Assignment 2 Solutions

October 28, 2005

1. (15)
   (a) update works
       set salary = salary * 1.02
       where company_name = "IBM"
   (b) update works
       set salary = salary*1.05
       where company_name = "IBM"
       and employee_name in (select manager_name from manages) ;
   (c) delete from works
       where company_name = “Startup Corp”

2. (15) select manages.manager_name,avg(salary)
    # next two lines equivalent to inner join
    from manages m, works w
    where m.employee_name = w.employee_name
    group by manages.manager_name

Updates should not be allowed in this view because there is no way to
determine how to change the underlying data. For example, suppose the
request is "change the average salary of employees working for Smith to
$200". Should everybody who works for Smith have their salary changed
to $200? Or should the first (or more, if necessary) employee found who
works for Smith have their salary adjusted so that the average is $200?
Neither approach really makes sense.
3. (10)
create domain company_names char(20)
create domain city_names char(30)
create domain person_names char(20)
create domain state_names char(2)

create table employee
(employee_name person_names,
street char(30),
city city_names,
state state_names,
zip numeric(5),
primary key (employee_name))

create table works
(employee_name person_names,
company_name company_names,
salary numeric(8, 2),
primary key (employee_name,company_name),
foreign key (employee_name) references employee,
foreign key (company_name) references company)

create table company
(company_name company_names,
city city_names,
state state_names,
zip numeric(5),
primary key (company_name))

create table manages
(employee_name person_names,
manager_name person_names,
primary key (employee_name,manager_name),
foreign key (employee_name) references employee,
foreign key (manager_name) references employee(employee_name) )

4. (15)
(a) intersection
(select * from r1) intersect (select * from r2)
# can also be solved using 'not in' clause
(b) union
(select * from r1) union (select * from r2)

(c) join
select r1.X, r2.X, r3.Z
from r1, r2
where r1.Y = r2.Y
# can also be solved using inner join clause
(d) difference
\[(select * from r1) except all (select * from r2)\]
# can also be solved using 'not in' clause

5. For simplicity, we present a variant of the SQL syntax. As part of the
create table expression for address we include:

foreign key (name) references salaried-worker or hourly-worker

To enforce this constraint, whenever a tuple is inserted into the address
relation, a lookup on the name value must be made on the salaried-worker
relation and (if that lookup failed) on the hourly-worker relation (or vice-
versa).

6. (15) JDBC solution using metadata taking resultset as an argument. One
example:

```java
import java.sql.*;
import java.io.*;
import java.text.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class metaDataPostgreSQL extends HttpServlet{
    String theURL = "jdbc:postgresql:test1";

    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException
    {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html><head>");
        String title = "Connect to PostGreSQL using MetaData";
        out.println("<title>" + title + "</title>");
        out.println("</head>");
        out.println("<body bgcolor="white">");
        out.println("<a href="/..helloworld.html">");
        out.println("<img src="/..images/code.gif" height=24 " +
            "width=24 align=right border=0 alt="view code"</a>");
        out.println("<a href="/..index.html">");
        out.println("<img src="/..images/return.gif" height=24 " +
            "width=24 align=right border=0 alt="return"</a>");
        out.println("<h1>" + title + "</h1>");
        out.println("<P>The url: " +theURL+"</P>" );
        dumpDatabase(out);
        tableFinish(out);
    }
```
```java
out.println("</body></html>" );

public String metaDataFunction(ResultSet results) throws SQLException {
    String returnString = "" ;
    ResultSetMetaData metaData = results.getMetaData();
    int numColumns = metaData.getColumnCount();
    while (results.next()) {
        returnString += "<TR>";
        for(int i = 1 ; i <= numColumns ; i++){
            returnString += "<TD>" ;
            switch(metaData.getColumnType(i)){
                case Types.NUMERIC: returnString += String.valueOf(results.getInt(i)) ; break ;
                case Types.DOUBLE: returnString += String.valueOf(results.getDouble(i)) ; break ;
                case Types.DATE: returnString += (results.getDate(i)).toString() ; break ;
                case Types.VARCHAR: returnString += (results.getString(i)) ; break ;
                default: System.out.println("type: "+ metaData.getColumnType(i)+" not defined") ;
                returnString += String.valueOf(results.getString(i)) ; break ;
            }
        }
        returnString += "</TD>" ;
    }
    returnString += "</TR>
" ;
    return returnString ;
}

public String tableHeader(ResultSet set) throws SQLException {
    String returnString = "" ;
    returnString +="<TABLE BORDER>" ;
    ResultSetMetaData metaData = set.getMetaData();
    int numColumns = metaData.getColumnCount();
    for(int i = 1 ; i <= numColumns ; i++){
        returnString += "<TD>" ;
        returnString += metaData.getColumnName(i) ;
        returnString += "</TD>" ;
    }
    returnString +="</TR>\n" ;
    return returnString ;
}

public void tableFinish(PrintWriter out){
    out.println("</TABLE>" );
}
```

public void dumpDatabase(PrintWriter out) {
    try {
        // Here we load the JDBC driver.
        Class.forName("org.postgresql.Driver");
        } catch (ClassNotFoundException e) {
            out.println("Unable to load driver!");
            return;
        }
    try {
        // Here I'm connecting to my database
        /* using the standard JDBC URL for PostgreSQL
        * (jdbc:postgresql:) followed by the name of
        * our database, vsbabu. I'm also passing the
        * username "vsbabu" and the password ". */
        Connection pSQL = DriverManager.getConnection(theURL, "ndunn", "");
        /* This is where we create a Statement object
        * and execute a SQL query using it. */
        Statement query = pSQL.createStatement();
        /* Here we'll loop through the ResultSet,
        * printing out state code and name */
        try {
            out.print(tableHeader(results));
            out.print(metaDataFunction(results));
        }
        catch(SQLException se){ out.println("SQL Exception: " + se); }
        // Close all of our JDBC resources.
        results.close();
        query.close();
        pSQL.close();
    } catch (SQLException se) {
        // Give feedback for any SQL errors.
        System.out.println("A SQL Exception: " + se.getMessage());
        se.printStackTrace(out);
    }
}

7. (15)
create view v as
select r1.A1, r2.A2, r3.A2*1.05
from R r1, R r2, R r3
# multiple inner join also works