Interactive Applications for Teaching Basic Concepts of Database and Internet Programming

Richard Cooper

Outline

- Motivation
- Examples of hidden techniques
- Examples of interactive software
- The need for a common toolkit

Teaching Practical Topics

Teaching about many **practical computing topics** is best done through **practical experience**

- programming exercises
- experience with databases
- building distributed and network applications

This approach is hampered by the nature of **commercial software**

- much is **hidden**
- this is vital or the products would not be usable
- this is notably true for database systems and networks

The Main Problem

Commercial systems **hide** the technique

- users are given interfaces which allow them to express what they want, not how it is achieved
- all the difficult stuff is hidden away

Many things are done **automatically**

- user decisions are transformed into the underlying mechanisms which are then executed
- DB queries become optimised relational algebra
- browsers generate HTTP messages

There are **no direct interfaces** to the underlying mechanisms

Database Examples I

The mapping process from conceptual schema to implementation schema

- either automated using design tool
- or done by hand which means knowledge needed before you can do it
- there is no means of seeing it working

Normalisation

- concepts difficult to understand
- no way of visualising anomalies
- no way of entering functional dependencies
- or of seeing a database being normalised

Database Examples II

Query Processing

- you can enter queries (SQL or graphically) and see the result
- intervening processes not shown
- formal languages cannot be entered directly
- optimisation techniques not visualisable

Transaction Management

- no way of visualising transaction processing
- cannot see concurrency control mechanisms
- or of showing recovery, rollback, etc.

Internet Examples I

Static web site structure and client side scripts can be seen through View Source

- but dynamic web site structure cannot be seen

HTTP messages are invisible

- as are cookies, session and context objects, etc.

Database connection is invisible

- It would useful to show how programs access the database, merge in HTML templates and produce web pages

Internet Examples II

XHTML cleaning is not visible

- visualisation of hierarchy
- display of errors

Web Service Construction

- SOAP visualisation
- how do you turn a module into a web service

XML Programming Examples

It would be nice to see how:

- SAX programs step through a file and produce output
- DOM trees are built
- XSLT processes XML files
- a comparison of DTDs and XML Schema

Examples of Interactive Software

Teaching Relational Database System

- ER, Normalisation, Relational Algebra, etc.

Teaching Web Database System

- Component diagram, browser, feedback panel

XML programming

- Showing how DOM, SAX and XSLT produce output

JDBC programming

Web Services

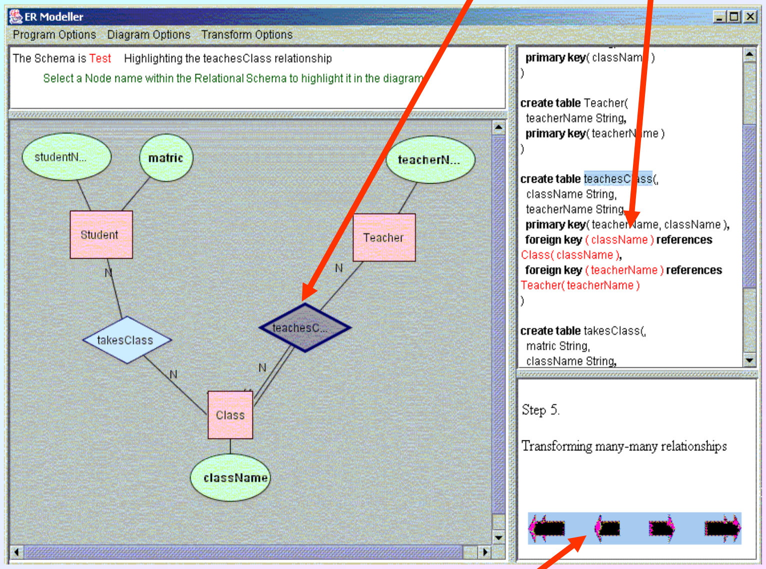
TRDB

This is a Java program with these modules

- a core module which implements tables, views, etc.
- an ER diagram tool
  - permits drawing the diagram
  - shows the stages of turning this into tables
  - allows correspondences to be shown between ER fragments and table fragments
- Relational Algebra and Calculus interfaces
  - expressions can be entered and results seen
- SQL interface
  - with mapping to relational algebra
- Transaction Visualiser
- Normalisation Visualiser

TRDB

Correspondence



The Schema is Test Highlighting the teachesClass relationship

Select a Node name within the Relational Schema to highlight it in the diagram

```
 primary key( className )
)

create table Teacher(
 teacherName String,
 primary key( teacherName )
)

create table teachesClass(,
 className String,
 teacherName String,
 primary key( teacherName, className ),
 foreign key ( className ) references
Class( className ),
 foreign key ( teacherName ) references
Teacher( teacherName )
)

create table takesClass(,
 matric String,
 className String,
```

Step 5.

Transforming many-many relationships

Progress Control

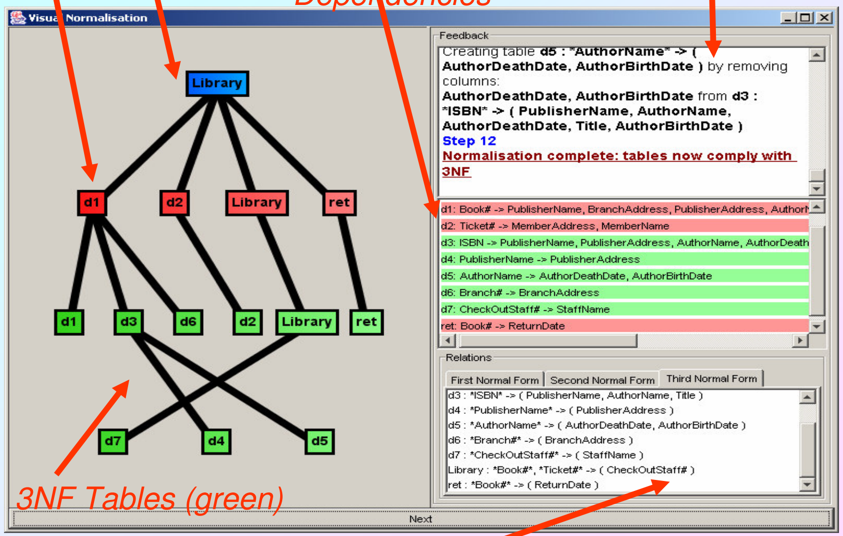
Normalisation

Universal Relation (blue)

2NF Tables (red)

Dependencies

What is happening



Feedback

Creating table **d5 : *AuthorName* -> (AuthorDeathDate, AuthorBirthDate)** by removing columns: **AuthorDeathDate, AuthorBirthDate** from **d3 : *ISBN* -> (PublisherName, AuthorName, AuthorDeathDate, Title, AuthorBirthDate)**

Step 12

Normalisation complete: tables now comply with 3NF

d1: Book# -> PublisherName, BranchAddress, PublisherAddress, AuthorN

d2: Ticket# -> MemberAddress, MemberName

d3: ISBN -> PublisherName, PublisherAddress, AuthorName, AuthorDeath

d4: PublisherName -> PublisherAddress

d5: AuthorName -> AuthorDeathDate, AuthorBirthDate

d6: Branch# -> BranchAddress

d7: CheckOutStaff# -> StaffName

ret: Book# -> ReturnDate

Relations

First Normal Form | Second Normal Form | Third Normal Form

d3 : *ISBN* -> (PublisherName, AuthorName, Title)
d4 : *PublisherName* -> (PublisherAddress)
d5 : *AuthorName* -> (AuthorDeathDate, AuthorBirthDate)
d6 : *Branch#* -> (BranchAddress)
d7 : *CheckOutStaff#* -> (StaffName)
Library : *Book#*, *Ticket#* -> (CheckOutStaff#)
ret : *Book#* -> (ReturnDate)

3NF Tables (green)

Resulting tables

TWDB

This visualises the process of using a dynamic web site

- shows the transfer of data
- allows current state of all HTTP messages and objects to be displayed
- provides pseudo-code and PHP version of dynamic pages

TWDB Interface Components

A browser window

- permits normal browsing

A queryable diagram

- showing the main components
  - selectable to show their current state in the feedback panel
- animated to show sequence of actions

A feedback panel

- showing what the web software is doing
- showing current state of the objects
- text hyperlinked to a glossary

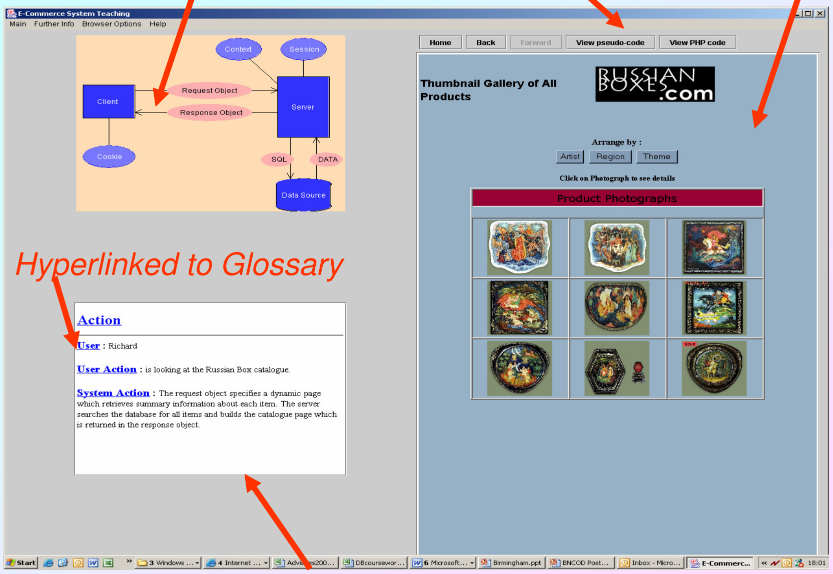
Other TWDB Interface Components

Each dynamic page can be viewed

- as pseudo code
- as PHP code

There are short tutorials on the main techniques

TWDB



Queryable Diagram

View Code

Browser

Hyperlinked to Glossary

Feedback Panel

Action

User : Richard

User Action : is looking at the Russian Box catalogue.

System Action : The request object specifies a dynamic page which retrieves summary information about each item. The server searches the database for all items and builds the catalogue page which is returned in the response object.

Thumbnail Gallery of All Products

Arrange by :

Click on Photograph to see details

XML Programming

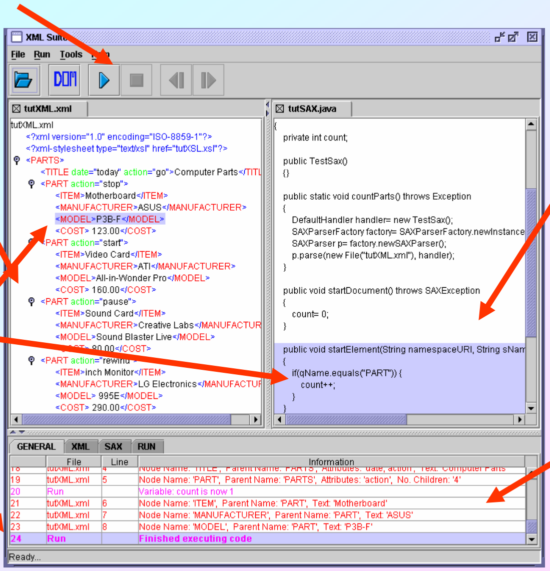
The software supports three XML programming styles

- SAX
- DOM
- XSLT

All are demonstrated in similar ways

- the XML file and the program are shown in separate panes
- as the program runs, the current line is highlighted
- the XML line is also highlighted
- a feedback panel shows what is happening

XML Programming



Progress Control

XML File

Highlight Current Position

SAX Program

Feedback Panel

```
tutXML.xml
  <?xml version="1.0" encoding="ISO-8859-1"?>
  <?xml-stylesheet type="text/xsl" href="tutXSL.xsl"?>
  <PARTS>
     <TITLE date="today" action="go">Computer Parts</TITL
     <PART action="stop">
        <ITEM>Motherboard</ITEM>
        <MANUFACTURER>ASUS</MANUFACTURER>
        <MODEL>P3B-F</MODEL>
        <COST> 123.00</COST>
     <PART action="start">
        <ITEM>Video Card</ITEM>
        <MANUFACTURER>ATI</MANUFACTURER>
        <MODEL>All-in-Wonder Pro</MODEL>
        <COST> 160.00</COST>
     <PART action="pause">
        <ITEM>Sound Card</ITEM>
        <MANUFACTURER>Creative Labs</MANUFACTUR
        <MODEL>Sound Blaster Live</MODEL>
        <COST> 80.00</COST>
     <PART action="rewind">
        <ITEM>inch Monitor</ITEM>
        <MANUFACTURER>LG Electronics</MANUFACTUR
        <MODEL> 995E</MODEL>
        <COST> 290.00</COST>
```

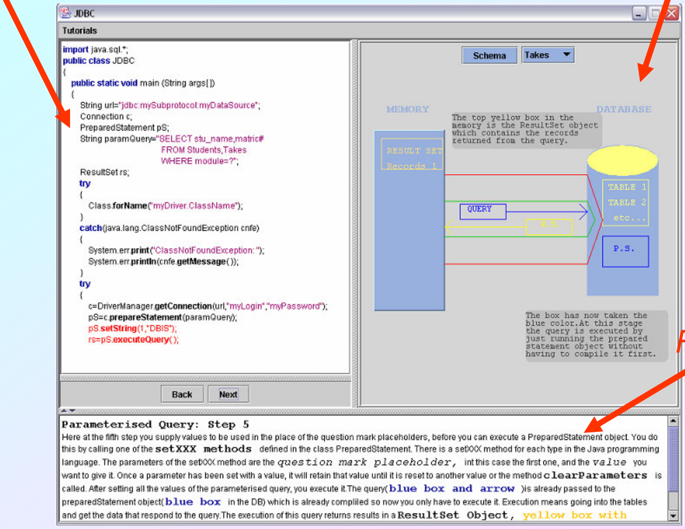
```
{
   private int count;

   public TestSax()
   {}

   public static void countParts() throws Exception
   {
      DefaultHandler handler= new TestSax();
      SAXParserFactory factory= SAXParserFactory.newInstance
      SAXParser p= factory.newSAXParser();
      p.parse(new File("tutXML.xml"), handler);
   }

   public void startDocument() throws SAXException
   {
      count= 0;
   }

   public void startElement(String namespaceURI, String sNam
   {
      if(qName.equals("PART")) {
         count++;
      }
   }
```

	File	Line	Information
18	tutXML.xml	4	Node Name: 'TITLE', Parent Name: 'PARTS', Attributes: 'date, action', Text: 'Computer Parts'
19	tutXML.xml	5	Node Name: 'PART', Parent Name: 'PARTS', Attributes: 'action', No. Children: '4'
20	Run		Variable: count is now 1
21	tutXML.xml	6	Node Name: 'ITEM', Parent Name: 'PART', Text: 'Motherboard'
22	tutXML.xml	7	Node Name: 'MANUFACTURER', Parent Name: 'PART', Text: 'ASUS'
23	tutXML.xml	8	Node Name: 'MODEL', Parent Name: 'PART', Text: 'P3B-F'
24	Run		**Finished executing code**

JDBC Programming

- The way a JDBC program is shown by:
  - building up the program is segments
  - producing a diagram of the client-server interaction
  - giving feedback on what is happening

JDBC Programming



```
import java.sql.*;
public class JDBC
{
  public static void main (String args[])
  {
    String url="jdbc:mySubprotocol:myDataSource";
    Connection c;
    PreparedStatement pS;
    String paramQuery="SELECT stu_name,matric#
                       FROM Students,Takes
                       WHERE module=?";
    ResultSet rs;
    try
    {
      Class.forName("myDriver.ClassName");
    }
    catch(java.lang.ClassNotFoundException cnfe)
    {
      System.err.print("ClassNotFoundException: ");
      System.err.println(cnfe.getMessage());
    }
    try
    {
      c=DriverManager.getConnection(url,"myLogin","myPassword");
      pS=c.prepareStatement(paramQuery);
      pS.setString(1,"DBIS");
      rs=pS.executeQuery();
```

The top yellow box in the memory is the ResultSet object which contains the records returned from the query.

The box has now taken the blue color.At this stage the query is executed by just running the prepared statement object without having to compile it first.

Parameterised Query: Step 5

Here at the fifth step you supply values to be used in the place of the question mark placeholders, before you can execute a PreparedStatement object. You do this by calling one of the `setXXX methods` defined in the class PreparedStatement. There is a setXXX method for each type in the Java programming language. The parameters of the setXXX method are the *question mark placeholder,* int this case the first one, and the *value* you want to give it. Once a parameter has been set with a value, it will retain that value until it is reset to another value or the method `clearParameters` is called. After setting all the values of the parameterised query, you execute it.The query(`blue box and arrow`)is already passed to the preparedStatement object(`blue box` in the DB) which is already compliled so now you only have to execute it. Execution means going into the tables and get the data that respond to the query.The execution of this query returns results in a `ResultSet Object,` `yellow box with`

Growing Program

Queryable Diagram

Feedback Panel

Creating a Generic Toolset

These applications all showing a number of similar processes and are showing

- equivalence between designs
- mapping processes
- software as it runs
- the state of system components

The goal is to extract those features into a reusable toolkit

- so that fresh applications can be assembled from the toolkit

Common Interface Elements

- Interactive Text Panels
- Interactive Diagrams
- Staged Execution
- Correspondences
- Feedback Panel
- Hyperlinked Glossaries

A Structure for Java Interfaces I

Fragment

- a component of a document which has a distinctive existence
- can be highlighted, added, removed, etc.
- iconic and textual fragments

Document

- a structured set of fragments

Parser

- generates a document according to a syntax
  - static – i.e. generates document from complete file
  - dynamic – i.e. responds to user edits

A Structure for Java Interfaces II

Correspondence

– connection between two fragments denoting equivalence

Action

– description of a change to a document or the framework

- can be used in a menu/toolbar or in a staged execution

Staged Execution

– series of actions controlled by a progress bar

A Structure for Java Interfaces III

Style

– to manage the look of documents

Panels and Frames

– a location in which a document appears

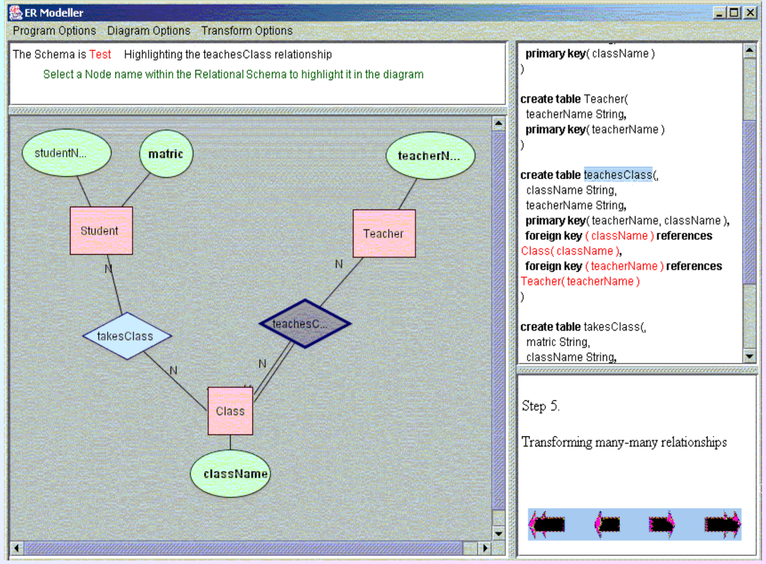
Framework

– a collection of panels, frames, menus and toolbars

Methodology

1. Set up a framework
2. Map panels to document type
3. Develop and install parsers
4. Identify correspondences
5. Create a set of actions and tie them to menus, toolbars, fragment interaction
6. Set up staged executions

Example



The Schema is Test Highlighting the teachesClass relationship

Select a Node name within the Relational Schema to highlight it in the diagram

```
 primary key( className )
)

create table Teacher(
 teacherName String,
 primary key( teacherName )
)

create table teachesClass(,
 className String,
 teacherName String,
 primary key( teacherName, className ),
 foreign key ( className ) references
Class( className ),
 foreign key ( teacherName ) references
Teacher( teacherName )
)

create table takesClass(,
 matric String,
 className String,
```

Step 5.

Transforming many-many relationships

Steps

1. Set up the framework of four panels
2. Create document classes for ER and SQL
3. Set up an interactive ER diagram parser
4. Set up correspondence control between the SQL and ER fragments
5. Set up menus
6. One of these calls the staged execution of the mapping process

Summary

Teaching complex systems is better done if the student can interact with the components

Software for a variety of database and internet systems has been achieved

This software has common features which can be extracted for re-use