Xi'an Jiaotong-Liverpool University
西交利物浦大學

MODULE CODE	EXAMINER	ACADEMIC UNIT	TEL
CPT103			

2nd SEMESTER 2023-24 Final EXAMINATION

Undergraduate – Year 2

Introduction to Databases

TIME ALLOWED: 2 Hours

INSTRUCTIONS TO CANDIDATES

1、 **This is a closed book examination.**
2、 **Total marks available are 100.**
3、 **Answer all questions.**
4、 **Answer should be written in the answer booklet(s) provided.**
5、 **Only English solutions are accepted.**
6、 **The university approved calculator - Casio FS82ES/83ES can be used.**
7、 **All materials must be returned to the exam supervisor upon completion of the exam. Failure to do so will be deemed academic misconduct and will be dealt with accordingly.**

Question 1 (35 marks)

Consider the following relations for bank card and transactions with some example data:

cards

card_no	balance	passport
1	6050	G203T
2	7000	G203T
3	280	A8910
4	5000	T2818

holders

passport	name
G203T	John
A8910	Anna
T2818	Chris

transactions

trans_id	from_card	to_card	amount	trans_time
1	1	2	1000	2021-03-21 19:21:21
2	3	4	1550	2023-02-01 08:16:00
3	4	1	50	2024-01-01 09:32:21

1) You are given the following SELECT queries. What are the results of application of these queries to the tables above? (3 marks each)

1)
```
SELECT card_no, name FROM cards NATURAL JOIN holders
    ORDER BY card_no DESC;
```

2)
```
SELECT name FROM holders
    WHERE passport LIKE '%T' OR passport LIKE 'T_';
```

3)
```
SELECT trans_id FROM cards LEFT OUTER JOIN transactions
    ON (card_no = from_card)
    WHERE to_card IN
        (SELECT card_no FROM cards WHERE balance <= 5000);
```

4)
```
SELECT c1.passport, c2.passport
    FROM cards c1, cards c2
    WHERE c1.card_no = c2.card_no - 3;
```

5)
```
SELECT name, sum(balance) AS wealth
    FROM cards INNER JOIN holders USING (passport)
    GROUP BY name HAVING wealth > 1000;
```

2) Write an SQL statement to increase the balances of all cards by 1000.

(4 marks)

UPDATE Cards SET balance = balance + 1000;

3) Write an SQL statement to delete transactions that were done before the year of 2020

(4 marks)

DELETE From transactions WHERE trans_time < '2020-01-01 00:00:00';

4) Write an SQL statement to get all transactions that involve cards having more than 2000 balance. In the result, list trans_id and all card numbers involved in this transaction. (Note: "involve" means giving or receiving money)

(4 marks)

SELECT trans_id, from_card, to_card from transactions inner join cards on card_no = from_card or card_no = to_card Group By trans_id Having balance > 2000;

Distinct

5) Get the list of holders who have more than 1 bank cards. In the result, list the number of bank cards, passport numbers and holder names and sort the results by holder names in ascending order.

(4 marks)

SELECT trans_id, from_card, to_card from transactions join cards C_1 on from_card = C_1.card_no join cards C_2 on to_card = C_2.card_no where C_1.balance > 2000 or C_2.balance > 2000;

6) Get the name(s) of holder(s) who has the most money deposited in the bank. If a person has multiple cards, all balances must be counted in. In the result, list holders' passport numbers and total balances.

(4 marks)

(5).

```
SELECT h.Name, count(C.card_no) as bankcard_numbers, h.passport from holders h
inner join cards C on h.passport = C.passport GROUP BY h.Name, h.passport
Having count(card_no) > 1 ORDER BY h.Name ASC;
```

(6).

```
SELECT t.name, max(t.sum) from (SELECT DISTINCT h.name as name, sum(C.balance) as sum
from holders h left join cards C on h.passport = C.passport Group by h.name) as t;
```

```
SELECT t.name, t.sum
FROM (
    SELECT h.name, SUM(c.balance) AS sum
    FROM holders h
    LEFT JOIN cards c
    ON h.passport = c.passport
    GROUP BY h.name
) AS t
WHERE t.sum = (
    SELECT MAX(sum)
    FROM (
        SELECT h.name, SUM(c.balance) AS sum
        FROM holders h
        LEFT JOIN cards c
        ON h.passport = c.passport
        GROUP BY h.name
    ) AS subquery
);
```

Question 2 (20 marks)

1) What are the values of the expressions below in the context of 3-valued logic? (10 marks)
   1. NOT (FALSE OR Unknown) Unknown
   2. TRUE OR Unknown TRUE
   3. (11 - 11) AND (NOT Unknown) False
   4. Unknown + 12 Unknown
   5. Unknown <> Unknown Unknown

2) Please answer the following questions related to deadlock: (6 marks)
   a) Explain what is a deadlock.
   b) Give an example showing what is deadlock.
   c) How to detect deadlocks.

3) What concurrency problem does the following schedule present? Please explain the problem in detail.

T1	T2
Read(X) X = X + 5 Write(X)	
	Read(X) Read(Y) Sum = X+Y
Read(Y) Y = Y - 5 Write(Y)	

(4 marks)

Question 3 (20 marks)

Normalise the following table "T" into the 3rd Normal Form by clearly describing the normalisation process, i.e. the dependencies removed and how the table is split into sub-tables. Describe the primary key and functional dependencies for each resulting sub-tables.

A	B	C	D	E	F	G

Attributes (A, B) form the primary key and the functional dependencies are:

A, B → C, D, E, F, G

B → C, D, E, F

D → E, F

F → B

E → F

2NF: FD B→C,D,E,F is partial dependence on the primary key (A,B) on Table T.
After removing it, T is splited into T_1: (A,B,G) with primary key (A,B)
T_2: (B, C,D,E,F) with primary key (B).

3NF: column (E,F) is depend transitively on the primary key B via D on Table T_2.
After removing it, T_2 is splited into T_{2-1}: (B,C,D) with primary key (B).
T_{2-2}: (D,E,F) with primary key (D).
column F is depend transitively on the primary key D via E on Table T_{2-2}.
After removing it, T_{2-2} is splited into T_{2-2-1} (D, E) with primary key (D).
T_{2-2-2} (E,F) with primary key E.

Final Design: T_1(A,B,G) T_{2-1} (B,C,D) T_{2-2-1} (D,E) T_{2-2-2} (E,F)

Question 4 (35 marks)

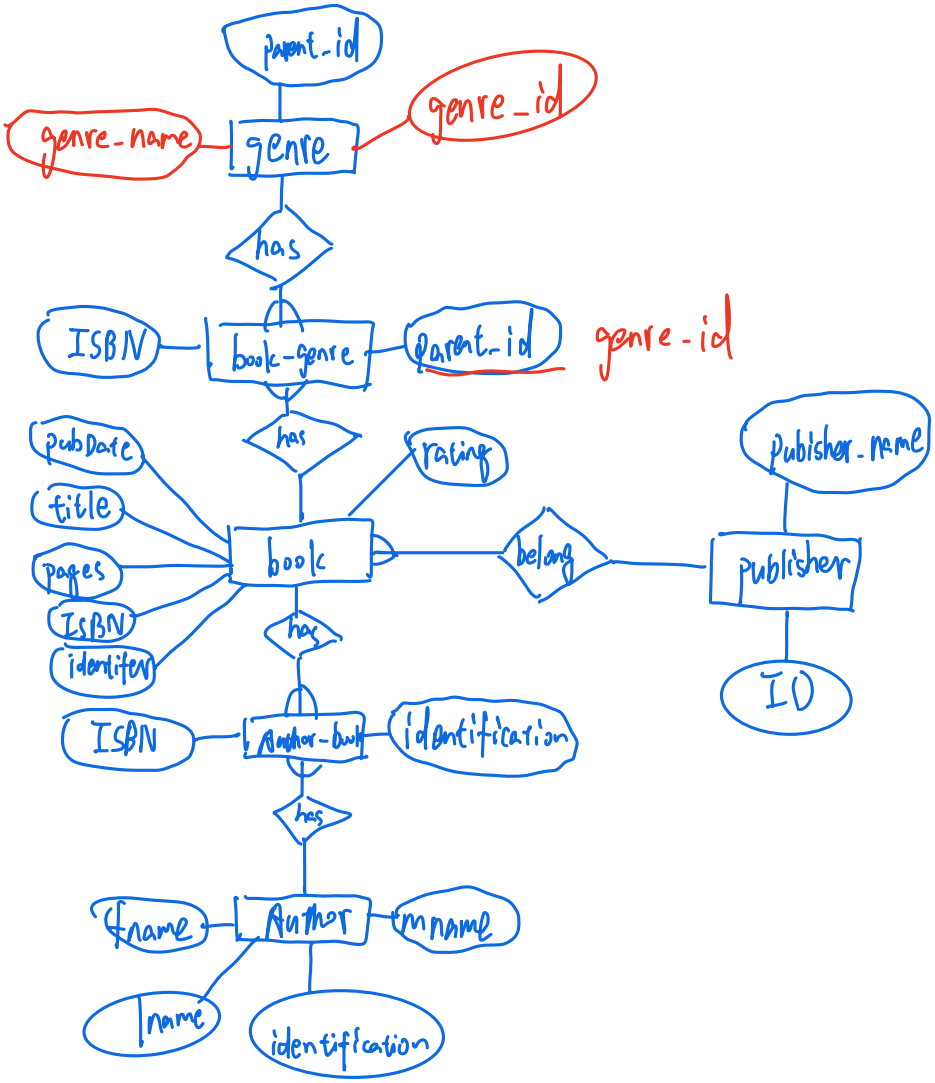
You are hired by a bookstore to develop a database for managing the information about books. The requirements are listed below:

1. Each book should have a unique book identifier, title, total pages, ISBN, published date, and the identification of the publisher. Each book belongs to a publisher and a publisher may have one or many books. If the value in the publisher column is NULL, it means the publisher is unknown at the time of recording the book.
   a. The rating should range from 1 to 5.
2. Each publisher should have a unique ID and a name.
3. Authors should have author identification, first name, middle name, and last name. Each author has one or many books while each book is written by one or multiple authors.
4. Each book may belong to one or more genres; a genre may have one or many books.
5. The genre data is hierarchical which is specified by an attribute called 'parent_id'.

Task 1: Draw the entity relationship diagram for the database. All M:N and 1:1 relationships must be properly dealt with. Note that domain constraints are not allowed in this question. (23 marks)

Task 2: Based on your solution to Task 1 above, write the SQL code to create the tables for the database. You should include all the specified attributes and specify the appropriate primary and foreign keys. Minor syntactical errors in your SQL code will not be penalised in the marking of this answer. (12 marks)

END OF Final EXAM




```
CREATE TABLE genre (
    parent_id INT DEFFAULT NULL,
    genre_id INT Primary KEY,
    genre_name VARCHAR(20),
    constraint fk_parent_genre foreign key (parent_id)
    references genre (genre_id) on update CASCADE on DELETE CASCADE
);
```

```
CREATE TABLE genre (
    parent_id INT primary key.
);

CREATE TABLE publisher (
    ID VARCHAR(20) primary key,
    publisher_name VARCHAR(250)
);

CREATE TABLE Author (
    fname Varchar(20),
    mname Varchar(20),
    lname Varchar(20),
    identification Varchar(50) primary key
);

CREATE TABLE book (
    pubDate Date,
    title varchar(100),
    pages INT,
    ISBN Varchar(50) Primary key,
    identifer Varchar(50) Unique key,
    rating INT,
    pub_ID varchar(20) DEFAULT Null,
    constraint fk_book_publisher foreign key (pub_ID)
    references publisher(ID) on DELETE SET Null
    on update CASCADE
);

CREATE TABLE book_genre (
    ISBN Varchar(50),
    parent_id INT
    constraint pk_book_genre_genre Foreign key
    (parent_id) references genre (parent_id);
    constraint pk_book_genre_book Foreign key
    (ISBN) references book (ISBN).
);

CREATE TABLE Author_book (
    ISBN Varchar(50),
    identification varchar(50),
    constraint pk_Author_book_Author Foreign key
    (ISBN) references book (ISBN),
    constraint pk_Author_book_book Foreign key
    (identification) references author (identification)
);
```