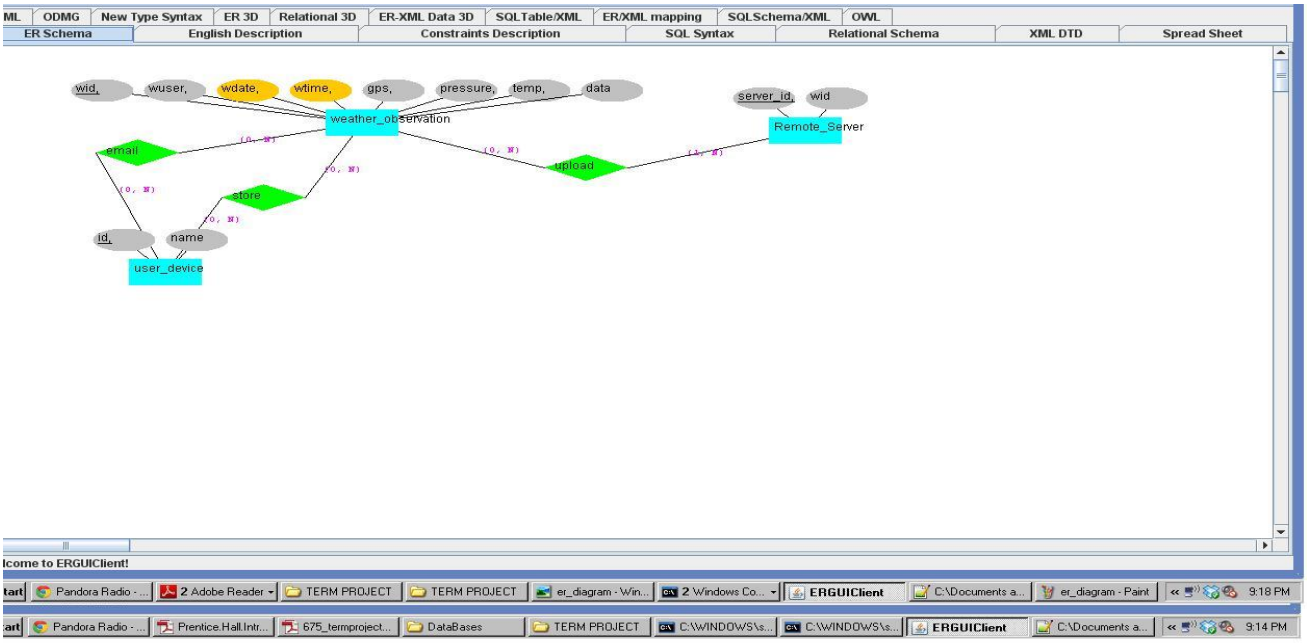
Ozan Gazi Onder
CsC 675
Term Project

Introduction: The following documents demonstrate the entire design and implementation cycle of the database Weather Observation. They include ER schemas, Relational Algebra, trees, necessary explanation, queries and their implementation with Microsoft SQL Server, Postgres , JDBC ODBC Interface and a scripting language(Perl). Moreover, it includes a graphical user interface that reads and display data with Java. Also, it displays a web application that reads and writes to the database in MySql that is implemented with PHP, html and CSS. The database has been created in a simple form such that it can be easily understood, and contains all the necessary tables with appropriate documentation.

The Weather Observation is a database that contains basic information of weather observations that are uploaded by different user or device(They are referred as user_device). A user can either email or store a weather observation data. Also, the weather observation can be uploaded to a remote server, separately.

ER DIAGRAM:



ER Diagram Documentation:

The following definitions are the English descriptions of the entities, relationships, attributes and constraints in the Weather Observation Database:

Weather_Observation

wid: Primary key, the ID of the observation (int)
wuser: foreign key references the id of user who stored or emailed the data.(varchar)
wtime: the time when the data is added. (time)

wdate: the date when the data is added (date)
gps: descibes the gps coordinates about this data. (varchar)
pressure: barometic pressure(unit mBar) (int)
temp: temeprature value(unit: fahrenheit F) (int)
data: additional data that user adds. It can be wind strenght, dsecription, comment or media object. (char)

user_device
id primary key, represent a user (int)
name: name and (optionally) last name of the user.(varchar)

store(or collect)
relationship that user adds data to the table Weather_Observation
wid: Primary key, also foreign key references Weather_Observation (int)
id: Primary key, also references id in user table. (int)

Contraints:
Each user can store at least zero at most N data to Weather_Observation.
At least 0 at most N data can be uploaded by a user.
Therefore our constraint will have a (0,N) relationship.

email
Relationship that user emails data to the table Weather_Observation
wid: Primary key, also foreign key references Weather_Observation(int)
id: Primary key, also references id in user table.(int)

Contraints:
Each user can store at least zero at most N data to Weather_Observation.
At least 1 at most N data can be uploaded by a user.

Therefore our constaint will have a (1,N) relationship.
wid: primary key also foreign key refences Weather_Observation.
sid: primary key also foreign key refrencing remoter_server.

Contraints: A remote server can have at least zero at most N number of weather data.
Key Contraint: There must be at least 1, at most 1 remote_sever for all the weather data.

remote_server the server contains the weather data
server_id: server id, primary key
wid: foreign key, references Weather_Observation

NULL VALUES ARE NOT ALLOWED
In this case, the user must enter valid data for all the entities of the Weather Observation table, and this includes pictures, pressure, temperature and gps coordinates.

NULL VALUES ARE ALLOWED:
The foreigtn keys can reference null values, and user does not have to enter input for all the attributes. For example, in the Weather_Observation table, wuser can reference null value, therefore, we can add a new weather data whose adder is not known.Similarly, we can allow null values in Store and Email tables, and therefore, they may contain null values word the wid attribute that is a foreign key represents Weather_Observation.

RELATIONAL ALGEBRA AND QUERY TREES

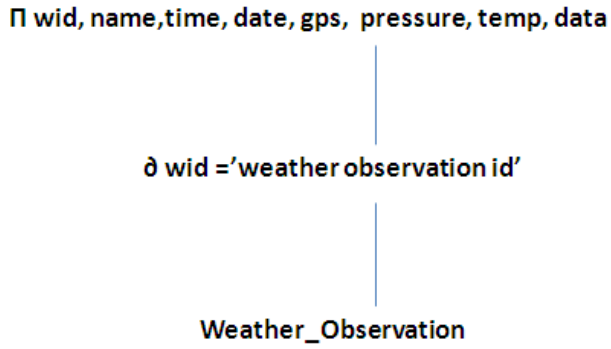
NOTE: First 5 questions are about inserting and deleting queries that does not involve relational algebra, that's why I started from question-6!!

NOTATIONS:
Π: projection
/ : division
∂: select
JOIN: join operator
∩: Intersection
– (Set Difference)

6-)

Given a Weather Observation-ID, retrieve all of the information needed to display that Weather Observation on a display.
By using the project operator, we will select the necessary columns to display the list of weather observations that has a specific weather observation ID, and we will retrieve this data by using select operation.

Π wid, name,time, date, gps, pressure, temp, data ((∂ wid ='weather observation id' Weather_Observation).

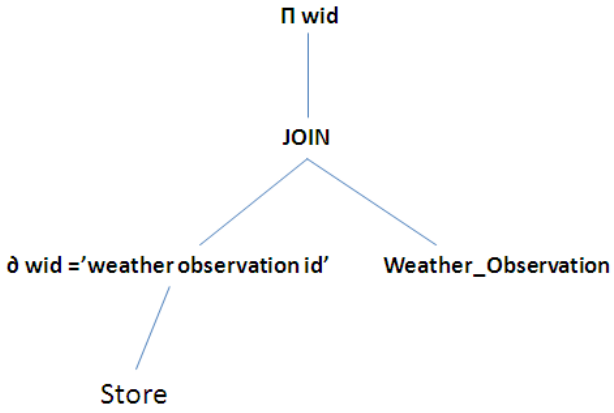


7-)

Retrieve the Weather Observation-ID's of all Weather Observations on the device *collected* by a specific user.

By using select operator, we will retrieve the id number of a specific user and use join operator to see if the user stored data to the Weather_Observation table.

Π wid(∂ id ='specific user id' Store) JOIN Weather_Observation)

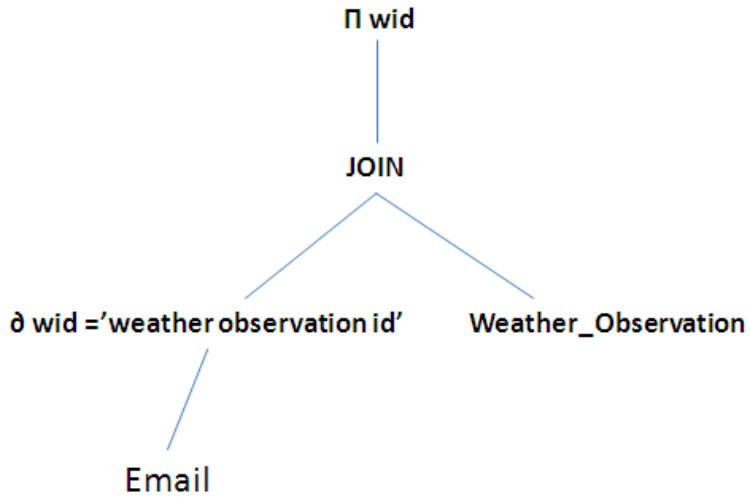


8-)

Retrieve the Weather Observation-ID's of all Weather Observations on the device *emailed* by a specific user.

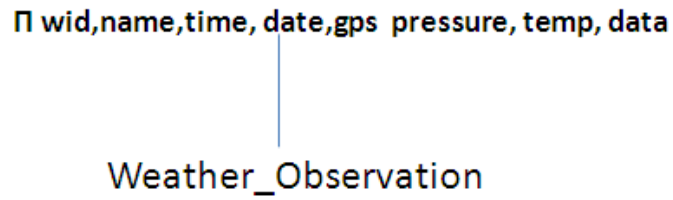
By using select operator, we will retrieve the id number of a specific user and use join operator to see if the user emailed data to the Weather_Observation table.

Π wid(∂ id ='specific user id' Email) JOIN Weather_Observation)



9-)
Retrieve the Weather Observation-ID’s of *all* Weather Observations currently stored in the SQLite Database on the user’s device.

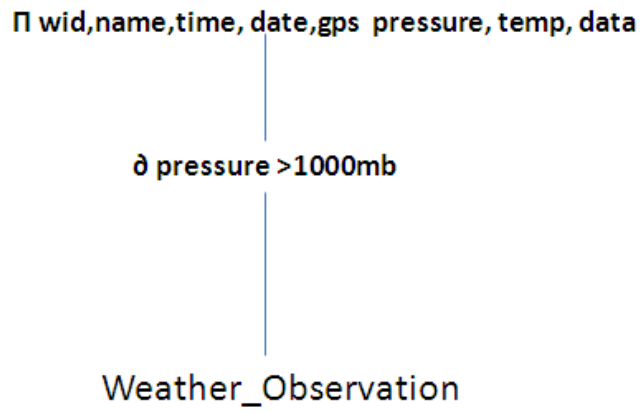
Π wid,name,time, date,gps pressure, temp, data (Weather_Observation).



10-)Retrieve the Weather Observation-ID’s of all Weather Observations on the device
with an associated type of data & range of values (for example, barometric pressure > 1000 mbar).

By using select operator, we will retrieve those whose pressure attribute is greater than 1000mb then we will use project operator to display all the columns.

Π wid,name,time, date,gps pressure, temp, data ((∂ pressure > 1000mb Weather_Observation).



Implementation of some sort of digital rights management system for stored photographs

Digital rights Management involves in controlling copying, viewing, displaying and trading electronic contents. If our database system were developed for commercial purposes, we would get permission from the copyright owners in order to display and share their intellectual properties. Otherwise, it could have been ended up with copyright infringement which violates the laws of copyrighted materials.

However, this database project is created for academic purposes that do not include any commercial advantage; therefore, according to the Fair-use Doctrine, we can share and display the copyrighted materials for educational purposes.

TABLE QUERIES:

```
CREATE TABLE Weather_Observation
(
  wid int NOT NULL,
  wuser int NOT NULL,
  wtime time,
  wdate date,
  gps varchar(20),
  pressure int,
  temp int,
```

```
  data varchar(50),

  PRIMARY KEY(wid),
  FOREIGN KEY(wuser) REFERENCES user_device

);

CREATE TABLE user_device
(
  id int NOT NULL,
  name varchar(20),

  PRIMARY KEY(id)


);

CREATE TABLE Store
(
  wid int NOT NULL,
  id int  NOT NULL,
  PRIMARY KEY(wid,id),
  FOREIGN KEY(wid) REFERENCES Weather_Observation,
  FOREIGN KEY(id) REFERENCES user_device

);

CREATE TABLE Email
(
  wid int NOT NULL,
  id int  NOT NULL,
  PRIMARY KEY(wid,id),
  FOREIGN KEY(wid) REFERENCES Weather_Observation,
  FOREIGN KEY(id) REFERENCES user_device

);

CREATE TABLE Remote_Server
(
  server_id int NOT NULL,
  wid int  NOT NULL,
  PRIMARY KEY(server_id),
  FOREIGN KEY(wid) REFERENCES Weather_Observation
```

);

Tables in Microsoft SQL Server

User_Device

100	ozan
101	ege
102	mike
103	john
104	oyku
105	jack

(6 row(s) affected)

Weather_Observation

wid	wuser	wtime	wdate	gps	pressure	temp	data
1	100	11:03:00.0000000	2012-05-05	40 43' 43''	1000	100	pictures/1.jpg
2	101	05:50:00.0000000	2001-03-03	31 6' 12''	2000	60	pictures/2.jpg
3	102	05:05:00.0000000	2010-10-10	20 '34 ''45	3000	25	pictures/3.jpg
4	100	17:30:00.0000000	1515-11-01	23 60' 38''	850	45	pictures/4.jpg
5	103	04:16:00.0000000	2009-09-09	25 4' 44"	1000	34	pictures/5.jpg
6	104	04:05:00.0000000	2003-04-04	32 43' 65"	1580	66	pictures/6.jpg
7	104	04:04:00.0000000	2003-07-03	43 98' 99"	500	50	pictures/7.jpg
8	103	03:05:00.0000000	2005-05-05	67 54' 67"	750	80	pictures/8.jpg

EMail

wid	id
1	100
2	101
3	102
4	100

Store

wid	id
5	103
6	104
7	104
8	103

Remote_Server

server_id	wid
1	3

2 1

MYSQL QUERIES AND RESULTS

QUESTION 1:

Create (enter) a new Weather Observation by storing all of the required data from both the device & user into the onboard SQLite database and return the Weather Observation-ID for the new observation.

INSERT INTO Weather_Observation (wid,wuser,wtime,wdate,gps,pressure,temp,data)
VALUES (9,103,'04:16:00','05.05.2003','43 43 76" ', 1600, 45, 'pictures/9.jpg');

```
SELECT * FROM Weather_Observation
WHERE wid=9

wid    wuser       wtime            wdate      gps          pressure    temp
data

9      103         04:16:00.0000000 2003-05-05 43 43 76"   1600        45
pictures/9.jpg

(1 row(s) affected)
```

QUESTION 2:

Delete a Weather Observation given its Weather Observation-ID.

DELETE FROM Weather_Observation
WHERE wid = 9

(1 row(s) affected)

QUESTION 3:

Move a Weather Observation to the remote archive, given its Weather Observation-ID.

INSERT INTO Remote_Server(server_id, wid)
VALUES (3, 5);

server_id	wid
1	3
2	1

3 5

QUESTION 4:

Change the value of a stored Weather Observation data value given its Weather Observation-ID and the type of data value to change (e.g. GPS coordinates)

```
UPDATE Weather_Observation
SET gps = '54 67" 78" '
WHERE wid = 4;
```

==========

```
SELECT * FROM Weather_Observation
WHERE wid=4;
```

4 100 17:30:00.0000000 1515-11-01 54 67" 78" 850 45 pictures/4.jpg

QUESTION 5:

Add a new type of observation data (e.g. solar energy strength) and value to a Weather Observation given its Weather Observation-ID.

```
UPDATE Weather_Observation
SET data = data +',  Solar Strenght = 1577.32PV'
WHERE wid = 4;
```

==========

```
select data from Weather_Observation
where wid=4;
```

data
pictures/4.jpg, Solar Strenght = 1577.32PV

QUESTION 6:(THE UNITS WILL BE ADDED WHEN CREATING THE APPLICATION!)

Given a Weather Observation-ID, retrieve all of the information needed to display that Weather Observation on a display

```
select * from Weather_Observation
```

==========

1	100	11:03:00.0000000	2012-05-05	40 43' 43''	1000	100	pictures/1.jpg
2	101	05:50:00.0000000	2001-03-03	31 6' 12''	2000	60	pictures/2.jpg
3	102	05:05:00.0000000	2010-10-10	20 '34 ''45	3000	25	pictures/3.jpg
4	100	17:30:00.0000000	1515-11-01	54 67" 78"	850	45	pictures/4.jpg,
Solar Strenght = 1577.32PV							
5	103	04:16:00.0000000	2009-09-09	25 4' 44"	1000	34	pictures/5.jpg
6	104	04:05:00.0000000	2003-04-04	32 43' 65"	1580	66	pictures/6.jpg
7	104	04:04:00.0000000	2003-07-03	43 98' 99"	500	50	pictures/7.jpg
8	103	03:05:00.0000000	2005-05-05	67 54' 67"	750	80	pictures/8.jpg

QUESTION 7:

Retrieve the Weather Observation-ID's of all Weather Observations on the device ***collected*** by a specific user.

```
SELECT  DISTINCT W.wid
FROM Weather_Observation W, Store S, user_device U
WHERE W.wuser = U.id AND U.id = S.id AND U.name='john';

=================
wid
-----------
5
8

(2 row(s) affected)
```

QUESTION 8:

Retrieve the Weather Observation-ID's of all Weather Observations on the device ***emailed*** by a specific user.

```
SELECT  DISTINCT W.wid
FROM Weather_Observation W, Email E, user_device U
WHERE W.wuser = U.id AND U.id = E.id AND U.name='mike';

==================

wid
-----------
3

(1 row(s) affected)
```

QUESTION 9:

Retrieve the Weather Observation-ID's of ***all*** Weather Observations currently stored in the SQLite Database on the user's device

SELECT wid FROM Weather_Observation;

wid

1
2
3
4
5
6
7
8

(8 row(s) affected)

QUESTION 10:(UNITS WILL BE ADDED WHEN CREATING THE APPLICATION!)

Retrieve the Weather Observation-ID's of all Weather Observations on the device with an associated type of data & range of values (for example, barometric pressure > 1000 mbar).

SELECT wid FROM Weather_Observation
WHERE pressure > 1000;

wid

2
3
6

(3 row(s) affected)

QUERIES AND RESULTS IN VIRUAL BOX

Tables:

```
csc675@csc675VB ~/DemoScripts $ psql
Password:
psql (9.2.4)
Type "help" for help.

csc675=> select * from weather_observation;
 wid | wuser | wtime |   wdate    |    gps     | pressure | temp |      data
_
   1 |   100 | 12:23 | 2012-03-05 | 45 34" 43 |     1000 |  100 | pictures/1.jpg
   2 |   101 | 04:23 | 2011-03-05 | 22 32" 43 |     2000 |   60 | pictures/2.jpg
   3 |   102 | 14:23 | 2003-06-05 | 22 12" 69 |     3000 |   25 | pictures/3.jpg
   4 |   100 | 05:23 | 2009-06-11 | 10 12" 70 |      850 |   45 | pictures/4.jpg
   5 |   103 | 04:16 | 2010-10-11 | 10 25" 33 |     1000 |   34 | pictures/5.jpg
   6 |   104 | 17:16 | 2009-10-11 | 10 50" 50 |     1580 |   66 | pictures/6.jpg
   7 |   104 | 13:40 | 2001-01-04 | 10 40" 50 |      500 |   50 | pictures/7.jpg
   8 |   103 | 03:40 | 2005-05-05 | 67 54" 32 |      750 |   80 | pictures/8.jpg
(8 rows)

lines 1-12/12 (END)_
```

```
Password:
psql (9.2.4)
Type "help" for help.

csc675=> select * from user_device;
 id  | name
-----+------
 100 | ozan
 101 | ege
 102 | mike
 103 | john
 104 | oyku
 105 | jack
(6 rows)

csc675=> select * from store;
 wid | id
-----+-----
   5 | 103
   6 | 104
   7 | 104
   8 | 103
(4 rows)

csc675=> _
```

```
 wid | id
-----+-----
   5 | 103
   6 | 104
   7 | 104
   8 | 103
(4 rows)

csc675=> select * from email;
 wid | id
-----+-----
   1 | 100
   2 | 101
   3 | 102
   4 | 100
(4 rows)

csc675=> select * from remote_server;
 server_id | wid
-----------+-----
         1 |   3
         2 |   1
(2 rows)

csc675=> _
```

Virtual Box Queries

QUESTION 1:

```
csc675=> insert into weather_observation(wid, wuser, wtime,wdate,gps,pressure,
temp,data) values(9,103,'12:12','03/03/03', '23 45" 56"', 1600, 50,
 'pictures/9.jpg');
INSERT 0 1
csc675=> select * from weather_observation;
 wid | wuser | wtime |   wdate    |    gps     | pressure | temp |      data
-----+-------+-------+------------+------------+----------+------+--------------
--
   1 |   100 | 12:23 | 2012-03-05 | 45 34" 43  |     1000 |  100 | pictures/1.jp
g
   2 |   101 | 04:23 | 2011-03-05 | 22 32" 43  |     2000 |   60 | pictures/2.jp
g
   3 |   102 | 14:23 | 2003-06-05 | 22 12" 69  |     3000 |   25 | pictures/3.jp
g
   4 |   100 | 05:23 | 2009-06-11 | 10 12" 70  |      850 |   45 | pictures/4.jp
g
   5 |   103 | 04:16 | 2010-10-11 | 10 25" 33  |     1000 |   34 | pictures/5.jp
```

QUESTIONS 2,3,4:

```
csc675=> delete from weather_observation where wid=9;
DELETE 1
csc675=> insert into remote_server(server_id, wid) values(3,5);
INSERT 0 1
csc675=> select * from remote_server;
 server_id | wid
-----------+-----
         1 |   3
         2 |   1
         3 |   5
(3 rows)

csc675=> update weather_observation set gps = '54 67" 78" ' where wid=4;
UPDATE 1
csc675=> select * from weather_observation where wid=4;
 wid | wuser | wtime |   wdate    |     gps     | pressure | temp |      data

-----+-------+-------+------------+-------------+----------+------+-------------
---
   4 |   100 | 05:23 | 2009-06-11 | 54 67" 78"  |      850 |   45 | pictures/4.j
pg
(1 row)

lines 1-5/5 (END)
```

Question 5, 6:

```
LINE 1: select data from weather observation where wid=4;
                         ^
csc675=> update weather_observation set data = ' solar strenght=34566PV'
    where wid=4;
UPDATE 1
csc675=> select data from weather_observation where wid=4;
          data
-------------------------
  solar strenght=34566PV
(1 row)

csc675=> select * from weather_observation;
 wid | wuser | wtime |   wdate    |     gps     | pressure | temp |          dat
a
-----+-------+-------+------------+-------------+----------+------+-------------
------------
   1 |   100 | 12:23 | 2012-03-05 | 45 34" 43   |     1000 |  100 | pictures/1.j
pg
   2 |   101 | 04:23 | 2011-03-05 | 22 32" 43   |     2000 |   60 | pictures/2.j
pg
   3 |   102 | 14:23 | 2003-06-05 | 22 12" 69   |     3000 |   25 | pictures/3.j
pg
   5 |   103 | 04:16 | 2010-10-11 | 10 25" 33   |     1000 |   34 | pictures/5.j
```

Question 7, 8

```
csc675=> select distinct w.wid from weather_observation w, store s,
user_device u where w.wuser = u.id and u.id=s.id and u.name='john';
 wid
-----
   8
   5
(2 rows)

csc675=> select distinct w.wid from weather_observation w, email e, user_device
u where w.wuser = u.id and u.id = e.id and u.name='mike';
 wid
-----
   3
(1 row)

csc675=> _
```

QUESTION 9, 10

```
   3
(1 row)

csc675=> select wid from weather_observation;
 wid
-----
   1
   2
   3
   5
   6
   7
   8
   4
(8 rows)

csc675=> select wid from weather_observation where pressure > 1000;
 wid
-----
   2
   3
   6
(3 rows)

csc675=> _
```

JDBC on Virual Box: The following program executes a query which retrieves the name attributes of the table user_device.

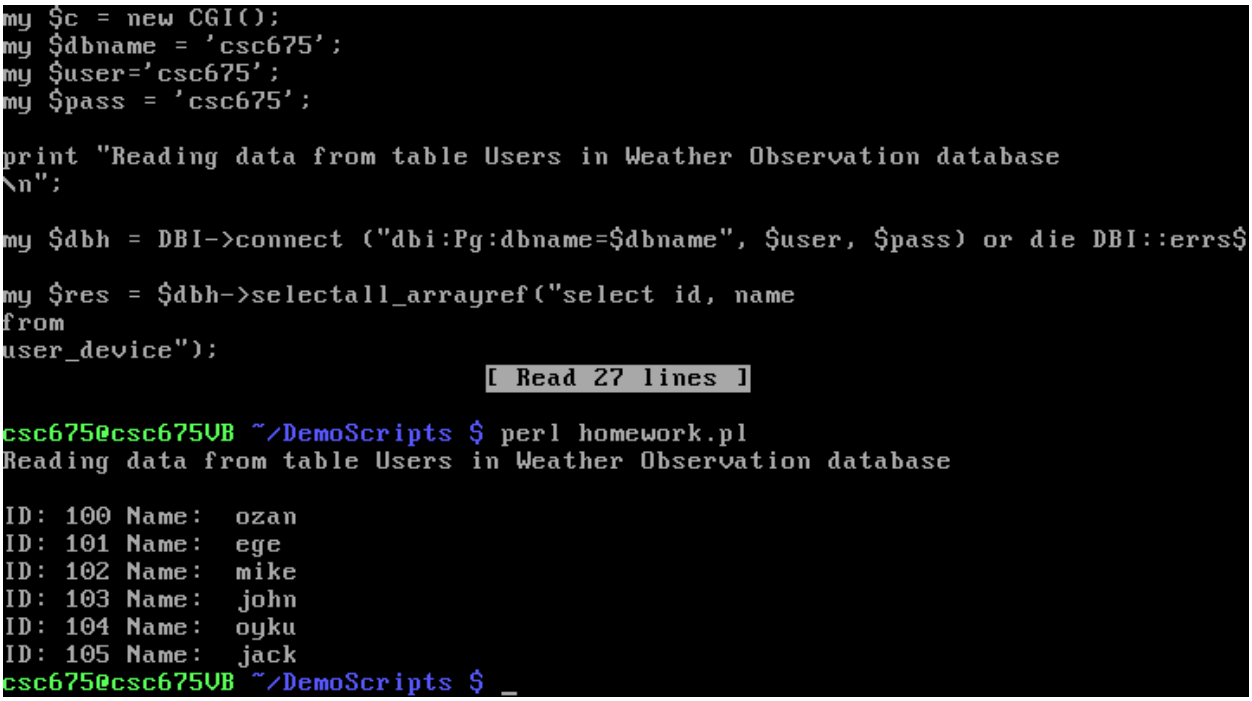
```
  GNU nano 2.3.2                File: test.java

import java.sql.*;
import java.text.*;
import java.io.*;

public class test
{
    Connection       db;        // connection object
    Statement        sql;       // statement to run queries with

    // the constructor does all the work in this simple example

    public test(String argv[])
        throws ClassNotFoundException, SQLException
    {
        String database = argv[0];
        String username = argv[1];
        String password = argv[2];

        // load the JDBC driver for PostgreSQL
        Class.forName("org.postgresql.Driver");
                               [ Read 72 lines ]
^G Get Help  ^O WriteOut  ^W Where Is  ^V Next Page ^U UnCut Text^-| First Line
^X Exit      ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos   ^-? Last Line
  GNU nano 2.3.2                File: test.java

    public test(String argv[])
        throws ClassNotFoundException, SQLException
    {
        String database = argv[0];
        String username = argv[1];
        String password = argv[2];

        // load the JDBC driver for PostgreSQL
        Class.forName("org.postgresql.Driver");

        // connect to the datbase server over TCP/IP
        // (requires that you edit pg_hba.conf
        // as shown in the "Authentication" section of this article)
        db = DriverManager.getConnection("jdbc:postgresql:"+database,
                                         username,
                                         password);

        // create a statement for later use
        sql = db.createStatement();


^G Get Help  ^O WriteOut  ^W Where Is  ^V Next Page ^U UnCut Text^-| First Line
^X Exit      ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos   ^-? Last Line
```


GNU nano 2.3.2 File: test.java
ResultSet results = sql.executeQuery(theQuery);
if (results != null)
{
while (results.next())
{
System.out.println("Name:"+results.getString("name"));
}
}
else
{ System.out.println("No rows found");
}
results.close();
db.close();
}
public static void showUsage()
{
^G Get Help ^O WriteOut ^W Where Is ^V Next Page ^U UnCut Text M-\ First Line
^X Exit ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos M-? Last Line
}
}
else
{ System.out.println("No rows found");
}
results.close();
db.close();
}
public static void showUsage()
{
csc675@csc675VB ~/DemoScripts $ javac test.java
csc675@csc675VB ~/DemoScripts $ java test "csc675" "postgres" "csc675postgres"
Now executing query: "select name from user_device"
Name:ozan
Name:ege
Name:mike
Name:john
Name:oyku
Name:jack
csc675@csc675VB ~/DemoScripts $

SCRIPTING LANGUAGE PERL, EXECUTING QUERY

```
  GNU nano 2.3.2              File: homework.pl

#!/usr/bin/perl

use DBI;
use CGI;
use CGI::Carp qw(fatalsToBrowser);
use strict;

my $c = new CGI();
my $dbname = 'csc675';
my $user='csc675';
my $pass = 'csc675';

print "Reading data from table Users in Weather Observation database
\n";

my $dbh = DBI->connect ("dbi:Pg:dbname=$dbname", $user, $pass) or die DBI::errs$

my $res = $dbh->selectall_arrayref("select id, name
from
user_device");
                          [ Read 27 lines ]
^G Get Help  ^O WriteOut  ^W Where Is  ^V Next Page ^U UnCut Text^- First Line
^X Exit      ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos   M-? Last Line
```

```
  GNU nano 2.3.2              File: homework.pl


print "Reading data from table Users in Weather Observation database
\n";

my $dbh = DBI->connect ("dbi:Pg:dbname=$dbname", $user, $pass) or die DBI::errs$

my $res = $dbh->selectall_arrayref("select id, name
from
user_device");

for my $row(@$res)
{
        print 'ID: ' ,@$row[0], ' Name:  ',@$row[1], "\n";
        }
$dbh->disconnect();




                          [ Read 27 lines ]
^G Get Help  ^O WriteOut  ^W Where Is  ^V Next Page ^U UnCut Text^- First Line
^X Exit      ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos   M-? Last Line
```



JDBC ODBC WITH GUI(GUI for Extra Credit Question)

```
package javadatabase;
import java.util.*;
import java.sql.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;


/**
 *
 * @Ozan Gazi Onder
 * extra credit
 * program that allows user to add and display weather_Observation data in GUI
 */
public class Wgui extends javax.swing.JFrame {

    /**
     * Creates new form Wgui
```

```
 */
public Wgui() {
   initComponents();
}


@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

   jScrollPane1 = new javax.swing.JScrollPane();
   area = new javax.swing.JTextArea();
   displayButton = new javax.swing.JButton();
   nameField = new javax.swing.JTextField();
   gpsField = new javax.swing.JTextField();
   pressureField = new javax.swing.JTextField();
   tempField = new javax.swing.JTextField();
   dataField = new javax.swing.JTextField();
   jLabel1 = new javax.swing.JLabel();
   jLabel2 = new javax.swing.JLabel();
   jLabel3 = new javax.swing.JLabel();
   enterButton = new javax.swing.JButton();

   setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

   area.setBackground(new java.awt.Color(0, 0, 0));
   area.setColumns(20);
   area.setForeground(new java.awt.Color(255, 255, 102));
   area.setRows(5);
   jScrollPane1.setViewportView(area);

   displayButton.setBackground(new java.awt.Color(51, 51, 255));
   displayButton.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
   displayButton.setForeground(new java.awt.Color(255, 255, 255));
   displayButton.setText("display data");
   displayButton.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
         displayButtonActionPerformed(evt);
      }
   });

   nameField.setText("Name");

   gpsField.setText("gps coordinates");
```

```
        pressureField.setText("pressure");

        tempField.setText("temperature");

        dataField.setText("additional data");

        jLabel1.setText("Name");

        jLabel2.setText("gps");

        jLabel3.setText("pressure");

        enterButton.setBackground(new java.awt.Color(102, 102, 255));
        enterButton.setFont(new java.awt.Font("Tahoma", 0, 12)); // NOI18N
        enterButton.setForeground(new java.awt.Color(255, 255, 255));
        enterButton.setText("SUBMIT");
        enterButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                enterButtonActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jScrollPane1)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addGap(54, 54, 54)
                .addComponent(displayButton, javax.swing.GroupLayout.PREFERRED_SIZE, 160,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap(80, Short.MAX_VALUE)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jLabel1, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.PREFERRED_SIZE, 38, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel2, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.PREFERRED_SIZE, 38, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel3, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.PREFERRED_SIZE, 38, javax.swing.GroupLayout.PREFERRED_SIZE))
```

```
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
              .addComponent(gpsField, javax.swing.GroupLayout.PREFERRED_SIZE, 125,
javax.swing.GroupLayout.PREFERRED_SIZE)
              .addComponent(nameField, javax.swing.GroupLayout.PREFERRED_SIZE, 125,
javax.swing.GroupLayout.PREFERRED_SIZE)
              .addComponent(pressureField, javax.swing.GroupLayout.PREFERRED_SIZE, 125,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
              .addGroup(layout.createSequentialGroup()
                .addGap(79, 79, 79)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                  .addComponent(dataField, javax.swing.GroupLayout.PREFERRED_SIZE, 132,
javax.swing.GroupLayout.PREFERRED_SIZE)
                  .addComponent(tempField, javax.swing.GroupLayout.PREFERRED_SIZE, 132,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(149, 149, 149))
              .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(enterButton, javax.swing.GroupLayout.PREFERRED_SIZE, 234,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(60, 60, 60))))
        );
        layout.setVerticalGroup(
          layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
          .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
            .addContainerGap()
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
              .addComponent(nameField, javax.swing.GroupLayout.PREFERRED_SIZE, 24,
javax.swing.GroupLayout.PREFERRED_SIZE)
              .addComponent(tempField, javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE)
              .addComponent(jLabel1))
            .addGap(7, 7, 7)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
              .addComponent(gpsField, javax.swing.GroupLayout.PREFERRED_SIZE, 23,
javax.swing.GroupLayout.PREFERRED_SIZE)
              .addComponent(dataField, javax.swing.GroupLayout.PREFERRED_SIZE, 29,
javax.swing.GroupLayout.PREFERRED_SIZE)
              .addComponent(jLabel2))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
```

```
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(pressureField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jLabel3)
                .addComponent(enterButton, javax.swing.GroupLayout.PREFERRED_SIZE, 36,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 61,
Short.MAX_VALUE)
            .addComponent(displayButton)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 213,
javax.swing.GroupLayout.PREFERRED_SIZE))
        );

        pack();
    }// </editor-fold>

    //input to the user_device and weather
    private void enterButtonActionPerformed(java.awt.event.ActionEvent evt) {
        String name = nameField.getText();
        WeatherObservation w = new WeatherObservation();
         String n = nameField.getText();
        String gps = gpsField.getText();
        int pres= Integer.parseInt(pressureField.getText());
        int temp = Integer.parseInt(tempField.getText());
        String data = dataField.getText();

        w.insertUser(n);
        w.insertData(n, gps, pres, temp, data);
    }

    //button on click, displays the list of weather observation data
    private void displayButtonActionPerformed(java.awt.event.ActionEvent evt) {
      String query;

      try
        {
          query = "SELECT * FROM Weather_Observation";
          Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
          Connection con = DriverManager.getConnection("jdbc:odbc:homework6");
           Statement st = con.createStatement();
           ResultSet rs = st.executeQuery(query);

           //display format in the text area
```

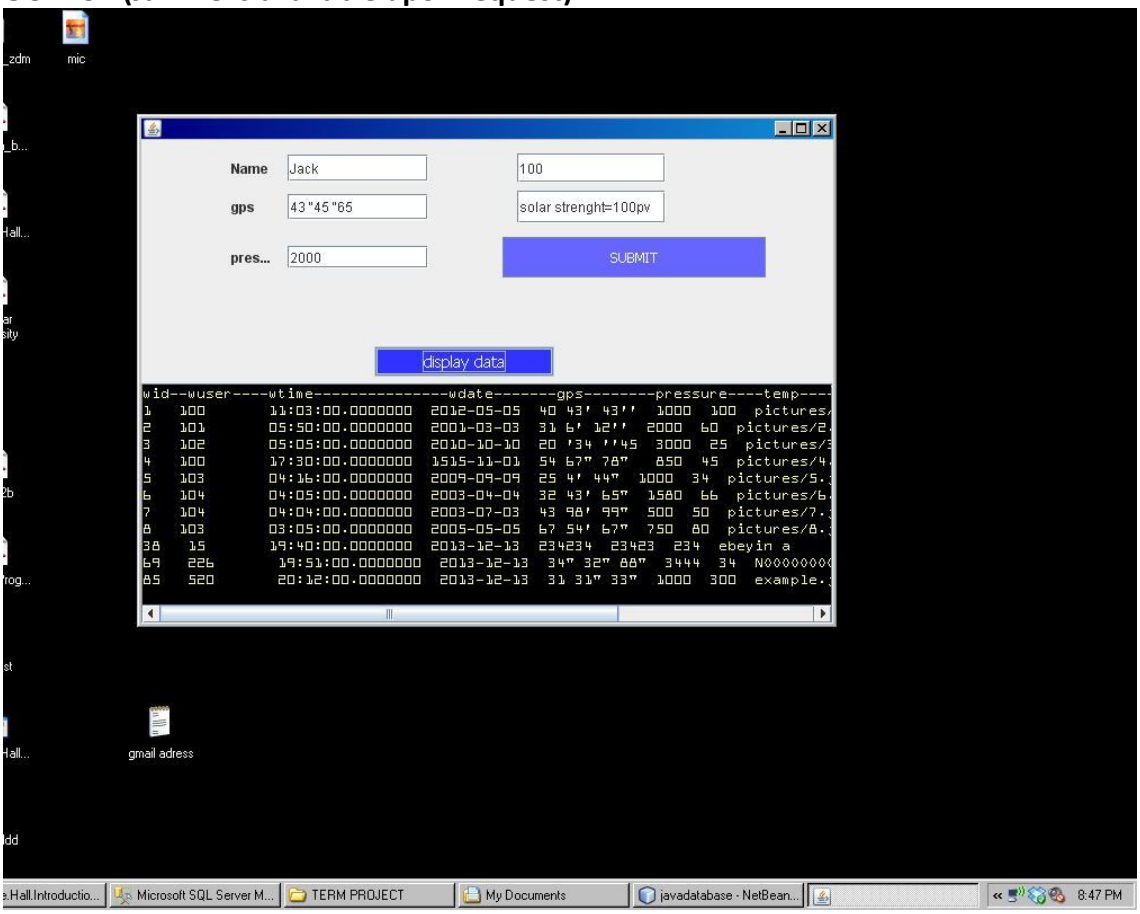
```
        String i1="", i2="", i3="", i4="", i5="", i6="", i7="", i8="";
        area.append("wid--wuser----wtime---------------wdate-------gps--------pressure----temp----
data\n");
        while(rs.next())
        {
          i1=rs.getString(1);
          i2=rs.getString(2);
          i3=rs.getString(3);
          i4=rs.getString(4);
          i5=rs.getString(5);
          i6=rs.getString(6);
          i7=rs.getString(7);
          i8=rs.getString(8);
          area.append(i1+"  "+i2+"     "+i3+"  "+i4+"  "+i5+"  "+i6+"  "+i7+"  "+i8);
          area.append("\n");
        }

      }
      catch(Exception ex)
      {
       ex.printStackTrace();
      }

   }

  //
   public static void main(String args[]) {

      /* Create and display the form */
      java.awt.EventQueue.invokeLater(new Runnable() {
          public void run() {
             new Wgui().setVisible(true);
          }
      });
   }
   // Variables declaration
   public javax.swing.JTextArea area;
   public javax.swing.JTextField dataField;
   public javax.swing.JButton displayButton;
   public javax.swing.JButton enterButton;
   public javax.swing.JTextField gpsField;
   private javax.swing.JLabel jLabel1;
   private javax.swing.JLabel jLabel2;
   private javax.swing.JLabel jLabel3;
```

```
    private javax.swing.JScrollPane jScrollPane1;
    public javax.swing.JTextField nameField;
    public javax.swing.JTextField pressureField;
    public javax.swing.JTextField tempField;
    // End of variables declaration
}


**********************************

//CLASS Weather_Observation that reads data
//and adds them to the database
package javadatabase;

import java.sql.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import java.util.Date;
import java.text.DateFormat;
import java.text.SimpleDateFormat;


public class WeatherObservation
{
    public Random rand = new Random();
    public int userId=0;

    public WeatherObservation()//consructor
    { }

    //reads the data and insert it to the database

    public void insertUser(String name)
    {

        userId = rand.nextInt(1000); //random userId is assigned
        String query;


        //create a new user in the database
       try
       {
```

```
        query = "insert into user_device(id, name) values ("+userId+", '"+name+"')";
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        Connection con = DriverManager.getConnection("jdbc:odbc:homework6");
         Statement st = con.createStatement();
         ResultSet rs = st.executeQuery(query);
         con.close();

      }
      catch(Exception ex)
      {
       ex.printStackTrace();
      }

   }

   //insert data to the table Weather_Observation
   public void insertData(String name, String gps, int pressure, int temp, String data)
   {
      int wid = rand.nextInt(100); //random wid assigned
      DateFormat timeFormat = new SimpleDateFormat("HH:mm");
      DateFormat dateFormat = new SimpleDateFormat("MM-dd-yyyy");
      Date currentDate = new Date();

       String date = (String)dateFormat.format(currentDate);
       String time = (String)timeFormat.format(currentDate);
       String query;

      try
      {
       query = "insert into Weather_Observation(wid, wuser, wtime, wdate, gps, pressure, temp,
data)"
            +" values ("+wid+","+userId+",'"+time+"','"+date+"',
'"+gps+"',"+pressure+","+temp+",'"+data+"' )";
         Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
         Connection con = DriverManager.getConnection("jdbc:odbc:homework6");
         Statement st = con.createStatement();
         ResultSet rs = st.executeQuery(query);
         con.close();

      }
      catch(Exception ex)
      {
       ex.printStackTrace();
      }
```

```
    }

}
```

OUTPUT (Jar File is available upon request)



EXTRA CREDIT(WEB APP)

The following program is executed on the browser, user can enter a new observation data, or see all the data that is stored in the database. Mysql in phpmyadmin is used to store weather observation data. Php is used to implemend back and database connection and CSS is used for the front and in order to display the data.

```
<!DOCTYPE html>
<html>
  <head>
    <link href="bootstrap.css" rel="stylesheet">
    <link href="signin.css" rel="stylesheet">

  </head>

  <body>
    <div class="container">

    <form class="form-signin" form action="db.php" method="post">
      <h2 class="form-signin-heading">Weather Observation</h2>


      <button class="btn btn-lg btn-primary btn-block" type="submit">Display Data</button>
    </form>


      <form class="form-signin" form action="db2.php" method="post">
      <h2 class="form-signin-heading">Add your observation data</h2>

      Name:
      <input type="text" class="form-control" name ="name" required autofocus>
      User Id Number:
     <input type="text" class="form-control" name ="wuser" required autofocus>
     GPS Coordinates
     <input type="text" class="form-control" name ="gps"  required autofocus>
     Barometric Pressure(mB)
     <input type="text" class="form-control" name ="pressure" required autofocus>
     Temperature(F)
     <input type="text" class="form-control" name ="temp" required autofocus>
     Additional Data(F)
     <input type="text" class="form-control" name ="data" required autofocus>
      <button class="btn btn-lg btn-primary btn-block" type="submit">Add</button>
    </form>



  </div> <!-- /container -->
</body>
```

```
</html>
```

Db.php

```
$username = "oonder";
 $password="abc";
 $hostname="localhost";
 $db="student_oonder";


   $con = mysqli_connect($hostname, $username, $password, $db)
         or die("Unable to connect");

   $result = mysqli_query($con,"SELECT * FROM Weather_Observation");

   echo "<table border='1'>
   <tr>
   <th>WEATHER-ID</th>
   <th>USER</th>
   <th>TIME</th>
   <th>DATE</th>
   <th>GPS COORDINATES</th>
   <th>PRESSURE</th>
   <th>TEMPERATURE</th>
   <th>ADDITIONAL DATA</th>
   </tr>";
   while($row = mysqli_fetch_array($result))
   {
      echo "<tr>";
      echo "<td>" . $row['wid'] . "</td>";
      echo "<td>" . $row['wuser'] . "</td>";
      echo "<td>" . $row['wtime'] . "</td>";
      echo "<td>" . $row['wdate'] . "</td>";
      echo "<td>" . $row['gps'] . "</td>";
      echo "<td>" . $row['pressure'] . " mBar"."</td>";
      echo "<td>" . $row['temp'] . "F". "</td>";
      echo "<td>" . $row['data'] . "</td>";
      echo "</tr>";


   }



mysqli_close($con);
```

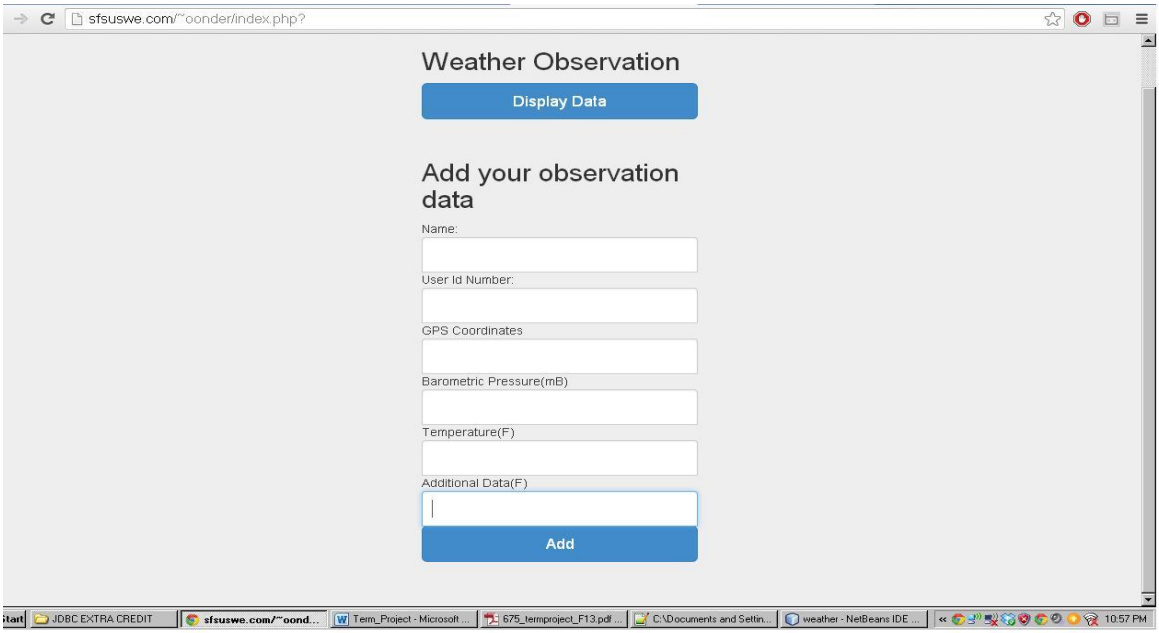
```
?>
```

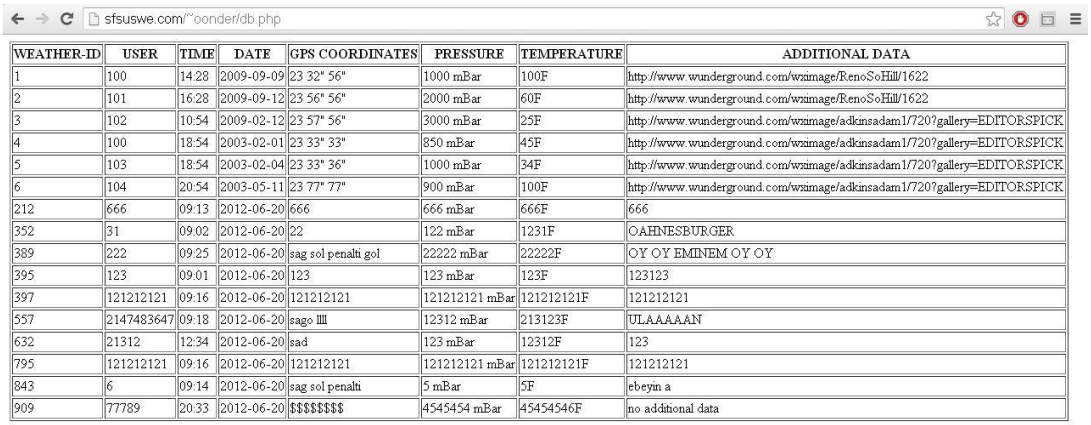
Db2.php

```
<html>
   <head>
     <link href="bootstrap.css" rel="stylesheet">
     <link href="signin.css" rel="stylesheet">

     <?php

 $username = "oonder";
 $password="abc";
 $hostname="localhost";
 $db="student_oonder";

 //wid be assigned randomly
   $wid = (rand(200,1000));
   $name = $_POST["name"];
   $user_id = $_POST["wuser"];
   $gps = $_POST["gps"];
   $pressure = $_POST["pressure"];
   $temp = $_POST["temp"];
   $data = $_POST["data"];

   //current date and time functions
   $time = gmDate("H:i") ;
   $date = date('m.d.Y');



   $con = mysqli_connect($hostname, $username, $password, $db)
          or die("Unable to connect");


   $result = mysqli_query($con,"INSERT INTO user_device (id, name)
                   VALUES ('{$user_id}', '{$name}')")
                    or die("Unable to connect");

   $result = mysqli_query($con,"INSERT INTO Weather_Observation (wid, wuser,
                         wtime, wdate, gps, pressure, temp, data)
                   VALUES ('{$wid}', '{$user_id}', '{$time}', '{$date}',
                         '{$gps}', '{$pressure}', '{$temp}', '{$data}')")
                    or die("Unable to connect");

   if($result) echo"Added!";
```

```
   mysqli_close($con);


?>

   </head>

   <body>
    <div class="container">

    <form class="form-signin" form action="index.php" method="post">

     <button class="btn btn-lg btn-primary btn-block" type="submit">Home Page</button>
    </form>

        </div> <!-- /container -->
  </body>
</html>
```

OUTPUT





WEATHER-ID	USER	TIME	DATE	GPS COORDINATES	PRESSURE	TEMPERATURE	ADDITIONAL DATA
1	100	14:28	2009-09-09	23 32" 56"	1000 mBar	100F	http://www.wunderground.com/wximage/RenoSoHill/1622
2	101	16:28	2009-09-12	23 56" 56"	2000 mBar	60F	http://www.wunderground.com/wximage/RenoSoHill/1622
3	102	10:54	2009-02-12	23 57" 56"	3000 mBar	25F	http://www.wunderground.com/wximage/adkinsadam1/720?gallery=EDITORSPICK
4	100	18:54	2003-02-01	23 33" 33"	850 mBar	45F	http://www.wunderground.com/wximage/adkinsadam1/720?gallery=EDITORSPICK
5	103	18:54	2003-02-04	23 33" 36"	1000 mBar	34F	http://www.wunderground.com/wximage/adkinsadam1/720?gallery=EDITORSPICK
6	104	20:54	2003-05-11	23 77" 77"	900 mBar	100F	http://www.wunderground.com/wximage/adkinsadam1/720?gallery=EDITORSPICK
212	666	09:13	2012-06-20	666	666 mBar	666F	666
352	31	09:02	2012-06-20	22	122 mBar	1231F	OAHNESBURGER
389	222	09:25	2012-06-20	sag sol penalti gol	22222 mBar	22222F	OY OY EMINEM OY OY
395	123	09:01	2012-06-20	123	123 mBar	123F	123123
397	121212121	09:16	2012-06-20	121212121	121212121 mBar	121212121F	121212121
557	2147483647	09:18	2012-06-20	sago lllll	12312 mBar	213123F	ULAAAAAN
632	21312	12:34	2012-06-20	sad	123 mBar	12312F	123
795	121212121	09:16	2012-06-20	121212121	121212121 mBar	121212121F	121212121
843	6	09:14	2012-06-20	sag sol penalti	5 mBar	5F	ebeyin a
909	77789	20:33	2012-06-20	$$$$$$$$	4545454 mBar	45454546F	no additional data

Summary and Conclusion: The Weather Observation Database consists of three entities: User_Device, Weather_Observation and Remote_Server. Since either a user or a device can add data to the Weather_Observation database, we had combined them as one single entity (for simplicity) and each user device, or user has their unique user_id that can be referenced from Weather_Observation Entity. Since a user or device can either email or manually store data, we had created two relationships that represent email and store tables. Furthermore, the data in the Weather_Observation table can be added to a remote server which also represent the entity Remote_Server that contains a unique ID and and ID number that also represents the data in the Weather_Observation table. Removing a user from email or store tables will not remove the data in the Weather_Observation table. The Weather_Observation is implemented in three Database management software: Microsoft SQL Server, Postgres, and MYSQL (for the web app).

Future Work: The database system can be extended such that different media objects can be uploaded to the server and can be retrieved upon request. If we decide to use the database for commercial purposes, we have to ask permission in order to use the copyrighted media objects to avoid copyright infringement.