

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.

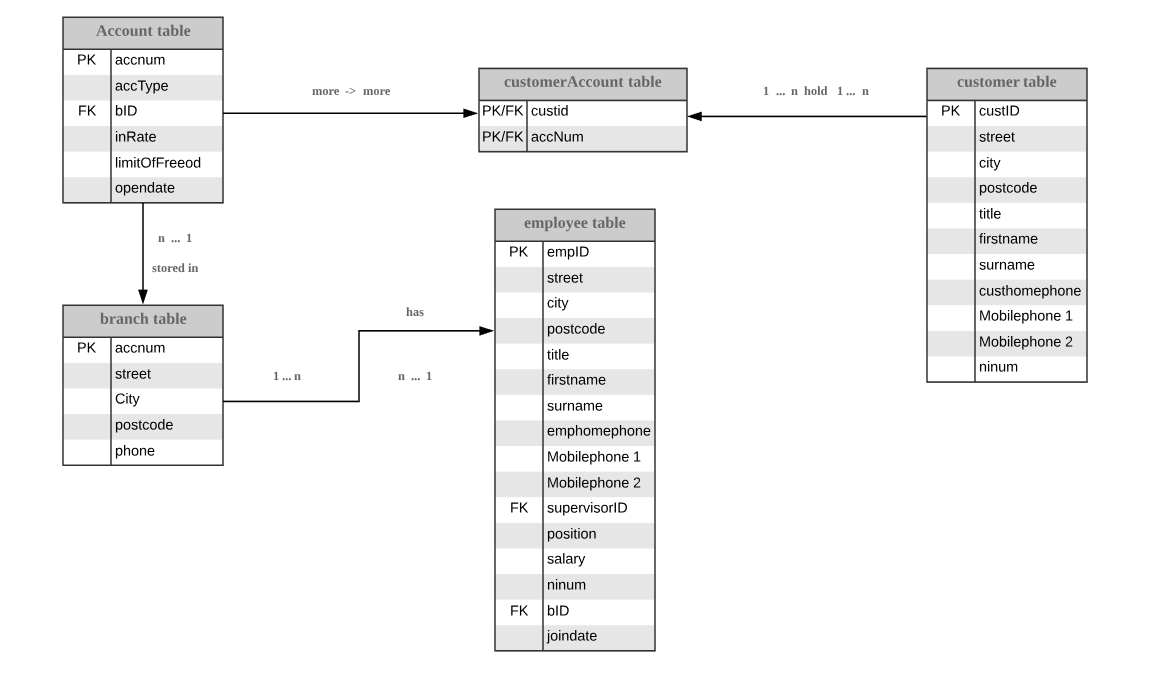
An employee has a job position (Head, Manager, Project Leader, Accountant, Cashier) and a salary, and works for exactly one branch. The date that the employee joined the bank is also recorded. Every employee has a supervisor at the same branch, except the head of the branch. The head of the branch is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

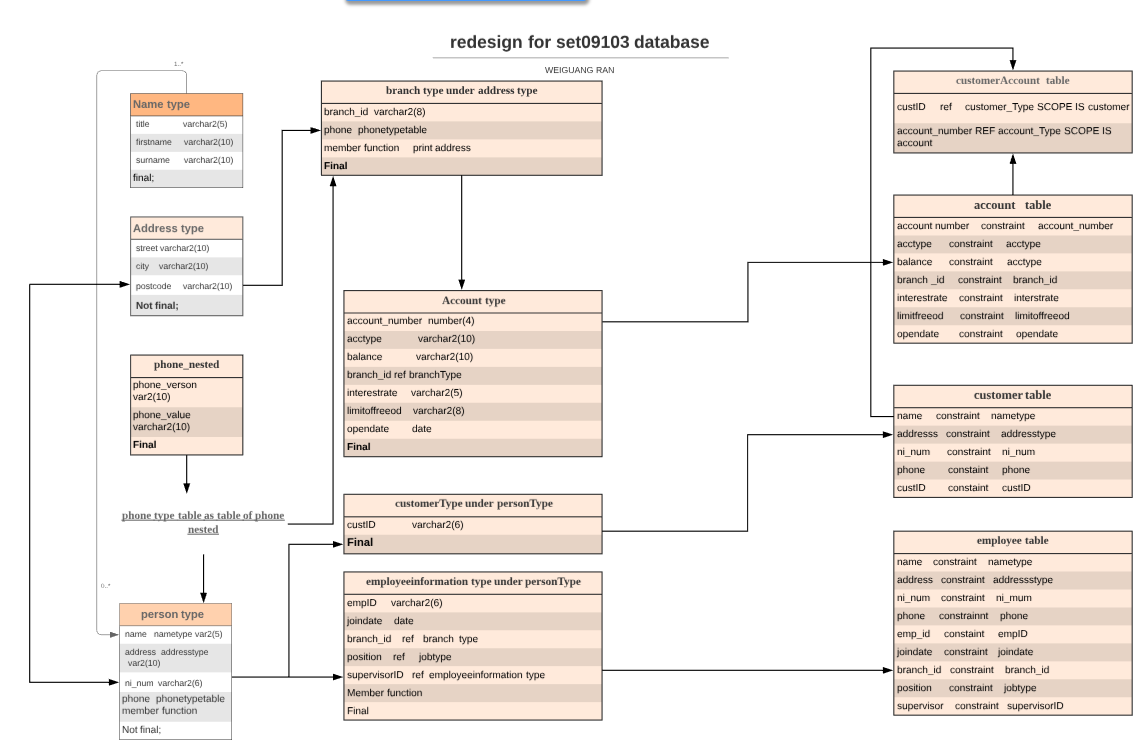
**1.Entity Relational Diagram of my coursework**



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**

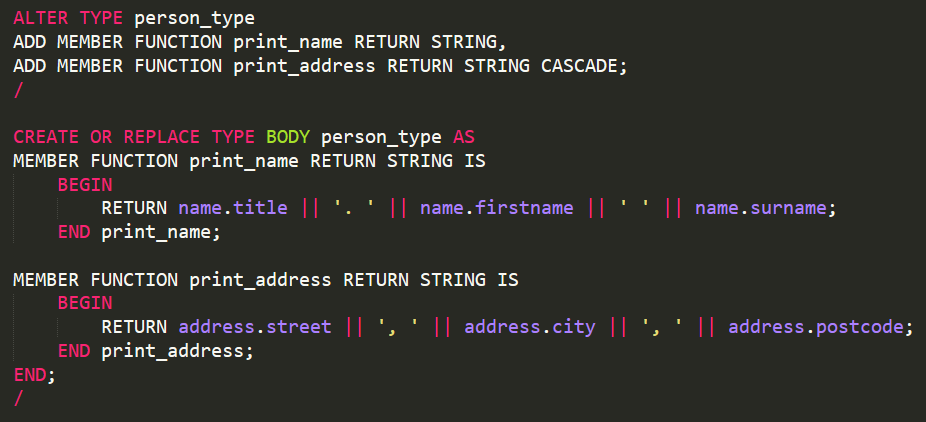
People can define constraints syntactically in two ways:(1) As part of the definition of an individual column or attribute. This is called **inline** specification. (2) As part of the table definition. This is called **out-of-line** specification.

**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**