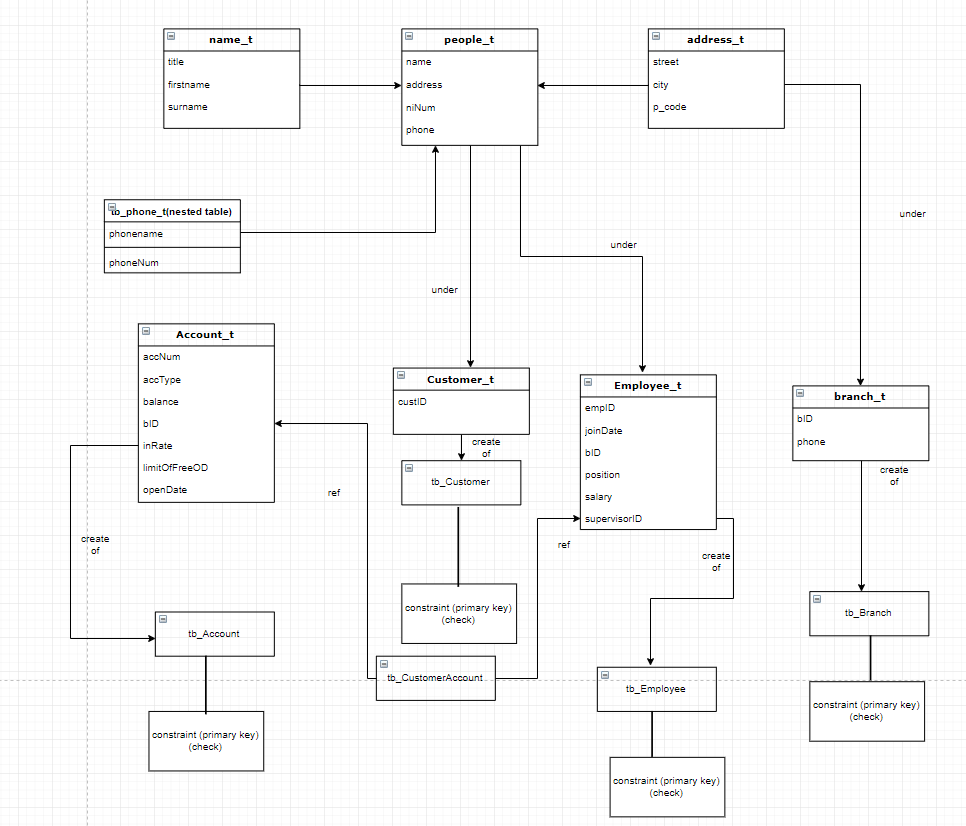
**SET09107: Coursework**

**Name: SHAOZHUO LI**

**Matriculation Number**: 40413818

**Task 1**:The ER diagram of database system.



**Task 2**:

After analysing about the features of object-relational database, I did the re-design task and this section will complain that the characteristics of object-relational database and re-designed database detailly. And also some critical discussion about my re-design.

There some features of the object-relational database, types, inheritance, collections and reference and constraint.

1. **Structured Types**

The first one is types, different with other relational database system, when we creating SQL database, the first thing we need to do is that we need to create the type. It is easy to find that there are some same attributes when we using relational database, so in order to decrease our the number of our codes, we use object-relational database. For example, we just need to do like that’create table tablename of typename’ to create our table, it is easier than we creating tables in relational database.

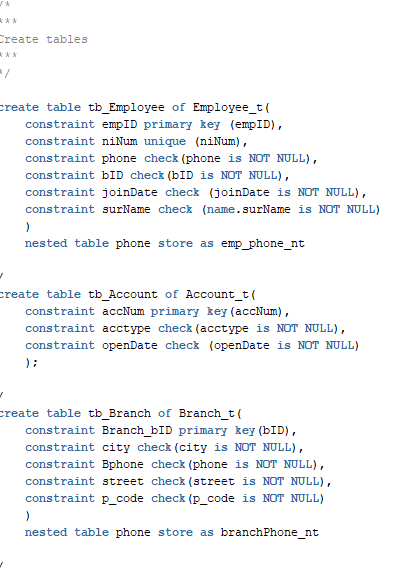
When I design the database, I created each type for the four table, I divided my types into five parts, phone\_type, people\_type, address\_type, name\_type and account\_type and use ‘under’ to create subtypes, the purpose of that is decrease the repeated codes. For example, customer\_type, branch\_type, employee\_type are all have the ‘street’, ‘city’ and ‘postcode’ attributes. What’s more, they also have the name properties, so the name type can also be used to create the three types. And the datatypes in SQL database are also easier to set in SQL database than in other relational database. For example, the VARCHAR2 type can be used to set all most all attributes.

1. **Inheritance**

The inheritance means that after creating one type, we can use this type to create another type when there are same properties during the two types. The code is that ‘create or replace type type2 under type1 ()’. We called the first type ‘supertype’, the second type ‘subtype’. And the attributes in the first type are also in the second type, and we can add some new and unique attributes for the second type, but these columns don’t appear in the father type. For example, I created ‘employee\_type’ and ‘customer\_type’ use the ‘people\_type’ and the’branch\_type’ under the ‘address\_type’. That is the inheritance, we can use it to create many types quickly and it makes our database more structured. When we want to create one subtype under supertype, the supertype must be not final. And one thing need to remembered is that SQL doesn’t support multiple inheritance.

1. **Reference and methods**

SQL database is able to create and refer objects, reference is the way that can only be used in create table. The function of it is like the ‘foreign key’ in relational database. The referenced table must have an object identifier for each tuple. And there are three functions of references, ref(), value() and deref(), the function of ref() is that inserting the data, and it can also be used to find references. Using value() to find objects in one table and using deref() ro return references. In this coursework, the customeraccount table is created by using reference, and the important thing when we creating one referenced table is ref type not table. The code of creating one referenced table is that: create table tablename (cloumn1 ref type1 scope is table\_type1). And another useful characteristic is methods, which can be viewed as the association with structured types. There are two kinds of methods, member methods and the static methods. What’s more, methods can be part of type definition of one structured type and the methods body need to be created separately. And we can define methods to achieve the function we want. In my re-design, I also used member methods to get the name( get\_name ) and address( get\_address), here are the code of them.



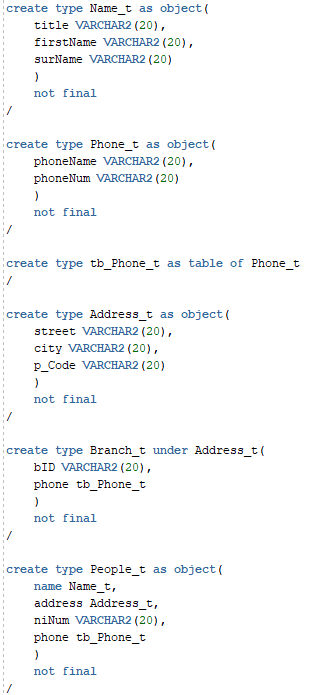
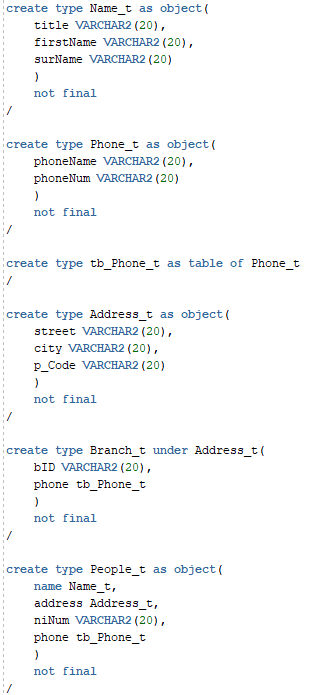
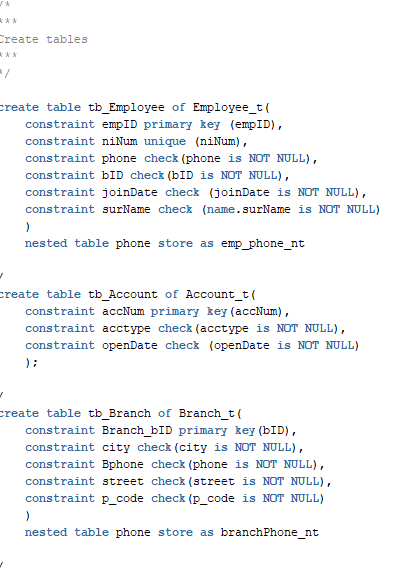
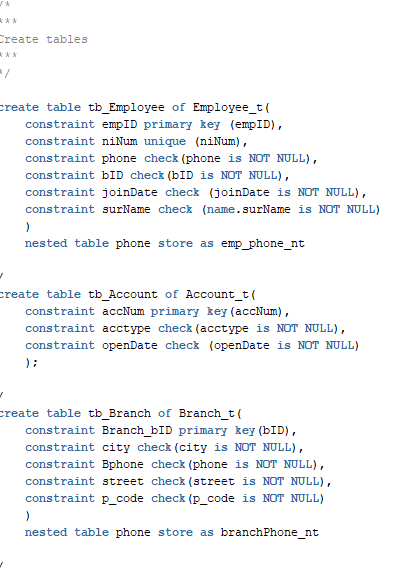
1. **Collection and Constraint**

Constraints is also one unique function in SQL, it has four kinds, primary key constraint, unique constraint, check constraint and not null constraint. Primary key constraint is used to identify the primary key of one table, requiring the primary columns are unique, which is similar with the relational database. Primary key constraint can be set when creating one table, and can also be added after creating one table. Check constraint is used to check the data we inserted and make sure the data is in range. For example, it can make sure that the value of city is one of Edinburgh, Glasgow and St. Andrews. Not null constraint is used to make sure the value of some columns never be empty. For example, the employeeID cannot contain NULL value. Unique constraint is used to ensure that the value can not be same in one column. Collections in oracle database has two data types, varrays and nested-table. Varrays are variable-length ordered lists and the maximum size must be specified when an attribute of type varray is defined and cannot be changed. Nested-tables are tables created in one table and they are unordered and both varrays and nested-tables must be created type definition firstly.

**Task 3**:

1. **Create types and tables**

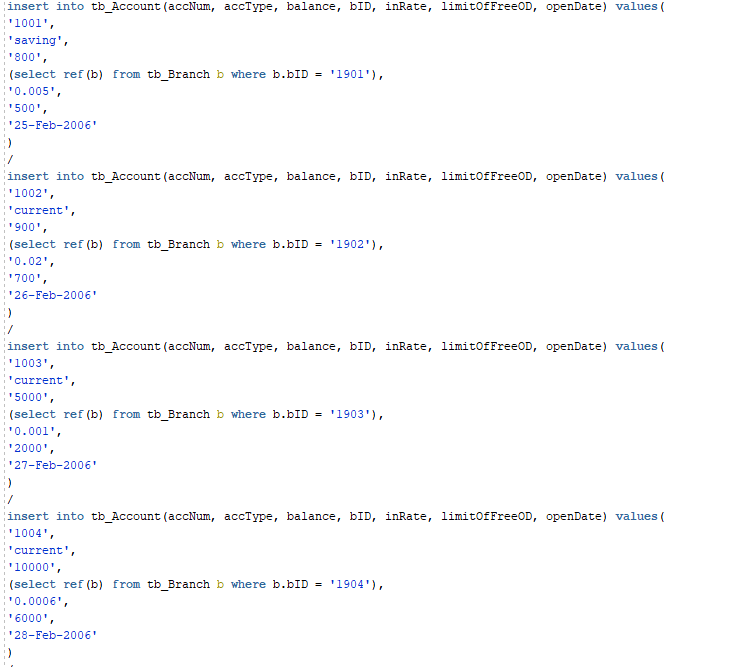
I have created 8 types and one nested-table-type and the five tables

1. **Inserting data**

Here are some pictures of my inserting data



**Task 4**:

Question a

select e.name.title||' '||

e.name.firstName||' '||

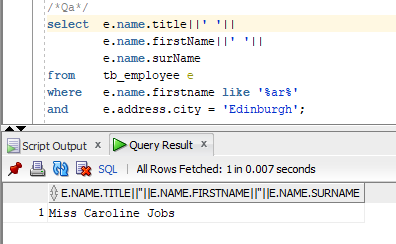
e.name.surName

from tb\_employee e

where e.name.firstname like '%ar%'

and e.address.city = 'Edinburgh';

/\* Output \*/



Question b

select a.bID.bID as "BID",

a.bID.city as "CITY",

count(a.accType) as "NUMBER"

from tb\_Account a

where accType = 'saving'

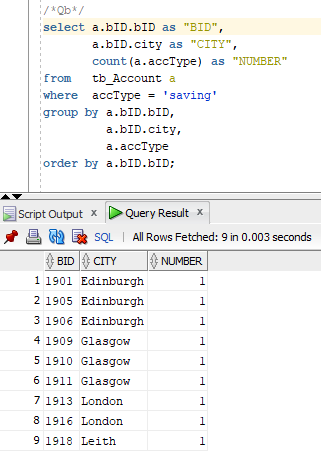
group by a.bID.bID,

a.bID.city,

a.accType

order by a.bID.bID;

/\* Output \*/



Question c

alter type People\_t

add member function get\_name return varchar2,

add member function get\_address return varchar2 cascade;

/

create or replace type body People\_t as

member function get\_name return VARCHAR2 is

begin

return name.title|| ' '|| name.firstName || name.surName;

end get\_name;

member function get\_address return VARCHAR2 is

begin

return address.street|| ', '|| address.city || ' ' || address.p\_Code;

end get\_address;

end;

/

select

c.accNum.bID.bID AS BID,

c.custID.get\_name() AS Name,

c.accNum.balance AS Balance

from (

select

c.accNum.bID.bID AS bID,

c.accNum.accType AS acctype,

MAX (c.accNum.balance) AS Maxbalance

from

tb\_CustomerAccount c

where

c.accNum.accType = 'saving'

group by c.accNum.bID.bID, c.accNum.accType

) balance

join tb\_CustomerAccount c

on

c.accNum.bID.bID = balance.bID

and

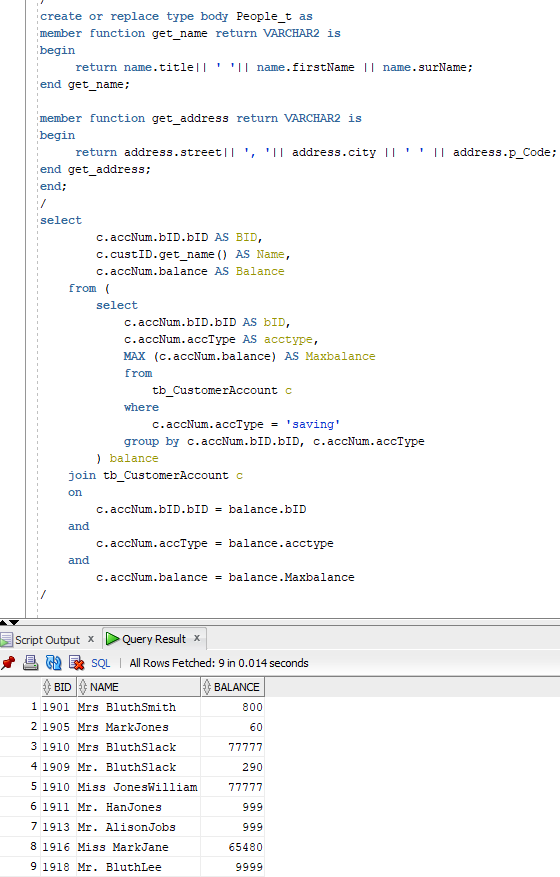
c.accNum.accType = balance.acctype

and

c.accNum.balance = balance.Maxbalance

/

/\* Output \*/



Question d

select

e.get\_address() as Workaddress

from

tb\_Employee e, tb\_CustomerAccount c

where

c.custID.name.firstName = e.name.firstName

and

c.custID.name.surName = e.name.surName

and

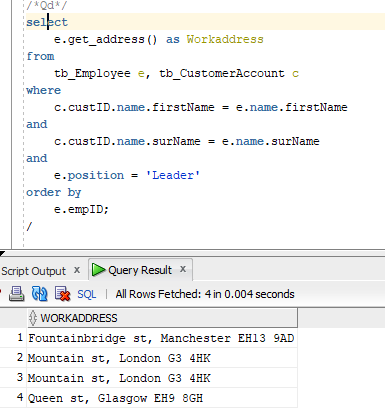
e.position = 'Leader'

order by

e.empID;

/

/\* Output \*/



Question e

select

c.accNum.bID.bID AS BID,

c.custID.get\_name() AS Fullname,

c.accNum.limitOfFreeOD AS freeOD

FROM (

SELECT c.accNum.bID.bID AS branch\_id,

MAX(c.accNum.limitOfFreeOD) AS maxOD

FROM tb\_CustomerAccount c

GROUP BY c.accNum.bID.bID

) maxOD, tb\_CustomerAccount c

WHERE c.accNum.limitOfFreeOD = maxOD.maxOD

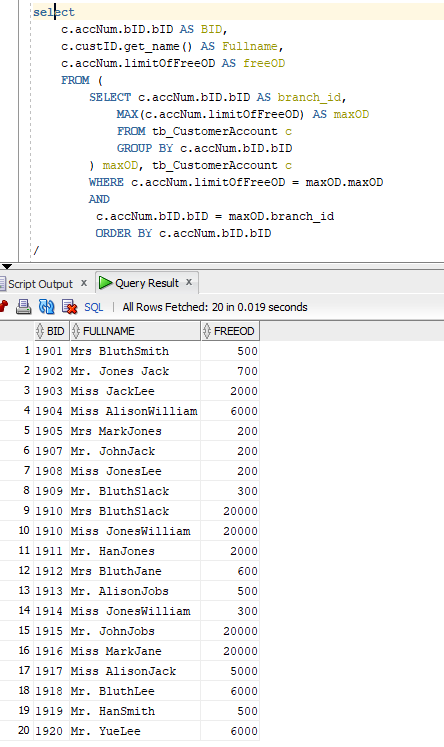
AND

c.accNum.bID.bID = maxOD.branch\_id

ORDER BY c.accNum.bID.bID

/

/\* Output \*/



Question g

select

count(e.get\_name()) as NumberOfEmployee,

e.supervisorID.get\_name() as SupervisorOfEmployee

from tb\_Employee e

where e.supervisorID.name.surName = 'Smith'

and e.supervisorID.name.title = 'Mr.'

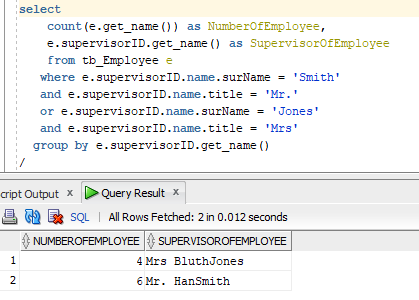
or e.supervisorID.name.surName = 'Jones'

and e.supervisorID.name.title = 'Mrs'

group by e.supervisorID.get\_name()

/

/\* Output \*/



**Task 5**:

After finish the coursework, I had a better understand of object-relational database and this section will talk about both the advantages and disadvantages of these two kinds of database.

Object-relational database

Object-relational database has four clear features structured types, inheritance, references and methods and collections and constraints. All of these characteristics make it as a whole and it decreases the time we spend on creating tables and use methods we can finish some complex problems that we can’t achieve them using relational database. However, it is a little bit difficult to use it correctly, collection and constraints and methods and references are hard to master and some codes are hard to understand. For example, when I using ref() to insert data, I need to know clearly about the relationships between two tables otherwise, I may

Relational database

Relational database is easy to understand and the statement is simple, but it doesn’t have design function to solve problems and there are so many duplicate attributes which makes our statement disordered and the structure is not clear. Compare to using relational database, I prefer to use object-relational database as I create one database with nice structure and is easy to change it.

**Task 6**.

drop table tb\_Employee;

drop table tb\_Account;

drop table tb\_Branch;

drop table tb\_Customer;

drop table tb\_CustomerAccount;

drop type Account\_t force;

drop type Name\_t force;

drop type Address\_t force;

drop type Branch\_t force;

drop type People\_t force;

drop type Employee\_t force;

drop type Customer\_t force;

drop type Phone\_t force;

drop type tb\_Phone\_t force;