**Advanced Database Systems Report (SET09107)**

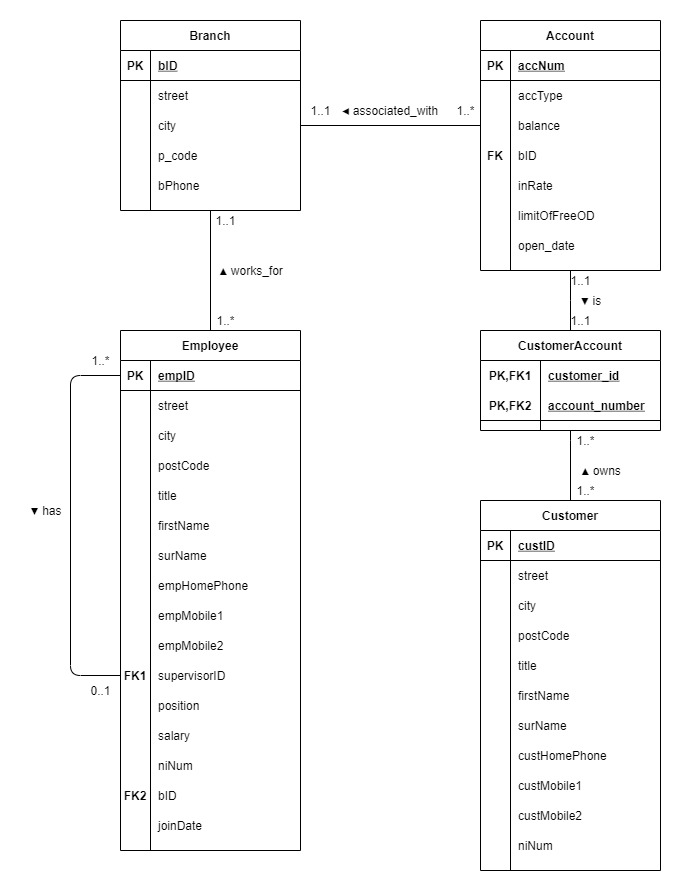
Valeri Vladimirov

40399682

School of Computing, Edinburgh Napier University, Edinburgh

40399682@napier.live.ac.uk

1. **ER Diagram**

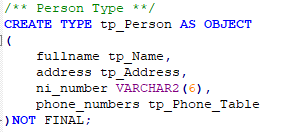


This is the ER diagram corresponding to the relational database schema and the scenario. Each employee has exactly 1 supervisor and works for exactly 1 branch. However, if the employee has a job position as ‘Head’ he doesn’t have a supervisor therefore the relationship is 0..1. A supervisor can supervise 1 or more employees. A branch can have 1 or many employees and 1 or many accounts can be created in the branch. Each account is associated with only 1 branch and has a 1 to 1 relationship with the Customer Account table. Each customer can have 1 or more customer accounts and each customer account can be a joint account (used by more than 1 customers).

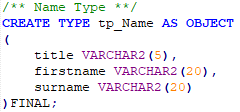
1. **Re-design**
   1. **Structured Types**

The new design of the database uses the object relational approach having 7 types, 3 subtypes and 6 tables. Using types and subtypes improves code redundancy as it can be seen from the Employee and Customer subtypes, which inherit properties from their Person supertype. The use of types in the object relation approach also reduces complexity, makes the code more reusable, readable and reliable, while also improving the quality and performance. Here are all of the structured types used:

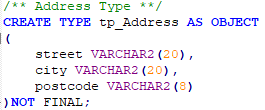
**Person Type** – a structured type having attributes for fullname, address, ni\_number and phone\_numbers. This type is a supertype from which the Employee and Customer types inherit. The Person type was declared as NOT FINAL meaning that subtypes are allowed. The Person type has 3 methods: PrintName, PrintAddress, PrintPhoneNumbers. PrintName and PrintAddress are mainly used to make the printing of the properties faster and with less code in the SELECT statements. The PrintPhoneNumbers method was used to print all of the phone numbers of the person on one line (used for Question F).



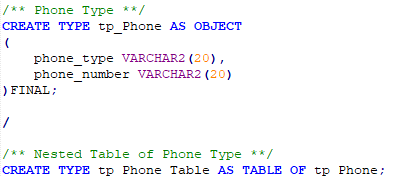
**Name Type –** a structured type having attributes for title, name and surname. It is a composite attribute which is used in other tables in the database. An example is the Person Type fullname attribute, which can be seen above.



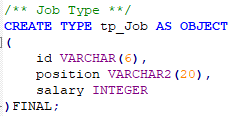
**Address Type** – a structured type having attributes for street, city, and postcode. This type is a supertype from which the Branch type inherits from. The Address type was declared as NOT FINAL meaning that subtypes are allowed. The Address type also has a method for PrintAddress, which reduces code in the SELECT statement (an example use is Question D).



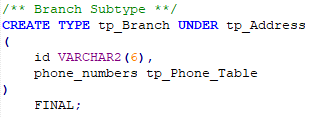
**Phone Type** – a structured type having attributes for phone\_type and phone\_number and also another type used to create a nested table for the customers, employees or branches which will have more than 1 number. Why this has been done will be explained in the Collections section.



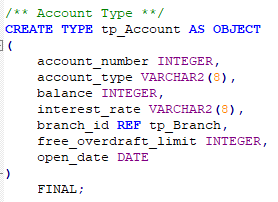
**Job Type** – a structured type having attributes for id, position and salary. It is also a composite attribute for which there will be references in the Employee type. Declared as FINAL since there won’t be any subtypes of it.



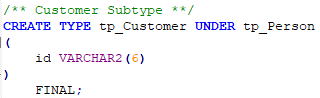
**Branch Type** – a structured type having attributes for id and phone\_numbers. It is a subtype of the Address supertype inheriting its properties and methods. Declared as FINAL since there won’t be any subtypes of it.



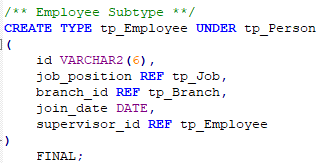
**Account Type** – a structured type having attributes for account\_number, account\_type, balance, interest\_rate, branch\_id, free\_overdraft\_limit and open\_date. It includes a reference to the Branch type and is declared as FINAL since there won’t be any subtypes of it.



**Customer Type** – a structured type having attributes for id. It is a subtype of the Person Type, which means that it inherits all of its properties and methods. Declared as FINAL since there won’t be any subtypes of it.



**Employee Type** – a structured type having attributes for id, job\_position, branch\_id, join date, supervisor\_id. It is a subtype of the Person Type, which means that it inherits all of its properties and methods. It also has an Award method, which is used for Question H and it counts how many years they have been working and how many employees they have supervised and then awards them based on the results. Declared as FINAL since there won’t be any subtypes of it. It also includes various references to the Job, Branch types and also a reference was used for the supervisor (referencing to the same type).



I have decided to create 6 tables to store the data from the structured types. When creating these tables I have applied various constraints to the attributes which can be seen in the Constraints section. The tables created are: Customer Table, Job Table, Branch Table, Employee Table, Account Table, Customer Account Table and can be seen in the DBCreating.sql file.

* 1. **Inheritance**

In the redesign of the database inheritance was used. This is due to the Object Relational approach, which is similar to Object Orientated Programming. Inheritance allows sub-types to inherit attributes from object types (supertypes) and this ensures reusability, reliability and code redundancy. An example of inheritance is how the Employee and Customer subtypes inherit attributes and methods from the Person Type.

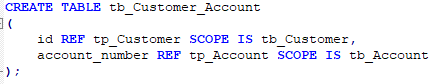


* 1. **References**

References are features of the object relational model which behave as foreign keys. They are used as logical pointers to a row object and one of their main advantages is that they replace JOINs in queries. There are many examples of REF used in the redesign of the database, such as referencing the Job type, Branch type. Reference has also been used for the supervisor (in Employee Type), which makes all of the information about the Employee’s supervisor available and quarriable. References are useful because they optimize the speed of queries and allow tables to contain more appropriate information.



The “SCOPE IS” functionality can be used to restrict the REF to actual object tables as it can be seen when creating the Customer Account table.



* 1. **Methods**

Member functions were used because they improve code redundancy (reduce repeated code) in the SELECT statements. They also make the code easier to understand. The methods are PrintName, PrintAddress, PrintPhoneNumbers and Award. More information about them was written above.

* 1. **Constraints**

Throughout the design I have used all of the constraints covered in the lectures. They were used to specify rules for the data in the tables, which ensure reliability and accuracy. They also improve the quality, performance and data integrity of the data.

**Primary Key Constraint** – was used to identify the primary key for the table, ensuring that these columns are unique.



**“Check” Constraint** – was used to validate incoming columns at row insert time, ensuring they match a given criteria. This constraint was used with the Not Null Constraint and for the account type checking if it is Current or Savings.





**Not Null Constraint** – as mentioned above it was used with the Check and it is used to specify that a column may never contain a NULL value.



**Unique Constraint** – was used to ensure that there are no duplicate entries for all the column values in the table.



* 1. **Collections**

Nested tables are one of the two collection types which Oracle supports. Nested tables (tables within tables) have been used for the current redesign of the database. They were used for the phone type, because there will be circumstances in which an employer, customer or a branch can have more than one phone number. Nested tables provide various advantages such as having no limit (no upper bound) for how many phone numbers there are, and they increase the flexibility. Nested tables are unordered lists, and they prohibit NULL values when there aren’t any phone numbers for the column.



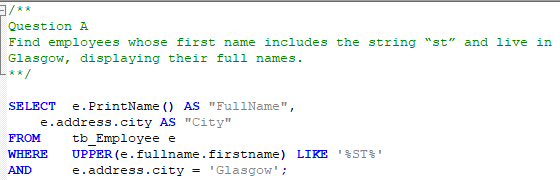


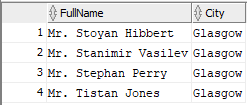
* 1. **Alternative Design**

One alternative design would have been using Varrays rather than nested tables for the phone numbers. However, varrays were not implemented because they require a maximum size of the array to be specified which was considered a limitation for the current design and reduced the flexibility. The advantages of nested tables over varrays were that they have no limitation for the number of phone numbers entered (no upper bound) and that they are unordered lists (varrays are ordered lists), also they can store different data types. This makes the nested table more flexible and preferable for the current design.

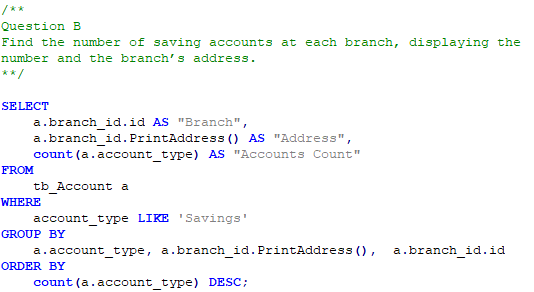
Firstly, I thought of using Employee and Customer as types and not subtypes. However, the final design is with Employee and Customer being subtypes of the Person supertype, this decision was made because of the hierarchical approach and because it improves code redundancy. Also, inheritance had to be used for the object relational approach, and these subtypes were perfect examples.

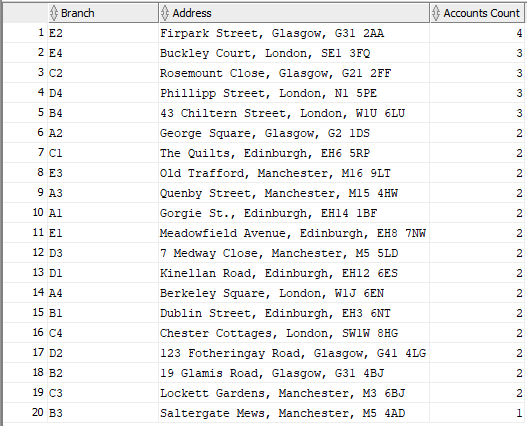
1. **SQL Statements**
   1. The first question uses the PrintName() method and searches for first names that have the ‘ST’ string, and the city is Glasgow. Examples show that it finds occasions where ‘ST’ is in the start of the name and in the middle.

****

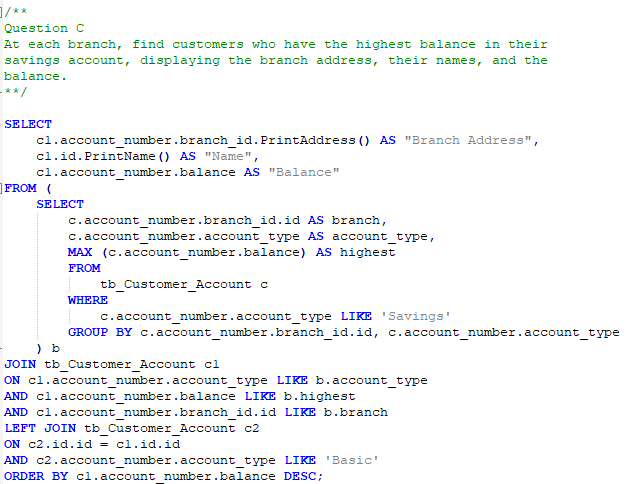
****

* 1. The second question uses the PrintAddress method to display the branch addresses. It searches for ‘Savings’ accounts in each branch and counts them. Grouping them and then ordering by descending. All 20 branches were included as examples.

****

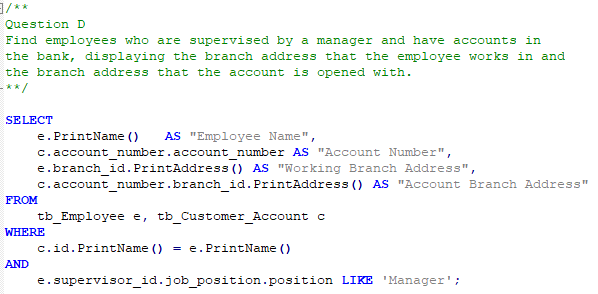
****

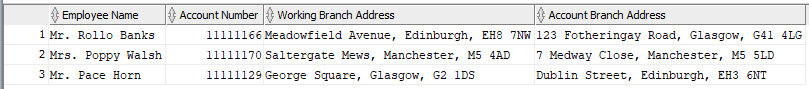
* 1. The third question uses both PrintAddress and PrintName methods. It has subquery (nested SELECT in a SELECT statement), which is used to find the highest balance for each ‘Savings’ account. There are examples for each branch (20).

****

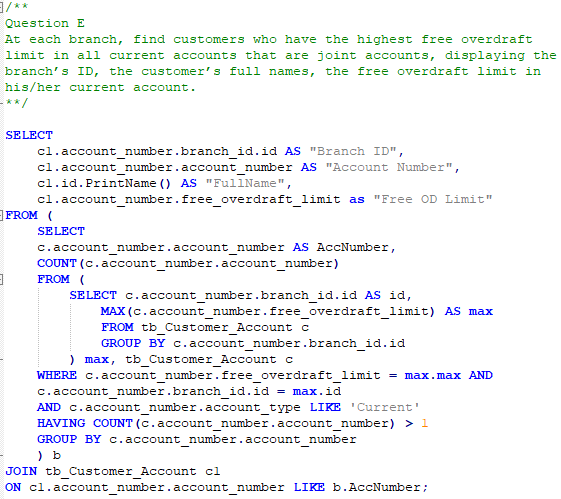


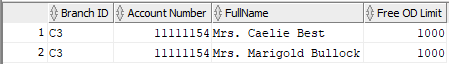
* 1. The fourth question uses 3 methods: PrintName and both PrintAddress for branch and person. It finds if an employee works at a branch by checking if the names are equal and then checking if the supervisor is a ‘Manager’.

****

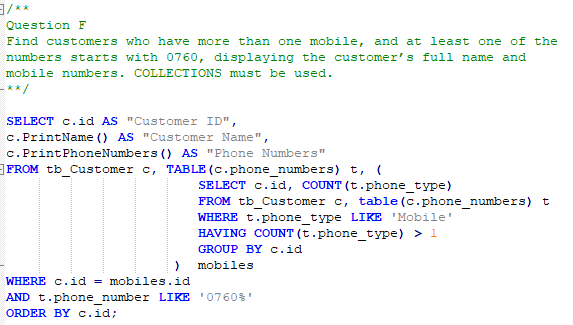


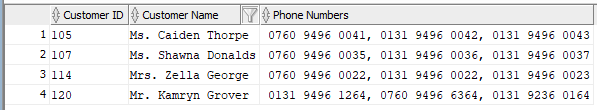
* 1. The fifth question uses a few nested SELECT statements. The most inner select is used to find the account in the branch with the highest free overdraft limit. The second select is used to find if the account is a ‘Current’ one and also to check if it is a joint account by using the Count function. In the example there were 2 joint accounts in C3 branch one with 500 and one with 1000 OD limit.

****

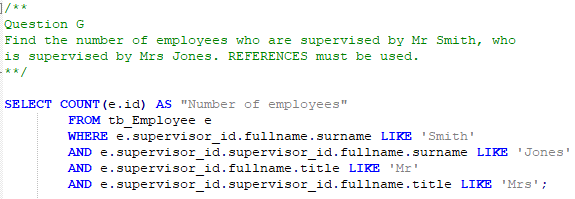
****

* 1. The sixth question uses the PrintName method and the PrintPhoneNumbers method. The print phone numbers method is used to display all numbers in 1 string rather than showing only 1 of the customers numbers and adds comas. This query also has a subquery which checks for who has more than 1 ‘Mobile’ numbers. Also checks if the number starts with ‘0760’.

****

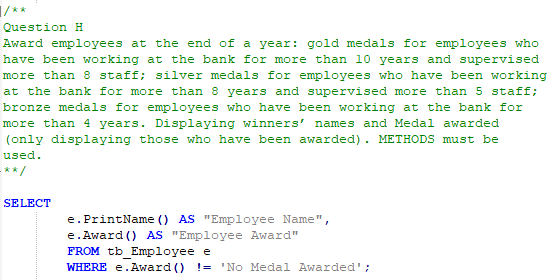
****

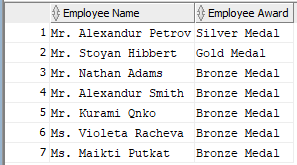
* 1. The seventh question counts the number of employees and uses References by checking which person is supervised by Mr Smith who is supervised by Mrs Jones.

****

****

* 1. The last question uses the PrintName and Award methods. The Award method is used to count the number of people an employee has supervised and count the years the employee has worked for. Then using IF statements deciding which medal the employee deserves. It also has a WHERE clause which doesn’t show the employees which don’t have a medal awarded.

****



1. **Advantages and disadvantages of the object-relational model**

One of the disadvantages of the object-relational model is that some users can find it quite difficult and complicated since it utilizes the functionalities of both the relational data model and object orientated data model. Opposite to the object-relational model is the relational model, which is simple to learn, easier to use and more widely adopted.

One advantage of the object relational model over the relational model is that in the object relational model you can create and use methods to print the real values and not the types. Throughout the coursework I have used multiple methods, examples include the PrintName(), Award(), PrintAddress(). These methods allow the faster printing of the attributes (in one column rather than multiple) and also less code in the SELECT statement, which means it improves the speed of the queries and the performance. If the relational model had been used this wouldn’t have been possible.

Another advantage of the object relational model over the relational one is the use of composite attributes. In the coursework an example of a composite attribute is the use of the Address(Street, City, Postcode) and Name(Title, Firstname, Surname) types in the Person type. The information is condensed into one column which saves time and memory, improving the queries performance.

The object relational model has the ability to map heterogeneous collections to tables, which the relational model cannot do. This makes possible to access and display different data together, an example of that is the Phone numbers nested table, allowing to save and display multiple phone numbers together.

Encapsulation is another advantage of the object relational model over the relational model. Encapsulation improves the speed and performance of the queries by storing code inside of the database and this is not possible with the relational model.

Overall, the relational model has a few advantages such as being a simple model, supporting data independence, having good ad-hoc query facilities, good storage management, good concurrency and is fast and efficient. However, it has its drawbacks as mentioned above and also it can’t run long duration transactions, express nested relationships, write methods, represent complex entities as a single unit, there is only a fixed number of types and can’t sufficiently express data that does not map well to tables.

That is why the object relational model was chosen. Apart from the advantages mentioned above other advantages of the model are OOP features such as inheritance. This was used for the Person class being a supertype and the Employee, Customer types being subtypes, also for branch and address. Inheritance promotes reusability, reliability and reduces code redundancy. References is another feature which came in useful when doing the coursework. References to other tables were used because they improve the speed and performance for queries, making it quicker to call other types in the queries. Other advantages of the model include upwards compatibility with existing relational languages, preserving relational foundations, allowing attributes of tuple to have complex types.