**5. IMPLEMENTATION AND TESTING**

**5.1.1 Coding Details**

The project has been divided into 4 modules.

1. Front End Tool- This Front End tool connects the user to the central database(Oracle Database). The user can input, update and delete the two scores of each student from this module in the database. While inserting records in the database the roll number is automatically incremented for the user if he/she requires that. Manual change to the ‘Roll No.’ field is also allowed.

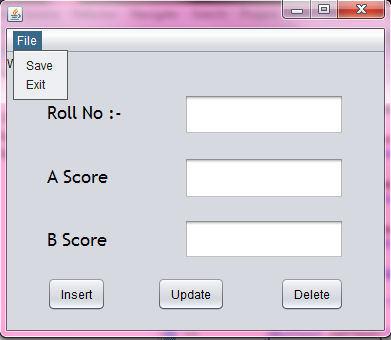


Figure – Front End Tool

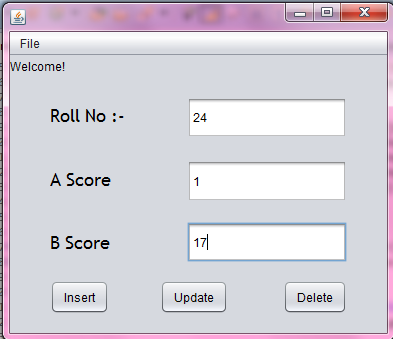


Figure –Inserting record

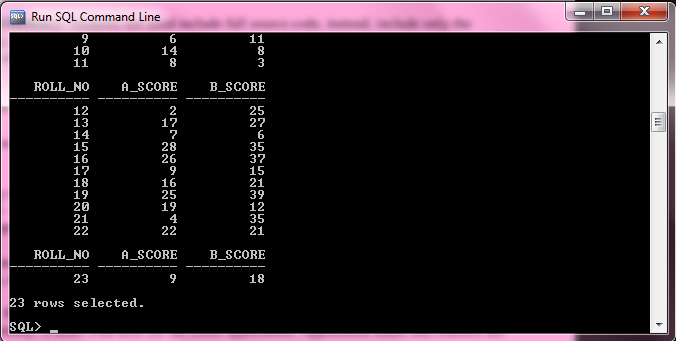


Figure –Database before insertion

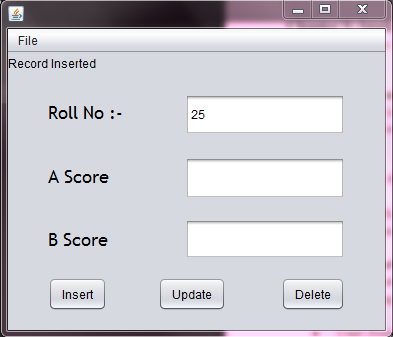


Figure –Tool after successful insertion

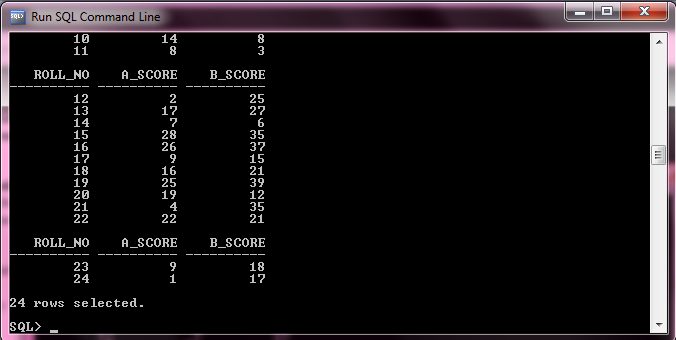


Figure –Database after insertion

*jButton1*.addActionListener(**new** ActionListener(){

**public** **void** actionPerformed(ActionEvent ae){

**try**

{

//Connection to Oracle Database Class.*forName*("oracle.jdbc.driver.OracleDriver");

Connection con=DriverManager.*getConnection*("jdbc:oracle:thin:@127.0.0.1:1521:XE","system","system");

String strqry="insert into StudentData values(?,?,?)";

PreparedStatement pst=con.prepareStatement(strqry);

pst.setInt(1,Integer.*parseInt*((*jTextField1*.getText()))); pst.setInt(2,Integer.*parseInt*((*jTextField2*.getText())));

pst.setInt(3,Integer.*parseInt*((*jTextField3*.getText())));

**int** res=pst.executeUpdate();

**if**(res>0)

{

**int** n=Integer.*parseInt*(*jTextField1*.getText());

String s="";

s+=(++n);

*jTextField1*.setText(s);

*jTextField2*.setText("");

*jTextField3*.setText("");

*jTextField2*.requestFocus();

*jLabel4*.setText("Record Inserted");

}

**else**

*jLabel4*.setText("Record Not Inserted");

}

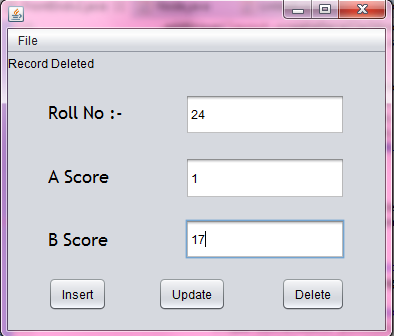


Figure –Deletion of record

*jButton3*.addActionListener(**new** ActionListener(){

**public** **void** actionPerformed(ActionEvent ae){

**try**

{

Class.*forName*("oracle.jdbc.driver.OracleDriver");

Connection con=DriverManager.*getConnection*("jdbc:oracle:thin:@127.0.0.1:1521:XE","system","system");

String strqry="delete from StudentData where Roll\_No=?";

PreparedStatement pst=con.prepareStatement(strqry);

pst.setInt(1,Integer.*parseInt*((*jTextField1*.getText())));

**int** res=pst.executeUpdate();

**if**(res>0)

{

*jTextField1*.setText("");

*jTextField2*.setText("");

*jTextField3*.setText("");

*jLabel4*.setText("Record Deleted");

}

**else**

*jLabel4*.setText("Record Not Deleted");

}

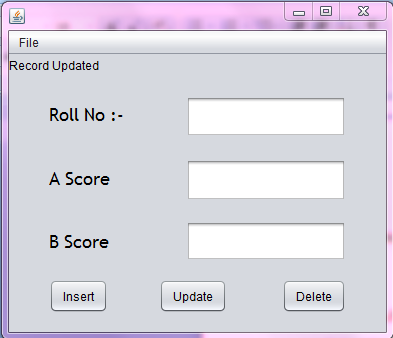
****

Figure **–** Updating Record

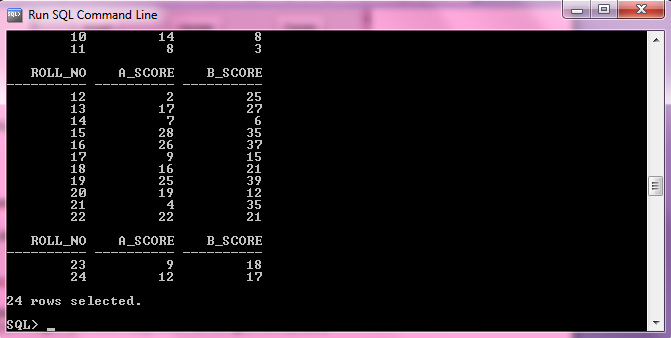
****

Figure –Database Updated

*jButton2*.addActionListener(**new** ActionListener(){

**public** **void** actionPerformed(ActionEvent ae){

**try**

{

Class.*forName*("oracle.jdbc.driver.OracleDriver");

Connection con=DriverManager.*getConnection*("jdbc:oracle:thin:@127.0.0.1:1521:XE","system","system");

String strqry="update StudentData set A\_Score=?, B\_Score=? where Roll\_No=?";

PreparedStatement pst=con.prepareStatement(strqry);

pst.setInt(3,Integer.*parseInt*((*jTextField1*.getText())));

pst.setInt(1,Integer.*parseInt*((*jTextField2*.getText())));

pst.setInt(2,Integer.*parseInt*((*jTextField3*.getText())));

**int** res=pst.executeUpdate();

**if**(res>0)

{

*jTextField1*.setText("");

*jTextField2*.setText("");

*jTextField3*.setText("");

*jLabel4*.setText("Record Updated");

}

**else**

*jLabel4*.setText("Record Not Updated");

}

1. ‘R’ Module-
2. Equalization module- This module now takes account of the initial clusters formed by the R module. The cluster centres and cluster number of each student is inserted into the program that optimize the clusters and allots equal number of students to each cluster.

**while**(flag=='N' || flag=='n')

{

System.***out***.print("Enter no of Students- ");

no=Integer.*parseInt*(br.readLine());

System.***out***.print("Enter no of clusters- ");

clstr=Integer.*parseInt*(br.readLine());

System.***out***.print("Therefore you have "+(no/clstr)+" no of students for each cluster. Enter (Y/y) or (N/n)- ");

flag=br.readLine().charAt(0);

**if**(flag=='Y' || flag=='y')

**break**;

}

The above code inputs and checks whether all the students can be allotted equally in each group or not.

Class.*forName*("oracle.jdbc.driver.OracleDriver");

c*on*=DriverManager.*getConnection*("jdbc:oracle:thin:@127.0.0.1:1521:XE","system","system");

String strqry="select \* from StudentData";

*pst*=*con*.prepareStatement(strqry);

*res*=*pst*.executeQuery();

**while**(*res*.next())

{

as=*res*.getInt(2);

bs=*res*.getInt(3);

**if**(j<=no)

{

System.***out***.print("Enter Cluster no. for Roll "+j+"- ");

clusterno=Integer.*parseInt*(br.readLine());

//calculating Euclidean distance of each student from the centre.

dist=Math.*sqrt*(Math.*pow*(cluster[0][clusterno]-as, 2)+Math.*pow*(cluster[1][clusterno]-bs, 2));

obj[clusterno].add(j++,dist);

}

}

The above code take input of each cluster number of the student then calculates the distance of each student from its respective cluster centres for future calculation.

**for**(**int** i=1;i<clstr;i++)

{

**if**(obj[i].count()>=(no/clstr))

{

/\*if present cluster has more objects than the required number it transfers the excess objects to the next cluster\*/

**while**(obj[i].count()!=(no/clstr))

{

data=obj[i].max();

Node ptr=obj[i].remove(data);

obj[i+1].add(ptr);

}

}

**else**

{

k=i+1;

**while**(obj[i].count()!=(no/clstr))

{

//if the cluster is empty then transfer is continued from the next cluster

**if**(obj[k].count()==0)

k++;

**else**

{

data=obj[k].max();

Node ptr=obj[k].remove(data);

obj[i].add(ptr);

}

}

}

}

The above code equalizes the cluster into equal heterogeneous groups.

**5.1.2 Coding Efficiency**

**5.2 Testing Approach**

**5.3 Unit Testing**

**5.4 Integrated Testing**

**5.5 Modification and Improvement**

Initial testing of the Equalisation algorithm generated few errors.

1. Transferring one object from would result in more than one object being transferred together.
2. Seldom would the clusters get exhausted pre-emptively leading to unequal and null groups.

Cause for errors.

1. Since each objects of every cluster links one other object removing the object having such links would result in removing all the successive object as well.
2. Transferring objects from clusters having lesser number of objects than what is required for equalization would transfer all the objects to the target cluster from the source cluster. This may lead to clusters having unsatisfied number of objects.

Modifications.

1. Since we are using Linked Data structure to store each object in each cluster, we had to reassign each link while removing an object anywhere other than the end of the list. This solved the problem of transferring multiple objects at a time.
2. We put in a check where if a cluster is empty and the target cluster does not have the required number of objects, the next cluster would transfer its objects to the target cluster.

Implementation Approaches: Define the plan of implementation, and the standards you have used in the

implementation.

Coding Details and Code Efficiency: Students not need include full source code, instead, include only the

important codes (algorithms, applets code, forms code etc). The program code should contain comments needed for

explaining the work a piece of code does. Comments may be needed to explain why it does it, or, why it does a

particular way.

You can explain the function of the code with a shot of the output screen of that program code.

 Code Efficiency: You should explain how your code is efficient and how you have handled code

optimisation.

Testing Approach: Testing should be according to the scheme presented in the system design chapter and should

follow some suitable model – e.g., category partition, state machine-based. Both functional testing and user-

acceptance testing are appropriate. Explain your approach of testing.

 Unit Testing: Unit testing deals with testing a unit or module as a whole. This would test the interaction of

many functions but, do confine the test within one module.

 Integrated Testing: Brings all the modules together into a special testing environment, then checks for errors,

bugs and interoperability. It deals with tests for the entire application. Application limits and features are

tested here.

Modifications and Improvements: Once you finish the testing you are bound to be faced with bugs, errors and you

will need to modify your source code to improve the system. Define what modification you implemented in the

system and how it improved your system.