# An Architecture for Archiving and Post-Processing Large, Distributed, Scientific Data Using SQL/MED and XML

Mark Papiani, Jasmin Wason, and Denis Nicole

[papiani@computer.org, jlw98r@ecs.soton.ac.uk, dan@ecs.soton.ac.uk]

High Performance Computing Group

Dept Electronics and Computer Science

University of Southampton

# Overview

This talk describes - An architecture and an implementation for an ***active*** web-based scientific data archive known as EASIA (Extensible Architecture for Scientific Information Archives).

- **Motivation - Large datasets from UK Turbulence Consortium**
- **Approach to problem of limited Bandwidth - distributed, active archive**
- **System Architecture - use of XML defined user interface / DATALINKs**
- **User Interface - Searching and Browsing**
- **SQL/MED**
- **Operations - Loosely coupled, distributed, standard server-side post-processing codes - incorporated via simple XML defined interfaces**
- **Java Code upload for secure server-side execution**

# Motivation (i)

**We have been working with the UK Turbulence Consortium to provide an architecture for archiving and manipulating the results of numerical simulations.**

**•larger grid sizes - United Kingdom's new national scientific supercomputing resource.**

**• One complete simulation, comprising perhaps one hundred timesteps, requires a total storage capacity of some hundreds of gigabytes.**

**• Improve collaboration between groups working on turbulence by providing a mechanism for dissemination of data to members of the turbulence modelling community.**

**⇒ Necessitates new Web-based mechanisms for storage, searching and retrieval of multi-gigabyte datasets that are generated for each timestep in a simulation.**

**⇒ In particular, an architecture is required that can minimise bandwidth usage whilst performing these tasks.**

# Motivation(ii): Active scientific data archives

• ***Caltech Workshop on Interfaces to Scientific Data Archives*** **- identified an urgent need for infrastructures that could manage and federate active libraries of scientific data.**

**Williams, R., Bunn, J., Reagan, M., and Pool, C., T. Workshop on Interfaces to Scientific Data Achives, California, USA, 25-27 March, 1998, Technical Report CACR-160, CALTECH, 42pp. http://www.cacr.caltech.edu/isda**

• ***Hawick and Coddington*** **- "An active data archive can be defined as one where much of the data is generated on-demand, as value-added data products or services, derived from existing data holdings". They also state that the information explosion has led to *a very real and practical need for systems to manage and interface to scientific archives*.**

**Hawick, K., A. and Coddington, P., D. Interfacing to Distributed Active Data Archives, Journal on Future Generation Computer Systems, to appear.**

# Approach (i)

**Starting point was to look at see if we could used a system we developed in 1996: DBbrowse - an automated web-based interface to an object-relational database management system (implemented using HTML/CGI/PERL).**

- **This generates a schema driven web interface, akin to QBE (Query-by-example) that incorporates searching and browsing of a database.**

- **Browsing is based on hypertext links in search results, that link to related data in related tables.**

- **Relationships are inferred by referential integrity constraints in the DB catalogue metadata.**

- **BLOB and CLOB types also contain hypertext links that rematerialise the underlying objects and return them to the user's browser with the appropriate MIME type set.**

High Performance Computing

# Approach (ii) - Experimental ftp bandwidth measurements

| Time | Direction of Transfer | Bandwidth (Mbit/s) | Estimated time to transfer small simulation data file | Estimated time to transfer large simulation data file |
|---|---|---|---|---|
| Day | To Southampton | 0.25 | 45m20s | 4h50m08s |
| Day | From Southampton | 0.37 | 30m38s | 3h16m02s |
| Evening | To Southampton | 0.58 | 19m32s | 2h05m03s |
| Evening | From Southampton | 1.94 | 5m51s | 37m23s |

**However - DBbrowse would need to be re-architected to cope with large datasets**

- **Experimental results demonstrated that using the Web to transfer datasets to a central archive and retrieve them is not feasible.**

- **Two file sizes - 85 MByte for a small simulation and 544 MByte large simulation. (Two current simulation resolutions being used by the UK Turbulence consortium.)**

- **University of Southampton currently - 10 Mbit/s connection to SuperJANET.**
- **Repeated measurements of transfer to/from Queen Mary & Westfield College, London (also 10 Mbit/s).**

- **Not surprisingly, evening is the best time to transfer files. Less predictable is the fact that supplying result files from Southampton will achieve significantly better performance than trying to send results to Southampton.**

University of Southampton

High Performance Computing

# Bandwidth Problems

**First problem!**

**User's Browser**  **Scientific Data Archive**

**Upload large dataset**

**Second problem!**

**User's Browser**  **Scientific Data Archive**

**Download large dataset**

⟹ **Our Solution:**

**1) Archive data where it is generated**

**2) Post-process - data reduction - 'operations' and code upload**

University of Southampton


# New approach (i)

• **Rewrite DBbrowse system using HTML/JavaScript on the client and Java Servlets/JDBC/Object-relational DBMS/XML on the server side.**

**Change the architecture to deal with large datasets in a low bandwidth environment:**

⟹ **Distribute the data and add user defined post-processing (loosely couple to the data via XML interfaces).**

**We demonstrate that the new DATALINK type, defined in the draft SQL:1999 (formerly SQL3) SQL Management of External Data Standard (SQL/MED), which facilitates database management of distributed external data, can help to overcome problems associated with limited bandwidth.**

**We show that a database can meet the apparently divergent requirements of storing both the relatively small simulation result metadata, and the large result files, in a unified way, whilst maintaining database security, recovery and integrity.**


# New approach (ii)

- **By managing data in this distributed way, the system allows post-processing of archived simulation results to be performed directly without the cost of having to rematerialise to files.**

- **This distribution also reduces access bottlenecks and processor loading.**

- **We also separate the user interface specification from the user interface processing**

  **We provide a tool to generate automatically a default user interface specification, in the form of an XML document, for a given database. The *XML user interface specification (XUIS)* file.**

- **Our architecture can archive not only data in a distributed fashion, but also applications. Applications are loosely coupled to the datasets (in a many-to-many relationship) via XML defined interfaces. They provide reusable server-side post-processing operations such as data reduction and visualisation.**

# Benefits of the EASIA architecture

• *The system can be accessed by users of the scientific archive, who may have little or no database or Web development expertise*. **Users are presented with a dynamically generated HTML query form that provides a search interface akin to Query by Example (QBE).**

• *The default interface specification adds a novel data browsing facility to maintain a Web-based feel.*

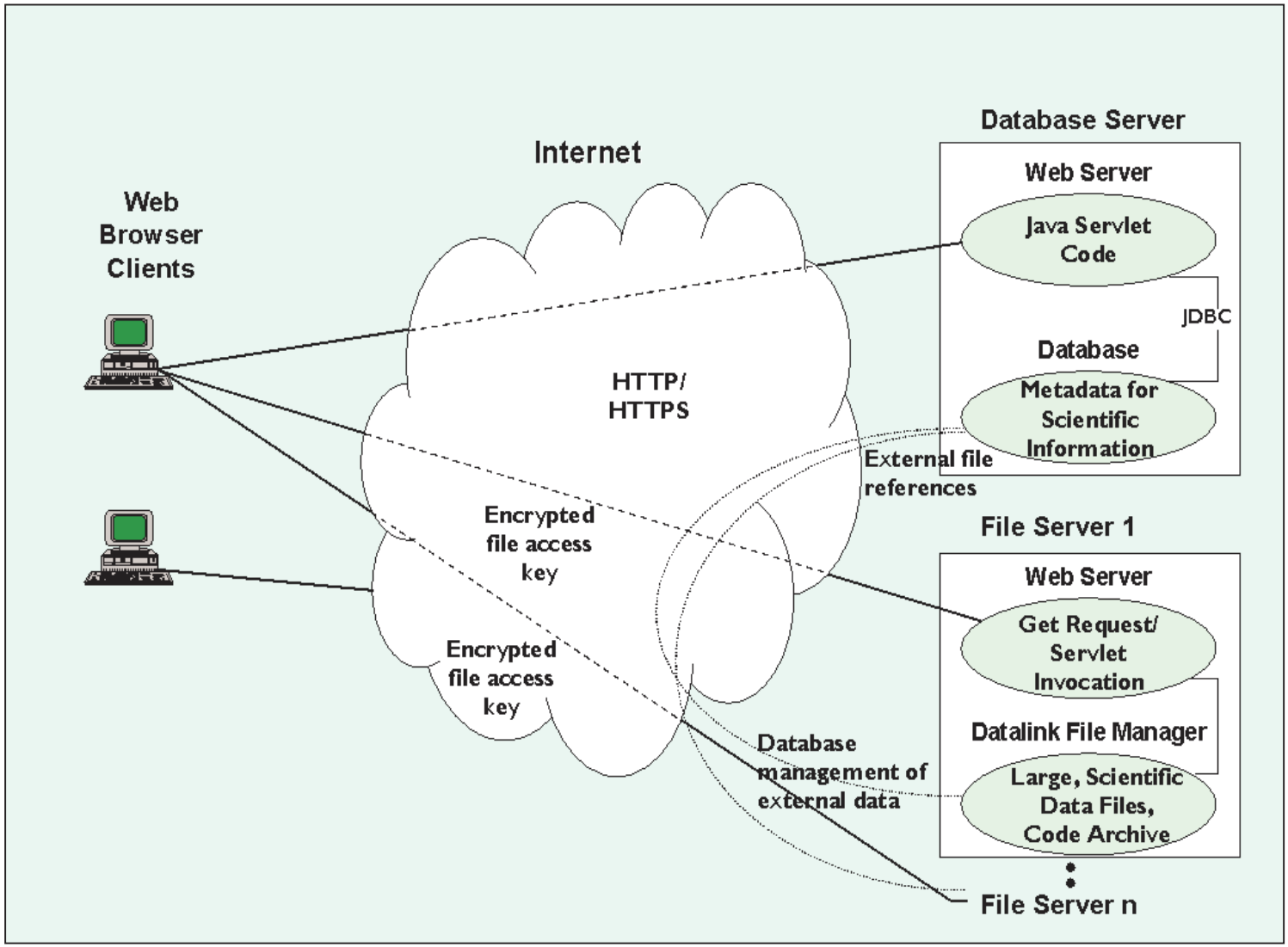
• *Large result files can be archived at (or close to) the point where they are generated.*

• **Because simulation results are stored in unmodified files,** *existing post-processing applications (e.g. FORTRAN codes), that use standard file I/O techniques, can be applied to the files without having to rewrite the applications.*

• *We are using our architecture to build a large scientific archive from commodity components, with many distributed machines acting as file servers for a single database.* **Security, backup and integrity of the file servers can be managed using SQL/MED. This arrangement can provide high performance in the following areas:**

- **Data can be distributed so that it is physically located closest to intensive usage.**
- **Data distribution can reduce access bottlenecks at individual sites.**
- **Each machine provides a distributed processing capability that allows multiple datasets to be post-processed simultaneously. Suitable user-directed post-processing, such as array slicing and visualisation, can significantly reduce the amount of data that needs to be shipped back to the user.**

# System architecture



# Architecture (ii)

**Database server host (located at Southampton University)**

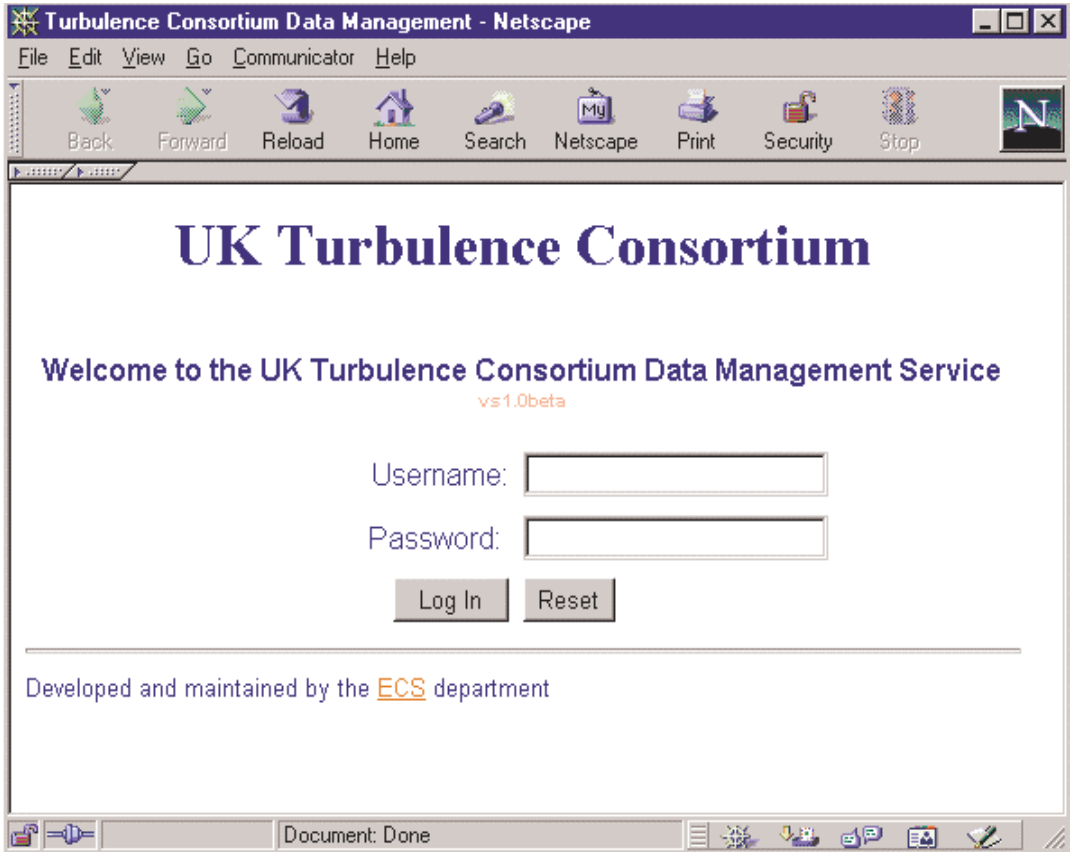
**Stores metadata describing the scientific information such as, simulation titles, descriptions and authors. This data is stored locally in the database and is accessed by our servlet code using Java Database Connectivity (JDBC)**

**File server hosts that may be located anywhere on the Internet**

**Store files referenced by attributes defined as DATALINK SQL-types. These file servers manage the large files associated with simulations, which have been archived where they were generated.**

**When the result of a database access yields a DATALINK value, our interface presents this to the user as a hypertext link that can be used to download the referenced file. The URL contains an encrypted key that is prefixed to the required file name. This data can also be post-processed.**

# Demo: http://www.hpcc.ecs.soton.ac.uk/~turbulence




username: *guest*
password: *guest*

Guest users:

- cannot download datasets
- cannot upload post-processing codes
- are limited in the types of operations they can run

# Searching and browsing the archive

**Users can begin to locate information in the scientific archive by searching or browsing data or by using a combination of both techniques.**

## Searching

**- Select a link to a query form for a particular table**

**On the query form, the user selects the fields to be returned. Also for each field present, restrictions including wildcards may be put on the values of the data.**

**Other features to aid direct searching - restrictions and sample values from drop-down lists - choices of attribute names, relation names and operators**

**- Alternatively request all data for a table**

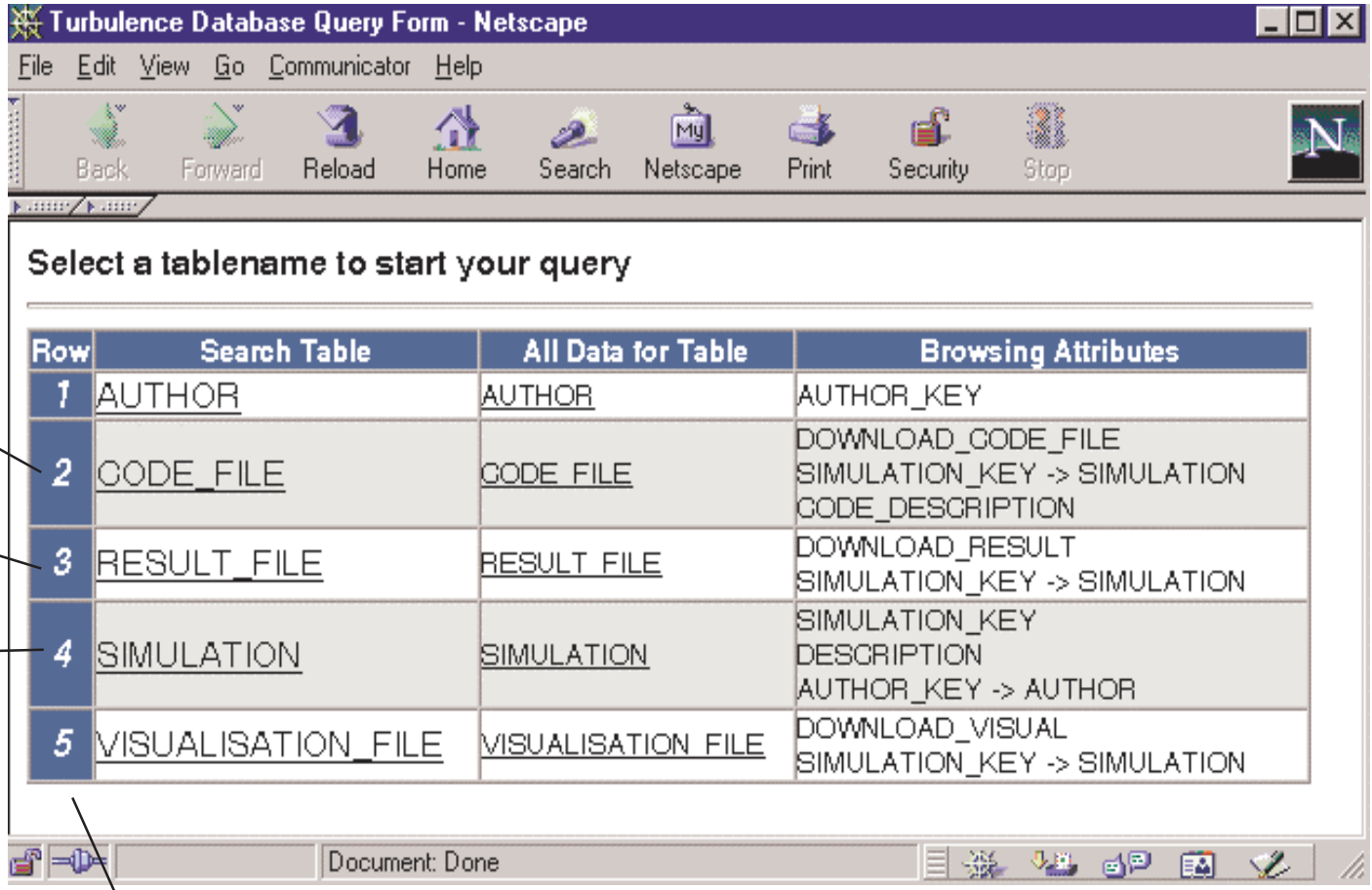
High Performance Computing

# Searching and browsing the archive

simulation codes and post-processing codes

datasets

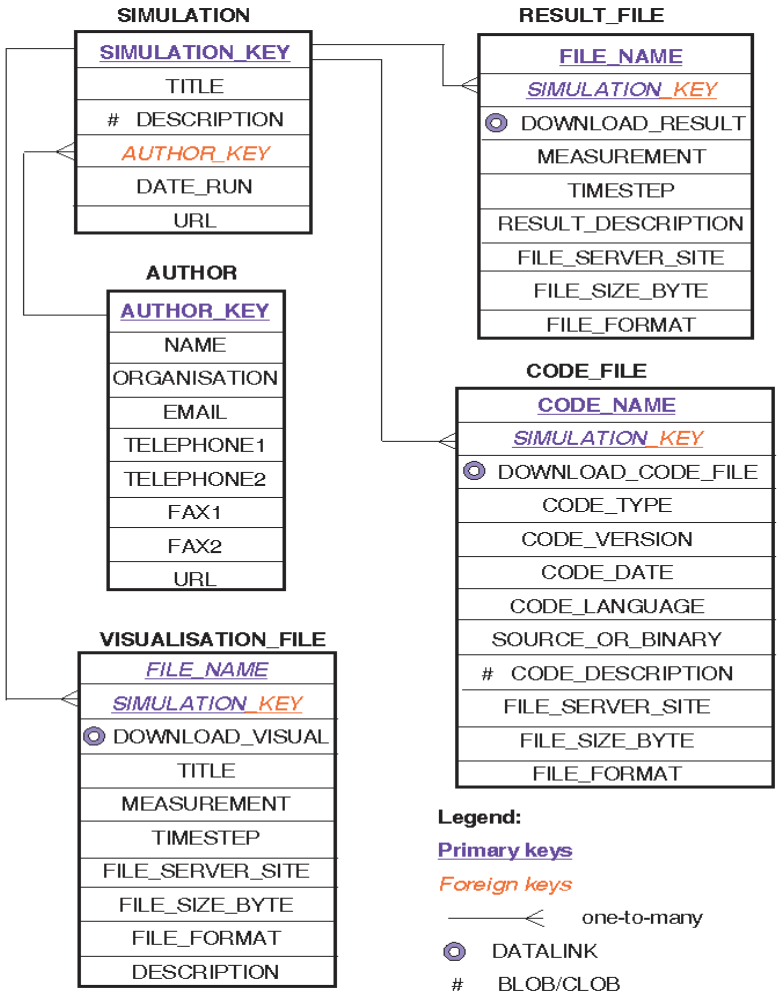
metadata identifying simulation




5 tables used for the Turbulence database

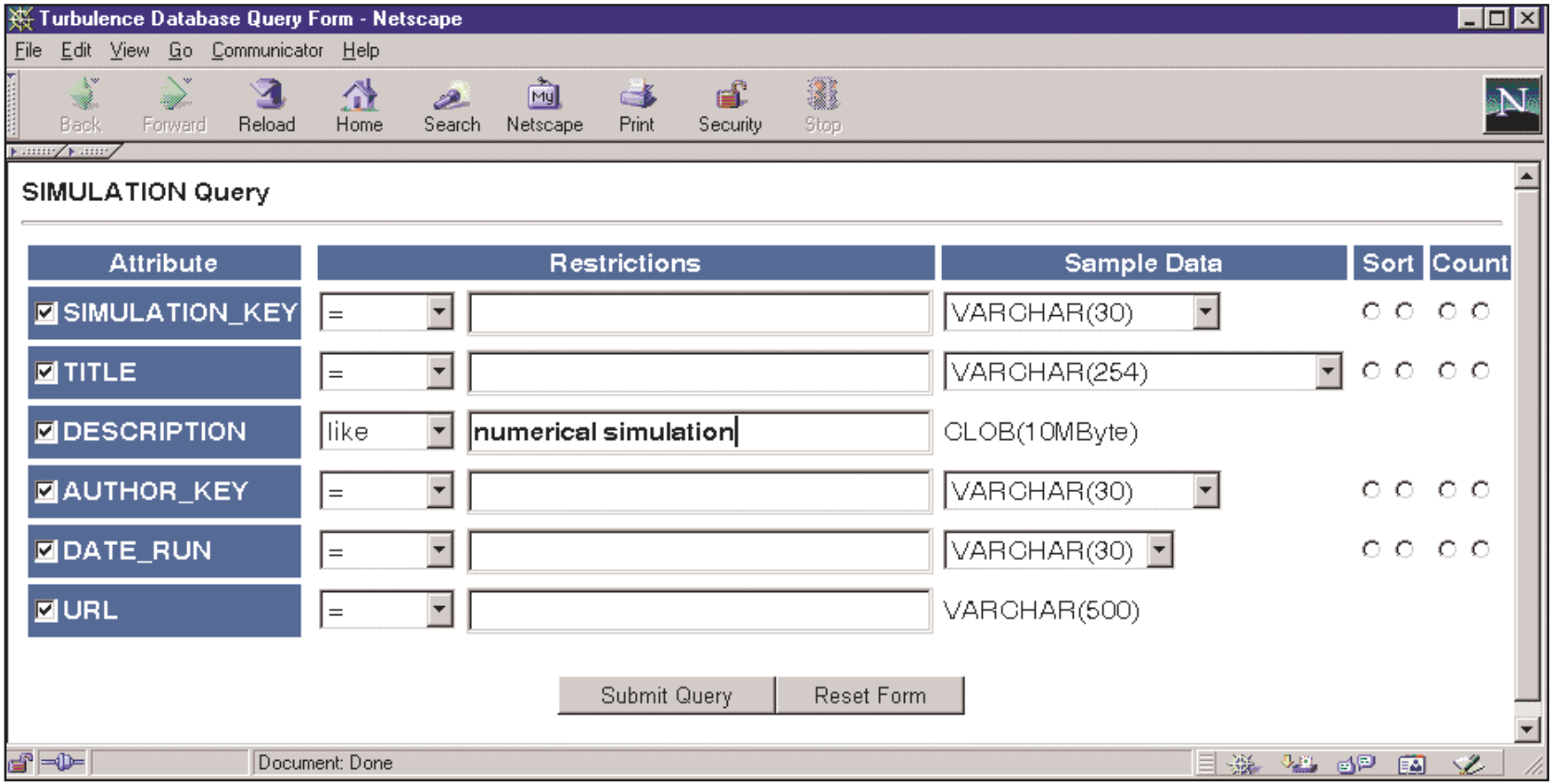
University of Southampton

# Sample database schema



High Performance Computing

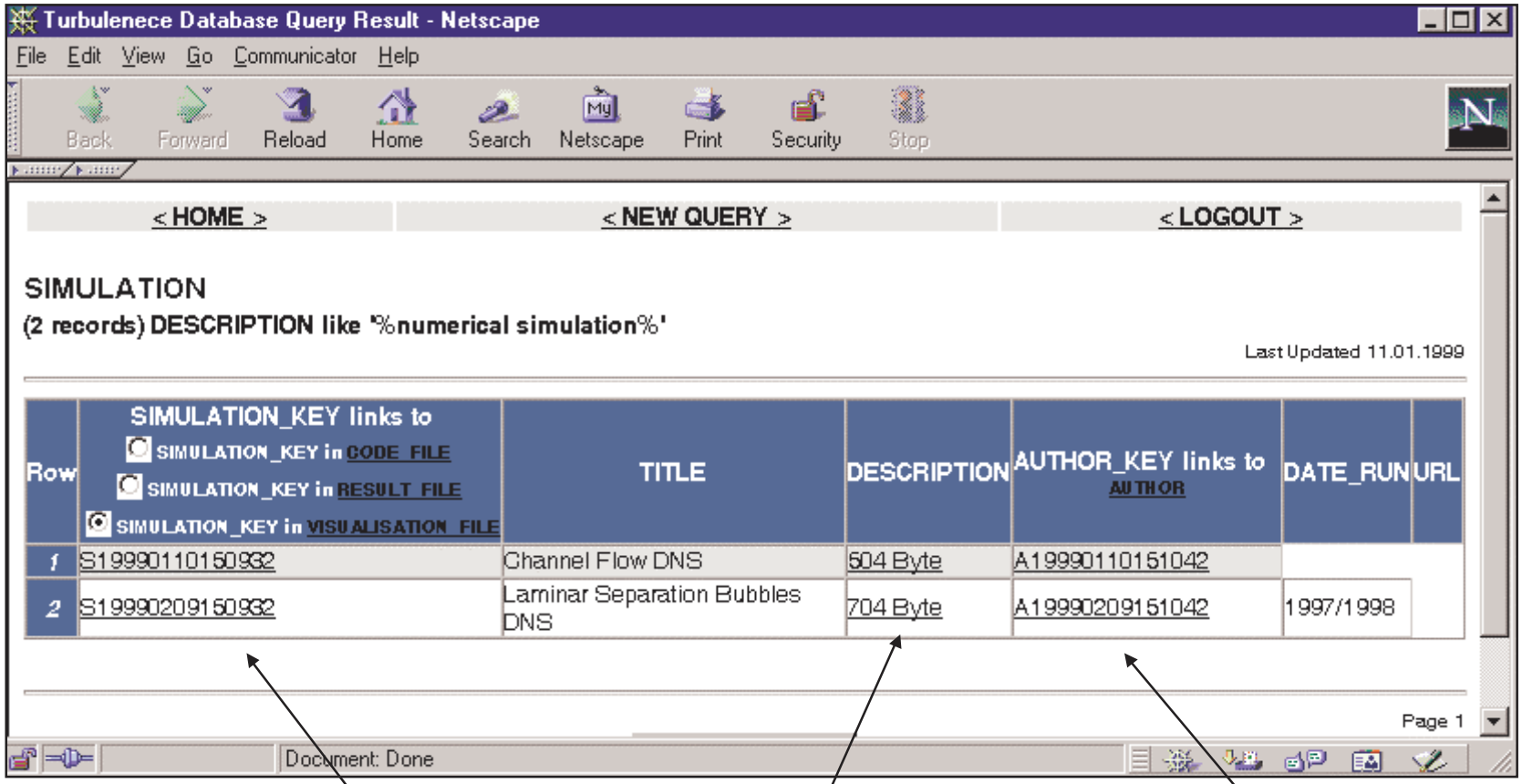
# Searching the archive




University of Southampton

High Performance Computing

# Result table from querying SIMULATION table




**Primary key browsing**

**CLOB browsing**

**Foreign key browsing**

University of Southampton

# Browsing

- **Foreign Key Browsing**

**(Selecting a link on an AUTHOR_KEY value will retrieve full details of the author)**

- **Primary Key Browsing**

**(SIMULATION_KEY links to three tables where it appears as a foreign key; the RESULT_FILE table, CODE_FILE table and VISUALIATION_FILE table. Selecting one of these values will return all the rows that the key appears in from one of the referenced tables)**

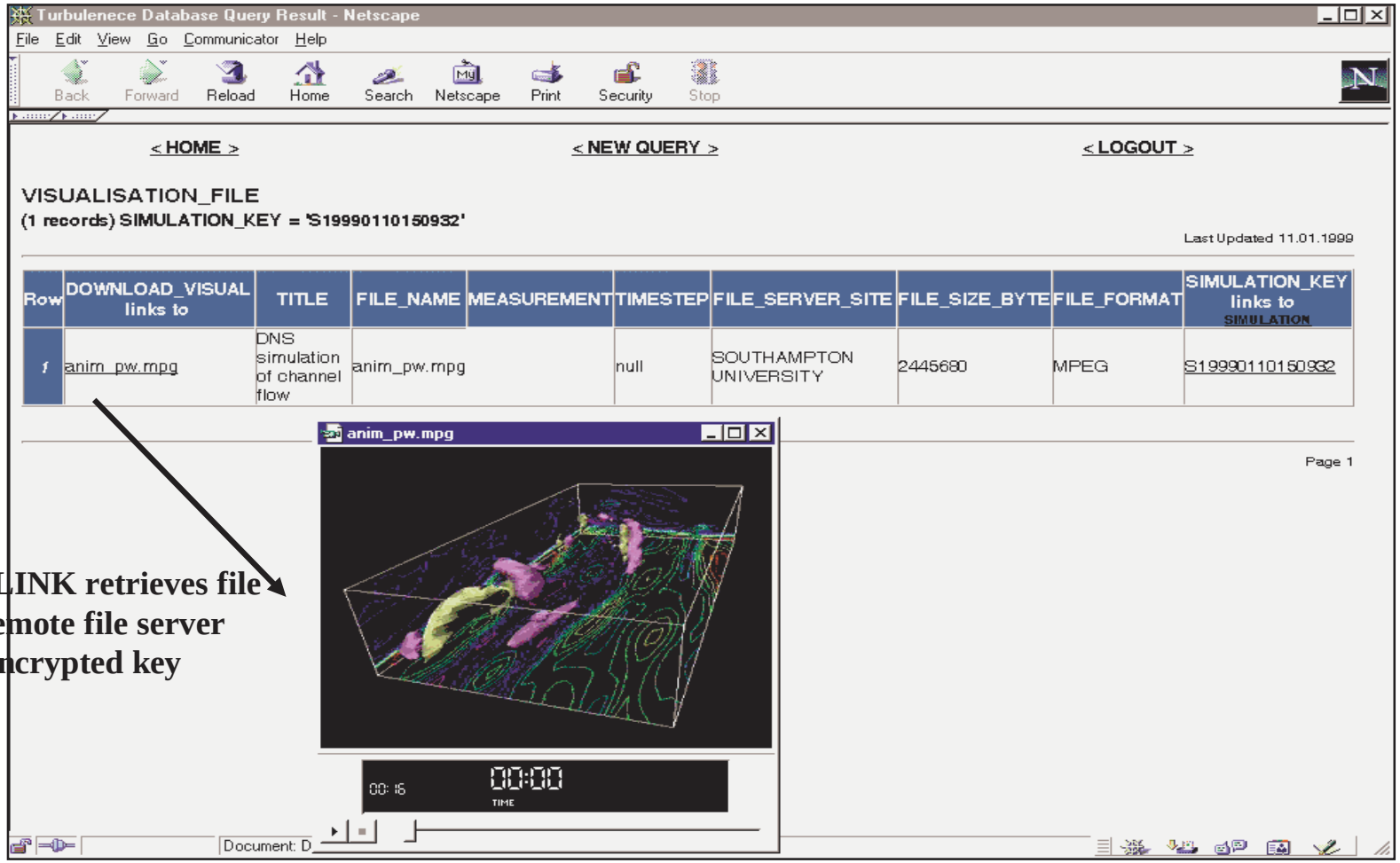
- **BLOB and CLOB**

**(Store small files that can be uploaded over the Internet. Hypertext link displays size of object - rematerialised and returned to the client)**

- **DATALINK Browsing**

**(Hypertext link displays size of object - contains an encrypted key, required to access the file from the remote file server)**

# DATALINK browsing




**DATALINK retrieves file from remote file server using encrypted key**

# SQL/MED (Management of External Data)

**ANSI and ISO have accepted the proposal for SQL Part 9: Management of External Data* this includes the specification of the DATALINK type.**

**ISO progressed SQL/MED to Committee Draft (CD) in December 1998 and it should become a standard in late 2000.**

**DATALINKs provide the following features for database management of external files:**

**• Referential Integrity (an external file referenced by the database cannot be renamed or deleted)**

**• Transaction Consistency (changes affecting both the database and external files are executed within a transaction. This ensures consistency between a file and its metadata)**

**• Security (file access controls can be based on the database privileges)**

**• Coordinated Backup and Recovery (the database management system can take responsibility for backup and recovery of external files in synchronisation with the internal data)**

*Mattos, N., Melton, J. and Richey, J. Database Language SQL-Part 9: Management of External Data (SQL/MED), ISO/IEC Committee Draft, CD 9075-9 (ISO/IEC JTC 1/SC 32 N00197), December, 1988. ftp://jerry.ece.umassd.edu/isowg3/dbl/YGJdocs/ygj023.pdf

# DATALINK Type (i)

```
CREATE TABLE RESULT_FILE (
  download_result DATALINK
    LINKTYPE URL
    FILE LINK CONTROL
    READ PERMISSION DB
    …
```

• **FILE LINK CONTROL -- a check should be made to ensure the existence of the file during a database insert or update.**

• **READ PERMISSION DB -- files can only be accessed using an encrypted file access token, obtained from the database by users with the correct database privileges**

• **Several other parameters and options are supported for the DATALINK type.**

# DATALINK Type (ii)

**A DATALINK value can be entered via a standard SQL INSERT or UPDATE statement.**

**The value takes the form:**
**http://host/filesystem/directory/filename**

**An SQL SELECT statement retrieves the value in the form:**
**http://host/filesystem/directory/access_token;filename**

**The file can then be accessed from the filesystem in the normal way using the name:**

**access_token;filename**

**or, by using the full URL if the file is placed on a Web server (as in EASIA). The access tokens have a finite life determined by a database configuration parameter. This can be set to expire after an interval.**


# XUIS - XML User Interface Specification File

**• System is started by initialising the Java servlet code (on DB server host) with an XUIS.**

**• Default XUIS can be created prior to system initialisation using a tool that we provide.**

**•Written in Java, uses JDBC to extract data and schema information from the database being used to archive simulation results.**

**• Default XUIS conforms to a DTD that we have created. The default XUIS can be customised prior to system initialisation.**

**• The XUIS contains table names, column names, column types, sample data values for each column, and details of primary keys and foreign keys.**

**• The XUIS also allows aliases to be defined for table and column names.**

**• XUIS defines interfaces for post-processing operations**

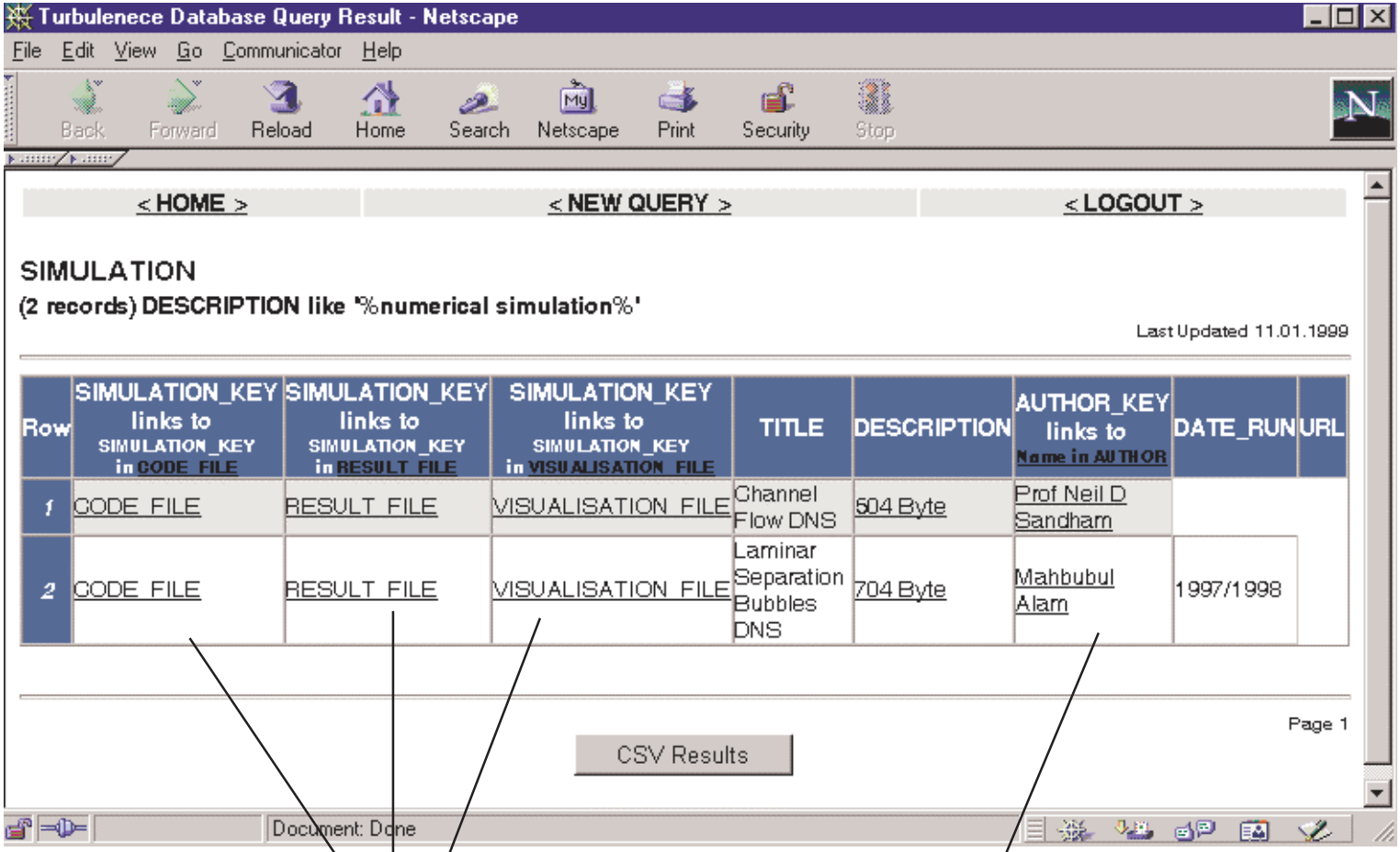
**•XUIS specifies if DATALINK column supports user upload of codes for execution against the stored data files**

# XUIS fragment

```
<table name="AUTHOR" primaryKey="AUTHOR.AUTHOR_KEY">
      <tablealias>Author</tablealias>
      <column name="AUTHOR_KEY" colid="AUTHOR.AUTHOR_KEY">
            <type><VARCHAR/><size>30</size></type>
            <pk><refby tablecolumn="SIMULATION.AUTHOR_KEY" /></pk>
            <samples>
                  <sample>A19990110151042</sample>
                  <sample>A19990209151042</sample>
            </samples>
      </column>
```

High Performance Computing

# Customisation in EASIA - through XUIS modification




Single primary key column replaced by multiple columns that identify linked tables (pk to fk relationship )

Foreign key (AUTHOR_KEY) replaced with data from a specified column (Name) in the

University of Southampton

# XUIS Customisation

```
<table name = "SIMULATION" primaryKey = "SIMULATION.SIMULATION_KEY">
  <tablealias>define alias for table name here</tablealias>
  <column name = "AUTHOR_KEY" colid = "SIMULATION.AUTHOR_KEY">

    <!--Foreign key link defined here, with possible
     substitute columns->
    <fk tablecolumn = "AUTHOR.AUTHOR_KEY"
     substcolumn = "AUTHOR.NAME"/>

    <samples>
      <sample>user defined sample 1</sample>
      <sample>user defined sample value 2</sample>
    </samples>
```

# Operations

**EASIA architecture allows the XUIS to be modified to allow post-processing applications that have been archived using DATALINK values to be dynamically executed server-side to reduce the data volume returned to the user.**

**These applications can consist of Java classes or any other executable format, suitable for the file server host on which the data resides, including C, FORTRAN and scripting languages.**

**These applications do not have to be specially written for our architecture (in fact, operations stored as DATALINKs can be downloaded separately for standalone execution elsewhere) and they can be packaged in a number of different formats including various compressed archive formats (such as tar.Z, gz, zip, tar etc.).**

**The only restriction is that the initial executable file accepts a filename as a command line parameter.**

**This filename will correspond to the name of a dataset to be processed.**

**Archived applications are associated with a number of archived datasets using a mark-up syntax that we have defined for 'operations' in the XUIS.**

**If the application allows other user-specified parameters, the syntax for operations has been defined so that an HTML form will be created to request these parameters at invocation time.**

# XUIS fragment for an operation (i)

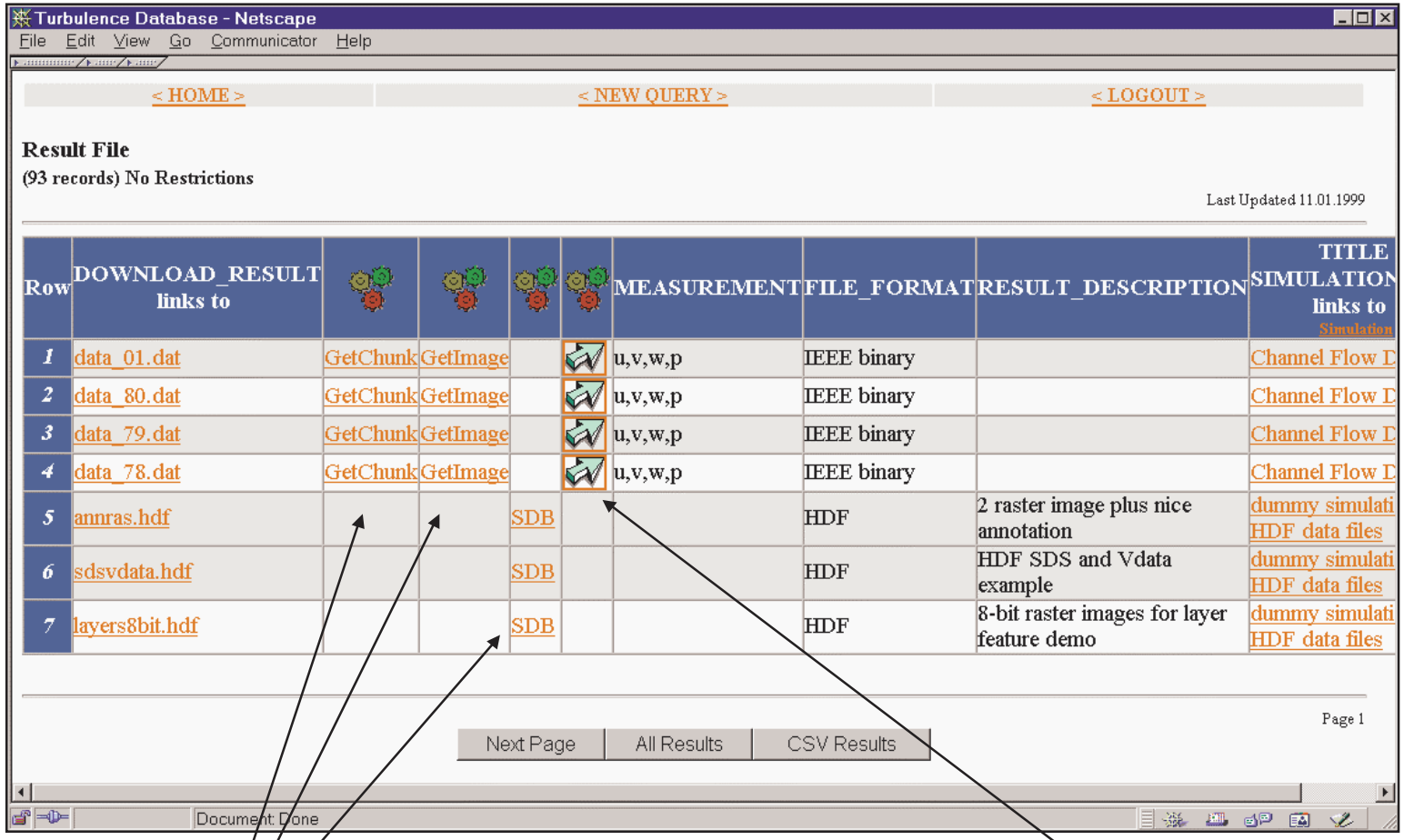
```
<column name="DOWNLOAD_RESULT"
        colid="RESULT_FILE.DOWNLOAD_RESULT">
  <type><DATALINK /></type>
  <operation    name="GetImage" type="JAVA" filename="GetImage.class"
                format="jar" guest.access="true" column="false">
        <if>
                <condition colid="RESULT_FILE.SIMULATION_KEY">
                        <eq>'S19990110150932'</eq>
                </condition>
        </if>
        <location>
                <database.result colid="CODE_FILE.DOWNLOAD_CODE_FILE">
                        <condition colid="CODE_FILE.CODE_NAME">
                                <eq>'GetImage.jar'</eq>
                        </condition>
                </database.result>
        </location>
        <parameters> ...
```

# XUIS fragment for an operation (ii)

```
<parameters>
   <param><variable>
        <description>Select the slice you wish to visualise:</description>
        <select name="slice" size="4">
                <option value="x0">x0=0.0</option>
                <option value="x1">x1=0.1015625</option>
                <option value="x2">x2=0.203125</option>
                ...
        </select>
   </variable></param>
   <param><variable>
        <description>Select velocity component or pressure:</description>
        <input type="radio" name="type" value="u">u speed</input>
        <input type="radio" name="type" value="v">v speed</input>
        <input type="radio" name="type" value="w">w speed</input>
        <input type="radio" name="type" value="p">pressure</input>
   </variable></param>
</parameters>
```

High Performance Computing

# Result table showing operations available for post-processing datasets

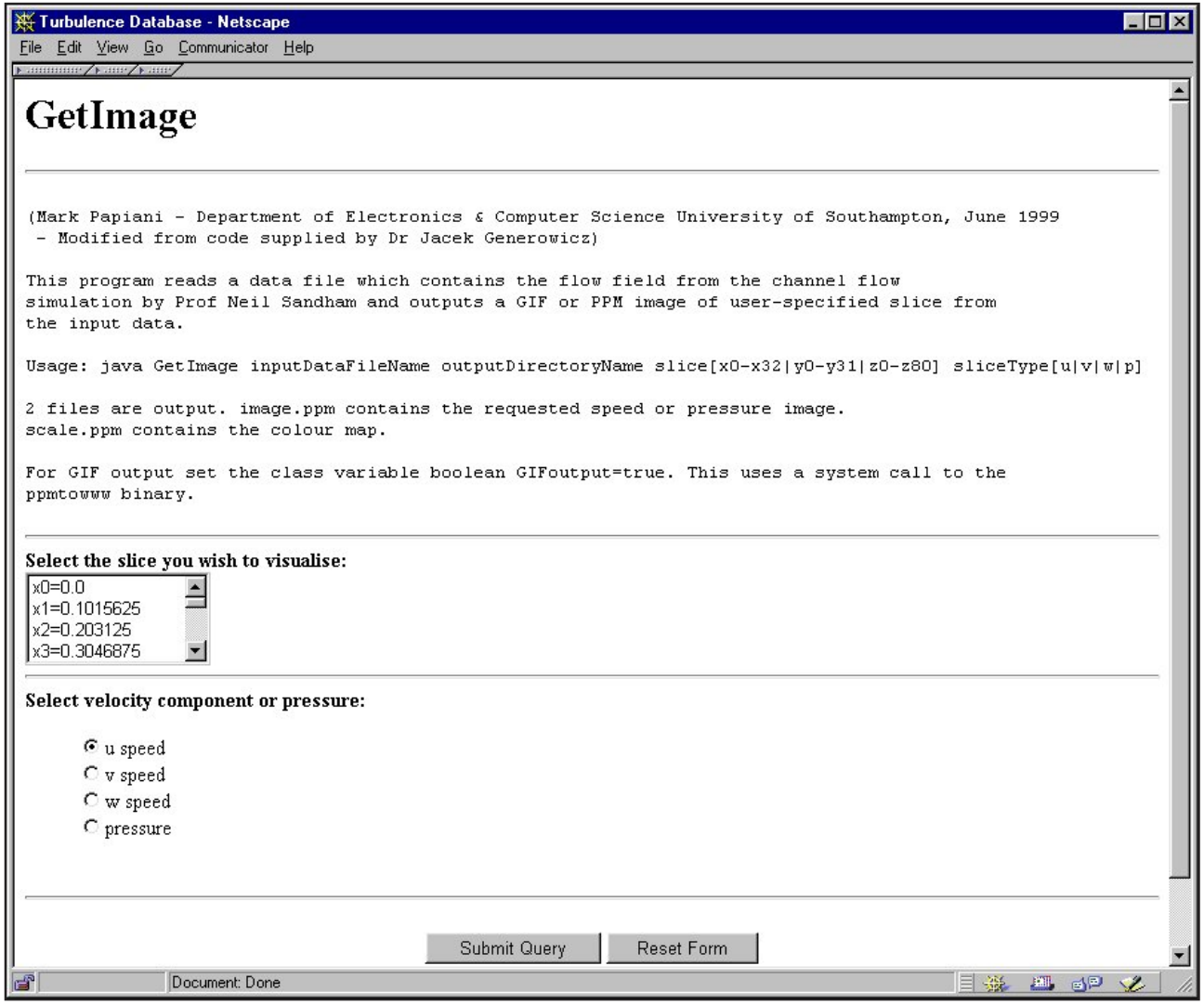



**Operations available on datasets (data and operations archived using DATALINKS)**

**Code upload available for server-side processing**

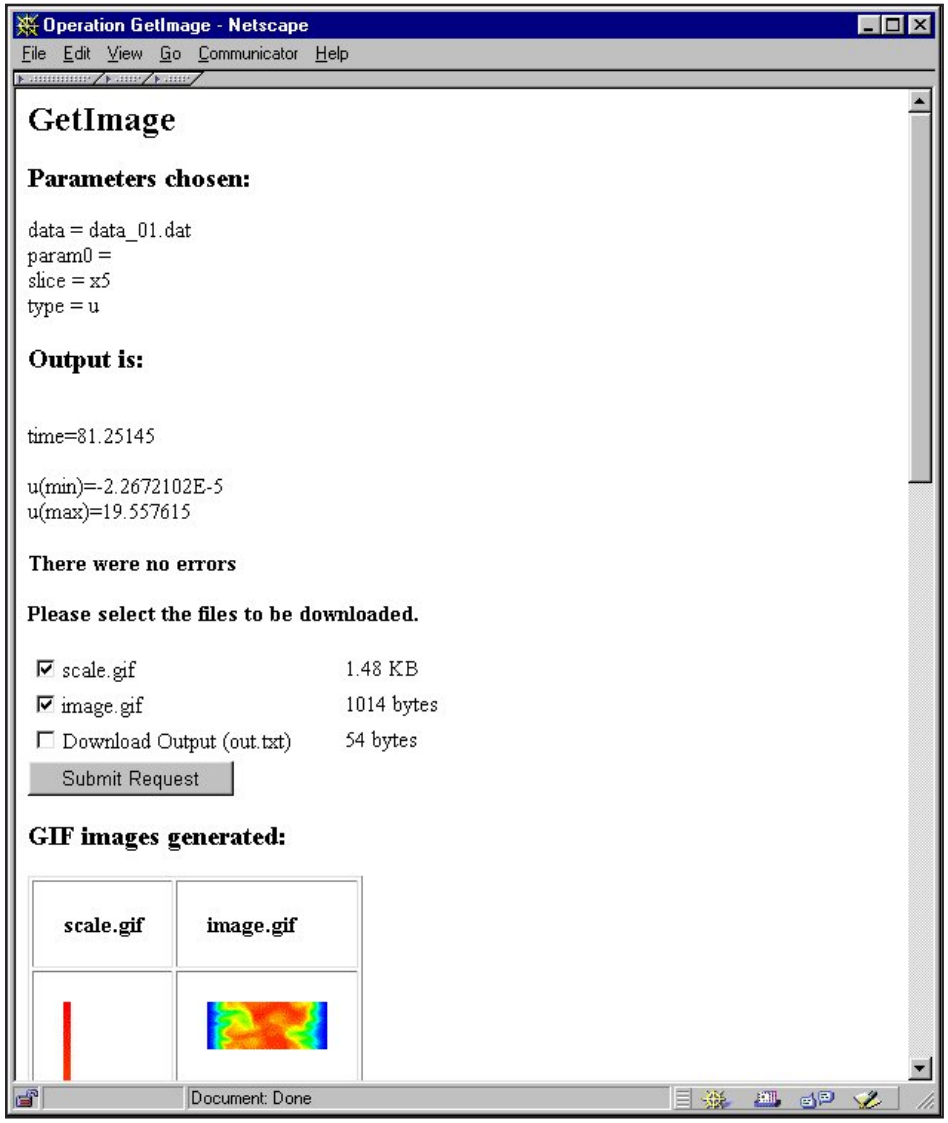
University of Southampton

# Input form for operation (generated according to XUIS)



High Performance Computing

# Output from operation execution




University of Southampton

# URL Operations

- **The XUIS can also specify operations as URLs**

- **These correspond to Servlet or CGI based post-processing services running on the same host as a particular DATALINK file server**

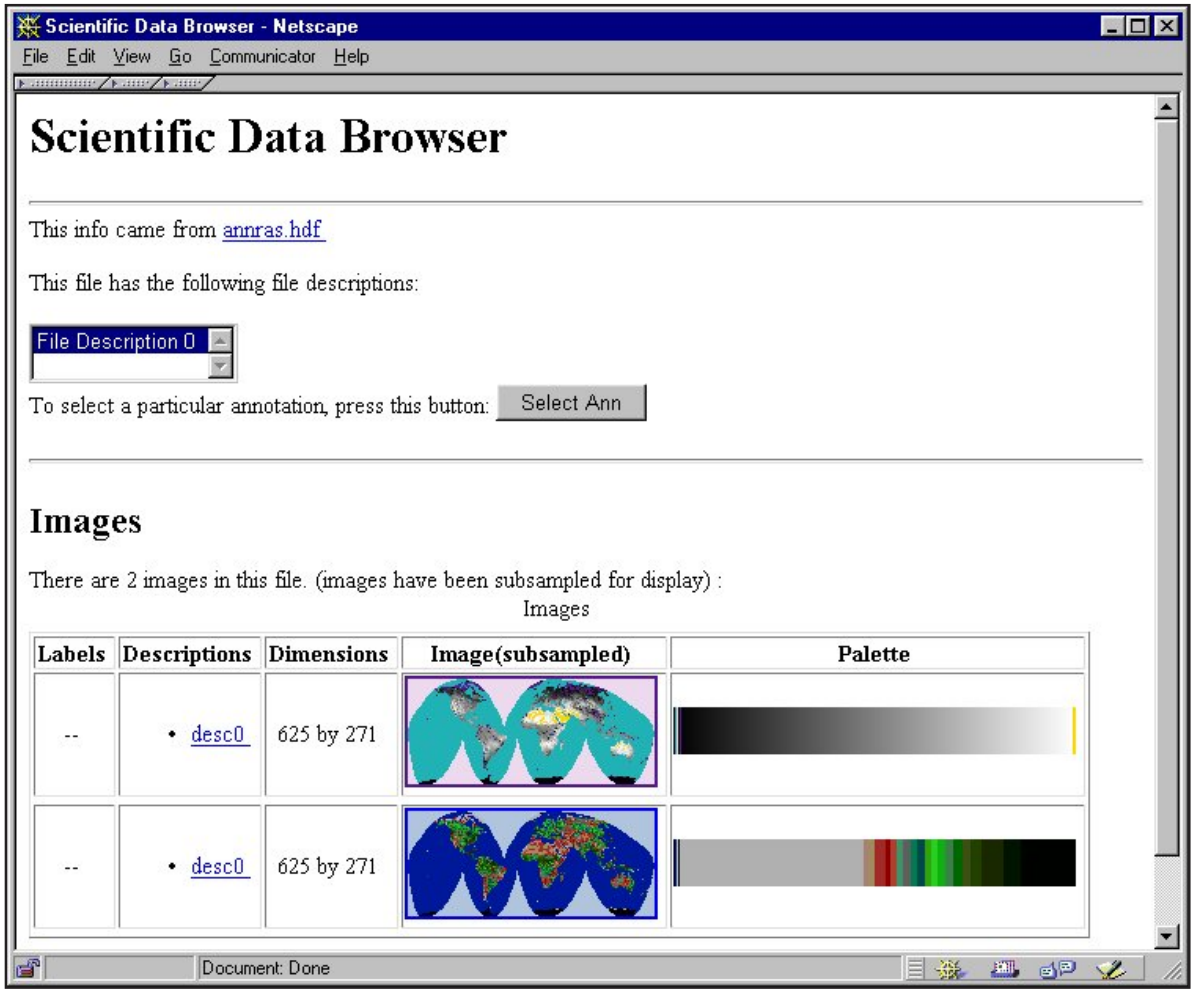
- **The following example shows how to include NCSA's Scientific Data Browser* CGI code as a service in EASIA - for post-processing HDF datasets - Simply included via XUIS modification**

***Yaeger, N. A Web Based Scientific Data Access Service: The Central Component of a Lightweight Data Archive, National Center for Supercomputing Applications, University of Illinois, Urbana-Champaign. http://hopi.ncsa.uiuc.edu/sdb/sdb.html**

# XUIS fragment for the SDB operation

```
<operation name="SDB" type="" filename=""
          format="" guest.access="true" column="false">
 <if>
      <condition colid="RESULT_FILE.FILE_FORMAT">
              <eq>'HDF'</eq>
      </condition>
 </if>
 <location>
      <URL>http://quagga.ecs.soton.ac.uk:8080/servlet/SDBservlet</URL>
 </location>
 <description>NCSA Scientific Data Browser</description>
</operation>
```

# NCSA'S SDB invoked on a dataset managed within our interface




# Post-processing via uploaded Java code

**Authorised users can upload Java code for secure server-side execution against datasets stored as DATALINKs on file server hosts**

**Code must accept filename as first command line parameter**

**Code must write output to relative filenames**

# Code upload for server-side execution - XUIS

```
<table name = "RESULT_FILE"
 primaryKey = "RESULT_FILE.FILE_NAME RESULT_FILE.SIMULATION_KEY">
  <column name="DOWNLOAD_RESULT"
    colid= "RESULT_FILE.DOWNLOAD_RESULT">
    <type><DATALINK/></type>
    <!-- Code  upload is allowed against this DATALINK, but not by
     guest users. A Java jar file can be run against the data-->
    <upload type="JAVA" format="jar" guest.access="false"
      column="false">
    <!--Only allow this operation on attributes in this column that
     meet the following conditions-->
    <if>
      <condition colid="RESULT_FILE.SIMULATION_KEY">
         <eq>'S19990110150932'</eq>
      </condition>
      <condition colid="RESULT_FILE.MEASUREMENT">
        <eq>'u,v,w,p'</eq>
      </condition>
    </if>
  </upload>
 </column>
```


# Implementation of Operations/Code Upload (i)

**Initial idea - a specially written operation *startup* servlet running within the JWS would dynamically load the required Java class (using *Java Reflection*).**

**Any output would be written to a temporary directory that had a unique name based on the user's servlet session identifier (and time/date information).**

**However, it is extremely difficult to redirect any file output to the temporary directory using this mechanism.**

**Although it is straightforward in Java to redirect any output directed to standard output or standard error to files in the temporary directory, it was not possible to get the *startup* servlet to redirect any other file output from the user's code (which used relative path names as mentioned previously)**


# SUN BUG Report -Setting the current directory from Java

**The following bug report is taken from the Sun's Java Developer Connection Bugs Database (http://developer.javasoft.com/developer/jdchome.html)**

**Bug Id: 4307856**
**Submit Date: Jan 27, 2000**
**Description: There should be a way for a Java application (not an applet) to set the current directory. This feature has a number of uses but it's mainly necessary to run certain applications launched from the Java application. This has been sorely missing for years and it's causing a lot of problems for developer that are then forced to use ugly hacks to work around the problem.**
**Workaround: None.**
**Evaluation: The Java platform API specifically does not allow the working directory to be changed, since having such mutable global state would vastly complicate writing multithreaded programs (4307856). For the purpose of creating a subprocess that runs in a different directory, however, a new variant of the Runtime.exec method was introduced in J2SDK 1.3 (Kestrel, see RFE 4156278).**


# Implementation of Operations/Code Upload (ii)

**Batch file approach supports changing directory and post-processing codes written in other languages.**

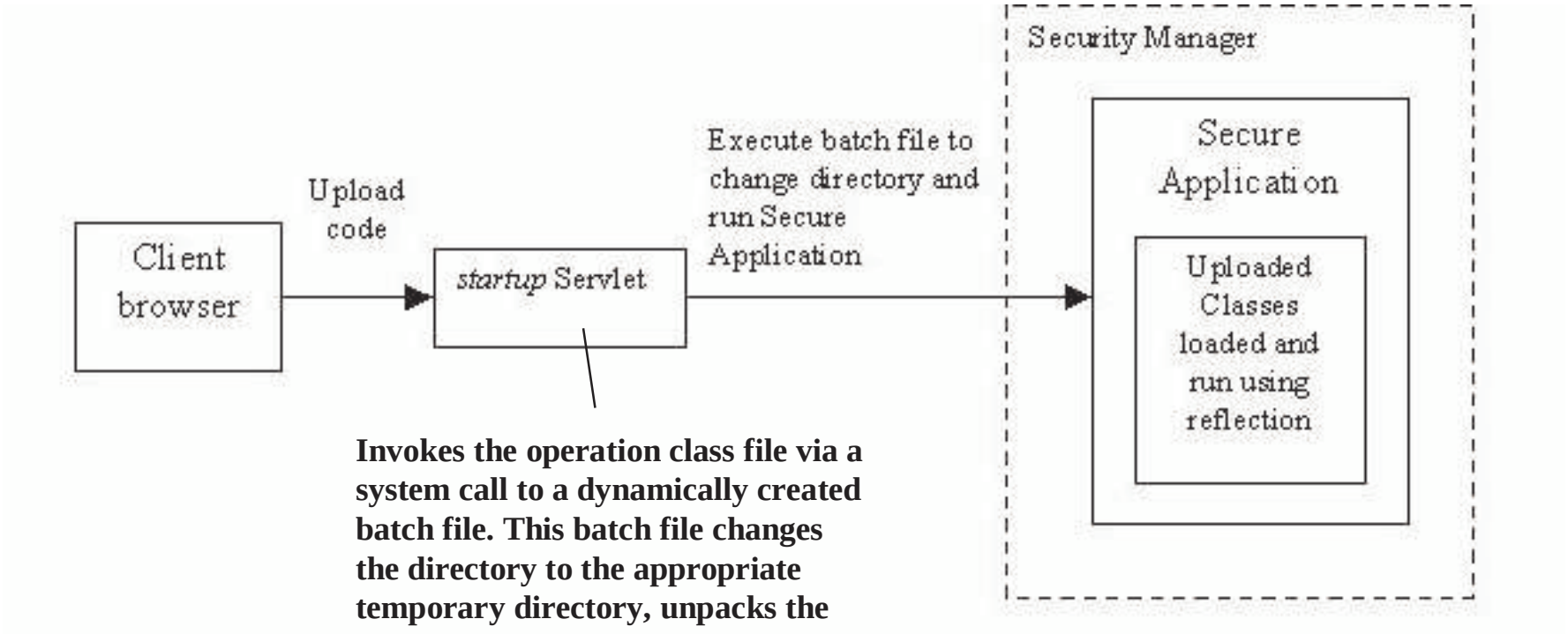
**Batch file is dynamically created by the startup servlet and contains commands to unpack operation into temporary directory and appropriate commands to invoke second Java interpreter or non-Java post-processing code.**

**The batch file mechanism is also used to run uploaded post-processing codes. This is required to not only fix the file output problem, but also to implement the 'sandboxing' restrictions required in this case.**

**Startup servlet then creates a script that changes to the required temporary directory, unpacks the uploaded jar archive, and then invokes another Java interpreter with another special secure application class that is used to implement the 'sandboxing' for uploaded code.**

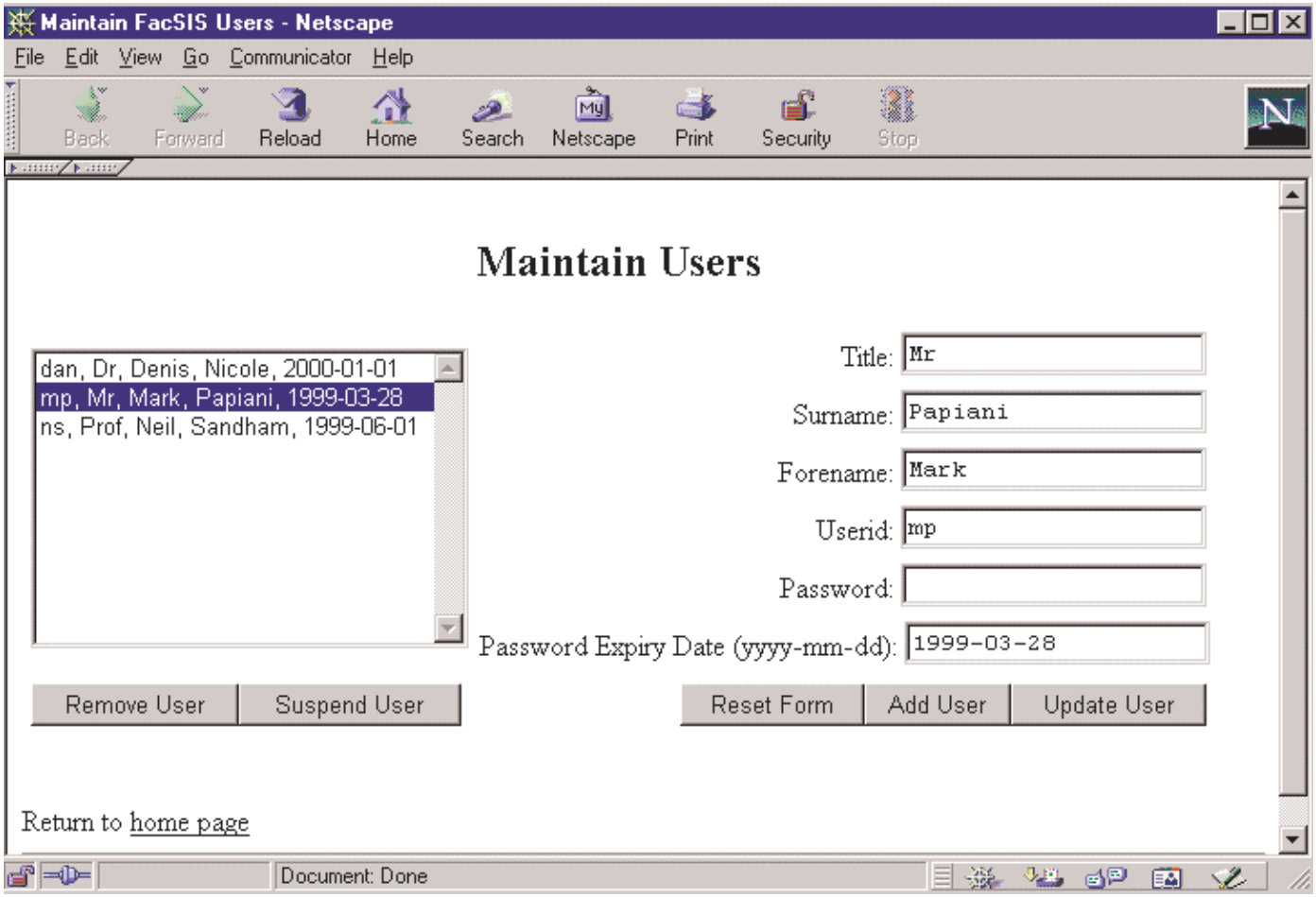
**This special class declares appropriate security restrictions and then dynamically loads and runs the user's uploaded code using Java reflection (applied to the user's input corresponding to the name of the class to run).**

# Implementation of Code Upload




**Invokes the operation class file via a system call to a dynamically created batch file. This batch file changes the directory to the appropriate temporary directory, unpacks the code if it is stored in an archive format (such as jar or zip) and then invokes another Java interpreter to run the operation class file for the user's requested post-processing code**

# Web-based user management



# Future

- Caching operations results
- Runtime monitoring of operation progress
- Store operation statistics (execution time, output details) for benefit of future users
- Can other languages be uploaded for secure execution?
- Extend XUIS DTD for more complex operation specification
  - operation chaining
  - operations applied to multiple datasets
  - Interactive applet based operations

# Summary (i)

- **Demonstrated an architecture for an active digital library to meet a requirement of the UK Turbulence Consortium *to make available to authorised users, large result files from numerical simulations, with a total storage requirement in the hundreds of gigabyte range.***

- **Greatly reduce bandwidth requirements by *avoiding costly network transfers associated with uploading data files to a centralised site* and by allowing *data reduction through post-processing* - archive applications along with data, which can be dynamically invoked to post-process the data.**

- **Data and code *distribution reduces retrieval bottlenecks/processing loading* at individual sites.**

- **Complete architecture including Web-based user interface for archiving large, distributed files whilst maintaining database security, integrity and recovery.**

    - **Help users locate scientific data files of interest, using an *intuitive searching and browsing mechanism* in keeping with a Web-based look and feel.**

    - ***Automate the interface construction* so that it requires little database or Web development experience to install and access. Generic, schema-driven system that can be used to manage many different types of large, distributed data archives.**

# Summary (ii)

**Automated construction -- *XML user interface specification (XUIS)* - generated by tool**

**Separating the *user interface specification* from the *user interface processing* can provide a number of further advantages:**

- **Customisation - Schema driven user interface can be customised (aliases for table and column names , different sample values, tables and attributes can also be hidden from view).**

- **User defined relationships between tables - Hypertext links to related data can be specified in the XML even if there are no referential integrity constraints defined for the database.**

- **Personalisation - Different Users (or classes of user) can have different XML files - different user interfaces to the same data.**

- **Operations can be associated with database columns - *standard reusable server-side post-processing codes loosely coupled to the datasets via XML defined interfaces***