The University Of New South Wales

Solutions for Final Exam

June 2000

COMP9311/3311

Database Systems

Time allowed: **3 hours**
Total number of questions: **13**
Total number of marks: **100**

Textbooks, lecture notes, etc. are **not** permitted.
Calculators may **not** be used.
Questions are **not** worth equal marks.
Answer **all** questions.
You can answer the questions in any order.
Start each question on a **new page**.
You may **not** take this question paper out of the exam.

Questions 1-10 are multiple-choice and are worth a total of *(20 marks)*. Answer each question **on this exam paper** by circling just one letter corresponding to the correct choice. A correct answer is worth 2 marks. An incorrect answer is worth -1 marks. Putting no answer is worth 0 marks. Note: each question has exactly one correct answer; circling multiple letters counts as an incorrect answer.

Question 1

In data modelling, a *key* is

a) an index to access tuples in a relation

b) a set of attributes that sorts a relation

c) **Answer:** a minimal set of attributes that uniquely identifies an entity

d) a set of entities and relations that defines an enterprise

Question 2

Consider the following typed relational schema:

```
Student(id:integer, name:string, address:string, degree:string)
Subject(id:string, title:string, syllabus:string)
Enrolled(studentID:integer, subjectID:string, grade:real)
```

Which of the following SQL queries will return the names of all students who are enrolled in the subject COMP1001?

a)
```
select Student.name from Student,Subject,Enrolled
 where Student.id=Subject.id and Subject.title='COMP1001'
```

b) **Answer:**
```
select Student.name from Student,Enrolled
 where Student.id=Enrolled.studentID and Enrolled.subjectID='COMP1001'
```

c)
```
select Student.name from Student,Subject,Enrolled
 where Subject.id='COMP1001'
```

d)
```
select Student.name from Student,Enrolled
 where Enrolled.subjectID='COMP1001' and NOT NULL Student.id
```

Question 3

Using the schema from question 2, what is represented by the following relational algebra expression: $\pi_{name}(\sigma_{(degree=\texttt{'MCompSci'} \wedge subjectID=\texttt{'COMP9311'})}(Student \bowtie_{id=StudentID} Enrolled))$

a) The names of all students who are enrolled in COMP9311.

b) The names of all students who are either MCompSci students or are enrolled in COMP9311.

c) **Answer:** The names of all MCompSci students who are enrolled in COMP9311.

d) The names of all MCompSci students who are not enrolled in COMP9311.

Question 4

What is the result of the operation A/B for the relations:

A =

x	y	z
2	3	4
2	3	5
3	4	5
4	2	5
3	4	6
4	3	6
4	3	7

B =

z
5
6

a)

x	y
2	3
3	4
4	2
4	3

b)

x	y
2	3
4	2
4	3

c) **Answer:**

x	y
3	4

d)

x	y
2	3
4	3

Question 5

For the schema in question 2, the `select` statement in the PL/SQL block:

```
declare
    mark REAL;
begin
    select grade into mark from Enrolled;
    ...
end
```

will most likely fail because:

a) the keyword `declare` is not written in upper-case letters

b) **Answer:** it will attempt to store multiple grade values into the `mark` variable

c) you cannot extract a value from a table directly into a PL/SQL variable

d) the query will not fail; it will succeed even if there are 1000 enrolment records

Question 6

An `IN OUT` parameter for a PL/SQL function or procedure

a) cannot be modified by the procedure/function that receives it

b) is used exclusively for performing input/output to the console

c) **Answer:** will be used to return a result but also has a useful initial value

d) is only useful to pass `CURSOR`s from one procedure/function to another

Question 7

Which of the following file organisations for the `Student` relation from question 2, does *not* provide any assistance in answering the query `select * from Student order by Student.id`?

a) a heap file sorted on the `id` field

b) a file with a B-tree index on `Student.id`

c) **Answer:** a hashed file where the hash key is `Student.id`

d) a file with a dense primary index on `Student.id`

Question 8

For a hashed file containing $B = 1000$ data pages, the best-case cost of performing an equality search for a specific hash key value is

a) **Answer:** 1 page

b) 2 pages

c) $log_2 B$ pages

d) B pages

Question 9

A relation $R(ABCDE)$, with functional dependencies $AB \rightarrow C,\ C \rightarrow E,\ D \rightarrow C$ has candidate keys:

a) AB only

b) **Answer:** ABD only

c) ABC, ABD, D

d) AB, ABC, ABD, BCD

Question 10

A database transaction is guaranteed to be *atomic* when

a) it requires multiple updates

b) it performs only `INSERT` operations

c) it is executed by a single database process

d) **Answer:** it performs all of its update operations, or none of them

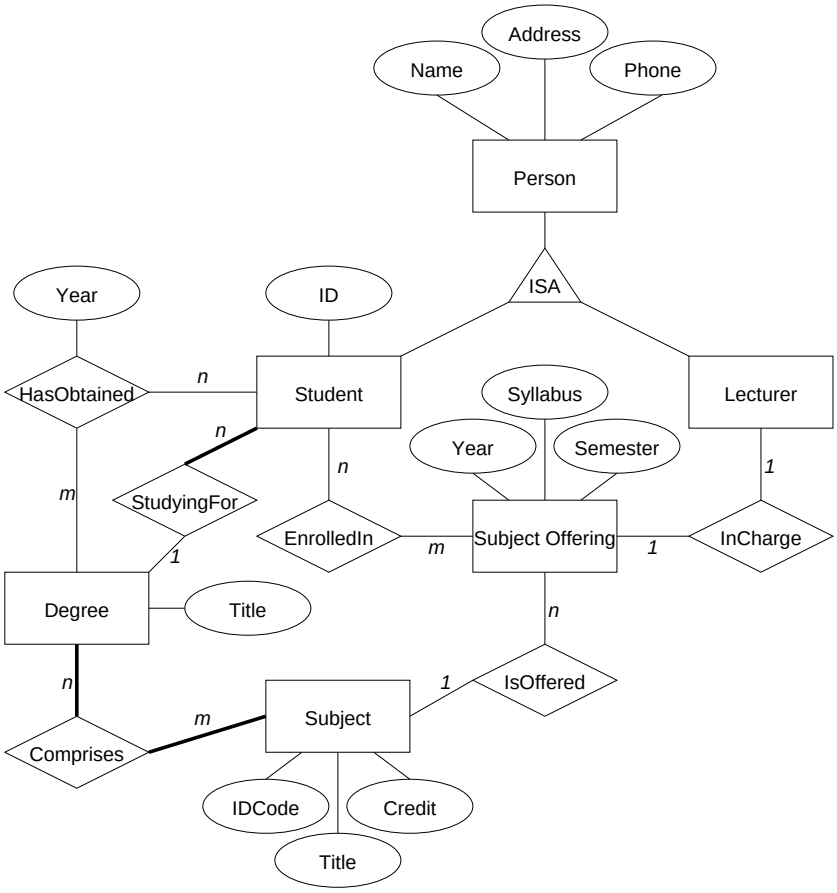
Question 11

(20 marks total) Draw an ER diagram to model the following scenario:

- the College of Old South Wales offers a number of degrees
- each degree has a title, a major, and a duration (in years)
- a degree is composed of a sequence of individual subjects
- each subject has an id code, a title, and a credit point value
- each time a subject is offered, it also has a lecturer and a syllabus
- the syllabus may vary slightly from offering to offering
- every subject offering occurs in a particular year and semester
- every person associated with the college has a name, an address and a phone number
- each student has a unique identifying number (student id)
- past students may have acquired several degrees
- past students obtained their degree in a specific year
- each current student is enrolled in exactly one degree
- every current student is either enrolled in subjects or on leave
- students enrol in exactly four subjects each semester
- each lecturer has a staff number and an office
- lecturers may be involved in several subjects each semester, or may be on leave
- lecturers will be In Charge of at most one subject each semester

Your diagram *must* show cardinalities and participation constraints. You must also state any assumptions you make that are not mentioned in the list above.

Suggested solution:



The above solution is a minimal solution, and ignores some subtle aspects of the question; nevertheless, it would get full marks.

Marking notes:

Award marks for the following features (the names don't have to be the same as mine):

- must have `Person`, `Student`, `Lecturer`, `Subject`, `Offering`, `Degree` entities with appropriate attributes (2 marks each)
- must have subclass relationship between `Student`, `Lecturer` and `Person` (1 mark)
- must have n:m `EnrolledIn` relationship between `Student` and `Offering` (1 marks)
- must have 1:1 `InCharge` relationship between `Lecturer` and `Offering` (1 marks)
- must have n:m `Comprises` relationship between `Subject` and `Degree`, with participation constraints on both branches (2 marks)
- must have n:1 `StudyingFor` relationship between `Student` and `Degree`, with participation constraint on `Student` (2 marks)
- must have n:m `HasObtained` relationship between `Student` and `Degree` (1 marks)

- if the `Subject` and `Offering` entities are combined, mark as correct, except deduct 4 marks from the total
- if they have done something different, but justified it with a plausible assumption, don't deduct marks

Other valid variations on this might include:

- subclassing students into `Current` and `Past` students
- subclassing `Current` students into `Studying` and `OnLeave` (or using an attribute to indicate this)
- subclassing `Lecturers` into `Teaching` and `OnLeave` (or using an attribute to indicate this)
- using double-line notation to indicate participation
- using stars or arrows to indicate relationship arity
- using arcs on lines to indicate subclasses

Question 12

(40 marks total) Consider the following relational schema describing musicians, bands, instruments, songs and albums for popular music:

```
create table Musician (
    name       varchar(30) primary key,
    memberOf   integer,
    age        integer,
               foreign key (memberOf) references Band(id)
);
create table Band (
    id         integer     primary key,
    name       varchar(50),
    website    varchar(50)
);
create table Song (
    catNo      integer     primary key,
    title      varchar(50),
    duration   real
);
create table Album (
    serialNo  integer     primary key,
    title      varchar(50),
    band       integer,
    producer   varchar(30),
    year       integer,
               foreign key (band) references Band(id)
);
create table Performs (
    musician   varchar(30),
    song       integer,
    instrument varchar(20),
               primary key (musician,song),
               foreign key (musician) references Musician(name),
               foreign key (song) references Song(catNo)
);
create table AppearsOn (
    song       integer,
    album      integer,
    trackNo   integer,
               primary key (song,album),
               foreign key (song) references Song(catNo),
               foreign key (album) references Album(serialNo)
);
```

(continued over page)

You may assume that

- each musician only ever plays with one band
- an album is a CD containing a number of songs performed by one band
- there is only one performance of each song, but it can appear on many albums
- the `trackNo` field indicates where a song appears on the album (1^{st}, 2^{nd}, 3^{rd}, ...)
- there is a band called "The Beatles"
- Paul McCartney is a musician who is a member of "The Beatles"
- there is precisely one answer for each query that asks oldest/longest/etc.

a) Write SQL queries to answer the following:

i) *(4 marks)* Who is the oldest musician and how old are they?

Suggested solution:

```
select name,age
from   Musician
where  age = (select max(age) from Musician);
```

Marking notes:

- mark using the scheme for marking programs

ii) *(4 marks)* What different instruments does Paul McCartney play on albums by The Beatles?

Suggested solution:

```
select distinct instrument
from   Album,Band,Performs,AppearsOn
where  Band.name = 'The Beatles' and Band.id = Album.band
       and AppearsOn.album = Album.serialNo
       and AppearOn.song = Performs.song
       and Performs.musician = 'Paul McCartney'
```

iii) *(5 marks)* What is the longest opening track (i.e. first song) on any album?

Suggested solution:

```
select title
from   Song
where  duration =
       (
           select max(Song.duration)
           from   Song,AppearsOn
           where  Song.catNo = AppearsOn.song
                  and AppearsOn.trackNo = 1
       )
```

Marking notes:

- mark using the scheme for marking programs

iv) *(7 marks)* Which musicians perform on every song that their band has released?

Suggested solution:

```
select name
from   Musician
where  not exists (
           (
               select catNo
               from   Song,AppearsOn,Album,Band,Performs
               where  Song.catNo = AppearsOn.song
                      and AppearsOn.album = Album.serialNo
                      and Album.band = Band.id
                      and Band.id = Musician.memberOf
                      and Performs.song = Song.catNo
                      and Performs.musician = Musician.name
           )
           minus
           (
               select catNo
               from   Song,AppearsOn,Album,Band
               where  Song.catNo = AppearsOn.song
                      and AppearsOn.album = Album.serialNo
                      and Album.band = Band.id
                      and Band.id = Musician.memberOf
           )
       )
```

Marking notes:

- mark using the scheme for marking programs

b) *(20 marks)* Implement a PL/SQL procedure that takes the name of an album and produces a list of tracks from that album, complete with information about who played what instrument on each track. This information should be presented in the following format:

Generic Format	Specific Example
Album Title `by` *Band Name*	`The White Album by The Beatles`

`1.` *Track Title*
Instrument: *Musician*
Instrument: *Musician*
`...`
	`1. Back in the U.S.S.R.`
	`Guitar: John Lennon`
	`Guitar: George Harrison`
	`Bass: Paul McCartney`
	`Drums: Ringo Starr`

`2.` *Track Title*
Instrument: *Musician*
Instrument: *Musician*
`...`
	`2. Dear Prudence`
	`Guitar: George Harrison`
	`Piano: John Lennon`

N`.` *Track Title*
Instrument: *Musician*
Instrument: *Musician*
`...`
	`3. Glass Onion`
	`Bass: Paul McCartney`
	`Guitar: George Harrison`
	`Drums: Ringo Starr`

State any assumptions that you make and please try to format your answers neatly.

Marking notes:

- mark using the scheme for marking programs
- the procedure can be called anything reasonable (related to track list)
- it doesn't matter if the `create or replace` is not present
- if they try to use nested loops, they'll get in a big mess
  - 10-14 marks for a plausible-looking nested-loops version
  - make sure that the nested queries are valid PL/SQL

Suggested solution:

```
create or replace
procedure trackList(album in varchar)
is
    band varchar(50);  -- or  band Band.name%type;
    currentTrack int;  -- current track as we scan track list

    -- Collect (track#,song,musician,instrument) tuples
    -- for all songs, arranged in order of track number
    cursor results is
        select AppearsOn.trackNo as trackID,
               Song.title as songTitle,
               Musician.name as playerName,
               Performs.instrument as instrumentPlayed
        from   Album, Song, AppearsOn, Musician, Performs
        where  Album.title = album
               and Album.serialNo = AppearsOn.album
               and AppearsOn.song = Song.catNo
               and AppearsOn.song = Performs.song
        order by AppearsOn.trackNo
        ;
begin
    -- Get the name of the band who made the album
    select Band.name into band
    from   Album, Band
    where  Album.title=album and Album.band=Band.id;

    -- Display album/band header
    dbms_output.put_line(album || ' by ' || band);

    -- Iterate over (track#,song,musician,instrument) tuples
    -- displaying information as specified in question 2
    currentTrack := 0;
    for res in results loop
        -- If it's the first entry for this track,
        -- print a blank line and then the track title
        if (res.trackID <> currentTrack) then
            currentTrack = res.trackID;
            dbms_output.put_line(res.trackID || ' ' || songTitle);
        end if;
        --Print the instrument and musician
        dbms_output.put('   ' || res.instrumentPlayed);
        dbms_output.put_line(': ' || res.playerName);
    end loop;
end;
```

Question 13

(20 marks total)

a) *(10 marks)* Consider the relation $R(ABCD)$. For each of the following sets of functional dependencies, determine which are the candidate keys for R and state, with reasons, what is the strongest normal form that R satisfies. Show all working.

Marking notes:

- each sub-part is worth 5 marks
- the method for determining candidate keys is
  - take a set of attributes XYZ that you think might be a candidate key
  - determine the closure $XYZ+$
  - if $XYZ+$ includes all attributes in the relation, then XYZ is a candidate key
- 2 marks for getting the candidate key(s)
- 1 marks for showing some (vaguely correct) working
- the method for determining normal form is
  - consider the NF definitions, the given candidate keys, the given FDs
  - check whether the keys/FDs violates some of the conditions
- 1 mark for getting the correct NF
- 1 mark for saying why it's not a higher NF

i) $C \rightarrow D,\ C \rightarrow A,\ B \rightarrow C$

Suggested solution:

```
Since  B->C and C->A and C->D  B+ = {ABCD}, so B is a candidate key.
Nothing else is a candidate key.

It's not BCNF, because the LHS of some FDs does not involve the
candidate key (e.g. C->A, C->D).
It's not 3NF because the LHS of some FDs does not involve the
candidate key AND the RHS is not part of a key either (e.g. C->D).
It is is 2NF (doesn't violate 2NF rules).
```

ii) $AB \rightarrow C,\ AB \rightarrow D,\ C \rightarrow A,\ D \rightarrow B$

Suggested solution:

```
AB is a candidate key, since AB+ = {ABCD}, trivially (i.e. direct from FDs)
CD is a candidate key, since CD+ = {ABCD}, trivially (i.e. direct from FDs)
Less obvious is BC, because C->A and then we have AB, etc.
Similary, AD is a candidate key, because D->B and then we have AB, etc.

It's not BCNF, because the LHS of some FDs does not involve a
candidate key (e.g. C->A)
It is in 3NF (doesn't violate 3NF rules).
```

b) *(5 marks)* Consider the relation `Student(id:integer, name:string, course:string)` where the *id* field is a primary key; there are 80,000 tuples in this relation, with 100 tuples per page; the tuples are stored in an unordered heap file; the file has a B-tree index on the *id* field; each node of the B-tree occupies a single page; the B-tree has an internal branching factor of 100 and tuple-ids are stored only in the leaf nodes.

Describe how the query `(select name from Student where id=2223333)` would be answered under this file organisation and compute an estimate of how many pages (both data and index pages) would be read in answering this query.

Marking notes:

- they should mention at least some of the points from below
- the answer should be structured as a chain of reasoning followed by a result
- if the reasoning is plausible, and the result is valid, 5 marks
- if they use the right reasoning, but get wrong result, 4 marks
- the reasoning is half-baked, but correct result, 3 marks
- if no useful/valid reasoning, but the correct answer, 2 marks
- if something vaguely relevant to the problem, 1 mark

Suggested solution:

```
The B-tree will have a root node, one level of internal nodes (10,000)
and then the leaf nodes.
The query traverses a path from root to leaf in the B-tree, which requires
us to read 3 nodes (index pages).
There is exactly one matching tuple, so only one data page would be read.
This gives a total of 4 page reads to answer the query.
```

c) *(5 marks)* Construct (and draw) the precedence graph for the following schedule for the execution of three transactions (T_1, T_2, T_3), and state whether the schedule is conflict serializable.

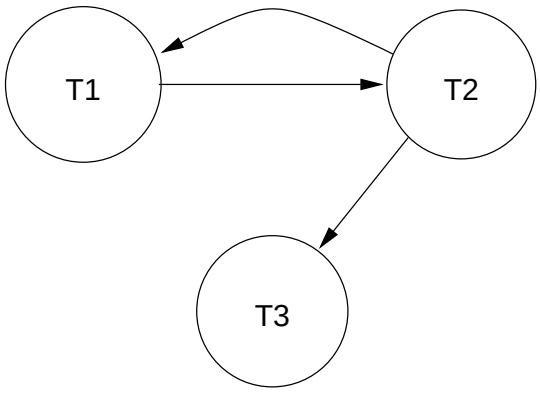
$Time$:	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
T_1:		$R(B)$				$W(B)$		
T_2:	$R(A)$		$W(A)$	$R(B)$				$W(B)$
T_3:					$R(A)$		$W(A)$	

Recall that $R(X)$ means that the transaction reads from the shared variable X and $W(Y)$ means that the transaction writes into the shared variable Y.

Marking notes:

- the method for solving this problem is to draw a precedence graph
  - each node represents a committed transaction T_n
  - there is a directed arc from T_i to T_j if
    - actions in T_i and T_j operate on a shared data item
    - an action of T_i precedes an action of T_j
    - at least one of the actions is a write
  - if the graph contains any cycles, then the schedule is not conflict serializable
- in the example
  - there is a conflict on A is between T_2 and T_3
  - all of T_2's actions on A precede T_3's actions on B
  - there is a conflict on B is between T_1 and T_2
  - some of T_1's actions on B, precede T_2's and others follow
- having a correct precedence graph (three nodes and three arcs), 4 marks
- stating the correct conclusion based on this, 1 mark

Suggested solution:



`The graph has a cycle and is therefore not conflict serializable.`