CMPU 2007

W228/206

DUBLIN INSTITUTE OF TECHNOLOGY
KEVIN STREET DUBLIN 8

BSc. (Honours) Degree in Computer Science

Year 2

Semester I Examinations 2013/2014

DATABASES I

Ms. D. Lawless
Dr. D. Lillis
Mr. K. Foley

Friday 10th January 13.00 p.m. – 15.00 p.m.

Question 1 is **compulsory.**
Answer question 1 **and** two of the other three questions.

Read case study 1 on Page 2 before attempting question 1.

Read the case study 2 on Page 2 before attempting questions 2 through 4.

There is a syntax table on the last page to assist you.

Case Study #1 - D-Taxi Company

D-Taxi is a small independent taxi company operating in Dublin. D-Taxi owns 25 taxis which it rents out to drivers on an annual basis. It currently has 75 drivers registered but has plans to expand its operation to three more cities with similar numbers in each.

D-Taxi needs some way of identifying each vehicle, its make, model, and year D-Taxi purchased it as well as the purchase price.

D-Taxi needs to identify each driver and store the driver name, address, licence number, date they started renting from D-Taxi and the annual rental fee they are charged. A license number must be stored for all drivers and this value must be unique.

Each driver must pay an annual rental fee in advance to D-Taxi giving them use of a vehicle for one 8 hour shift every day of the year. D-Taxi needs to keep track of a driver's rental history including which vehicle they rent on which date and much they earn each shift.

D-Taxi is responsible for maintaining the vehicles. If a vehicle is due for a service D-Taxi contacts a garage and arranges it. It needs to store for each service the date, the cost and the garage that serviced it.

At the end of each shift drivers give the money they have earned to D-Taxi. At the end of every week Q-Taxi calculates the amount owing to each driver based on the money earned from fares and applies a deduction of 5%. The drivers are then paid.

1. **(a)** List the primary entities for the D-Taxi company stating for each entity which attribute you would use as a primary key.

(5 marks)

(b) List the attributes for each of your entities, noting constraints (other than foreign keys).

(5 marks)

(c) Draw a full ERD, complete with attributes and their datatypes, underlining primary keys and marking foreign keys with (FK).

(10 marks)

(d) Assuming that you have created tables in your schema that correspond to your ERD, write a set of SQL statements to add the following information:

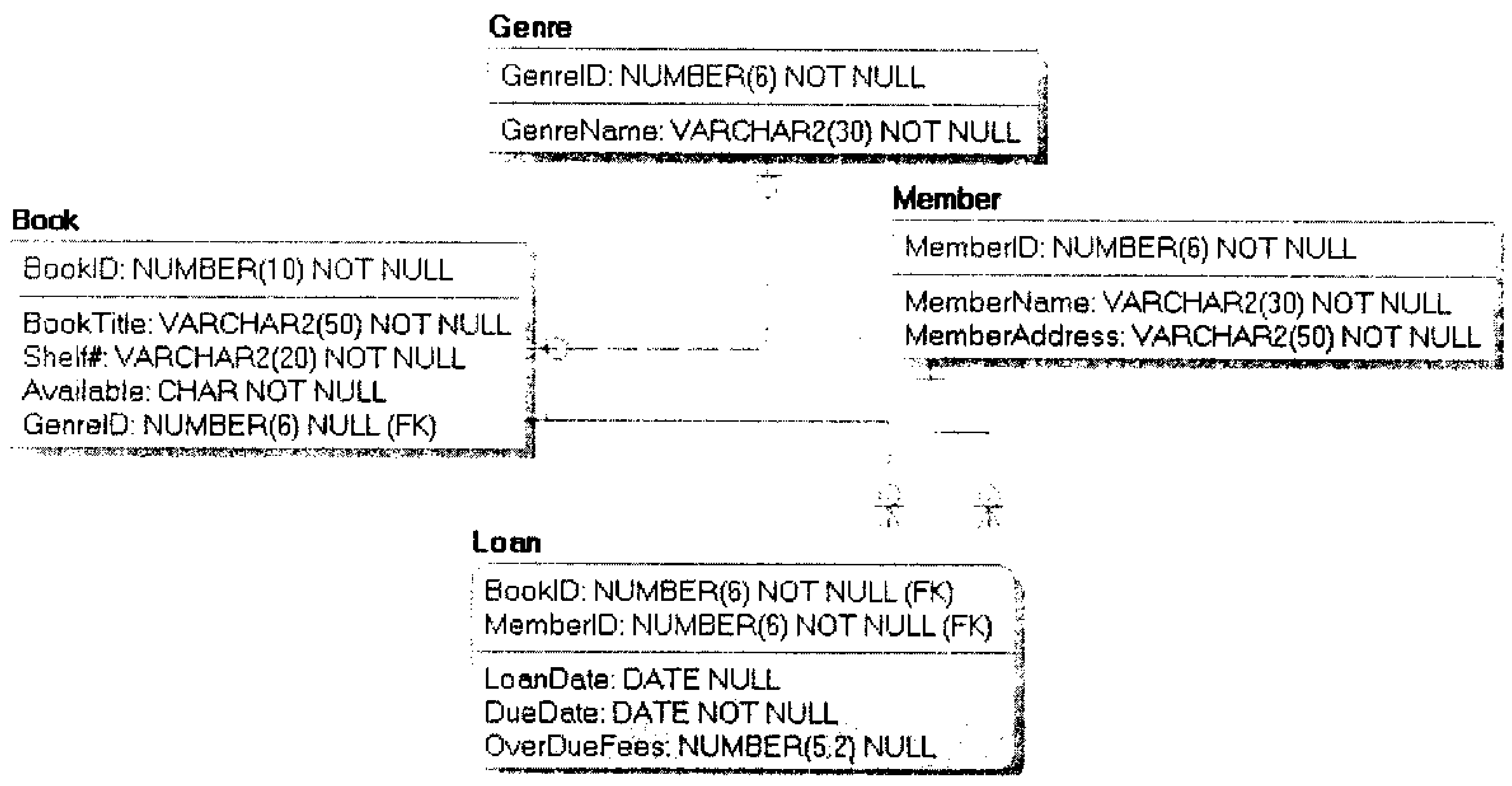
```
"The driver Hamish MacBeth, (driver's name) of 12 Lochdubh, Scotland,
(driver's address), licence number KA12340000 who registered with D-Taxi
on 1st January 2006, pays an annual rental fee of €1000 and has driven
three shifts as follows:

01-Jan-13, earning €150 using vehicle 12-D-2777, Toyota Corolla, bought
for €20000 on 1st January 2013

02-Jan-13, earning €300 using vehicle 10-D-1000, Ford Focus, bought for
€12000 on 1st January 2012

03-Jan-13 00 earning €400 using vehicle 09-D-1999, Ford Mondeo, bought for
€12000 on 1st January 2012"
```

(10 marks)

(e) Write an SQL statement to calculate the total deduction applied to MacBeth's earnings between Date 01-01-13 and 03-01-13. You should include some text in the select statement to indicate what information is being presented and format the deduction as euros.

(10 marks)

Case Study #2 – Library System ERD


Genre
GenreID: NUMBER(6) NOT NULL
GenreName: VARCHAR2(30) NOT NULL
Book
BookID: NUMBER(10) NOT NULL
BookTitle: VARCHAR2(50) NOT NULL
Shelf#: VARCHAR2(20) NOT NULL
Available: CHAR NOT NULL
GenreID: NUMBER(6) NULL (FK)
Member
MemberID: NUMBER(6) NOT NULL
MemberName: VARCHAR2(30) NOT NULL
MemberAddress: VARCHAR2(50) NOT NULL
Loan
BookID: NUMBER(6) NOT NULL (FK)
MemberID: NUMBER(6) NOT NULL (FK)
LoanDate: DATE NULL
DueDate: DATE NOT NULL
OverDueFees: NUMBER(5,2) NULL

Description: The Book table stores details of books in the library catalogued. Books are allocated a unique ID, the title, location on shelves and their availability is recorded. Each book is allocated to one genre. Details of genres are stored in the Genre table. Members are allocated a unique ID and their name and address are stored. Details of books members borrow are stored in the Loan table which stores the date of the loan, due date and any fees charged if the book is not returned on time. These details are stored in the Loan table together with the ID of the book borrowed and the member who borrowed it.

2. (a) (i) Can two books with the same identifier appear in the table Loan?

(ii) Write the query to return the book id, member id and date due for all loans.

(iii) Amend the query to include fees owed for all books showing only those that had overdue fees attached.

(iv) Amend the query above to sort this in due date order ascending.

(v) Write the SQL to return from the loan table the total owed by each member.

(5 x 2 marks)

(b) Amend the query for (a) (iii) above to include the name of the book and the name of the member sorted in descending order of member name. Explain how the join used works.

(10 marks)

(c) Write a query to return a list of the genres of books each member has borrowed. Each genre should appear at most once for a member. Explain how the join used works.

(10 marks)

3. (a) Write a query to return the book titles for all books borrowed since '01-JAN-2011' together with the ID and names of the members that borrowed them. Return the results in ascending order of date borrowed.

(10 marks)

(b) Write a query to return the ID and name of all members who have never borrowed a book explaining how the join works.

(10 marks)

(c) (i) Write the SQL to alter the relevant attribute to accommodate values of up to 9999.99 for overdue fees.

(ii) Write the SQL to add a constraint to ensure that availability of a book can only be indicated by values of Y or N.

(iii) Write the SQL to set the value of availability to be Y by default.

(iv) Write the SQL to add an attribute to the member table to store a member's email address.

(v) Write the SQL to add a constraint to ensure that email addresses have a unique value.

(5 x 2 marks)

4. **(a)** **(i)** Write an SQL statement to add a loan for member ID 100, borrowing book 100, with a due date of '01-Jan-2014' using the current date as the date of the loan.

(5 marks)

(ii) Identify what must exist in the database for this insert statement to succeed. Write the SQL to achieve this.

(5 marks)

(b) Write an SQL statement to update all loans for member ID 100, to have a due date of the loan date plus 14 days.

(10 marks)

(c) **(i)** Combine your answers for (a) and (b) above into a single *transaction* (not a single statement), making it permanent before exiting. Justify your answer.

(7 marks)

(ii) Explain how you would amend this transaction to reverse the changes made. Justify your answer.

(3 marks)

SYNTAX TABLE

ALTER TABLE *tablename*
{[ADD | MODIFY] *column-definition*}
ALTER TABLE *tablename*
{ADD CONSTRAINT constraint_name CHECK (*column_name condition*)}
CREATE TABLE *tablename*
({*column-definition*}
[PRIMARY KEY ({*column-name*},)]
{[FOREIGN KEY ({*column-name*}) REFERENCES *tablename*]})
COMMIT
DELETE FROM *tablename* WHERE *condition*
DROP [TABLE *tablename*|DROP VIEW *viewname*]
INSERT INTO *tablename* [{*column-name,*}] VALUES (*data-value-list*)
ROLLBACK WORK
SELECT [DISTINCT] *column-list* FROM *tablename*
[WHERE *condition*]
[ORDER BY *column-list*]
[GROUP BY *column-name*]
[HAVING *condition*]
Conditions : =,>,<,>=,<=,<>, BETWEEN .. AND.., IN *(list)*, IS NULL, LIKE
Logical operators: AND, OR, NOT
Set operations: UNION, MINUS, INTERSECT
SELECT SYSDATE FROM DUAL;
SELECT column-list FROM
<tablelist> WHERE <column-list> IN | ALL |ANY |EXISTS (Select statement)
ORDER BY ASC|DESC

```
SELECT [TableExpression [ INNER ] JOIN TableExpression
{
  ON booleanExpression |
  USING clause
}]
[TableExpression [LEFT|RIGHT] [ OUTER ] JOIN TableExpression
{
  ON booleanExpression |
  USING clause
}]
```

UPDATE *tablename*
[SET *column-name= <data-value>*] [WHERE *condition*]
Column-definition = column-name [CHAR [(*n*)] | VARCHAR(*n*) | NUMBER [*n,p*] | DATE | DATETIME] {[NOT NULL | UNIQUE | PRIMARY KEY]}