UNIVERSITY OF LONDON
IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 2004

BEng Honours Degree in Computing Part III
MEng Honours Degree in Information Systems Engineering Part IV
BSc Honours Degree in Mathematics and Computer Science Part III
MSc in Advanced Computing
for Internal Students of the Imperial College of Science, Technology and Medicine

This paper is also taken for the relevant examinations for the Associateship of the City and Guilds of London Institute
This paper is also taken for the relevant examinations for the Associateship of the Royal College of Science

PAPER C312=I4.4

ADVANCED DATABASES

Thursday 29 April 2004, 10:00
Duration: 120 minutes

Answer THREE questions

Paper contains 4 questions
Calculators not required

Section A (Use a separate answer book for this Section)

1 a Describe the block-based nested loop join using pseudocode.

b If we have two relations R and S with cardinalities of 100 and 50 respectively with tuple sizes of 1000 bytes for both, block sizes of 100 bytes, and memory amounting to 101 blocks, calculate the total number of disk IOs for the block-based nested loop join.

c If the role of R and S in the algorithm were reversed, but all the parameters remained the same as in (ii), recalculate the number of disk IOs required. Why is there a difference?

d If an index was placed on relation R and we used the tuple-based nested loop join, estimate how the performance would change.

e Describe how would the performance change if we added a further relation Q, of cardinality 10 and tuple size 100, into the tuple-based nested loop join?

The five parts carry, respectively, 20%, 20%, 20%, 15%, and 25% of the marks.

2 a Briefly describe the role of the Buffer Manager in the context of the DBMS engine architecture.

b A DBMS uses a buffer management system. If a tuple is in the memory it takes $20\mu s$ to access it. If it is in the buffer and not in memory it takes $60\mu s$ to load it into the memory. If the tuple is not in the buffer, 12ms are required to fetch the tuple from disk followed by $60\mu s$ to copy it to memory. The memory hit-rate is 0.7, the buffer hit-rate is 0.6. What is the average time in μs required to access a tuple on this system? (Explain your answer)

c

```
BEGIN TRANSACTION Booking
 UPDATE seats
 SET seat='reserved'
 WHERE seat_id=100;

 UPDATE credit_card
 SET bal=bal-ticket_price
 WHERE CCowner_id=500;
COMMIT TRANSACTION Booking
```

```
BEGIN TRANSACTION Cancel_booking
 UPDATE credit_card
 SET bal=bal+ticket_price-5
 WHERE CCowner_id=500;

 UPDATE seats
 SET seat='unreserved'
 WHERE seat_id=100;
COMMIT TRANSACTION Cancel_booking
```

i) Using histories notation, show how the both of the transactions above could be ordered so that they are deadlocked. Show the waits-for graph for this.

ii) Using histories notation, how would you order the transactions to ensure there was no deadlock.

d If we were running the transactions above and they deadlocked what would happen to the data held in the buffer of the transaction that was rolled back?

The four parts carry, respectively, 25%, 25%, 30%, and 20% of the marks.

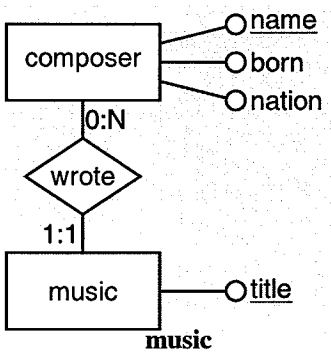
Section B (Use a separate answer book for this Section)

3 a Two export database schemas **music** and **art** are described below, which you should integrate into a single global ER model. Apart from the final ER model, your answer should clearly enumerate the transformations applied during the integration, using notation such as:

addEntity(⟪special⟫, [⟨x⟩ | ⟨x, y⟩ ← ⟪produced,work,artist⟫; y = 'nemo']

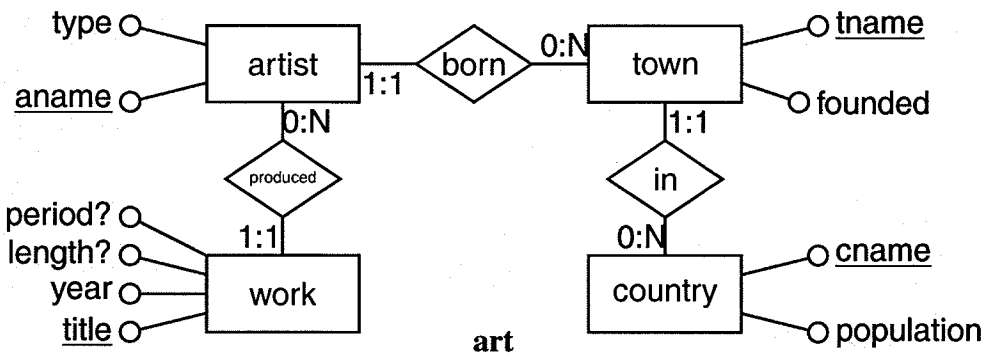
Your answer need *not* include the various extendEntity, extendAttribute, *etc* steps required at the end of schema conforming.

The **music** database contains details of major classical music works and the person who composed them. Each composer has a record of their date of birth and nation where their place of birth is currently located.



music

The **art** database contains details of major artists divided by type into painters and composers, where the composers are found to be the same set as is found in the **music** database, and the same pieces of music are recorded in **art** as are recorded in **music**. For pieces of music only, a record is made of the length of the work. Any work might be associated with a period such as *romantic* or *modern*. Each artist is associated with the **town** where they were born, and each **town** is associated with the country that the town is currently within.



art

b The XML data file shown below records which departments are located in certain buildings, by repeating under each building all the information held about a department. An XML element schema is also shown, with a few completed data values.

```
⟨campus⟩
  ⟨building bid="100" bname="Huxley"⟩
    ⟨dept did="1" dname="computing"/⟩
    ⟨dept did="2" dname="maths"/⟩
    ⟨dept did="3" dname="physics"/⟩
  ⟨/building⟩
  ⟨building bid="101" bname="Blackett"⟩
    ⟨dept did="3" dname="physics"/⟩
  ⟨/building⟩
  ⟨building bid="102" bname="Penny"⟩
    ⟨dept did="1" dname="computing"/⟩
  ⟨/building⟩
⟨/campus⟩
```

campus		
id	pid	ord
1	0	0

building				
id	pid	ord	bid	bname
2	1	1	100	Huxley
⋮				

dept				
id	pid	ord	did	dname
⋮				

i) Copy and complete the data values in the XML element schema for the XML data file, using any reasonable numbering scheme for the node ids.

ii) Design a relational schema in 3NF for the data, which ignores ordering information, and makes reasonable assumptions as to what might be the key value of each a relation. Show primary key definitions by underlining attributes, and foreign key definitions using the $\rightarrow$ notation.

iii) For each of your tables, write a relational algebra definition for the content of each table in terms of the tables in the element schema.

iv) Suggest a restructuring of the XML file, such that it avoids the redundancy it currently has, but keeps the current structure as far as possible. Sketch the new XML file, and outline how you would ensure it maintains its integrity using XML Schema key and keyref statements.

```
⟨xsd:key name="key_name"⟩
  ⟨xsd:selector xpath="xpath_expression"/⟩
  ⟨xsd:field xpath="xpath_expression"/⟩
⟨/xsd:key⟩
⟨xsd:keyref name="keyref_name" refer="key_name"⟩
  ⟨xsd:selector xpath="xpath_expression"/⟩
  ⟨xsd:field xpath="xpath_expression"/⟩
⟨/xsd:keyref⟩
```

v) What XPath significant aspect of the XML file does the element schema fail to represent? Can you suggest how that aspect could be represented by additions to the element schema?

The two parts carry equal marks

4a Convert the following relational database schema into an ER model. In the relational schema, primary key attributes are underlined, and foreign keys of a relation are listed after the relation definition. You should make and state any reasonable assumption(s) necessary during the conversion process.

person(pid, name, address)
vehicle(vid, built, owner_pid)
vehicle.owner_pid → person.pid
road(vid, driver_pid, reg_no)
road.vid → vehicle.vid
road.driver_pid → person.pid
air(vid, seats, range)
air.vid → vehicle.vid
flies_in(vid, pid)
flies_in.vid → air.vid
flies_in.pid → person.pid
car(vid, doors, cc)
car.vid → road.vid

b In a database replicated over eight servers, it is found that on average there are 1000 writes per hour, and 800 reads per hour. The following alternative distributed locking options are available, described in terms of the number of servers read and write locks are sent to:

Option	Read locks sent to	Write locks sent to
A	8	1
B	5	4
C	4	5
D	1	8

i) Briefly explain which (if any) of the options are invalid, in the sense of failing to correctly detect some conflicts.

ii) For those which you think are valid, calculate the total number of locks required to be made across the database per hour, and hence state which option will give the best performance.

iii) If the system needed to be one fault tolerant, how would you adjust the option of (ii) to still work correctly even with one failed server in the system?

c The following shows a temporal database, represented in the temporal structure, for the valid time records of a league table, recording positions of county cricket teams in an annual competition.

league	
team	pos
surrey	1
essex	2
sussex	3
somerset	4

$t = 0$

league	
team	pos
surrey	1
hants	2
sussex	3
essex	4

$t = 1$

league	
team	pos
hants	1
somerset	2
sussex	3
surrey	4

$t = 2$

league	
team	pos
somerset	1
sussex	2
durham	3
hants	4

$t = 3$

For each of the following three temporal relation algebra queries, list the output relation, and suggest what the query is intended to do (*e.g.* 'Finds all teams who have left the league').

i) at $t = 4$: $\blacksquare \pi_{\text{team}}$ league

ii) at $t = 3$: $(\pi_{\text{league1.team}}\ \text{league1}) \overset{\text{S}}{\times} \sigma_{\text{league2.team='essex'}}\ \text{league2}$

iii) at $t = 4$:
$(\blacklozenge \pi_{\text{league1.team,league2.team}}\ \sigma_{\text{league1.pos}>\text{league2.pos}}(\text{league1} \times \text{league2})) -$
$(\blacklozenge \pi_{\text{league1.team,league2.team}}\ \sigma_{\text{league1.pos}\leq\text{league2.pos}}(\text{league1} \times \text{league2}))$

iv) Write a TRA query to list the teams that have never won the league

Note that $\blacklozenge$ is the *sometime in the past* operator, $\blacksquare$ is the *always in the past* operator, $\bullet$ is the *previous time* operator, and $\overset{\text{S}}{\times}$ is the *since product* operator. Also note that league1 and league2 are aliases for the league table.

The three parts carry, respectively, 35%, 20%, and 45% of the marks.