The University Of New South Wales

Sample Final Exam

Based on November 2002 COMP3311 Exam

COMP9311

Intro to Database Systems

Time allowed: **3 hours**
Total number of questions: **5**
Total number of marks: **115**
Textbooks, study notes, calculators, mobile phones, etc.
are **not** permitted in this exam.
All questions are worth equal marks.
Answer **all** questions.
You can answer the questions in any order.
Answer these questions in the script book provided.
Do **not** write your answer in this exam paper.
Start each questions on a new page.
If you use more than one script book,
fill in your details on the front of *each* book.
You may **not** take this question paper out of the exam.

Name:

Student#:

Question 1

Answer the following multiple-choice questions. Each question has *four* alternatives, one of them is correct. Once you have chosen the correct alternative, circle your choice **in this book**.

Each question in this section is worth **2 marks** (20 marks total). There is a **penalty** of **-1 mark** for answering a question in this section incorrectly. There is no penalty for not answering a question.

a) Consider the following ODL definition for classes describing `Movie`s and the `Actor`s who star in them:

```
class Movie {
   attribute string title;
   attribute integer year;
   attribute integer length;
   relationship Set<Actor> stars inverse Actor::starredIn;
};
class Actor {
   attribute string name;
   attribute string address;
   ...
};
```

We wish to represent the n:m relationship between actors and the movies that they star in. Which one of the following `relationship` definitions would complete the `Actor` class definition and capture this relationship.

i) `relationship Movie starredIn inverse Movie::stars;`

ii) `relationship Movie starredIn inverse Set<Movie>::stars;`

iii) `relationship Set<Movie> starredIn inverse Movie::stars;`

iv) `relationship Set<Movie> starredIn inverse Set<Movie>::stars;`

b) Consider the SQL query:

```
select name,address from R where age < 30
```

Which of the following relational algebra expressions gives a correct translation of this query?

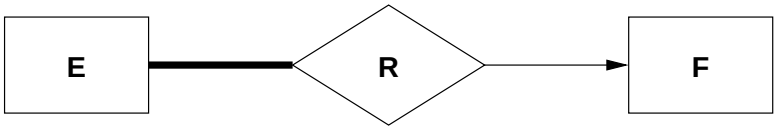
i) $\sigma_{(name,address)}(\pi_{(age<30)}R)$

ii) $\pi_{(name,address)}(\sigma_{(age<30)}R)$

iii) $(name, address) \bowtie R_{(age<30)}$

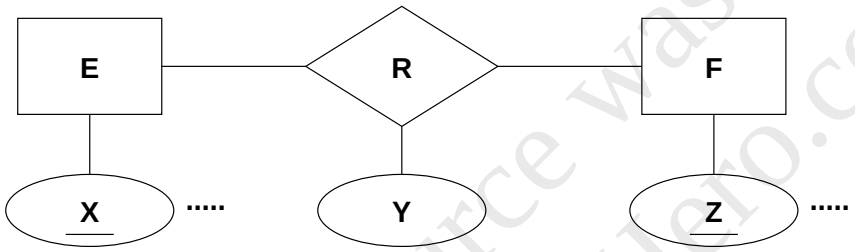
iv) $\pi_{(name,address)}(R \bowtie (age < 30))$

c) The following E/R diagram describes what kind of relationship between entities `E` and `F`?



i) one in which every `E` entity is related to exactly one `F` entity
ii) one in which some `E` entities are related to exactly one `F` entity
iii) one in which every `E` entity is related to one or more `F` entities
iv) one in which some `E` entities are related to one or more `F` entities

d) Consider the following E/R diagram:



How would the relationship `R` be represented in a relational design derived from this?

i) by combining all attributes from `E`, `F` and `R` in a single table
ii) as a table `R` containing an attribute for `Y` and foreign keys `X` and `Z`
iii) as a table `R` containing foreign keys `X` and `Z`, and placing attribute `Y` in `E`
iv) by placing a foreign key for `X` in `F`, a foreign key for `Z` in `E`, and attribute `Y` in `F`

e) Consider the relation $R = ABCD$ and the set F of functional dependencies on its attributes:

$$F = \{BC \to D,\ C \to A,\ AB \to C\}$$

What are the candidate keys of the relation R?

i) AB, AC
ii) BC, BD
iii) AB, BC
iv) AD, AB

f) Which one of the following statements is true?

i) 1NF enforces the least redundancy.
ii) A relation schema in 2NF is also in 3NF.
iii) BCNF allows transitive dependencies among attributes.
iv) In the standard relational model, every relation is in 1NF.

g) PostgreSQL-specific querstion deleted.

h) PHP-specific question deleted.

i) A *dense index* on attribute `A` ...

   i) has a single entry for each distinct A value and requires the data file to be sorted on A

   ii) may have multiple entries for each distinct A value and requires the data file to be sorted on A

   iii) has a single entry for each distinct A value and does not require the data file to be sorted on A

   iv) may have multiple entries for each distinct A value and does not require the data file to be sorted on A

j) Which of the following schedules *is* conflict-serializable?

   i) `T1:R(X) T2:R(X) T1:W(X) T2:W(X)`

   ii) `T1:R(X) T2:W(X) T1:W(X) T3:W(X)`

   iii) `T1:R(X) T2:R(Y) T3:W(X) T2:R(X) T1:R(Y)`

   iv) `T1:R(X) T1:R(Y) T1:W(X) T2:R(Y) T3:W(Y) T1:W(X) T2:R(Y)`

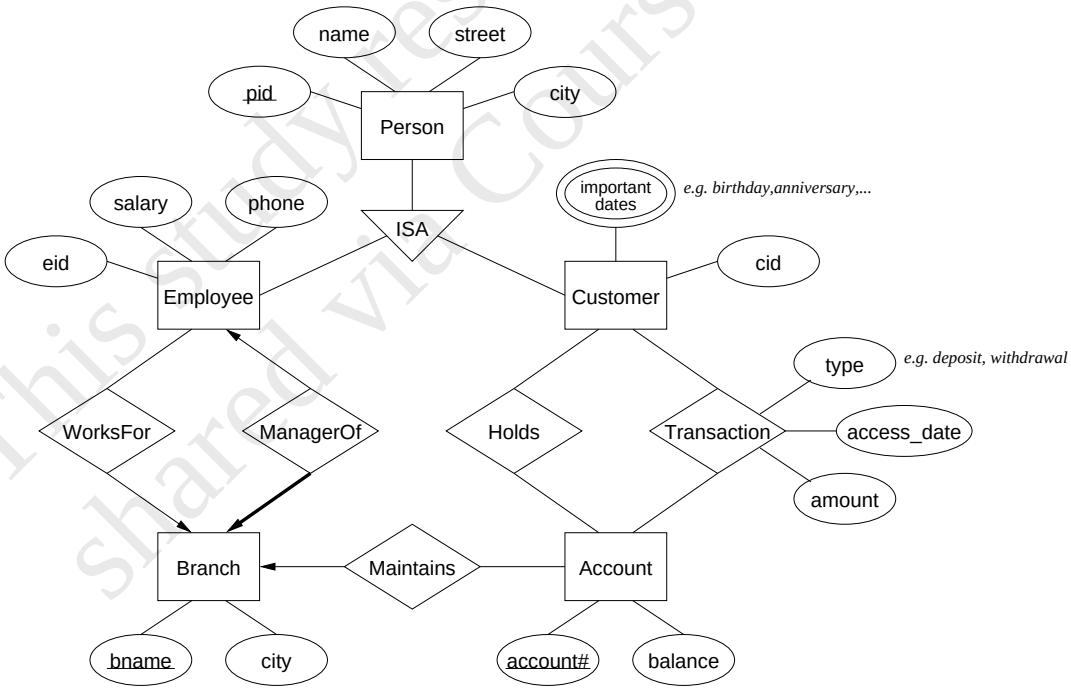
Question 2

(30 marks total)

a) *(10 marks)* Draw an ER diagram that captures all the following information:

- Patients are identified by an SSN, and their names, addresses and ages must be recorded.
- Doctors are identified by an SSN. For each doctor, the name, specialty and years of experience must be recorded.
- Each pharmacy has a name, address and phone number. A pharmacy must have a manager.
- A pharmacist is identified by an SSN, he/she can only work for one pharmacy. For each pharmacist, the name, qualification must be recorded.
- For each drug, the trade name and formula must be recorded.
- Every patient has a primary physician. Every doctor has at least one patient.
- Each pharmacy sells several drugs, and has a price for each. A drug could be sold at several pharmacies, and the price could vary from one pharmacy to anther.
- Doctors prescribe drugs for patients. A doctor could prescribe one or more drugs for several patients, and a patient could obtain prescriptions from several doctors. Each prescription has a date and quantity associated with it.

b) *(10 marks)* Convert the following E/R design (for a simple banking application) into a relational design. Give the relational design as a relational diagram with arrows to indicate the foreign key relationships. Underline all attributes that correspond to primary keys.



c) *(10 marks)* Express the relational model as a set of Oracle `create table` statements. Make any assumptions that you wish about attribute domains. Include all primary/foreign key constraints and any domain constraints that clarify your assumptions.

Question 3

(25 marks total)
Consider the following employee database, where the primary keys are underlined.

```
Employee(ename:string, street:string, city:string);
Works(employee:string, company:string, salary:real);
Company(cname:string, city:string);
Manages(employee:string, manager-name:string)
```

Give a single SQL statement for each of the following queries:

a) *(2 marks)* Find the names, street addresses, and cities of residence of all employees who work for "First Bank Corporation" and earn more than $40,000.

b) *(2 marks)* Find the names of all employees in the database who live in the same cities as the companies for which they work.

c) *(2 marks)* Give all managers of "First Bank Corporation" a 10 percent salary raise.

d) *(3 marks)* Find the names of all employees in the database who earn more than any employee of "Small Bank Corporation".

e) *(5 marks)* Assume that the companies may be located in several cities. Find the names of all companies located in every city in which "Small Bank Corporation" is located.

f) *(5 marks)* Find the name of the company that has the most employees.

g) *(6 marks)* Find those companies whose employees earn a higher salary, on average, than the average salary at "First Bank Corporation", display those companies' names in ascending order.

Question 4

(20 marks total)
Consider the following (partially-complete) relational schema:

```
Received(id:integer, publisher:integer, amount:integer, isbn:text, received:timestamp)
Editions(isbn:text, title:text, publisher:integer, published:date, ...)
Store(isbn:text, storeAmount:integer, soldAmount:integer)
Publisher(pid:integer, name:text, address:text, ...)
```

a) *(8 marks)* Suppose that a customer wants to order (multiple copies of) a book. Write a PL/SQL function `check_order()` that accepts the ISBN of a book and the number of copies requested, and determines whether the order can be met. The function should:

- check for a non-existent ISBN (invalid ISBN)
- check whether the store contains sufficient copies of the book
- if so, work out the number of copies remaining after the order is filled
- if not, give the shortage (i.e. the number of additional copies needed to meet the order)

Sample output from the `check_order()` function:

```
SQL> select check_order('0-8053-1755-6', 100);
The isbn does not exist.

SQL> select check_order('0-8053-1755-7', 100);
Not enough copies in stock. The shortage is 32.

SQL> select check_order('0-8053-1755-8', 100);
Completion of this order would leave 66 copies of 0-8053-1755-8.
```

b) *(12 marks)* Define a PL/SQL trigger function `receive_supply()` which is called after each `insert` or `update` operation is performed on the `Received` table. The `received_supply()` function should check to make sure that each added book contains a valid publisher `id` and a valid book `isbn`. It should then do the following:

- for an `insert`, include the number of copies received in the total amount in store
- for an `update`, if the change involves the book, then restore the `Store` entry for the old book and update the `Store` entry for the new book
- it should also calculate a new `Received.id` (one higher than the previous highest) and ensure that it is placed in the `id` field of the new tuple.
- it should also set the time-stamp for the new tuple to the current time.

Under this scheme, tuples would be inserted into the Received table as:

```
insert into Received(isbn,publisher,amount) values ('0-8053-1755-4',9335,200);
```

Question 5

(20 marks total)
Consider the following (simplified) relational schema for university study:

```
Student(id:integer, family:string, given:string, degree:string, enrolled:date)
Course(id:integer, code:string, session:string, title:string, syllabus:string)
Enrolment(student:integer, course:string, mark:real, grade:string)
```

a) *(6 marks)* For each of the following SQL queries, write an efficient relational algebra expression that might be used to implement the query. To make the expressions clearer, you may use as many named intermediate temporary relations as you wish. Correct, but grossly inefficient, relational algebra expressions will be awarded only half marks.

i)
```
select given,family from Student
```

ii)
```
select * from Enrolment where student=2233456
```

iii)
```
select given,family,course
from   Enrolment, Student
where  Enrolment.student = Student.id
```

iv)
```
select e.code, e.session, c.title, e.mark, e.grade
from   Enrolment e, Course c, Student s
where  e.course = c.id and e.student = s.id and s.id = 2234567
```

b) *(14 marks)* Join-cost question deleted. Not relevant for COMP9311.