Still More SQL – Alter, Create, and ResultSetMetaData

Now suppose that the address book previously created is to be used by a club that charges dues. We can add a column to the database using the **Alter** command. We can write

"Alter Table AddressTable Add Dues decimal (10, 2)"

The general form of the Alter command is

"Alter Table table-name Add column-name datatype"

```
package addresses;

import java.sql.*;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

/*	AddColumn adds a column to the database. */
public class AddColumn extends HttpServlet
{
	public void doGet (HttpServletRequest request, HttpServletResponse response)
	{
		try
		{
			// Get a jdbc-odbc bridge and connect to addresses.mdb.
			Class.forName ("sun.jdbc.odbc.JdbcOdbcDriver");
			Connection con = DriverManager.getConnection ("jdbc:odbc:addresses");

			PrintWriter out = response.getWriter ();
			Page.createHeader (out, "Address List");

			// Get the name of the new column and its datatype.
			String columnName = request.getParameter ("columnName");
			String datatype = request.getParameter ("datatype");

			// Create a statement and query the database.
			Statement stmt = con.createStatement ();
			String query = "Alter Table AddressTable Add " + columnName + " " + datatype;
			int success = stmt.executeUpdate (query);
			if (success == 0) out.println ("Alter error.");
			else out.println ("Column inserted.");
			stmt.close ();

			Page.createFooter (out);
		} catch (IOException ex) {System.out.println ("IO Exception.");}
		  catch (ClassNotFoundException exc) {System.out.println ("Class Not Found Exception");}
		  catch (SQLException exs) {System.out.println ("SQL Exception");}
	} // doGet
} // class AddColumn
```

The data type specifies what type of data the column can hold. The table below contains the most common data types in SQL. The table was copied from the web site, http://www.w3schools.com. (This site has a number of excellent tutorials on SQL as well as other topics. It is a very good way to learn how to create web applications.)

Data Type	Description

integer(size)
int(size)
smallint(size)
	Hold integers only. The maximum number of digits is specified in parenthesis.

decimal(size,d)
numeric(size,d)
	Hold numbers with fractions. The maximum number of digits is specified in "size". The maximum number of digits to the right of the decimal is specified in "d".

float(n) real double	Floating point number with n binary digits of precisions. 32-bit floating point number. 64-bit floating point number.
char(size)	Holds a fixed length string (can contain letters, numbers, and special characters). The fixed size is specified in parenthesis.
varchar(size)	Holds a variable length string (can contain letters, numbers, and special characters). The maximum size is specified in parenthesis.
date(yyyymmdd)	Holds a date

When a member pays her dues, the amount can be entered using the Update command. To do this, change the update method above to

```
	"Update AddressTable Set Dues = " + dues + " Where Name = '" + name + "'"
```

It is important to note, that the numeric field, dues, is not surrounded by single quotes. Only strings are. So anytime you have a numeric field, leave out the single quotes.

We can also create a new table to be added to the database. The command here is

```
	"Create Table table_name (column_name1 datatype1, column_name2 datatype2, …)"
```

If the club wants to create a table with its officers, this can be done with

```
	"Create Table Officers (Name varchar (30), Office varchar (30))"
```

This will create a new table in the addresses database with two columns, Name and Office. Both are strings of variable size not to exceed 30 characters. Then to add data, use the Insert command, for example

```
	"Insert Into Officers" + " Values ('" + name + "', '" + office + "')"
```

where name and office have been provided by the user.

```
public void doGet (HttpServletRequest request, HttpServletResponse response)
{
	try
	{
		// Get a jdbc-odbc bridge and connect to addresses.mdb.
		Class.forName ("sun.jdbc.odbc.JdbcOdbcDriver");
		Connection con = DriverManager.getConnection ("jdbc:odbc:addresses");

		Statement stmt = con.createStatement ();
		String query = "Create Table Officers (Name varchar (30), Office varchar (30))";
		int success = stmt.executeUpdate (query);
		if (success != 0) out.println ("Table created.");
		else out.println ("Create error.");
		stmt.close ();
```

```
        } catch (IOException ex) {System.out.println ("IO Exception.");}
          catch (ClassNotFoundException exc) {System.out.println ("Class Not Found Exception");}
          catch (SQLException exs) {System.out.println ("SQL Exception");}
} // createTable

// Insert the new data into the database.
public doGet (HttpServletRequest request, HttpServletResponse response)
{
        try
        {
                // Get a jdbc-odbc bridge and connect to addresses.mdb.
                Class.forName ("sun.jdbc.odbc.JdbcOdbcDriver");
                Connection con = DriverManager.getConnection ("jdbc:odbc:addresses");

                // Get the data to be inserted.
                String name = request.getParameter ("name");
                String office = request.getParameter ("office");

                // Create the statement and execute the update.
                Statement stmt = con.createStatement ();
                String query = "Insert Into Officers" + " Values ('" + name + "', '" + office + "')";
                int success = stmt.executeUpdate (query);
                if (success != 0) out.println ("Data inserted.");
                else out.println ("Insert error.");
                stmt.close ();
        } catch (IOException ex) {System.out.println ("IO Exception.");}
          catch (ClassNotFoundException exc) {System.out.println ("Class Not Found Exception");}
          catch (SQLException exs) {System.out.println ("SQL Exception");}
} // insertNewData
```

With two tables, you can perform more complicated queries that get data from both tables. For example, you can get the e-mail address and telephone number of the club president. Since we now have two tables, it is necessary to specify which table each column name refers to. As in Java, this is done using a period to determine the *path*.

```
        "Select * Officers.Name, AddressTable.Email, AddressTable.Telephone "
        + "From AddressTable, Officers"
        + " Where  Officers.Office = 'President' "
```

StringBuffers

Many of the queries really should be done using a StringBuffer. A StringBuffer is like a String, but it can be modified. Each time something is appended to a String, a new object is allocated and the old one is left for the garbage collector to delete. Unlike Strings, a StringBuffer must be instantiated. Its constructor can either have no parameters or some initial capacity. If the former, the initial size will be 16 characters. The previous query would be better written as:

```
        StringBuffer queryString = new StringBuffer (50);
        queryString = queryString.append ("Insert Into Officers Values ('" )
                        .append (name).append ("', '").append (office).append ("')");
        String query = queryString.toString ();
```

ResultSetMetaData

MetaData can be used to obtain information about a database table such as the number of columns and the names of the columns. This can be useful if you wish to display a table but do not know everything about it. You can write a generic method that can be used for any database table, given the table's name.

```
// displayData sends a copy of the database to the client formatted as an html table.
public void displayData (PrintWriter out, Connection con, String tableName)
{
    try
    {
        Statement stmt = con.createStatement ();
        String query = "Select * From " + tableName;
        ResultSet rs = stmt.executeQuery (query);
        ResultSetMetaData metaData = rs.getMetaData ();
        int numberColumns = metaData.getColumnCount ();

        out.println ("<table border='1' cellspacing='10'>");
        // Use the table name as the caption for the html table.
        out.println ("<caption>" + tableName + "</caption>");

        // Display the column names in the heading.
        out.println ("<tr>");
        for (int count = 1; count <= numberColumns; count ++)
            out.println ("<th>"+metaData.getColumnName (count)+"</th>");
        out.println ("</tr>");

        // Display all the data in the table.
        while (rs.next ())
        {
            out.println ("<tr>");
            for (int count = 1; count<= numberColumns; count ++)
                out.println ("<td>"+rs.getString (count)+"</td>");
            out.println ("</tr>");
        }
        out.println ("</table>");
        stmt.close ();
    } catch (SQLException es) {System.out.println ("SQL Exception");}
} // displayData
```

References

1. Marty Hall & Larry Brown, *Core Servlets and Java Server Pages*, First Edition, Sun Microsystems Press/Prentice-Hall PTR Book, 2003.
2. Karl Moss, *Java Servlets Developer's Guide*, McGraw-Hill/Osborne, 2002.
3. W3Schools Online Web Tutorials, http://www.w3schools.com.