UNIVERSITI SAINS MALAYSIA

First Semester Examination
Academic Session 1996/97

October/November 1996

CIS401 - Database Management and Design

Duration : [3 hours]

INSTRUCTION TO CANDIDATE:

- Please ensure that this examination paper contains **EIGHT** questions in **FIVE** printed pages before you start the examination.
- Answer **ALL** questions. If you choose to answer the questions in English, at least one question must be answered in Bahasa Malaysia.

ENGLISH VERSION OF THE QUESTION PAPER

1. (a) For each of the following, write SQL statements to create views where needed and to grant the indicated privileges for the university database:

STUDENT(STUDID, STUNAME, MAJOR, CREDITS)
FACULTY(FACID, FSCNAME, DEPT, RANK)
CLASS(COURSE#, FACID, SCHED, ROOM)
ENROLL(COURSE#, STUID, GRADE)

(i) Give permission to read the tables STUDENT and CLASS to user U1. The user may also read ENROLL, but without the GRADE attribute.

(ii) Give permission to read and modify the tables FACULTY and CLASS to user U2. This user may authorize user U3 to read and modify CLASS, but not FACULTY.

(iii) Give permission to read and modify the tables STUDENT, CLASS, and ENROLL to user U4. This user may authorize users U5 to read these tables and to grant this right to other users.

(50/100)

(b) Describe, by using an example, a simple and polyalphabetic encryption method.

(50/100)

2. (a) Explain what a domain constraint is and list the most popular domain constraints.

(40/100)

(b) Define the following database with the use of domain constraints and entity integrity constraints:

customer(customer-name, street, customer-city),
branch(branch-name, assets, branch_city),
deposit(branch-name, account-number, customer-name, amount)
borrow(branch-name, loan-number, customer-name, amount)

(30/100)

(c) Write assertions stating that:

(i) balance amount is not less than zero;

(ii) loan amount is not greater than 1000;

(iii) a customer name existing in the deposit relation exists in the customer relation, too.

(30/100)

3. Consider the following two transactions:

```
T1:   read(A);
      read(B);
      B := A+B;
      write(B);
T2:   read(B);
      read(A);
      A := A + B;
      write(A);
```

(a) Add lock and unlock instructions to transactions T1 and T2 so that they observe the two-phase locking protocol. (30/100)

(b) Write a schedule for these transactions, which produces a deadlock. (30/100)

(c) Write a serializable time-stamp protocol for these transactions, indicate time-stamp values for each read/write operation. (40/100)

4. (a) Assume a system having a log with immediate updates has the following log entries, ending with a system crash:

```
<R, begin>
<R, X, 1, 5>
<R, Y, -1, 0>
<R, commit>
<S, begin>
<S, Z, 8, 12>
<Checkpoint record>
<S, X, 5, 10>
<T, start>
<T, Y, 0, 15>
<S, commit>
-------- system crash ----------
```

(i) Which transactions, if any, need to be redone? (10/100)

(ii) Which transactions, if any, need to be undone? (10/100)

(iii) Which transactions, if any, are not affected by the crash? (10/100)

(b) Now assume the system uses an incremental log with deferred updates:

(i) Rewrite the log entries for the transactions in Exercise 1 for this logging method.

(40/100)

(ii) Which transactions, if any, need to be redone after the failure?

(10/100)

(iii) Which transactions, if any, need to be undone after the failure?

(10/100)

(iv) Which transactions, if any, are not affected by the crash?

(10/100)

5. (a) Model a relevant part of a university administration. In particular, apply the abstraction concepts of aggregation - e.g., students enrolling in courses being taught by particular professors - and generalization with the following specialization hierarchy: students and university staff are a specialization of persons and professors and secretaries are a specialization of university staff; indicate the relationships *"is-a"* and *"part-of"*

(50/100)

(b) List and explain main features of an object-oriented model.

(50/100)

6. (a) Describe the object type definition frame, indicating the purpose of each type definition section.

(30/100)

(b) Design an object-oriented tourist database storing data about cities, hotels, monuments, and restaurants in the following way:

- a hotel is characterized by *name*, *rank (stars)*, set of *free room numbers*, set of *guests* which is a set pairs <person_name, room number>, and operations of *reserving a room* for a particular guest and *checking-out*;
- a city is characterized by *name, set of streets, set of hotels*, and operations of *creating a new hotel* and *finding rooms* of needed rank;
- a monument is characterized by *name, address, admission fee*, and *statistics* about visitors;
- a restaurant is characterized by *city, rank (stars)*, and *menus*;
- a tourist city is regarded as a special kind of city and is characterized, in addition, by a set of *monuments* and an operation of *creating a new monument*.

Operation implementations may be omitted.

(70/100)

...5/-

7. (a) Give a definition and describe different kinds of persistence.

(30/100)

(b) Declare and implement operations and bodies of an object type "complex" with the following operation:

- initializer Complex creating a complex number using two float numbers;
- get the real part of a complex number;
- get the imaginary part of a complex number;
- add a complex number to a given complex number;
- subtract complex number from a given complex number;
- compare two complex numbers for equality.

(70/100)

8. (a) Consider the relation that is fragmented horizontally by plant-number:

EMPLOYEE (NAME, ADDRESS, SALARY, PLANT-NUMBER)

Assume each fragment has two replicas, one stored at the New York site and one stored locally at the plant site. Describe a good processing strategy for the following queries entered at the San Jose site:

(i) Find all employees at the Boca plant.

(20/100)

(ii) Find the highest-paid employee at each of the following sites Toronto, Montreal.

(20/100)

(iii) Find the lowest-paid employee in the company.

(20/100)

(b) Describe the purpose and technique of the vertical fragmentation in a distributed database system

(40/100)

- oooOooo -