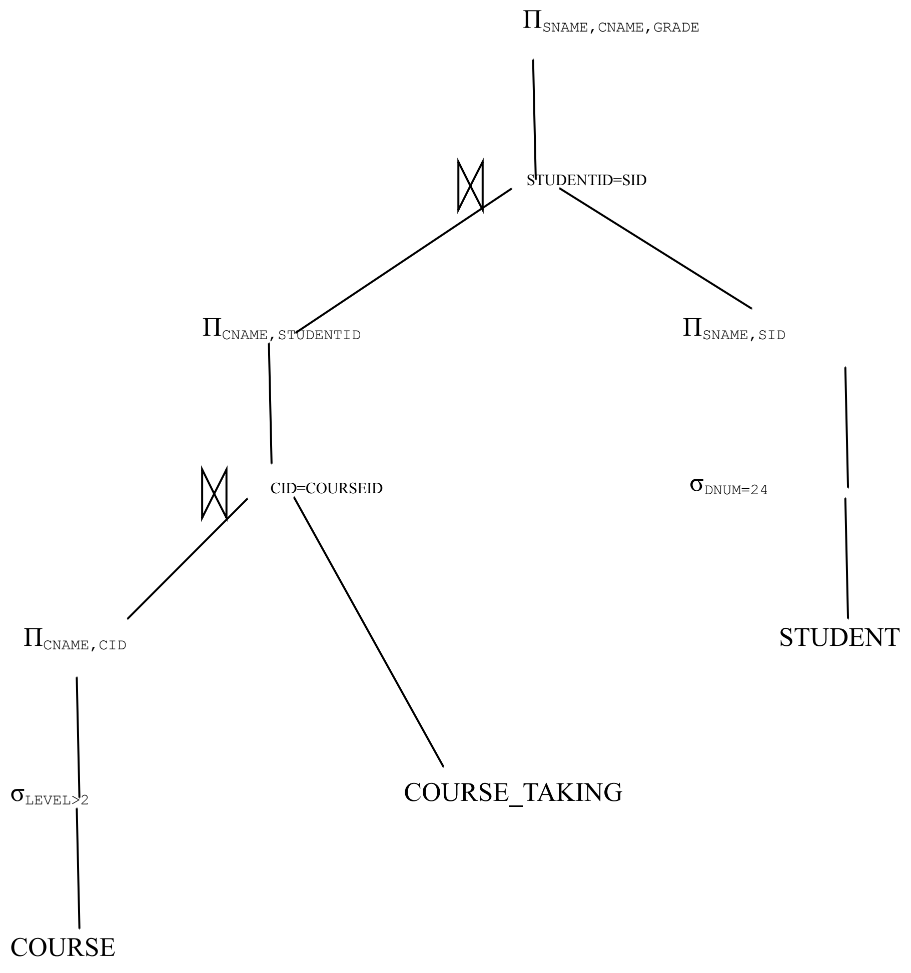
Answer:



Marking scheme: partially correct gets 1 point.

b) Apply the optimization rules to the above query tree and come up with the most optimized query tree using those rules. State the necessary assumptions for your decision. **(5 points)**

Assumptions: There are less courses with level larger than 2 compared to the number of students in department 24.



Marking scheme: One optimization step missing will get 1 point deducted and some typo or missing symbols will also cause 1-2 points deducted.

<u>Problem Six</u>: Transaction Processing and Concurrency Control (25 points)

1)

(a) What is the lock compatibity table? (2 marks)

(b) How to evaluate serializability of concurrent transactions A and B? (2 marks)

(c) Can share and exclusive lock guarantee serializability in concurrency control? (2 marks)

(d) Justify your answer with examples. (6 marks)

Answers:

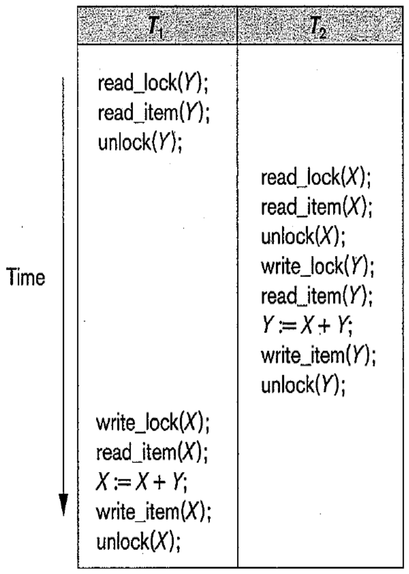
(a) Lock compatibility table is as follows: where Y means compatible, and N means not compatible, X-lock means exclusive lock, U-lock means update lock and S-lock means share-lock.

	X-lock (X)	U-lock (U)	S-lock (S)	No-lock (__)
X	N (not compatible)	N	N	Y (compatible)
U	N	N	Y	Y
S	N	Y	Y	Y
___	Y	Y	Y	Y

(b) The serialiability of concurrent transactions A and B can be computed as:
If A_and_B = A_then_B or B_then_A, then A and B are in serializability.

(c) Lock compatible table cannot guarantee serializability in concurrency control.

(d) For example, in the following two concurrent transaction T1 and T2
Initial value x=20 y=30, T1-then-T2 X=50 Y=80,
T2-then-T1 x=70 Y=50, T1-and-T2 X=50 Y=50, non-serializble



T_1	T_2
read_lock(Y);	
read_item(Y);	
unlock(Y);	
	read_lock(X);
	read_item(X);
	unlock(X);
	write_lock(Y);
	read_item(Y);
	$Y := X + Y$;
	write_item(Y);
	unlock(Y);
write_lock(X);	
read_item(X);	
$X := X + Y$;	
write_item(X);	
unlock(X);	

2)

Assume a system having an incremental log on transactions A, B, C and D on records X, Y, Z, N and P with immediate updates has the following log entries, ending with a system crash.

```
<A, starts>
<A, X, 1, 5>
<B, starts>
<B, Z, 8, 12>
<B, commits>
<B, starts>
<A, Y, 3, 4>
<check point record>
<D, N, 2, 5>
<D, commits>
<C starts>
<C, P, 5, 6>
---- System Crash ----
```

(a) Under what situations and using what kinds of method for you to perform database recovery. (3 marks)

(b) Which transactions, if any, need to be redone from where with "Redo" operation? Which transactions, if any, need to be undone to where with "Undo" operation? Which transactions, if any, are not affected by the crash? (2 marks)

(c) Assuming all goes well on recovery, what gets written to the database file and what gets written to the log? (2 marks)

(d) Assuming the recovery fails, what gets written to the database file and what gets written to the log? (2 marks)

(e) If the system used an incremental log with deferred updates, which transactions, if any, need to be redone? Which transactions, if any, need to be undone? Which transactions, if any, are not affected by the crash before the real update? (2 marks)

(f) What are the benefits of deferred updates? What are the costs of too many checkpoints? (2 mark)

Answers:

(a) Roll Forward database recovery is needed whenever there is media failure such as disk I/O error in the existing database which makes it unworkable for the production system.

Roll Backward database recovery is needed whenever there is system down such as power failure which makes uncertainty to the state of the transactions after last checkpoint.
Roll Forward with Roll Back database recovery is needed whenever there is media error in the existing database and uncertainty of the state of incomplete transactions in the log file.

(b) Redo D from last check point to commit, undo C from system crash to last check point, B and A are unaffected.

(c) Log file:

```
<C, undo P>
<C, other updates>
<C, commits>
<D, redo N>
<D , N, 2, 5>
<D, commits>
```

database file:
Transactions A, B, and D updated

(d) Log file:
<C undo P>
<D, redo N>
:
: repeat undo and redo without success unit
database file:
Transactions B and A updated

(e) redo D, not need to undo any, B and A are unaffected

(f) Deferred update can improve performance by updating to a temporary file.
Too many checkpoints will decrease transaction processing performance.

--- END ---