COMP 231 Database Management Systems		Name:	
Midterm Examination		Student ID:	
October 26, 2001 10:00 – 10:45		Lab section:	

1) [5] Given a relation R, which of the followings can uniquely identify the tuples in R? Circle the correct one(s).

- [✓ primary key of R]
- [✓ candidate key of R]
- [✓ super key of R]
- foreign key of R
- [✓ R itself]
- partial key of R

2) [30] The following relation records the biological father and mother of people. If a person has a child, his person_id will be used as the child's father_id or mother_id, depending on the sex of the person.

Parent (person_id, father_id, mother_id)

i) **[8]** Write an SQL query to retrieval the grandfather of ~~"John Lee"~~ of the person with persion_id=1234.

```
select p.father_id
from  Parent c, Parent p
where c.father_id=p.person_id
and   c.person_id=1234
```

Know how to use alias to answer the question: 4 pts
Use of 'p' correctly (didn't use c instead of p): 2 pts
Use of 'c' and 'p' correctly (didn't reverse them): 2 pts

In English, both the father of your father and the father of your mother are your grandfather. The answer above is enough. I provide the complete solution for your reference.

```
select p.father_id
from Parent c, Parent p
where ( c.father_id=p.person_id
or      c.mother_id=p.person_id )
and   c.person_id=1234
```

Alternative, you can get the father of your father and the father of your mother with two SQL statements and union the result together:

```
( select … … c.father_id=p.person_id …)
UNION
( select … … c.mother_id=p.person_id )
```

ii) **[7]** Answer I) using tuple relation calculus.

```
{ p[father_id] | p ∈ Parent ∧ (∃ c ∈ Parent)
(c[father_id]=p[person_id] ∧ c[person_id]=1234 }
```

You can use either notations: c[father_id] or c.father_id

iii) **[15]** Write an SQL query to retrieval all the descendents (children, grand children, great grand children, etc) of ~~"Mary Cheung".~~ of the person with persion_id=1234.

Since this query requires recursion, you need to use temporary variable and loop to compute the result. That is, you need embedded SQL.

Embedded SQL is used (regardless of correctness): 7 pts
Declaration: 4 pts
Looping and use of variables: 4 pts

```
EXEC SQL
   declare c cursor for
   select person_id
   from Parent
   where mother_id=:id
   or    father_id=:id
END-EXEC
```

This gets the sons and daughters of 1234 and recursively 1234's grand sons and grand daughter.
I won't deduct any points if the students only gets the male branch only or the female branch only.

```
id-list=append(1234)

while ( id-list is not empty ) {
   id=get-element(id-list)
   EXEC SQL open c END-EXEC
   EXEC SQL fetch c into :descendent-id END-EXEC
   while ( result tuple exist ) {
      printf ("%d\n", descendent-id);
      id-list=append(descendent-id)
      EXEC SQL fetch c into :descendent-id END-EXEC
   }
}
EXEC SQL close c END-EXEC
```

- I use a combination of C and pseudo code syntax, but any understandable pseudo code is OK.
- I also assume that the variables id and descendent are defined elsewhere
- I assume that a list structure (id-list) has been implemented, the append function appends an element at the end of the list and the get-element functions obtains an element from the list. The list can be implemented with an array structure. *The details are optional, and no point will be deducted without the details. Some pseudo code illustrating the concept of looping down the family hierarch is enough.*

3) [30] You are given a library database consisting of the relations below.

BOOK	(<u>ISBN</u>, Author-ID, Author-Name, Title, No-Of-Copies)
STUDENT	(<u>Student-ID</u>, Name, Address)
LOAN	(<u>Student-ID, ISBN</u>, Due-Date)

ISBN is a unique serial number for books. Author-ID and Student-ID are HKID numbers.

a) **[10]** Write ***an SQL delete statement*** to delete all books that have zero copy in the library.

```
delete from book
where No-Of-Copies = 0
```

b) **[10]** Write ***an SQL query*** to list the student Ids of students who borrowed all the books authored by "Dik Lee".

```
SELECT Student_ID
FROM STUDENT
WHERE NOT EXISTS(
    SELECT *
    FROM BOOK
    WHERE Author_Name='Dik Lee'
    AND ISBN NOT IN(
        SELECT ISBN
        FROM LOAN
        WHERE STUDENT.Student_ID = LOAN.Student_ID))
```

c) [10] Write an SQL query to list the student Ids of students who have all of the borrowed books overdue. (In your SQL query, you can use Due-Date > "26/10/2001" to test if a book is overdue.)

```
SELECT Student_ID
FROM STUDENT
WHERE NOT EXISTS(
    SELECT *
    FROM LOAN
    WHERE LOAN.Student_ID = STUDENT.Student_ID
    AND Due_Date >= "26/10/2001")
```

Note that in b and c, we use

```
select studend_id from student where …
```

It would examine each student and see if the student satisfies the condition in the where clause.

You can also use:

```
select student_id from loan where …
```

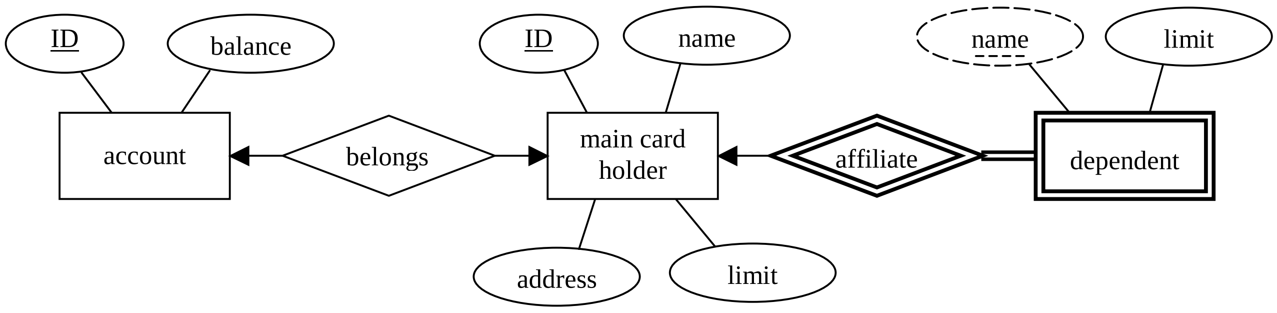
so that each student who had borrowed a book would be checked for the condition in the where clause. However, you must use alias since loan will then be used twice. Also, you have to use distinct since a student-id will appear in the loan table for as many times as the number of books he/she borrowed.

3) [35] You are hired by a credit-card company to design a database. After conducting an analysis on the requirements, you come to the following conclusions:

- Cardholders can be either main cardholders or dependent cardholders.
- A dependent cardholder must be affiliated with (i.e., sponsored or supported by) one and only one main cardholder.
- A main cardholder has an account, while a dependent cardholder uses the same account as his/her sponsor.
- Accounts have unique account Ids. Each account records the balance (i.e., unpaid expenses) of the account.
- For main cardholders, the database records their ID#, which are unique, names, addresses, and credit limits (i.e., the maximum amount of money they can charge to their credit cards).
- For dependent cardholders, the database records their names and credit limits.

Conditions that are obvious from a real world operation are not listed above. In your design, write down any assumptions you have made.

Draw an ER diagram for the database. Indicate clearly the cardinalities, keys and existential constraints.



We assume that an account has only one main cardholder, and a main cardholder can have only one account with the credit card company. This is not explicitly stated in the question, but will simplify the design.