# Java RMI Written Assignment

## Abstract

The assignment aims to execute a distributed system. We design a computer manufacturer and distributer which is called Banana. Three main products smartphone, laptops and desk computers are sold in Banana owned shops.

1. Design a Java Branch Client that can use RMI to communicate with Branch Server, Load Balance Server and Product Server
2. Design a Branch Server to provide the shop's production information for Branch Client's query
3. Design a Product Server to provide the HQ's production information for Branch Client's query
4. Design a Load Balance Server to split the traffic of Branch Client's query request to three different running Product Server based on product type.

## Design

1. Database
   - Use MySQL as the Database Server
   - Database Name is banana.
   - Database User name is bananadbadm
   - Database User password is Banana@2018
   - There are six tables as below in banana database:
     - Ptype table: The table is used to store the product type. There are three types ptypeid 1 for ptypename smartphone, ptypeid 2 for ptypename laptop, ptypeid 3 for ptypename desktop computers. The table definition is as below:

| Field | Type | Null | Key | Extra |
|---|---|---|---|---|
| ptypeid | smallint | NO | Primary Key | auto_increment |
| ptypename | varchar(20) | NO | | |

     - Product table: The table is used to store the product basic information. Field pid is the id of the product. Field pname is the product name. Ptypeid is the type id of product. The table definition is as below:

| Field | Type | Null | Key | Extra |
|---|---|---|---|---|
| pid | int | NO | Primary Key | auto_increment |
| pname | varchar(25) | NO | | |
| ptypeid | smallint | NO | | |

     - Productstatus table: The table is used to store the product status information. Field pid is the id of the product. Field stock is the amount of product in the HQ. Field demand is the requirement of amount of product from customers. The table definition is as below:

| Field | Type | Null | Key | Extra |
|---|---|---|---|---|
| pid | int | NO | Primary Key | |
| stock | int | NO | | |
| demand | int | NO | | |

     - Branch table: The table is used to store the branch shop basic information. Field branchid is the id of branch shop. Field branchname is the name of branch shop. The table definition is as below:

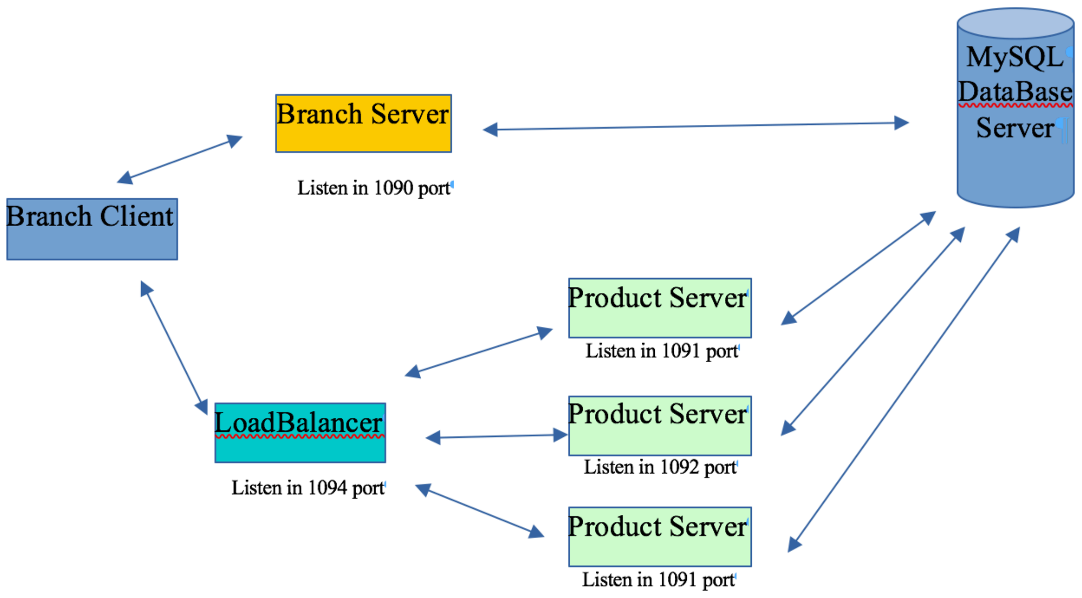
| Field | Type | Null | Key | Extra |
|---|---|---|---|---|
| branchid | int | NO | Primary Key | Auto increment |
| branchname | varchar(25) | NO | | |

- Branchproductstatus table: The table is used to store the branch shop product information. Field branchid is the id of branch shop. Field pid is the id of product. Field sold is the amount of product that branch shop sold. Field stock is the amount of product in the HQ. Field demand is the requirement of amount of product from customers. The table definition is as below:

| Field | Type | Null | Key | Extra |
|---|---|---|---|---|
| branchid | int | NO | Primary Key | |
| pid | int | NO | Primary Key | |
| sold | int | NO | | |
| stock | int | NO | | |
| demand | int | NO | | |

2. System Design

- Branch Client retrieve the shop's information about the statistics of stock, demand and sold of product from the Branch Server.
- Branch Client also retrieve the HQ's product information about the statistics of stock, the demand of products via Load Balance Server
- There are three product servers to start up. These three servers is respectively responsible for product information's query on three types products. The tree types include smartphones, laptops and desktop computers.
- Load Balance Server acts as both Java RMI Server and Client. Firstly, Load Balance Server acts as the Java RMI Server to accept the connection from Branch Client. Then the Load balance Server will look up the product type from MySQL database to determine which product server to connect. On the other hand, Load Balance Sever is responsible for acting as the Java RMI Client to connect to the appropriate products server to retrieve product information. Once get the product information that product sever will reply, and the Load Balance Server will pass through the reply to the Branch Client.
- For demo purpose, five servers and one client are all in one machine, five servers will listen in different port for simulating the communication environment of five servers and one client. Branch Server will listen at 1090 port. Load Balancer will listen at 1094 port. Three product servers will listen at three different ports for load balancing. The three product servers are illustrated as below:
  - The product server listens at 1091 port that will responsible for process the query request of smartphone.
  - The product server listens at 1092 port that will responsible for process the query request of laptop.
  - The product server listen at 1093 port that will responsible for process the query request of desktop.
- The architecture diagram of system design is as below:



**Implementation**

- System Implementation: There are four parts;
- Product Server:
  - ProductStatus.java: Define the product information class to represent the database table productstatus. This is also the definition of object that Product Server reply to Load Balance Server.
  - ProdcutStatusInterfaec.java: Define the RMI interface and method getProductStatus for the client to Call.
  - ProductStatusImpl.java: Implement the RMI interface and method getProductStatus. The method getProductStatus will connect to MySQL Database to get the product information base on product id and return the ProductStatus object of ProductStatus type.
  - ProductServer.java: Create Java RMI registry and bind the interface to ip address and port and listen the connection.
- Load Balance Server:
  - LoadBalancerInterface.java: Define the RMI interface and method getProductStatus for the client to Call.
  - LoadBalacnerImpl.java: Implement the RMI interface and method getProductStatus.The method getProductStatus will connect to MySQL Database to get the product type base on product type id and the ProductStatus object of ProductStatus type.
  - LoadBalancer.java: Create Java RMI registry and bind the interface to ip address and port and listen the connection.
- Branch Server:
  - BrachnProductStatus.java: Define the branch shop product information class to represent the database table branhchproductstatus. This is also the definition of object that Branch Server reply to Branch Client.
  - BranchProdcutStatusInterfaec.java: Define the RMI interface and method getBranchProductStatus for the client to Call.
  - BranchProductStatusImpl.java: Implement the RMI interface and method getBranchProductStatus. The method getBranchProductStatus will connect to MySQL Database to get the branch shop product information base on product id and branchid. Finally, it will return the BranchProductStatus object of BranchProductStatus type.

        - BranchServer.java: Create Java RMI registry and bind the interface to ip address and port and listen the connection.
    - Branch Client:
        - BranchClient.java: Get the command parameters to determine to connect Load Balance Server or Branch Server and the host and port to connect. Once the connection is built up, the programme will get and bind the server's RMI interface and call the method and pass through the parameters which is required.

1. Installation Gudie
    - Database Setup
        - Install and setup a MySQL or MariaDB Database Server.
        - Use following command to create a database which is named banana , and default character set is utf8;
            - Create database banana default character set=utf8;
        - Import the attached sql schema creation and data file to do database banana and use the following command that can also do the job:
            - mysql -u root -p banana < banana.sql
        - Use following command to create the database user account bananadbadm and password bananadbadm for Java RMI Application's usage:
            - create user 'bananadbadm'@'localhost' identified by 'Banana@2018';
        - Use following commands to grant the all privileges to do database user bananadbadm and acknowledge the database system to flush the privileges to let the newly assigned privileges to be effective.
            - grant all on banana.* to 'bananadbadm'@'localhost';
            - flush privileges;

2. User Guide
    - The whole application was packed as a single jar file which is named dcprj1.jar that include the MySQL JDBC driver.
    - Using the following command to start the Branch Server. The parameter 0.0.0.0 means to listen all of available network interface:
        - java -cp ./dcprj1.jar com.grace.app.jrmi.BranchServer 0.0.0.0 1094
    - Using the following command to start the three Product Servers respectively to listen the connection at 1091,1092 and 1093 port,
        - java -cp ./dcprj1.jar com.grace.app.jrmi.ProductServer 0.0.0.0 1091
        - java -cp ./dcprj1.jar com.grace.app.jrmi.ProductServer 0.0.0.0 1092
        - java -cp ./dcprj1.jar com.grace.app.jrmi.ProductServer 0.0.0.0 1093
    - Using the following command to start the Load Balance Server, Load Balance Server will listen at 1090:
        - java -cp ./dcprj1.jar com.grace.app.jrmi.LoadBalancer 0.0.0.0 1090
    - Using the following command for Branch Client to connect to server:
        - Connect to Branch Server (port 1094) to get the branch id 1's shop(the last two parameter)the product information about product id 1(the last parameter),127.0.0.1 is the local host's ipaddr:
            - java -cp ./dcprj1.jar com.grace.app.jrmi.BranchClient b 127.0.0.1 1094 1 1
        - Connect to Balance Server (port is 1090) to get the HQ's product information about product id 1(the last parameter),127.0.0.1 is the local host's ipaddr as product id 1's product type id is 1. The Balance Server will connect to the Product Server that listen at port 1091 to get the product information:

        - java -cp ./dcprj1.jar com.grace.app.jrmi.BranchClient p 127.0.0.1 1090 1
    - Connect to Balance Server (port is 1090) to get the HQ's product information about product id 5(the last parameter),127.0.0.1 is the local host's ipaddr,Because product id 5's product type id is 2,The Balance Server will connect to the Product Server that listen at port 1092 to get the product information:
        - java -cp ./dcprj1.jar com.grace.app.jrmi.BranchClient p 127.0.0.1 1090 5
    - Connect to Balance Server(port is 1090) to get the HQ's product information about product id 9(the last parameter),127.0.0.1 is the local host's ipaddr,Because product id 9's product type id is 3,The Balance Server will connect to the Product Server that listen at port 1093 to get the product information:
        - java -cp ./dcprj1.jar com.grace.app.jrmi.BranchClient p 127.0.0.1 1090 9

**Analysis**

- Advantages and Disadvantages: All of works in one server and use port to simulate the multi-server's environment's design that ease the complex environment setup and demo. However, it is difficult to deploy to multi-servers. Product Server, Load Balance Server and Branch Server preserve the ability to deploy to different servers. There is a limitation is that the Database connection setting is hard-code in these server's code. It can be solved to read an external database setting file.

- Problems: In the process of design the Load Balance Server, I was confused the interconnect's design of Load Balance's RMI interface type and Product Server's RMI interface type. This lead to the compile error and run time error. Finally, I clarify the difference of interconnection, then finish the design.

**Conclusion**

The whole servers and client include MySQL DataBase Server which now limit to must run at the same server. Listen and Connection ip address and port all input by command's parameter. It is good to let the all parameters which are set to store in individually configuration files. The all applications can read the configuration file's setting and start itself automatically. Then we can let each server (include MySQL Database Server) to running in its standalone computer and unique ip address. That will meet the truly requirement of Distributed Computing.

On the other hand, the Load Balance Server's load balance scheme is limited to one product type map to one server. The scheme is just Load Sharing. Based on the current request processing number, it's a better scheme to distribute the requests to the servers that have at least request processing number. The least-requests-number scheme needs to do more complex design and programming.

**Reference**

1.Docs.oracle.com. (2018). Trail: RMI (The Java™ Tutorials). [online] Available at: https://docs.oracle.com/javase/tutorial/rmi/ [Accessed 16 Feb. 2018].
2. www.tutorialspoint.com. (2018). Java RMI Quick Guide. [online] Available at: https://www.tutorialspoint.com/java_rmi/java_rmi_quick_guide.htm [Accessed 16 Feb. 2018].
3. Dev.mysql.com. (2018). MySQL :: Download Connector/J. [online] Available at: https://dev.mysql.com/downloads/connector/j/ [Accessed 16 Feb. 2018].