

ADVANCED DATABASE SYSTEM CW1

### WEIGUANG RAN 40412989@live.napier.ac.uk

SET09107

15 March 2019

**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.

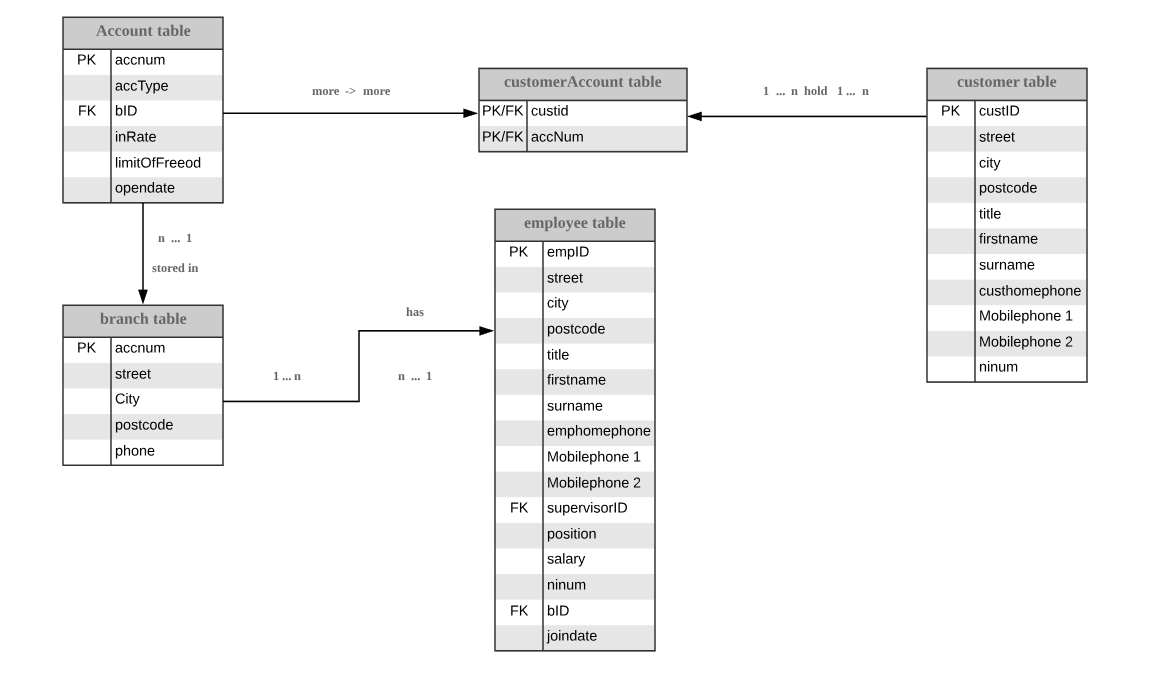
An employee has a job position (Head, Manager, Project Leader, Accountant, Cashier) and a salary, and works for exactly one branch. The date that the employee joined the bank is also recorded. Every employee has a supervisor at the same branch, except the head of the branch. The head of the branch is 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.

**Key words: WEIGUANG RAN, SET09107, ORACLE 11g, sql developer, Database**

**Contents**

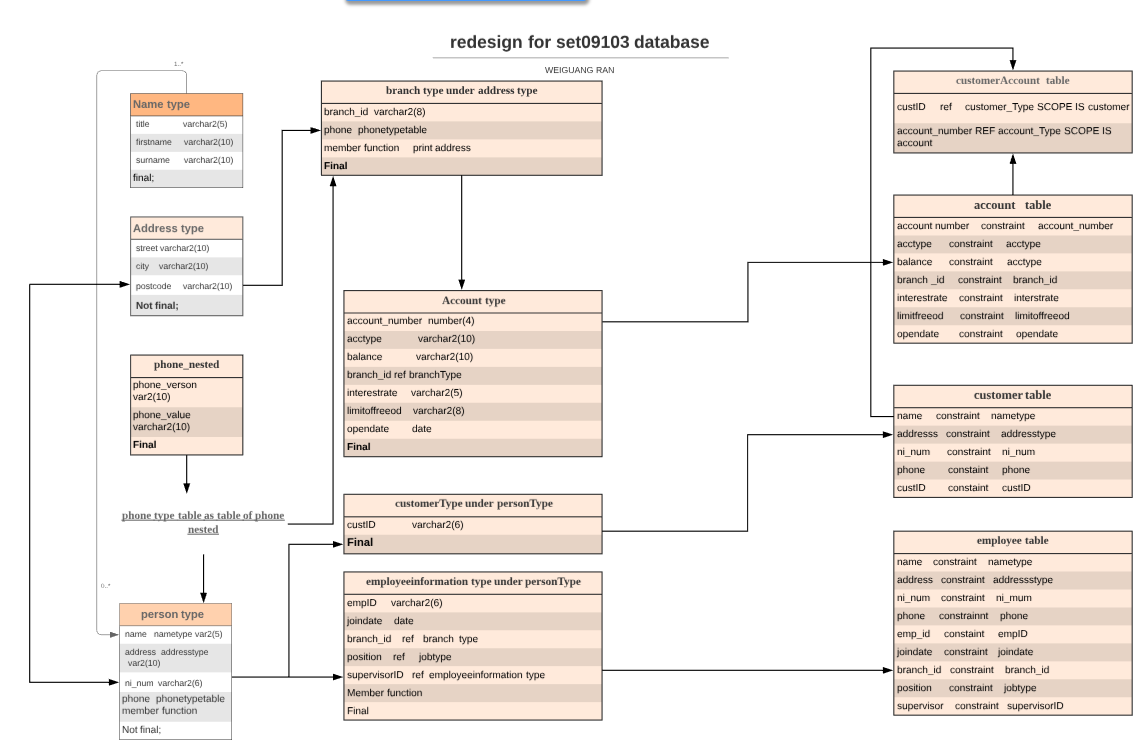
**1.Entity Relational Diagram of my coursework**



1. **Re-design the database**

Aim of re-design: The purpose of redesigning the database is to make the data import faster and more reasonable by creating reasonable objects and tables. In this process, the length, value and frequency of the data need to be considered. A good design is the key to the success of the database.

* 1. **re-design flow picture**

****

* 1. **How to re-design the database**

**2.2.1 Data flow**

Sub-type ---> Type ---> Sub-table ---> table

**2.2.2 Type**

**Definition of object type:** An **object type** is “a user-defined composite datatype that encapsulates a data structure along with the functions and procedures needed to manipulate the data”(1). Object type included two parts —— Object Type Specification and Object Type Body:

**Object Type Specification** : An interface between an object and an application that defines the public properties and methods of the object.

**Object Type Body:** used to implement the public method defined by the object type specification.

**Why use object types:** Object types can let people break down a large system into logical entities. This let people create software components that are modular, maintainable, and reusable across projects and teams.(2)It can help people define different object types and make each operate slightly differently. One more thing is that Object types allow for realistic data modeling.

**Type in this database:** In this database, originally I create 8 types and 5 tables with different properties, but when I insert data and doing test, I found jobs for different people made me confused, I need update the type and table, so I **add job type(have jobID, jobtitle and salary) and job table below job type**, then insert different jobs(head, manager, accounting and cashier into job table, this can be shown in my sql.doc. So finally I have 9 types and 6 tables.

**2.2.3 References**

**Definition of ref:** Thepurpose of ref takes as its argument a correlation variable(table alias) associated with a row of an object table or an object view. A ref value is returned for the object instance that is bound to the variables or row.(3) Reference is very strongly to use in object type, ref is kind of like foreign keys, taking information from other place(like type or other tables) to a new table, for example, in my database, when I want to use branchID data(in the branch type) in my account table, I can just write **select ref(b) from branch b where b.branch\_id = 'xxx'.** And then I get get different data connect which the branchid which i write down.

**2.2.4 Nested table**

A nested table is a table in a table. A nested table is a collection of rows that are represented in the main table as one of them. A nested table can contain multiple rows for each record in the primary table. In a sense, it is a way to store a one-to-many relationship in a table. And in this database, I create phone numbers as the nested table, firstly, I create phone\_nested type:

**CREATE TYPE phone\_nested AS OBJECT (**

**phone\_version VARCHAR2(20),**

**phone\_value VARCHAR2(20))**

**FINAL;**

Then,

**CREATE TYPE phone\_type\_table AS TABLE OF phone\_nested;**

When I create table and want to store phone numbers, I can write coding just like:

**CONSTRAINT phonenumber CHECK (phone IS NOT NULL),**

**NESTED TABLE phone STORE AS emp\_phone\_nested\_table;**

The characteristics of the nested table:

1. Object reuse: If you write object-oriented code, you have an increased chance of reusing previously written code modules. Similarly, if you create object-oriented database objects, you increase the chance that database objects can be reused.

2. Standard support: If you create standard objects, then their chances of being reused will increase.

3. Define the access path: For each object, the user can define the procedures and functions that run on it so that the data can be combined with the method of accessing the data. With access paths defined in this way, you can standardize data access methods and improve object reusability.

**2.2.5 Constraint**

The purpose of constraint is that defining an **integrity constraint--**a rule that restricts the values in a database. Normally there are six different types of constraint:(4)

**NOT NULL, Unique, primary key, foreign key, check** and **REF**

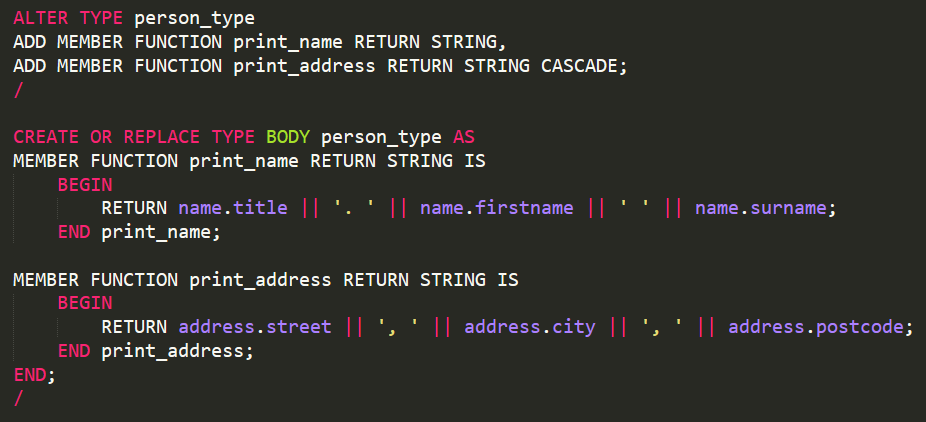
People can define constraints syntactically in two ways:(1) As part of the definition of an individual column or attribute. This is called **inline** specification. (2) As part of the table definition. This is called **out-of-line** specification.

**2.2.6 Object Methods**

Object methods, also known as subprograms, are functions or procedures that you can declare in an object type definition to implement behavior that you want objects of that type to perform. An application calls the subprograms to invoke the behavior.

Subprograms can be written in PL/SQL or virtually any other programming language. Methods written in PL/SQL or Java are stored in the database. Methods written in other languages, such as C, are stored externally.(5)

When you create an object methods, there is a fixed format, like **Create Function xxx, Begin xxxx end;** for example, In my database, there is a type like this:



Member function: member function is a special feature of oracle PL/SQL, this function is handle and run calculations on the data of the type that the member function is featured inside. And I use this method in my database for these things below:

1. In the person type, people can select the full name of employees, not the first, title or second name, the member function can put them together and show up at the same time. **Coding can be found below the person\_type.**
2. In the address type, I create a full address function, just the same like full name, this function allow system to show the full address, like: **64 parkhead avenue, Edinburgh, EH114SE**, not like just **64 parkheadavenue**, or just **Edinburgh** or just **postcode. Coding can be found below the address\_type.**

**2.2.7 Varrays**

The varray (variable size array) is one of the three types of collections in PL/SQL (associative array, nested table, varray). The varray's key distinguishing feature is that when you declare a varray type, you specify the maximum number of elements that can be defined in the varray.

The different between varrays and nest table:

1. Nested tables are unbounded, while mutable arrays have the largest size.
2. Individual elements can be removed from a nested table, but they cannot be removed from the varray.
3. Variables are stored online by Oracle (in the same tablespace), while nested table data is stored in a storage table, which is a system-generated database table associated with a nested table.
4. Nested tables support indexes, while mutable arrays do not support indexes.

But finally in my database, I decide to use nest table compare with the varrays. The most important reason is that the nested table support indexes, because if the database used for the website, phone number must one thing which customer often search, so using nest table will improve a lot speed. And one more reason is that I am more familiar with nest table.

**2.3 How the new database works**

Most part of the types and tables are one to one relational, for example, just like employee\_type under the employee table, the data which store in this type also only search by employee table. Customer type/Customer table, job type/job table the same as well. But the database also has one to more and more to more relational as well, for instance, phone type and person type is not one to one relational, the data which stored in there not only ref for one table, can be used for other tables as well. The table which named employee table job table and customer table are important, if an employee is a manager, he can be the supervisor to manage more people than a cashier or other lower level position works, if insert other position into the database. What is more, when I was doing the task, I found the method for assigning the awards need to refactored the table, so at last I add two new columns, which are used for the name of target tasks and the number of target tasks, also here I need to use if—else loop structure.

At first, I thought the job type is not necessary, I can just insert different job data, but when I insert data, I found the oracle cannot judge and allow insert more than one rows into one column at the same time, so I create the job type and job table to store the job data individually.

**2.4 Table screenshots**

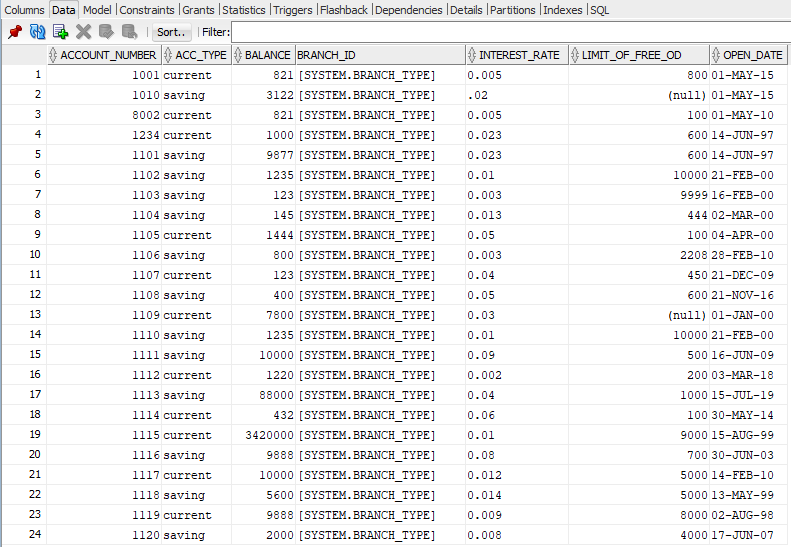


Figure1: Account table

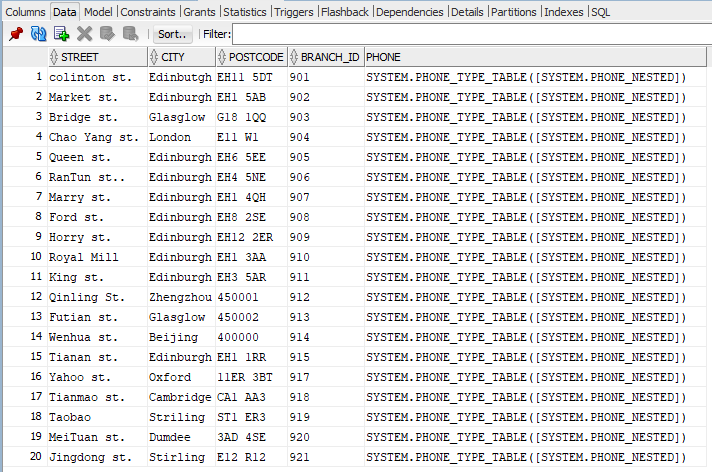


Figure2: branch table

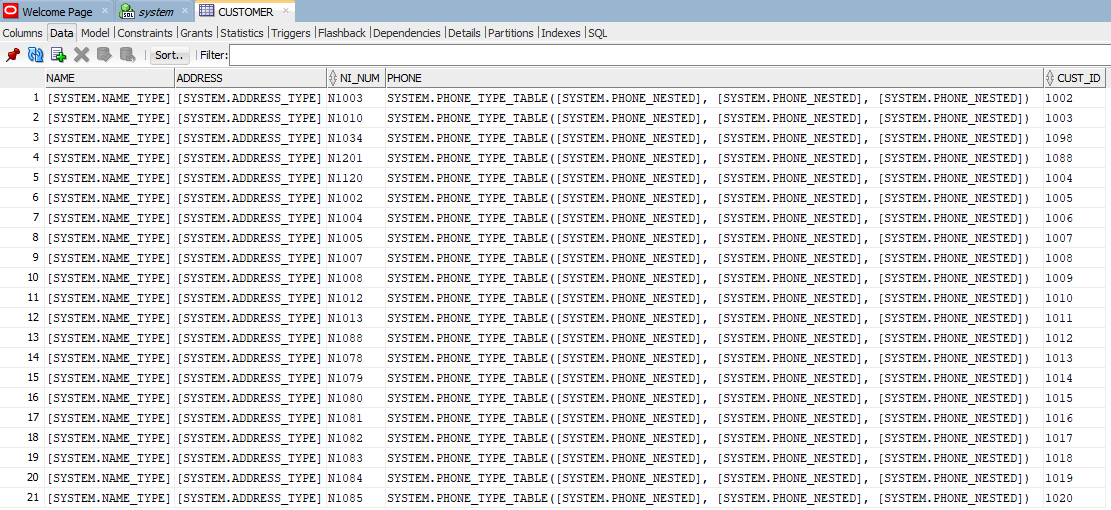


Figure3: Customer table

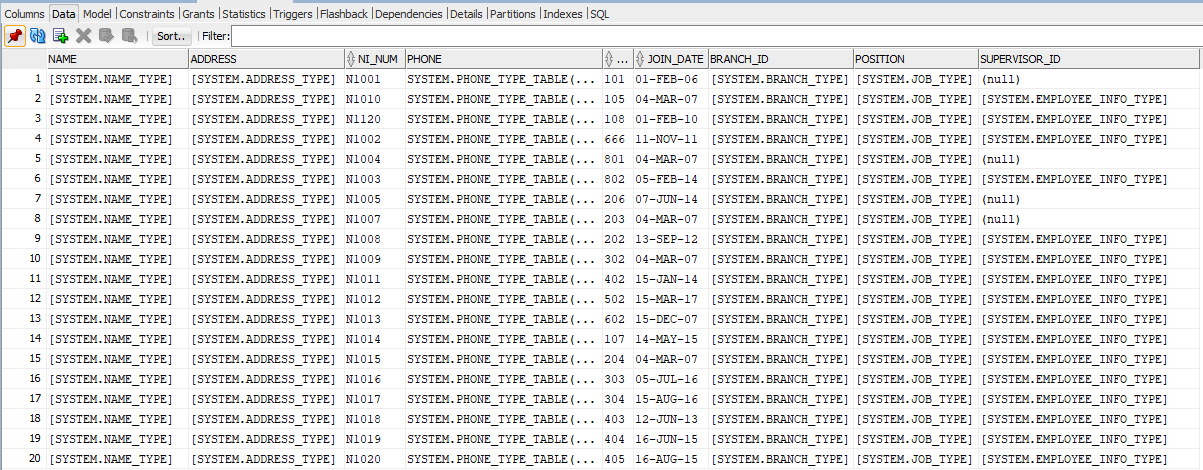


Figure4: Employee table



Figure5: Customer\_account table

1. **Questions & Answers**

**3.1 Question1:**

Draw an ER diagram corresponding to the relational database schema and the scenario. (5 marks)

|  |
| --- |
| SELECT e.name.title || '. ' || |
|  | e.name.firstName || ' ' || |
|  | e.name.surname || ' is living in ' || |
|  | e.address.city |
|  | AS "Employees: 'ar' and Edinburgh" |
|  | FROM employee e |
|  | WHERE INSTR(e.name.firstname,'ar')>0 |
|  | AND e.address.city = 'Edinburgh'; |

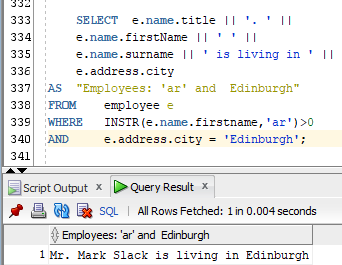


Figure6: Question1

**3.2 Question2:**

|  |
| --- |
| SELECT |
|  | a.branch\_id.branch\_id AS "Branch ID", |
|  | a.branch\_id.city AS "Address", |
|  | count(a.acc\_type) AS "Number of savingsaccounts" |
|  | FROM |
|  | account a |
|  | WHERE |
|  | acc\_type = 'saving' |
|  | GROUP BY |
|  | a.acc\_type, a.branch\_id.city, a.branch\_id.branch\_id |
|  | ORDER BY |
|  | a.branch\_id.branch\_id ASC; |

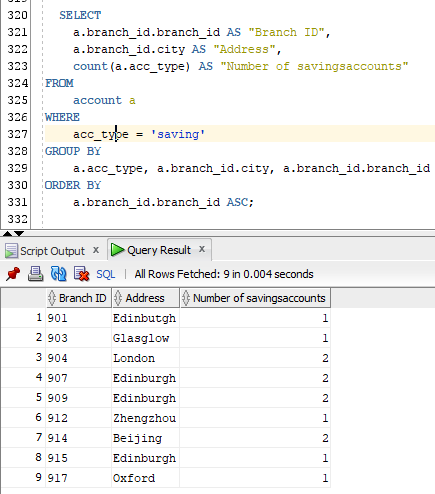


Figure7: Question2

**3.3 Question3:**

ALTER TYPE person\_type

ADD MEMBER FUNCTION print\_name RETURN STRING,

ADD MEMBER FUNCTION print\_address RETURN STRING CASCADE;

/

CREATE OR REPLACE TYPE BODY person\_type AS

MEMBER FUNCTION print\_name RETURN STRING IS

BEGIN

RETURN name.title || '. ' || name.firstname || ' ' || name.surname;

END print\_name;

SELECT

c.account\_number.branch\_id.branch\_id AS branchID,

c.cust\_id.print\_name() AS names,

c.account\_number.balance AS balance

FROM (

SELECT

c.account\_number.branch\_id.branch\_id AS branch\_id,

c.account\_number.acc\_type AS acc\_type,

MAX (c.account\_number.balance) AS max\_balance

FROM

customer\_account c

WHERE

c.account\_number.acc\_type = 'saving'

GROUP BY c.account\_number.branch\_id.branch\_id, c.account\_number.acc\_type

) balance

JOIN customer\_account c

ON

c.account\_number.branch\_id.branch\_id = balance.branch\_id

AND

c.account\_number.acc\_type = balance.acc\_type

AND

c.account\_number.balance = balance.max\_balance

LEFT JOIN customer\_account t2

ON t2.cust\_id.cust\_id = c.cust\_id.cust\_ID

AND t2.account\_number.acc\_type = 'Basic';

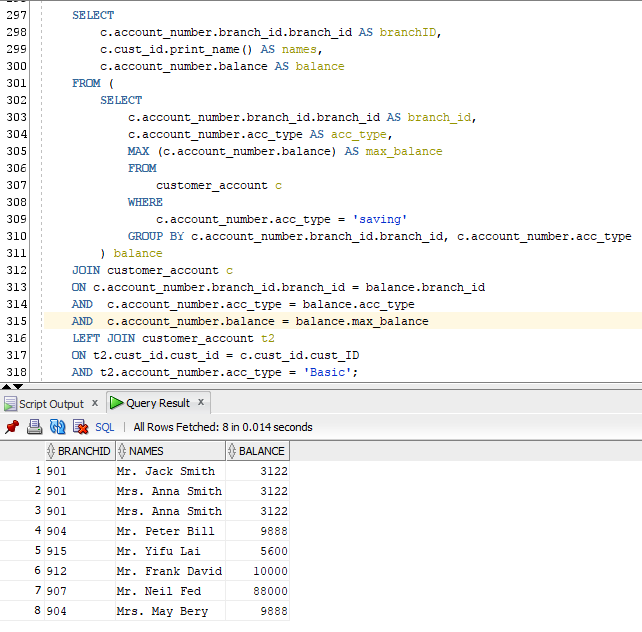


Figure8: Question3

**3.4 Question4:**

SELECT

'Emp ID: ' || e.emp\_id || ', ' ||

e.print\_name() AS employee,

c.account\_number.account\_number AS "Account Number",

e.supervisor\_id.print\_name() AS "Supervisor",

e.supervisor\_id.position.job\_title AS "Supervisor Job Title"

FROM

employee e, customer\_account c

WHERE

c.cust\_id.name.firstname = e.name.firstname

AND

c.cust\_id.name.surname = e.name.surname

AND

e.supervisor\_id.position.job\_title = 'Manager'

ORDER BY

e.emp\_id ASC;

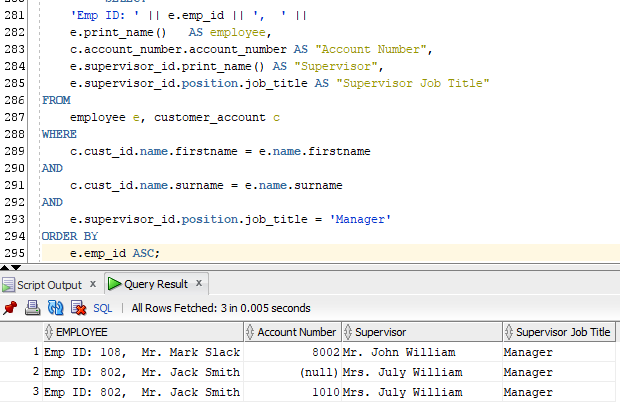


Figure9: Question4

**3.5 Question5:**

SELECT

c.account\_number.branch\_id.branch\_id AS branch\_id,

c.cust\_id.print\_name() AS full\_name,

c.account\_number.limit\_of\_free\_OD AS free\_od

FROM (

SELECT c.account\_number.branch\_id.branch\_id AS branch\_id,

MAX(c.account\_number.limit\_of\_free\_OD) AS maxOD

FROM customer\_account c

GROUP BY c.account\_number.branch\_id.branch\_id

) maxOD, customer\_account c

WHERE c.account\_number.limit\_of\_free\_OD = maxOD.maxOD AND

c.account\_number.branch\_id.branch\_id = maxOD.branch\_id

ORDER BY c.account\_number.branch\_id.branch\_id ASC;



Figure10: Question5

**5. Critical discussion**

**Advantages of the object-relational database:**

1. Have extended data types – object-relational database system allow users to define a new data type and corresponding operations based on application requirements. The new data types, once defined, is shared by all users just like the basic data type. For bank database, data always need update and have the new data type, so the object-relational database would be easier to change for DBA.
2. Support for complex objects-the object-relational database system can support complex objects in SQL, and implement processing such as querying complex objects. A complex object is an object that consists of multiple basic types or user-defined data types. Considering the database of bank always have many members and complex data types, so it is the best choice.
3. Support the concept of inheritance. Inheritance is an important concept of object-oriented technology. The object-relational database system can support the concept of subtype and super type, that means, support the concept of inheritance, such as the inheritance of attribute data and the inheritance of functions and procedures; With multiple inheritance, etc. It also supports important object-oriented ideas such as function overloading.
4. Provide a general rule system. Object-relational database systems provide a powerful and versatile rules system. In the traditional, it is generally triggered to ensure the integrity of the data in the database. The trigger is a form of the rule. This advantage can help different DBA to maintain the database.

**Advantage of the relational database:**

1. Maintain data consistency, data relational model based on relational model, structured storage, integrity constraint.
2. Due to the premise of the premise of the standard, the cost of data update is small(the same filed is basically only one place);
3. It is possible to perform complex queries such as join; This advantage is very Important for big database(like bank), because they usually need some complex queries including paging query and multi query.
4. There are many practical results and professional technical information (mature technology). Many database using sql relational language, like Mysql, db2, sql2000.

**Disadvantages of the objected-relational database:**

Although objected-relational database model have many advantages, it looks like connect both advantages of objected database and relational database. However, it just ‘look like’. The advantages also can be the disadvantages, because it combined with two different databases and take advantages of their capability, so sometimes it will become very complex and difficult to handle. As for the database of bank, it means when DBA changed or left, the new DBA would need a long time to familiar with the former database.

**Disadvantages of the relational database:**

(1) Transaction Consistency is not necessary nowadays: Relational databases have a large overhead in maintaining the consistency of things. Nowadays, many web2.0 systems do not have high read/write consistency on transactions, and transactions consistency is not so important now.

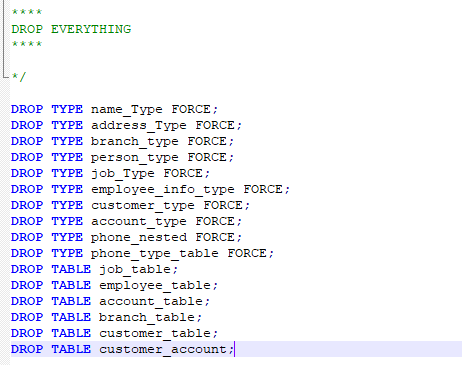
(2) Fixed table structure: very poor scalability, system upgrades, and increased functionality often mean huge changes in the data structure. This relational database is also difficult to cope with and requires new structured data storage.

(3) Complex SQL: any large data volume web system is very jealous of multiple large table association queries, as well as complex data analysis types of complex SQL report queries. And often the thing is that the primary key query of a single table, as well as a simple conditional paged query of a single table, the function of SQL is greatly weakened.

**Different between object-relational database and relational database:**

1. Method: Method is a special function which used for object-relational db but the relational db does not have it. A method is defined as “procedure or function that can operate on the attribute type.”(5) There are different kinds of functions in method, in this database, I use member function to show the full name and full address.
2. Types: Only object-relational database have types, often types are below the table, one type can contain many attributes, and the relational database just contain attributes in tables. Types have benefits and disadvantages, but just for bank database, using types is better.

**6. Drop everything**



**7. Conclusion**