SET9107: Advanced Database Systems

2019/2020

Ismael Souf

40326941

Abstract

A bank has several branches in the UK. It needs a database to store information about its local branches. Each branch is identified by a unique branch code, an address (street, city, post code), and a phone number. The customer accounts at each branch are also recorded.

Each customer account is identified by a unique account number, an account type (current or savings), and a balance. Each account has an interest rate (interest rate can be determined by yourself - any reasonable one will be fine). An account is also associated with exactly one branch. The date when the account is opened is recorded as well. An account must be classified as either a current or a savings account (but not both). A current account also has a limit of free overdraft (overdraft can be determined by yourself - any reasonable one will be fine). The free overdraft limit is set at the opening of an account.

Data about customers and employees is also recorded. All customers and employees have an associated National Insurance number (a tax payer's unique identification number), address (street, city, post code) and phone numbers (home number and mobile numbers). An employee cannot be a customer at the same branch where he/she works. An employee has a job position (Head, Manager, Project Leader, Accountant, Cashier) and a salary, and works for exactly one branch. The date that the employee joined the bank is also recorded. Every employee has a supervisor at the same branch, except the head of the branch. The supervisor is either the head, a manager or a team leader. The head of the branch is the only person who is not supervised by anyone at the same branch. A customer may have multiple accounts with the bank, and an account may be owned by multiple customers as a joint account.

Content

1. Introduction

The aim of the coursework is to re-design the database to capture more of the semantics of the application making use of the object-relational features as extensively as possible while still retaining the semantics of the application. In order to demonstrate the object-relational design decisions, it needs to implement the database and unambiguously answer the queries provided.

2. ER Diagram

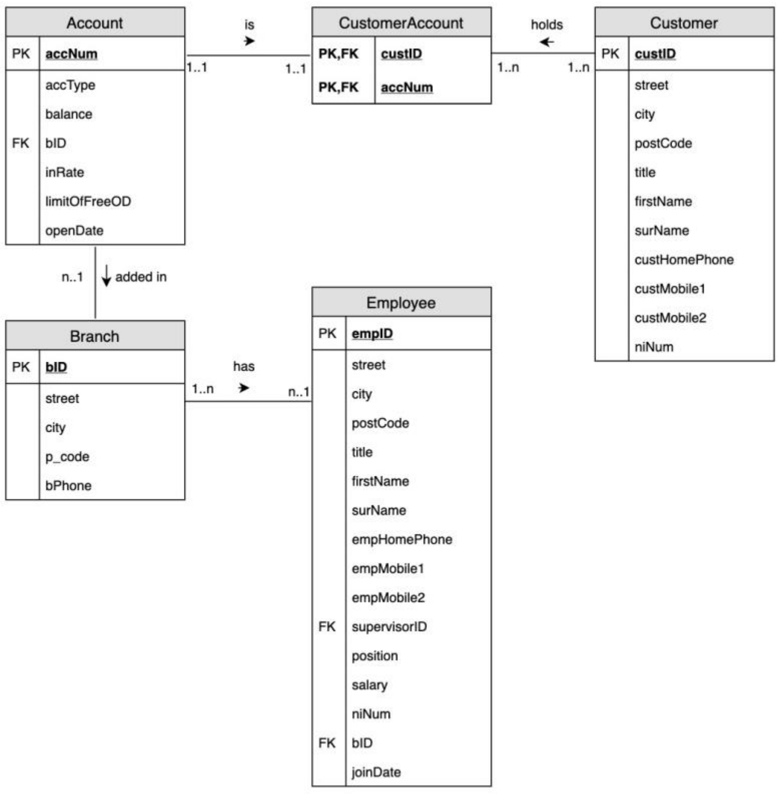


Fig1. ER Diagram of the bank application

3. Re-design the database

3.1 Object-relation model design

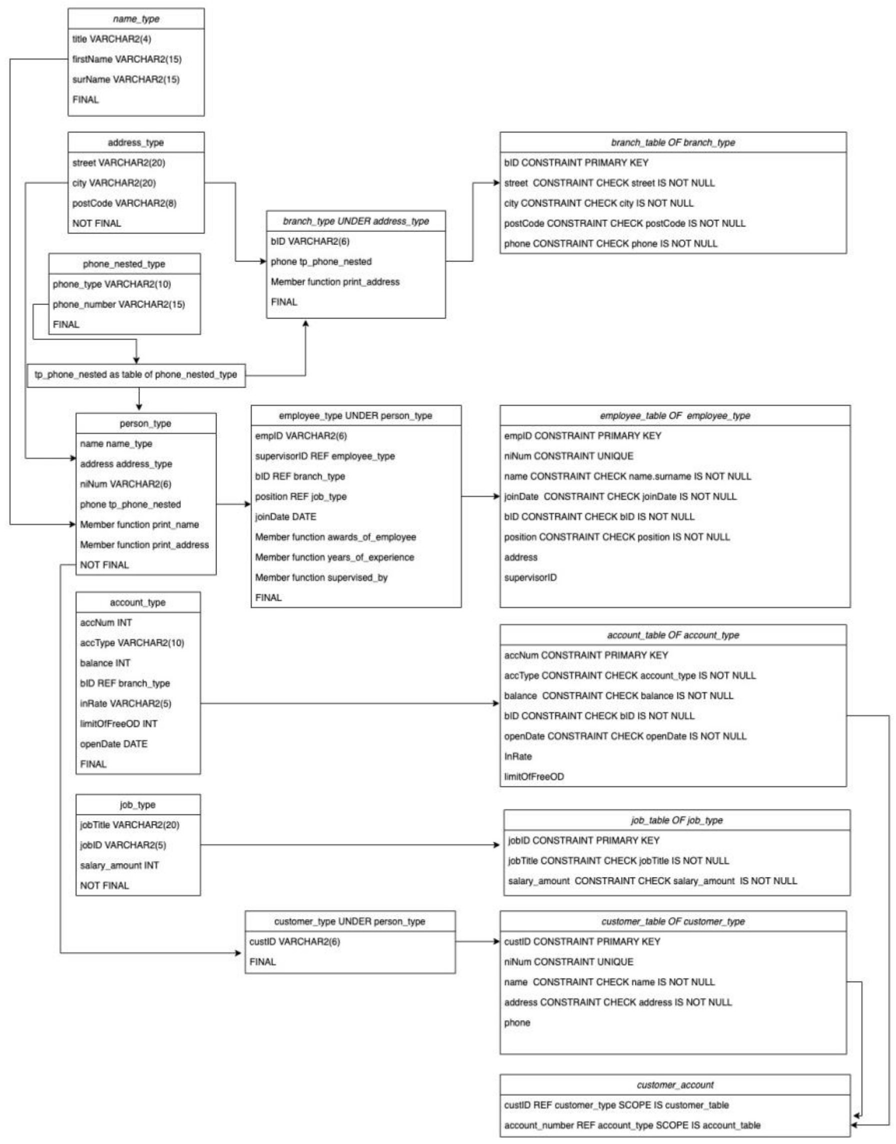

name_type
title VARCHAR2(4)
firstName VARCHAR2(15)
surName VARCHAR2(15)
FINAL
address_type
street VARCHAR2(20)
city VARCHAR2(20)
postCode VARCHAR2(8)
NOT FINAL
phone_nested_type
phone_type VARCHAR2(10)
phone_number VARCHAR2(15)
FINAL
tp_phone_nested as table of phone_nested_type
branch_type UNDER address_type
bID VARCHAR2(6)
phone tp_phone_nested
Member function print_address
FINAL
branch_table OF branch_type
bID CONSTRAINT PRIMARY KEY
street CONSTRAINT CHECK street IS NOT NULL
city CONSTRAINT CHECK city IS NOT NULL
postCode CONSTRAINT CHECK postCode IS NOT NULL
phone CONSTRAINT CHECK phone IS NOT NULL
person_type
name name_type
address address_type
niNum VARCHAR2(6)
phone tp_phone_nested
Member function print_name
Member function print_address
NOT FINAL
employee_type UNDER person_type
empID VARCHAR2(6)
supervisorID REF employee_type
bID REF branch_type
position REF job_type
joinDate DATE
Member function awards_of_employee
Member function years_of_experience
Member function supervised_by
FINAL
employee_table OF employee_type
empID CONSTRAINT PRIMARY KEY
niNum CONSTRAINT UNIQUE
name CONSTRAINT CHECK name.surname IS NOT NULL
joinDate CONSTRAINT CHECK joinDate IS NOT NULL
bID CONSTRAINT CHECK bID IS NOT NULL
position CONSTRAINT CHECK position IS NOT NULL
address
supervisorID
account_type
accNum INT
accType VARCHAR2(10)
balance INT
bID REF branch_type
inRate VARCHAR2(5)
limitOfFreeOD INT
openDate DATE
FINAL
account_table OF account_type
accNum CONSTRAINT PRIMARY KEY
accType CONSTRAINT CHECK account_type IS NOT NULL
balance CONSTRAINT CHECK balance IS NOT NULL
bID CONSTRAINT CHECK bID IS NOT NULL
openDate CONSTRAINT CHECK openDate IS NOT NULL
InRate
limitOfFreeOD
job_type
jobTitle VARCHAR2(20)
jobID VARCHAR2(5)
salary_amount INT
NOT FINAL
job_table OF job_type
jobID CONSTRAINT PRIMARY KEY
jobTitle CONSTRAINT CHECK jobTitle IS NOT NULL
salary_amount CONSTRAINT CHECK salary_amount IS NOT NULL
customer_type UNDER person_type
custID VARCHAR2(6)
FINAL
customer_table OF customer_type
custID CONSTRAINT PRIMARY KEY
niNum CONSTRAINT UNIQUE
name CONSTRAINT CHECK name IS NOT NULL
address CONSTRAINT CHECK address IS NOT NULL
phone
customer_account
custID REF customer_type SCOPE IS customer_table
account_number REF account_type SCOPE IS account_table

Fig2. Diagram of object-relational model

3.2 Description of design

This part of the report includes all object-relational features that are expected to be used in our database such as structured types, inheritances, references, methods, constraints and collections. The object-relational model combines features of the object-oriented model and relational model.

A structure type is a user-defined data type which contain one or more attributes. For example, in our design we can see different structures types such as "name_type" containing the attribute "title", "firstname", "surname. More structures types can be seen (eg. in the Fig2). In each type, we can find specification that allow the type to have a subtype. The first specification is called "FINAL" and is used to restrict the use of subtype. In the other hand, "NOT FINAL" indicates that the subtypes are allowed, which can be seen in the "address_type". The second feature used in our database is subtype. A subtype is generally related to the supertype. Subtypes can be created in order to give extra attributes to the supertype. For example, the type "employee_type" is considered as a subtype of "person_type". In fact, the subtype employee_type consider the all the attributes of person_type and add more attributes such as empID,joinDate… At the same type a supertype is an entity type that has a relationship with one or more subtypes, and it contains attributes that are common to its subtypes. This can be seen in the person_type which has two subtypes: employee_types and customer_types. Both subtypes inherit from the attributes of person_types. It is important to notice that the supertype must be "NOT FINAL". Finally, a supertype can be changed even after some subtypes have created by altering types.

The different types of the database helped us to create table which inherit from the supertypes, subtypes in order to make an object relational design.

A powerful feature that contributed to the object-relation model is reference. A reference is an object-oriented which provide the ability to create and refer to objects. References can only be scoped when defining a table. In order to replace joins, using references in a query is a

good option. Therefore, a reference can be used as a foreign key in 1-n relationships. In fact, three references have been used in the database: supervisorID, position and bID. These references have been set with a ref as a logical pointer to an instance object in the ref type.
For example, "supervisorID ref employee_table" it is possible to access everything related to the supervisor name, job, salary and the rest in the type. In the other hand, "scope is" has been used to restrict the references to actual object tables which can be seen in the customer_account for example.
References helped us remove the need of a JOIN function as required in an entity relational database while working with queries.

Methods also called member methods can be viewed as functions associated with structured types.
To implement a method, we need to create the PL/SQL code and specify it within a CREATE TYPE BODY statement.
In the database, methods have been added to existing types which can be considered as the member functions cited above. The cascade specification makes the alternations apply to existing dependent object tables.
Different member functions have been used in order to alter types of employee_types, person_types and branch_types. The member functions allow us in the employee_type to find the awards of employees by calculating their year of experience and look for the employees supervised.
Then, a member function print_name has been created in order to display the full name of employees. Eg. "Mrs Vanidha Smith" instead of title as "Mrs", firstname as "Vanidha "and surname as "Smith".
Finally, a member function print_address has been created in order to display the full address of the employees and customer but also the branches. For example, instead of displaying:
Street: "Crescent ", **City**: "Edinburgh", Postcode: "EH15 1AB",
the function allows us to display:
Address: "Crescent, Edinburgh, EH15 1AB".

Therefore, constraints are rules that restricts the values in a database. Among the six types of constraints I used 4 types in order to demonstrate the object-relational model. The first one is primary key constraint which identify a

not null and unique constraint in a single declaration. For example, in the database, in the "branch_table "the bID is a primary key constraint or in the "employee_table" empID is a primary key constraint.
Secondly, we can find unique constraint, it prohibits multiple rows from having the same value in the columns or combination of columns, but the values can be null. For example, an insurance number is unique for each employee or customer.
Then, check constraint validates incoming columns at row insert time. It has been used in order to check whether the values in the different tables are null. For example, "CONSTRAINT CHECK name IS NOT NULL".
A NOT NULL constraint restricts a database value from being null.
For example, a "firstname" may not be nul in the "employee_table".

Finally, collections are used in some of the most important performance optimization features or PL/SQL such as table functions in our case. They are PL/SQL functions that return collections and can be called in the FROM clause of a SELECT statement. Oracle supports two collection data types: Varrays and Nested tables. Nested tables are tables within table, in constrast to varrays, nested tables are unordered lists.
I have used nested tables in order to store two phones for the employees or customer. For example, a customer can have two phone number by using the table "tp_phone_nested".

The object-relational features that have been added into the database such as structure type, inheritance, methods, constraints or collections helped us extends the relational data model by including object orientation and constructs to deal with added data types.

3.3 Critical discussion on the rationale for the object-relational design

The basic objectives of an object-relational design are to prioritize a certain design cycle which include integrity, flexibility, performance and accessibility. In fact, after extending the ER model with notions of encapsulation, methods and object identity we have captured the semantics of object supported in object-oriented programming.

However, few details are important to be considered. For example, for the employees, even though the project leaders or managers have a better experience and are supervising more employees than other, their salary is equal. This is a problem in the database that need to be considered in order to give the opportunity to have an overview of the different projects done by the supervisor. Therefore, I decided to not add a constraint check on different attributes because of the real world. For example, in the "account_table" an interested rate could be null. The number datatype is the most common numeric datatype which is used in the world of Oracle and PL/SQL programming. This is one of the reasons I have decided to use it in order to find the awards of the employees. In fact, the only position that could be awarded are those who supervise the others such as Head, Project Leader and Manager.

3.4 Alternative possible object-relational

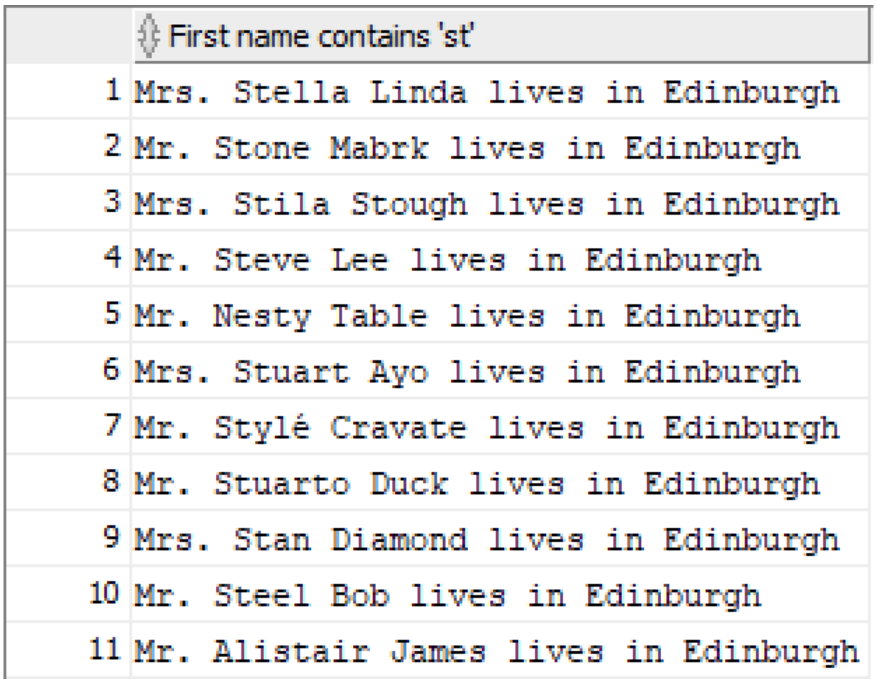
Two alternative object-relation has been possible. A foreign key constraint could have been used because it requires values in one table to match values in another table. For example, the bID is a foreign key in the "account_table", "employee_table" …
Therefore, instead of using nested tables it could have been Varrays. Varray tables are useful to SQL joins. However, a maximum size of the array must be specified when an attribute of type Varray is defined and cannot be used if the number of repeating items is unknown or very large.

4. SQL Statements

rCheck in the dbCreating for details on the methods.

4.1 Find employees whose first name includes the string “st” and live in Edinburgh, displaying their full names.

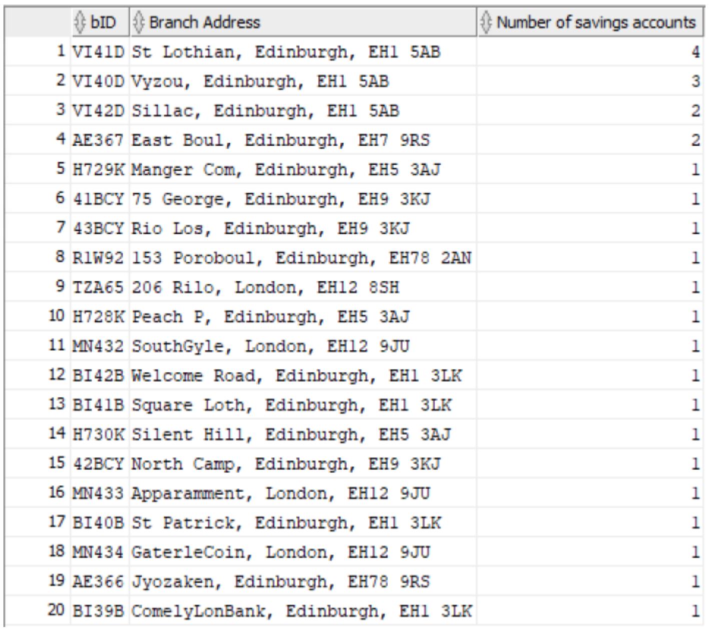
```
SELECT   e.name.title || '. ' ||
         e.name.firstName || ' ' ||
         e.name.surname || ' lives in ' ||
         e.address.city
AS       "First name contains 'st'"
FROM     employee_table e
WHERE    INSTR(lower(e.name.firstname),'st') > 0
AND      e.address.city = 'Edinburgh';
```



	First name contains 'st'
1	Mrs. Stella Linda lives in Edinburgh
2	Mr. Stone Mabrk lives in Edinburgh
3	Mrs. Stila Stough lives in Edinburgh
4	Mr. Steve Lee lives in Edinburgh
5	Mr. Nesty Table lives in Edinburgh
6	Mrs. Stuart Ayo lives in Edinburgh
7	Mr. Stylé Cravate lives in Edinburgh
8	Mr. Stuarto Duck lives in Edinburgh
9	Mrs. Stan Diamond lives in Edinburgh
10	Mr. Steel Bob lives in Edinburgh
11	Mr. Alistair James lives in Edinburgh

4.2 Find the number of saving accounts at each branch, displaying the number and branch's address.

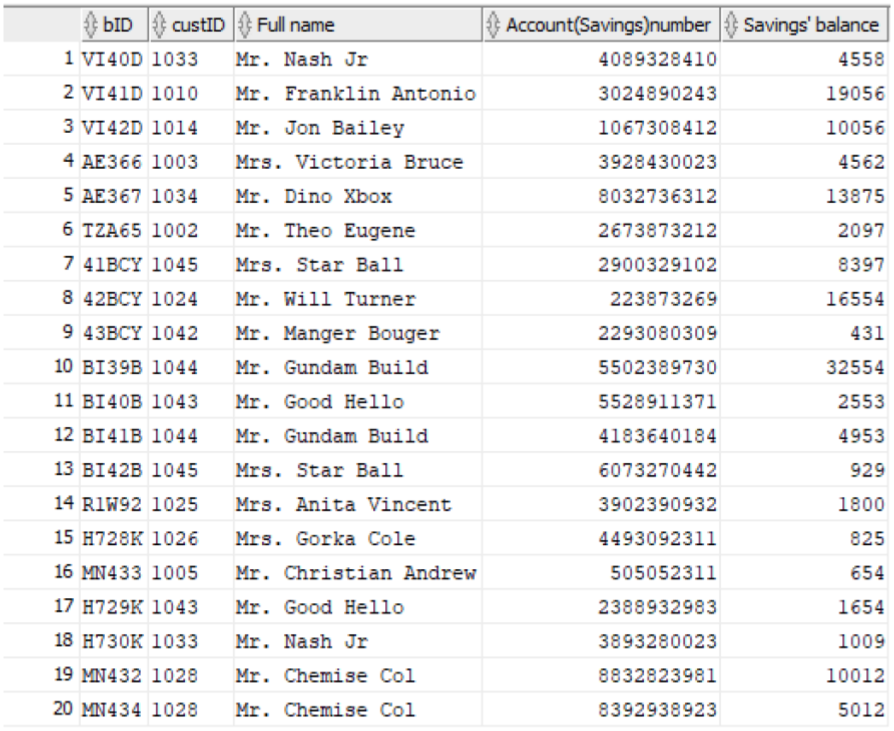
```
SELECT   a.bID.bID AS "bID",
         a.bID.print_address() AS "Branch Address",
         Count(a.acctype) AS "Number of savings accounts"
FROM     account_table a
WHERE    acctype = 'Savings'
GROUP BY a.acctype, a.bID.bID, a.bID.print_address()
             ORDER BY count(a.acctype) DESC;
```



	bID	Branch Address	Number of savings accounts
1	VI41D	St Lothian, Edinburgh, EH1 5AB	4
2	VI40D	Vyzou, Edinburgh, EH1 5AB	3
3	VI42D	Sillac, Edinburgh, EH1 5AB	2
4	AE367	East Boul, Edinburgh, EH7 9RS	2
5	H729K	Manger Com, Edinburgh, EH5 3AJ	1
6	41BCY	75 George, Edinburgh, EH9 3KJ	1
7	43BCY	Rio Los, Edinburgh, EH9 3KJ	1
8	R1W92	153 Poroboul, Edinburgh, EH78 2AN	1
9	TZA65	206 Rilo, London, EH12 8SH	1
10	H728K	Peach P, Edinburgh, EH5 3AJ	1
11	MN432	SouthGyle, London, EH12 9JU	1
12	BI42B	Welcome Road, Edinburgh, EH1 3LK	1
13	BI41B	Square Loth, Edinburgh, EH1 3LK	1
14	H730K	Silent Hill, Edinburgh, EH5 3AJ	1
15	42BCY	North Camp, Edinburgh, EH9 3KJ	1
16	MN433	Apparamment, London, EH12 9JU	1
17	BI40B	St Patrick, Edinburgh, EH1 3LK	1
18	MN434	GaterleCoin, London, EH12 9JU	1
19	AE366	Jyozaken, Edinburgh, EH78 9RS	1
20	BI39B	ComelyLonBank, Edinburgh, EH1 3LK	1

4.3 At each branch, find customers who have the highest balance in their savings account, displaying the branch ID, their names, and the balance.

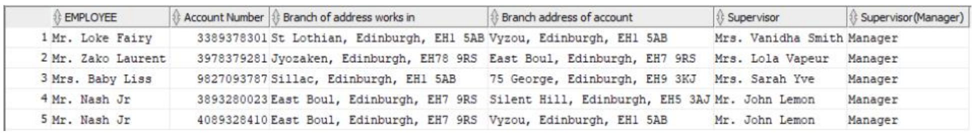
```
WITH balance
     AS (SELECT c.accnum.bid.bid      AS bID,
                c.accnum.acctype      AS acctype,
                /*Select the highest balance with MAX*/
                Max(c.accnum.balance) AS max_balance
         FROM   customer_account c
         WHERE  c.accnum.acctype = 'Savings'
         GROUP  BY c.accnum.bid.bid,
                   c.accnum.acctype)
SELECT c.accnum.bid.bid      AS "bID",
       c.custid.custid       AS "custID",
       c.custid.Print_name() AS "Full name",
       c.accnum.accnum       AS "Account(Savings)number",
       c.accnum.balance      AS "Savings' balance"
FROM   balance
       JOIN customer_account c
         ON c.accnum.bid.bid = balance.bid
            AND c.accnum.acctype = balance.acctype
            AND c.accnum.balance = balance.max_balance
```



	bID	custID	Full name	Account(Savings)number	Savings' balance
1	VI40D	1033	Mr. Nash Jr	4089328410	4558
2	VI41D	1010	Mr. Franklin Antonio	3024890243	19056
3	VI42D	1014	Mr. Jon Bailey	1067308412	10056
4	AE366	1003	Mrs. Victoria Bruce	3928430023	4562
5	AE367	1034	Mr. Dino Xbox	8032736312	13875
6	TZA65	1002	Mr. Theo Eugene	2673873212	2097
7	41BCY	1045	Mrs. Star Ball	2900329102	8397
8	42BCY	1024	Mr. Will Turner	223873269	16554
9	43BCY	1042	Mr. Manger Bouger	2293080309	431
10	BI39B	1044	Mr. Gundam Build	5502389730	32554
11	BI40B	1043	Mr. Good Hello	5528911371	2553
12	BI41B	1044	Mr. Gundam Build	4183640184	4953
13	BI42B	1045	Mrs. Star Ball	6073270442	929
14	R1W92	1025	Mrs. Anita Vincent	3902390932	1800
15	H728K	1026	Mrs. Gorka Cole	4493092311	825
16	MN433	1005	Mr. Christian Andrew	505052311	654
17	H729K	1043	Mr. Good Hello	2388932983	1654
18	H730K	1033	Mr. Nash Jr	3893280023	1009
19	MN432	1028	Mr. Chemise Col	8832823981	10012
20	MN434	1028	Mr. Chemise Col	8392938923	5012

4.4 Find employees who are supervised by a manager and have accounts in the bank, displaying the branch address that the employee works in and the branch address that the account is opened with.

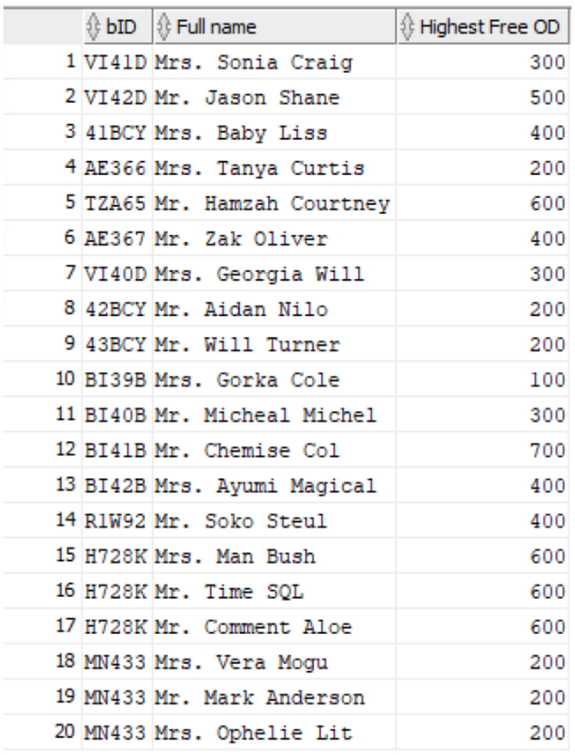
```
SELECT
    e.print_name()   AS employee,
    c.accnum.accnum AS "Account Number",
    e.bid.print_address() AS "Branch of address works
in",
    c.accnum.bid.print_address() AS "Branch address of
account",
    e.supervisorid.print_name() AS "Supervisor",
    e.supervisorid.position.jobtitle AS "Supervisor Job
Title"
FROM
    employee_table e, customer_account c
WHERE
    c.custid.name.firstname = e.name.firstname
AND
    c.custid.name.surname = e.name.surname
AND
    e.supervisorid.position.jobtitle = 'Manager'
```



	EMPLOYEE	Account Number	Branch of address works in	Branch address of account	Supervisor	Supervisor(Manager)
1	Mr. Loke Fairy	3389378301	St Lothian, Edinburgh, EH1 5AB	Vyzou, Edinburgh, EH1 5AB	Mrs. Vanidha Smith	Manager
2	Mr. Zako Laurent	3978379281	Jyozaken, Edinburgh, EH78 9RS	East Boul, Edinburgh, EH7 9RS	Mrs. Lola Vapeur	Manager
3	Mrs. Baby Liss	9827093787	Sillac, Edinburgh, EH1 5AB	75 George, Edinburgh, EH9 3KJ	Mrs. Sarah Yve	Manager
4	Mr. Nash Jr	3893280023	East Boul, Edinburgh, EH7 9RS	Silent Hill, Edinburgh, EH5 3AJ	Mr. John Lemon	Manager
5	Mr. Nash Jr	4089328410	East Boul, Edinburgh, EH7 9RS	Vyzou, Edinburgh, EH1 5AB	Mr. John Lemon	Manager

4.5 At each branch, find customers who have the highest free overdraft limit in all current accounts that are joint accounts, displaying the branch’s ID, the customer’s full names, the free overdraft limit in his/her current account.

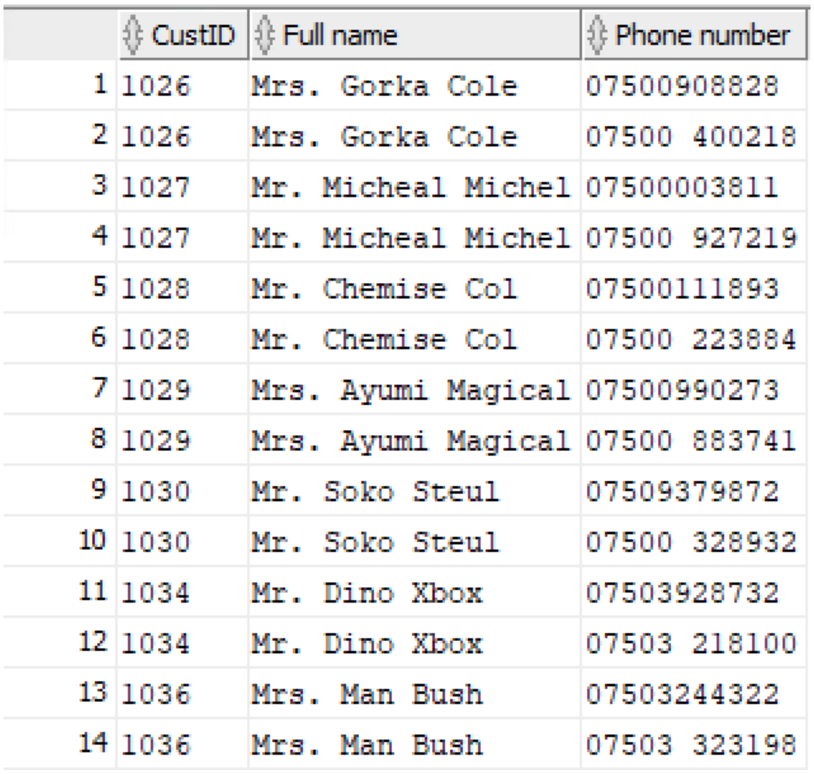
```
SELECT
    c.accnum.bid.bid AS "bID",
    c.custid.Print_name() AS "Full name",
    c.accnum.limit_of_free_od AS "Highest Free OD"
    FROM (
        SELECT c.accnum.bid.bid AS bID,
          Max(c.accnum.limit_of_free_od) AS highestFreeOD
            FROM customer_account c
            GROUP BY c.accnum.bid.bid
        ) highestFreeOD, customer_account c
WHERE
c.accnum.limit_of_free_od = highestFreeOD.highestfreeod
AND c.accnum.bid.bid = highestFreeOD.bid
AND c.accnum.acctype = 'Current'
```



	bID	Full name	Highest Free OD
1	VI41D	Mrs. Sonia Craig	300
2	VI42D	Mr. Jason Shane	500
3	41BCY	Mrs. Baby Liss	400
4	AE366	Mrs. Tanya Curtis	200
5	TZA65	Mr. Hamzah Courtney	600
6	AE367	Mr. Zak Oliver	400
7	VI40D	Mrs. Georgia Will	300
8	42BCY	Mr. Aidan Nilo	200
9	43BCY	Mr. Will Turner	200
10	BI39B	Mrs. Gorka Cole	100
11	BI40B	Mr. Micheal Michel	300
12	BI41B	Mr. Chemise Col	700
13	BI42B	Mrs. Ayumi Magical	400
14	R1W92	Mr. Soko Steul	400
15	H728K	Mrs. Man Bush	600
16	H728K	Mr. Time SQL	600
17	H728K	Mr. Comment Aloe	600
18	MN433	Mrs. Vera Mogu	200
19	MN433	Mr. Mark Anderson	200
20	MN433	Mrs. Ophelie Lit	200

4.6 Find customers who have more than one mobile, and at least one of the numbers starts with 0750, displaying the customer's full name and mobile numbers. COLLECTIONS must be used.

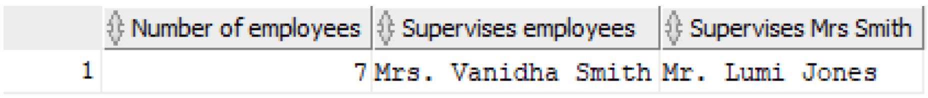
```
SELECT
    c.custid AS "CustID",
    c.print_name() AS "Full name",
    tp.phone_number AS "Phone number"
FROM
    (SELECT c.custid AS custID, Count(tp.phone_type) AS
mobile_count, phone_type AS phone_type
    FROM customer_table c, TABLE(c.phone) tp
    WHERE tp.phone_type = 'Mobile'
    AND tp.phone_number LIKE '0750%' GROUP BY c.custid,
phone_type) phone_number, customer_table c,
TABLE(c.phone) tp
WHERE c.custid = phone_number.custid
AND tp.phone_type = phone_number.phone_type
AND tp.phone_type = 'Mobile'
AND phone_number.mobile_count > 1
```



	CustID	Full name	Phone number
1	1026	Mrs. Gorka Cole	07500908828
2	1026	Mrs. Gorka Cole	07500 400218
3	1027	Mr. Micheal Michel	07500003811
4	1027	Mr. Micheal Michel	07500 927219
5	1028	Mr. Chemise Col	07500111893
6	1028	Mr. Chemise Col	07500 223884
7	1029	Mrs. Ayumi Magical	07500990273
8	1029	Mrs. Ayumi Magical	07500 883741
9	1030	Mr. Soko Steul	07509379872
10	1030	Mr. Soko Steul	07500 328932
11	1034	Mr. Dino Xbox	07503928732
12	1034	Mr. Dino Xbox	07503 218100
13	1036	Mrs. Man Bush	07503244322
14	1036	Mrs. Man Bush	07503 323198

4.7 Find the number of employees who are supervised by Mrs Smith, who is supervised by Mr Jones. REFERENCES must be used.

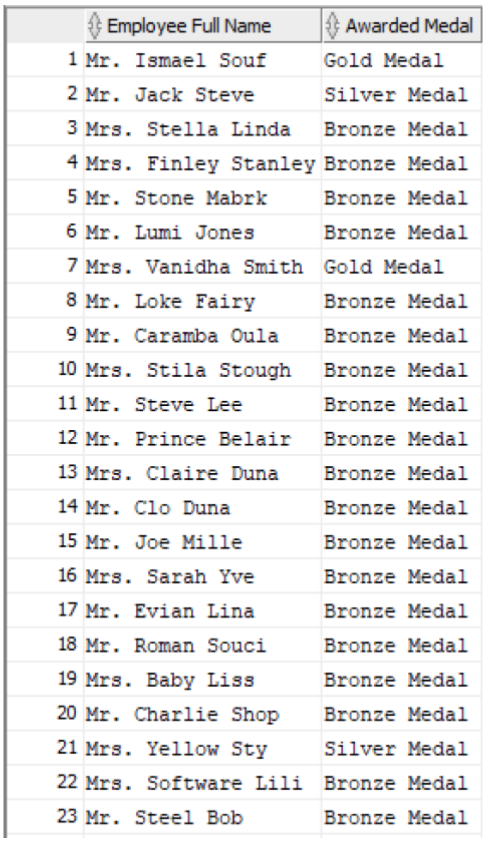
```
SELECT
/*Check in the dbCreating for details on the methods*/
    Count(e.print_name()) AS "Number of employees",
    e.supervisorid.Print_name() AS "Supervises
employees",
 (SELECT e.supervisorid.print_name() FROM employee_table
e WHERE e.supervisorid.name.surname = 'Jones') AS
"Supervises Mrs Smith"
    FROM employee_table e
    WHERE
        e.supervisorid.name.surname = 'Smith'
        AND e.supervisorid.empid = (
            SELECT e.empid FROM employee_table e
            WHERE
                e.supervisorid.name.surname = 'Jones')
    GROUP BY e.supervisorid.print_name() ;
```



	Number of employees	Supervises employees	Supervises Mrs Smith
1	7	Mrs. Vanidha Smith	Mr. Lumi Jones

4.8 Award employees at the end of a year: gold medals for employees who have been working at the bank for more than 12 years and supervised more than 6 staff; silver medals for employees who have been working at the bank for more than 8 years and supervised more than 3 staff; bronze medals for employees who have been working at the bank for more than 4 years. Displaying winners' names and Medal awarded (only displaying those who have been awarded). METHODS must be used.

```
SELECT
    e.print_name() AS "Employee Full Name" ,
/*Check in the dbCreating for details on the methods*/
    e.Awards() AS "Awarded Medal"
FROM
    employee_table e
WHERE
    e.Awards() != '0';
```



	Employee Full Name	Awarded Medal
1	Mr. Ismael Souf	Gold Medal
2	Mr. Jack Steve	Silver Medal
3	Mrs. Stella Linda	Bronze Medal
4	Mrs. Finley Stanley	Bronze Medal
5	Mr. Stone Mabrk	Bronze Medal
6	Mr. Lumi Jones	Bronze Medal
7	Mrs. Vanidha Smith	Gold Medal
8	Mr. Loke Fairy	Bronze Medal
9	Mr. Caramba Oula	Bronze Medal
10	Mrs. Stila Stough	Bronze Medal
11	Mr. Steve Lee	Bronze Medal
12	Mr. Prince Belair	Bronze Medal
13	Mrs. Claire Duna	Bronze Medal
14	Mr. Clo Duna	Bronze Medal
15	Mr. Joe Mille	Bronze Medal
16	Mrs. Sarah Yve	Bronze Medal
17	Mr. Evian Lina	Bronze Medal
18	Mr. Roman Souci	Bronze Medal
19	Mrs. Baby Liss	Bronze Medal
20	Mr. Charlie Shop	Bronze Medal
21	Mrs. Yellow Sty	Silver Medal
22	Mrs. Software Lili	Bronze Medal
23	Mr. Steel Bob	Bronze Medal

5. Advantages and disadvantages of the object-relational model against the relational model

An entity-relational model is a top-down approach to database design. The relational model is based on two simple concepts which are tables and relations. It involves relational keys such as Primary keys and foreign keys but also integrity constraints. In other words, it consists of a collection of entities and relationships among the entities. The relational model is a simple relation database and have a simple formal model and semantics whereas the object-relational combine features of the object-oriented model and relational model. The object-oriented model captures the semantics of objects supported in object-oriented programming. First of all, the relational model cannot use the expression of nested relationships. Object-relational model allows the attributes to have complex types, including non-atomic values such as nested relations. In fact, in our object-relational model we have easily used nested table in order to have two phone numbers which could be impossible to do with a relational model in order to answer the query involving collections. Therefore, in the relational model the users cannot defines types which is very important in our study. For example, only fixed numbers of built-in types are available. An important feature that has been used in our study is methods. Methods cannot be written in a relational model this is a huge disadvantage over the object-relational model. It appears that in a relational model, databases can sometimes be very complex with the amount of data and the relations between the data. This is one of the reason object-relational model has been used over the relational model. O-O concepts in the object-relational model give us the opportunity to work with complex objects when a relational-model cannot represent or manipulate complex entities as a single unit. A key feature of the object-relational model is the use of SCOPE IS when you declare the REF which has been used in our study. Finally, manipulating objects in PL/SQL is also an advantage over the relational model. For example, to access or change a value of an attribute dot notation has been used. The object-relational has various advantages over the relational-model by combining the features of object-oriented model and relational model critically discussed above.