# 資料庫系統作業三

409410050 資工二王謙靜

## 甲1.

No

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Recipient Name | Donor Organization | Organization Type | amount | |
| Alice | CCU | tobacco | | 1000 |
| Bob | NCKU | school | | null |
| Campbell | CCU | tobacco | | 1000 |

第一個query：Alice, Campbell

第二個query：Alcie, Bob, Campbell。

query1：找和campbell同donorOrganization的人

query2：

(select donorOrganization from donations B

where B.recipientName = A.recipientName)會找到所有的donorOrganization，之後做except會使donorOrganization為empty set，故做最後的not exist後會將所有人都輸出。

## 甲2.

No

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Recipient Name | Donor Organization | Organization Type | amount | |
| Alice | CCU | tobacco | | 1000 |
| Bob | NCKU | school | | null |

第一個query會輸出Alice

第二個query會輸出Alice, Bob

有null時不管大於小於都是false

## 甲3.

No

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Recipient Name | Donor Organization | Organization Type | amount | |
| Alice | CCU | tobacco | | 1000 |
| Alice | NCKU | school | | 100 |

第一個query輸出Alice。

第二個在where之後會只剩下(Alice, CCU, school, 1000)那一筆，到having會被刪掉，故此query沒有輸出。

## 甲4.

No

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Recipient Name | Donor Organization | Organization Type | | amount |
| Alice | CCU | tobacco | 1000 | |
| Alice | NCKU | school | 100 | |

第一個query：找沒有被tobacoo贈與的人名，故不會有任何輸出。

第二個query：找有被非tobacco贈與的人名，會輸出Alice。

## 乙 0(create table).

|  |  |
| --- | --- |
| CREATE TABLE TASTES(  Name varchar(30) NOT NULL,  Filling varchar(30) NOT NULL,  PRIMARY KEY(Name, Filling)  ); | |
| CREATE TABLE LOCATIONS(  LName varchar(30) PRIMARY KEY NOT NULL,  Phone varchar(15),  Address varchar(50)  ); | |
| CREATE TABLE SANDWICHES(  Location varchar(50) NOT NULL,  Bread varchar(15) NOT NULL,  Filling varchar(30) NOT NULL,  Price double,  PRIMARY KEY(Location, Bread, Filling),  Constraint Locations\_sandwiches\_fk FOREIGN KEY(Location)  REFERENCES LOCATIONS(LName)  ); | |
| CREATE TSATES | |
|  | |
| CREATE LOCATIONS | |
|  | |
| CREATE SANDWICHES | |
|  | |
| IN TASTES TABLE | In SANDWICHES TABLE |
| IN LOCATIONS TABLE | |

## 乙1.

|  |
| --- |
| SELECT Location  FROM SANDWICHES s  WHERE s.Filling in (SELECT Filling  FROM TASTES t  WHERE t.Name = "Jones"); |
|  |

## 乙2.

|  |
| --- |
| SELECT L.Location, COUNT(DISTINCT s.Name)  FROM SANDWICHES L, TASTES s  WHERE L.Filling = s.Filling  GROUP BY L.Location; |
|  |

## 丙 0(Create table).

|  |
| --- |
| CREATE TABLE Authors(  aname varchar(25)PRIMARY KEY NOT NULL,  age int NOT NULL,  affiliation varchar(25),  nationality varchar(25) NOT NULL,  CHECK (age >= 0 and age <= 999)  ); |
|  |
| CREATE TABLE Publishers(  pid int PRIMARY KEY NOT NULL,  pname varchar(50) NOT NULL  ); |
|  |
| CREATE TABLE Publications(  ISBN varchar(25) PRIMARY KEY NOT NULL,  title varchar(50) NOT NULL,  pdate date NOT NULL,  category varchar(25) NOT NULL,  pid int NOT NULL,  CONSTRAINT Publication\_pid\_fk FOREIGN KEY(pid)  REFERENCES Publishers(pid)  ); |
|  |
| CREATE TABLE Readers(  rid int PRIMARY KEY NOT NULL,  rname varchar(25) NOT NULL,  email varchar(50)  ); |
|  |
| CREATE TABLE PubAuthors(  ISBN varchar(25) NOT NULL,  aname varchar(25) NOT NULL,  PRIMARY KEY(ISBN, aname),  CONSTRAINT PubAuthors\_ISBN\_fk FOREIGN KEY(ISBN)  REFERENCES Publications(ISBN),  CONSTRAINT PubAuthors\_aname\_fk FOREIGN KEY(aname)  REFERENCES Authors(aname)  ); |
|  |
| CREATE TABLE Selling(  ISBN varchar(25) NOT NULL,  rid int NOT NULL,  quantity int NOT NULL,  PRIMARY KEY(ISBN, rid),  CONSTRAINT Selling\_ISBN\_fk FOREIGN KEY(ISBN)  REFERENCES Publications(ISBN),  CONSTRAINT Selling\_rid\_fk FOREIGN KEY(rid)  REFERENCES Readers(rid),  CONSTRAINT quantity\_limit  CHECK (quantity >= 0 and quantity < 1000000)  ); |
|  |

|  |  |
| --- | --- |
| Authors table | Publishers table |
| Publications table | Reader table |
| Selling tabel | Pubauthors table |

## table check(constraint)

|  |
| --- |
|  |
|  |

## 丙1.

|  |
| --- |
| SELECT pubau.aname  FROM PubAuthors pubau, Publications book  WHERE book.ISBN = pubau.ISBN and book.pdate >= '2001-01-01'  GROUP BY pubau.aname  HAVING COUNT(DISTINCT pubau.ISBN) >= 2  ORDER BY pubau.aname DESC; |
|  |

## 丙2.

|  |
| --- |
| SELECT pution.pid, push.pname, pution.ISBN, pution.title, SUM(sell.quantity) as sale  FROM Publishers push, Publications pution, Selling sell  WHERE push.pid = pution.pid and sell.ISBN = pution.ISBN  GROUP BY pution.ISBN  ORDER BY pution.pid, push.pname, pution.ISBN, pution.title, sale; |
|  |

## 丙3.

|  |
| --- |
| SELECT tmp4.title, publisher1.pname  FROM Publishers publisher1 NATURAL LEFT OUTER JOIN  (SELECT tmp3.title, tmp3.pid  FROM (SELECT tmp.pid, MAX(sale) as max\_sale  FROM (SELECT pushA.pid, SUM(sellA.quantity) as sale  FROM Publishers pushA, Publications putionA, Selling sellA  WHERE putionA.pdate >= "2000-01-01" AND pushA.pid = putionA.pid and sellA.ISBN = putionA.ISBN  GROUP BY putionA.ISBN) as tmp  GROUP BY tmp.pid) as tmp2,  (SELECT pushA.pid, pushA.pname, SUM(sellA.quantity) as sale, putionA.title  FROM Publishers pushA, Publications putionA, Selling sellA  WHERE putionA.pdate >= "2000-01-01" AND pushA.pid = putionA.pid and sellA.ISBN = putionA.ISBN  GROUP BY putionA.ISBN) as tmp3  WHERE tmp2.max\_sale = tmp3.sale and tmp2.pid = tmp3.pid) as tmp4  ORDER BY tmp4.title, publisher1.pname; |
|  |

## 丙4.

|  |
| --- |
| SELECT tmp2.category, pution.title, tmp2.max\_sale  FROM (SELECT putionA.category, MAX(tmp.sale) as max\_sale  FROM (SELECT sellA.ISBN, SUM(sellA.quantity) as sale  FROM Selling sellA  GROUP BY sellA.ISBN) as tmp, Publications putionA  WHERE putionA.ISBN = tmp.ISBN  GROUP BY putionA.category) as tmp2,  (SELECT sellA.ISBN, SUM(sellA.quantity) as sale  FROM Selling sellA  GROUP BY sellA.ISBN) as tmp3, Publications pution  WHERE tmp3.sale = tmp2.max\_sale and tmp3.ISBN = pution.ISBN  and tmp2.category = pution.category  ORDER BY tmp2.category, pution.title, tmp2.max\_sale; |
|  |

## 丙5.

|  |
| --- |
| SELECT pution.title  FROM Publications pution  WHERE pution.category = "technology" and pution.pdate >= '2000-01-01'  and pution.ISBN not in (SELECT pubAuthor.ISBN  FROM PubAuthors pubAuthor, Authors author  WHERE pubAuthor.aname = author.aname  and author.age >=50)  ORDER BY pution.title; |
|  |

## 丁0(create table).

|  |
| --- |
| CREATE TABLE Earthquake(  qname varchar(25) PRIMARY KEY NOT NULL,  qdate date NOT NULL,  latitude varchar(25) NOT NULL,  longitude varchar(25) NOT NULL,  magnitude double NOT NULL,  fault varchar(50)  ); |
|  |

|  |  |
| --- | --- |
| CREATE TABLE Cities(  cID int PRIMARY KEY NOT NULL,  cname varchar(25) NOT NULL,  latitude varchar(25) NOT NULL,  longitude varchar(25) NOT NULL,  population int  ); | |
|  | |
| CREATE TABLE Damage(  cityID int NOT NULL,  qname varchar(25) NOT NULL,  cost int,  liveslost int,  PRIMARY KEY(cityID, qname),  CONSTRAINT damage\_city\_fk FOREIGN KEY(cityID)  REFERENCES Cities(cID),  CONSTRAINT damage\_qname\_fk FOREIGN KEY(qname)  REFERENCES Earthquake(qname)  ); | |
|  | |
| Cities table | Damage table |
| Earthquake table | |

## 丁1.

|  |
| --- |
| SELECT eq.qname, eq.qdate  FROM Earthquake eq  WHERE eq.magnitude > 7.0; |
|  |

## 丁2.

|  |
| --- |
| SELECT AVG(city.population)  FROM Cities city  WHERE city.cname LIKE 'San%'; |
|  |

## 丁3.

|  |
| --- |
| SELECT damage.liveslost, damage.qname, city.cname  FROM Cities city, Damage, Earthquake eq  WHERE city.cID = damage.cityID AND eq.qname = damage.qname  AND eq.qdate < '1960-01-01'; |
|  |

## 丁4.

|  |
| --- |
| SELECT city.cid, SUM(tmp.cost) as total\_cost  FROM (SELECT damage.cityID as cid, damage.cost  FROM Earthquake eq, damage  WHERE eq.magnitude > 7.0 and eq.qname = damage.qname) as tmp  NATURAL RIGHT OUTER JOIN Cities city  GROUP BY city.cid; |
|  |

## 丁5.

|  |
| --- |
| SELECT eq.fault, AVG(eq.magnitude)  FROM Earthquake eq  GROUP BY eq.fault; |
|  |

## 丁6.

|  |
| --- |
| SELECT eq.qname, damage.cityID, damage.liveslost  FROM Earthquake eq NATURAL LEFT OUTER JOIN damage; |
| 另外做測試 |

## 丁7.

|  |
| --- |
| SELECT city.cname  FROM Cities city  WHERE city.population > (SELECT AVG(city2.population)  FROM Cities city2); |
|  |

## 戊0(create table).

|  |  |
| --- | --- |
| CREATE TABLE Manufacturer(  maker varchar(25) PRIMARY KEY NOT NULL  ); | |
| CREATE TABLE PC(  model varchar(25) PRIMARY KEY NOT NULL,  speed float ,  ram varchar(10) ,  hd varchar(10) ,  price int  ); | |
|  | |
| CREATE TABLE Laptop(  model varchar(25) PRIMARY KEY NOT NULL,  speed float ,  ram varchar(10) ,  hd varchar(10) ,  screen int ,  price int  ); | |
| CREATE TABLE Printer(  model varchar(25) PRIMARY KEY NOT NULL,  color varchar(25) ,  type varchar(25) ,  price int  ); | |
|  | |
| CREATE TABLE Product(  maker varchar(25) NOT NULL,  model varchar(25) NOT NULL,  type varchar(25),  PRIMARY KEY(maker, model),  CONSTRAINT product\_maker\_fk FOREIGN KEY(maker)  REFERENCES Manufacturer(maker)  ); | |
|  | |
| Printer table    Product table | Manufacturer table |
| PC table |
| Laptop table |

## constraint of product

為了保證所有model都在PC, Laptop, Printer其中一個table中，故寫trigger在product insert時觸發，將model放入對應的table中。

|  |
| --- |
| DELIMITER $$  CREATE TRIGGER product\_insert\_add  AFTER INSERT ON product  FOR EACH ROW  BEGIN  IF new.type = "pc" THEN  INSERT IGNORE INTO PC(model, speed, ram, hd, price)  VALUES (new.model, null, null, null, null);  END IF;  IF new.type ="laptop" THEN  INSERT IGNORE INTO Laptop (model,speed,ram,hd,screen,price)  VALUES (new.model, null, null, null, null, null);  END IF;  IF new.type ="printer" THEN  INSERT IGNORE INTO Printer (model,color,type,price)  VALUES (new.model, null, null, null);  END IF;  END$$  DELIMITER ; |
|  |

## 戊1.

|  |
| --- |
| SELECT DISTINCT pd.maker  FROM Product pd  WHERE pd.model in (SELECT PC.model  FROM PC  WHERE PC.speed >=3.0); |
|  |

## 戊2.

|  |
| --- |
| SELECT \*  FROM Printer  WHERE Printer.price = (SELECT MAX(p.price)  FROM Printer P); |
|  |

## 戊3.

|  |
| --- |
| SELECT \*  FROM Laptop  WHERE Laptop.speed < (SELECT MIN(PC.speed)  FROM PC); |
|  |

## 己0(create table).

|  |
| --- |
| CREATE TABLE Actor(  pid int PRIMARY KEY NOT NULL,  fname varchar(25) NOT NULL,  lname varchar(25) NOT NULL,  gender varchar(25)  ); |
|  |
| CREATE TABLE Movie(  mid int PRIMARY KEY NOT NULL,  name varchar(25) NOT NULL,  year int NOT NULL,  revenue int NOT NULL  ); |
|  |
| CREATE TABLE Directors(  did int PRIMARY KEY NOT NULL,  fname varchar(25) NOT NULL,  lname varchar(25) NOT NULL  ); |
|  |
| CREATE TABLE Casts(  pid int NOT NULL,  mid int NOT NULL,  role varchar(25) NOT NULL,  PRIMARY KEY(pid, mid, role),  CONSTRAINT actor\_pid\_fk FOREIGN KEY(pid)  REFERENCES Actor(pid),  CONSTRAINT movie\_mid\_fk FOREIGN KEY(mid)  REFERENCES Movie(mid)  ); |
|  |

|  |  |
| --- | --- |
| CREATE TABLE Movie\_directors(  did int NOT NULL,  mid int NOT NULL,  PRIMARY KEY(mid, did),  CONSTRAINT directors\_did\_fk FOREIGN KEY(did)  REFERENCES Directors(did),  CONSTRAINT moviedir\_mid\_fk FOREIGN KEY(mid)  REFERENCES Movie(mid)  ); | |
|  | |
| Acotr table | Casts table |

|  |  |
| --- | --- |
| Movie table | Movie\_directors table |
| Directors table |

## 己1.

|  |
| --- |
| SELECT d.fname, d.lname, COUNT(m.mid) as number  FROM Directors d, Movie\_directors md, Movie m  WHERE d.did = md.did and m.mid = md.mid  GROUP BY d.did  HAVING COUNT(m.mid) >= 200; |
|  |

## 己2.

|  |
| --- |
| SELECT mov.year, COUNT(tmp.mid)  FROM Movie mov NATURAL LEFT OUTER JOIN  (SELECT m.mid  FROM Movie m  WHERE m.mid not in (SELECT cast.mid  FROM Casts cast, Actor a  WHERE cast.pid = a.pid and a.gender != 'female')  ) as tmp  GROUP BY mov.year  ORDER BY mov.year; |
|  |