

Technical Architecture of the ICE System

Franz Brummer

25.04.08

Contents

Introduction and Overview.....	1
Components of the ICE System.....	2
Database.....	2
Applicationserver.....	3
Webpages.....	3
ICE Applicationserver Web Application.....	4
Cocoon Publishing Framework.....	4
Client.....	5
Old Client.....	5
New Client.....	6
Example Table Generation.....	6
Client Start and Authentication.....	7
Keyword Selection.....	7
Stock Selection.....	7
Table Definiton.....	7
Table Generation And Display.....	7

Introduction and Overview

The ICE system is a Java based multi tier system. It is platform independent on the server as well as on the client side.

Three distinct tiers (or layers) can be distinguished:

1. Database Management System (DBMS)
2. Application Server
3. Client

All data is stored in the DBMS. Almost any relational database system which is supported by a JDBC (Java Database Connectivity) compliant driver can be used. In actual use are MySQL, Oracle and we are testing PostgreSQL – up to now without problems. The central part of the ICE is the application server. It handles all client requests, queries the database if necessary and returns the results to the client for display.

The ICE application server can handle multiple simultaneous requests from different clients. Every client gets its own session on the server. For delivering static and dynamic webpages to the client the application server can be connected to a webserver like Apache.

Authentication and session management is done over the webbrowser (HTTP based). A role system for user permissions is implemented.

Clients can be thick clients (Java applets or webstart applications) or thin web based (dynamic web pages).

For the server operating system we recommend Debian GNU/Linuxⁱ or the Debian derived Ubuntuⁱⁱ.

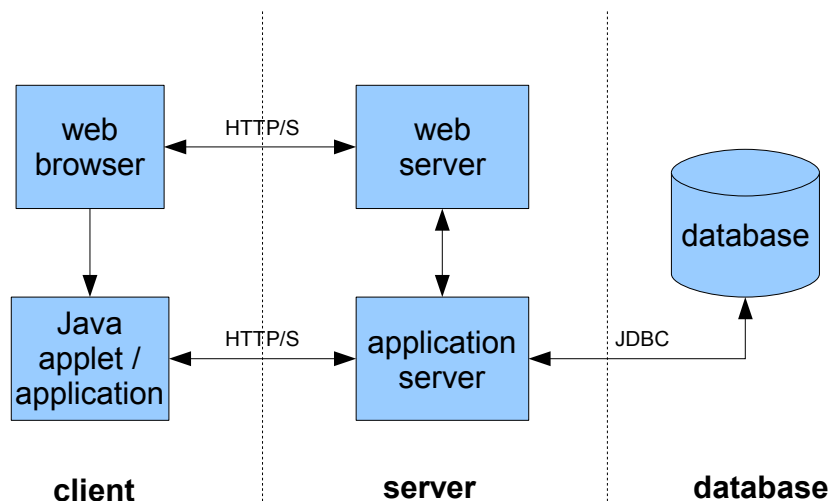


Figure 1: Components of the ICE System

Components of the ICE System

Database

The database contains a data warehouse of all the data in the ICE system. It consists of the facts tables with the actual data and meta data tables which describe the data. It also contains all the information about user accounts and standard tables (report tables saved by the users).

For an overview of the database see „Explanation of the ICE Database Structure“.

At the moment ICE is tested with Oracle, MySQLⁱⁱⁱ, PostgreSQL^{iv} and Informix database installations.

Data for the ICE has to be imported with special import tools that also update the meta data of the system. All data in the ICE is harmonized for the ICE key system.

The only components in the ICE that communicate directly with the database using SQL queries are the applicationserver and the tools for importing data or updating the metadata.

The database can run on the same server as the applicationserver (used mostly for testing and developing purposes) but usually is installed on a dedicated server.

For database administration tasks the database web interfaces phpMyAdmin^v (for MySQL) or phpPgAdmin^{vi} (for PostgreSQL) can be installed on the Apache webserver. This requires that the PHP module for Apache is installed.

Applicationserver

Technically the applicationserver consists of two web applications (ICE and Cocoon) running in a Java servlet container and a collection of static web pages. We use Apache Tomcat^{vii}, which is the reference implementation for servlet containers.

The applicationserver receives requests from the clients and responds to them. It manages the authentication and user sessions. It also manages the connection to the database, sends SQL queries to it and receives data from it.

The communication between the clients and the application server utilizes HTTP. A servlet in the ice web application receives HTTP requests with the commands from the clients and answers with HTTP responses that contain the results from the server.

Authentication is managed by Tomcat which gets the user login data from the database (jdbc database realm). The user has to login only once to use all web applications on the Tomcat server (single sign on). For every logged in user, session data is kept on the application server. Groups and roles can be associated with a login name to allow fine grained access control. The login credentials are provided by http basic access authentication. For additional security the https protocol can be used.

The Apache Tomcat is connected to the Apache webserver with the Apache Tomcat connector^{viii} (mod_jk) module. This allows to let Tomcat run behind the webserver with Apache serving static pages (as well as dynamic php pages like for example the phpMyAdmin web administration interface for MySQL databases) and redirecting all requests to the ice web applications to the tomcat servlet engine. The advantage of this approach is that Apache does what it is optimized for: serving web pages secure, and performant.

Webpages

The ICE System uses the Apache webserver^{ix} to serve static web pages. This can be used to build a portal site as entry point to the ICE System.

The ICE clients are then started from HTML links on the website as Java webstart applications or applets. There are also some administrative tools like the user account administration that can be linked to.

Collections of report tables (standard tables) can be made available in a hierarchic, directory-like structure on the site with a fine grained access control.

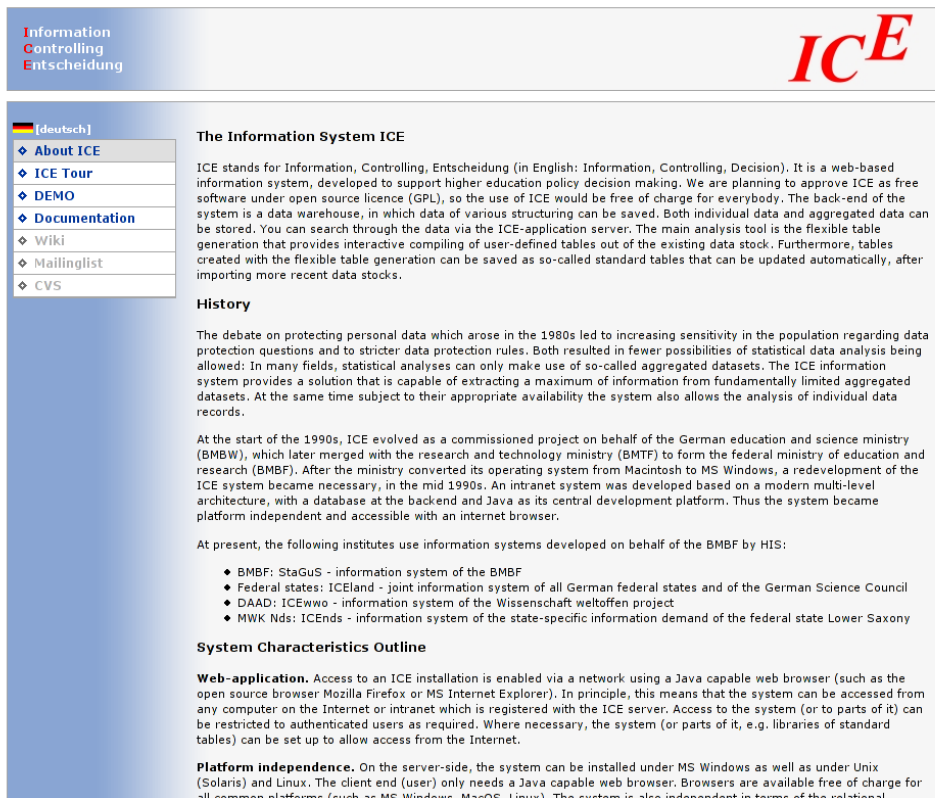


Figure 2: ICE Web Portal

ICE Applicationserver Web Application

The applicationserver receives HTTP requests from the clients. Every client has its own session on the server with state information for the current connection. Sessions time out after a certain period of inactivity.

Communication between the client and server can be by means of a proprietary text based protocol (the „old protocol“) or by sending and receiving of serialized Java objects (the „new protocol“).

If, for example, the new client asks for the data for a certain table definition, the definition object along with a container object for the data is sent to the applicationserver. The applicationserver analyzes the definition, queries the database, fills the container object with the data and sends it back to the client.

The client can ask the server for lists of the available keywords, keys and stocks in the database. It can send table definitions to the server and receive the data values and it can save table definitions together with their data values as standard tables.

Cocoon Publishing Framework

The Cocoon^x publishing framework is used to output tables in different formats. It transforms XML data with XSLT stylesheets and supports many different output formats like for example HTML, PDF, Excel, Gnumeric or XML.

Whenever the user defines a table or views a standard table the ICE system offers to output it in HTML, PDF or Excel using the Cocoon publishing framework. It runs as a web application inside the Tomcat servlet container alongside the ICE applicationserver.

Cocoon also renders the HTML pages while browsing the standard table collections of the users.

Client

The ICE system offers the user a choice of different clients:

Old Client

The old client is a java applet with an AWT (Abstract Windowing Toolkit – the first GUI framework for Java) GUI (Graphical User Interface).

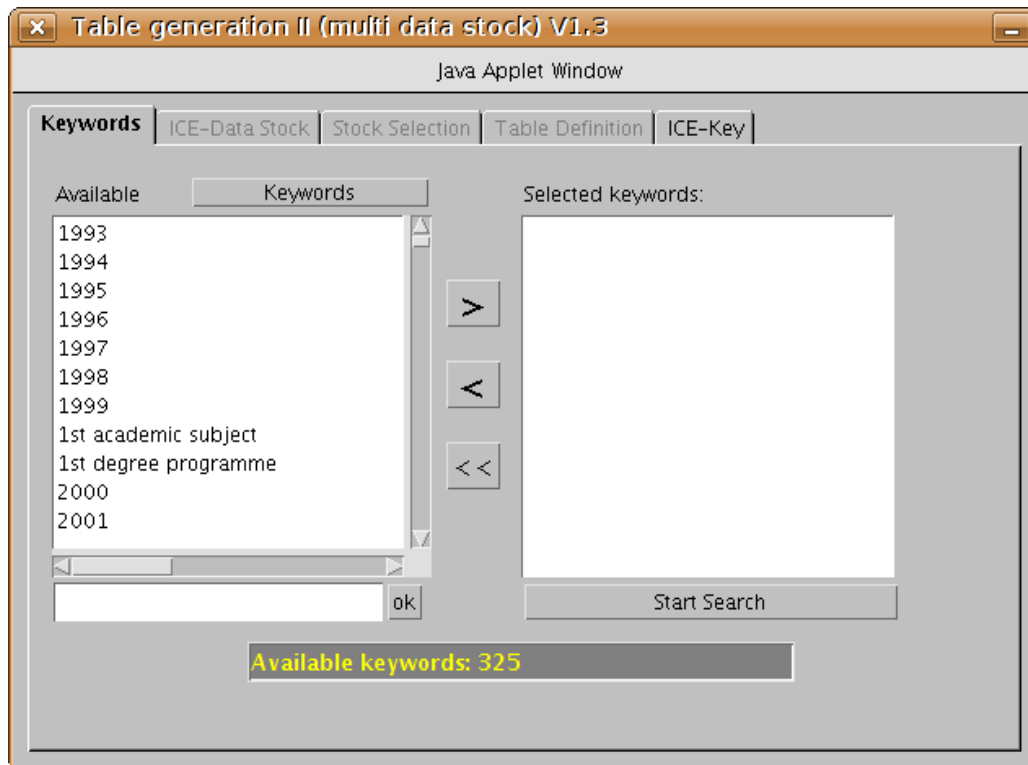


Figure 3: Old AWT Client

It runs under Java version 1.3 and above. The old client utilizes a proprietary text protocol over HTTP for communicating with the applicationserver. Its user interface is organized around tabs. The user starts by selecting keywords on the first tab. The system then searches for stocks which contain all the keywords and offers them for selection on the next tab. Once the user has selected a stock she can define a table definition on the table definition tab and send the definition to the server to receive the generated table.

New Client

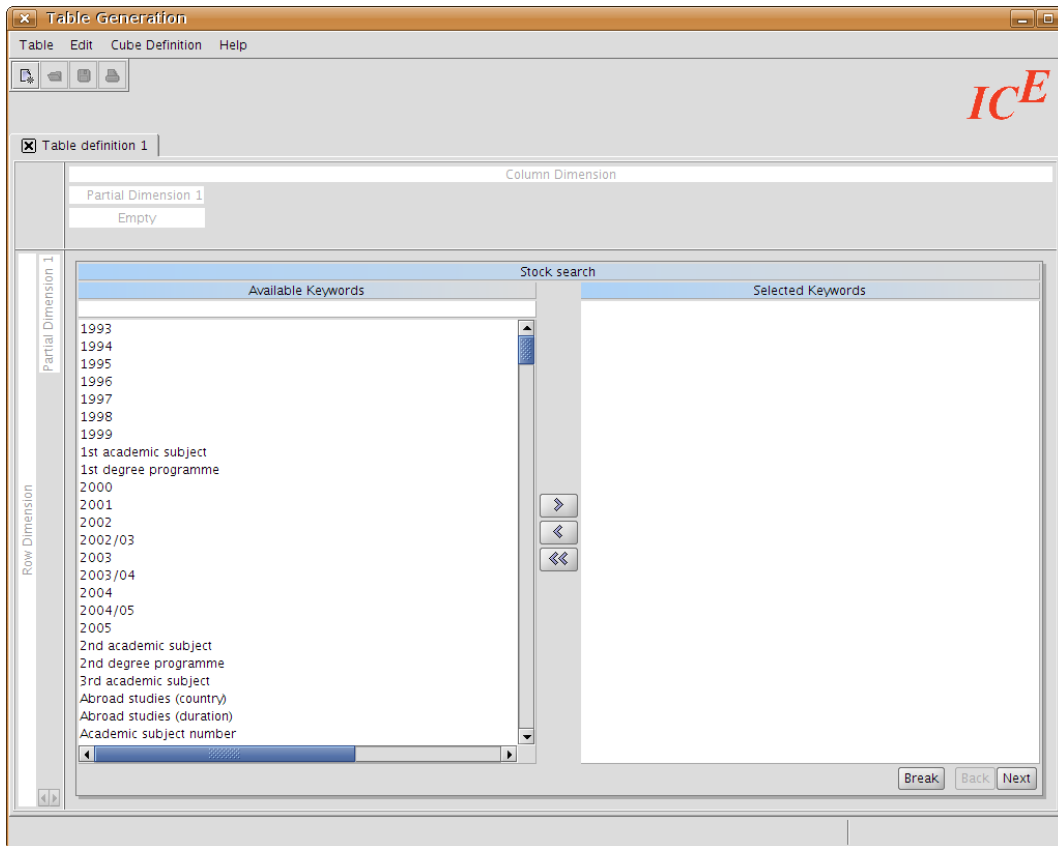


Figure 4: New Swing Client

The new client offers a modern Swing user interface with drag and drop, menus and a toolbar. It needs a JRE^{xi} (Java Runtime Environment) version 1.5 and above.

In contrast to the old client multiple table definitions can be open and manipulated simultaneously in a tabbed interface. The definitions are created by dragging and dropping characteristics or attributes into the different dimensions (row, column) so that the resulting row and column headers of the table can be previewed in realtime.

The new client communicates with the applicationserver by sending and receiving serialized Java objects. The way the new server and client communicate over the network can be configured very flexible: RMI (Remote method invocation), socket connections or HTTP can be utilized transparently. At the moment only HTTP is implemented.

Example Table Generation

To understand how the different components of the ICE System work together let's analyse the communication conducted during a typical session. This is just an overview of the events rather than a detailed description of everything that happens during the table generation.

Client Start and Authentication

The user opens the ICE portal website in the webbrowser. He can then click on a link to start the client. He is presented with a login dialogue that asks for his user name and password. Depending on the client version the dialogue is either the normal browser dialogue for http authentication or a custom dialogue window from the Java webstart application.

During the authentication the username and password are sent to the applicationserver. The server then opens a database connection and looks up the login data in the database. If the user name and password match with the database the server instantiates a session for the user and sends back a session identifier to the client. After the successful authentication either an applet or a webstart application window opens - depending on the client version.

Keyword Selection

Now the client displays the main application window to the user. The client has to load the list of available stocks, the list of all the keywords and the keys from the database to make these items available for selection by the user. So the client asks the server for these items who in turn makes queries to the database and sends back the results (depending on the client version, either as text strings or Java objects) to the client.

The user then selects keywords to define which stocks he needs. Again the selected keywords are sent to the server who queries the database for a list of the stocks which contain all the keywords and sends it to the client.

Stock Selection

On the client the list of stocks is presented to the user who can then select a stock. Whenever the user selects a stock (usually by clicking on the entry for the stock in the list) the client asks the server for all the characteristics and their attributes that are contained in the stock. An overview of the characteristics contained in a stock is then shown to the user by the client.

Table Definiton

When the user has found the right stock for his purposes he selects it for definition. He can then choose characteristics and attributes the stock contains for the row and column headers of the table.

Table Generation and Display

Once the user is satisfied with the table definition she sends it to the server along with the information in which format she would like to view the resulting table. Results can be displayed in a spread sheet like Java application (ICEcalc) as HTML in a browser window or as PDF. The server then queries the database for all values in the table. When all values are collected the server sends them to the Cocoon publishing framework to format the table for display. The formatted result table is then sent to the client.

- i Debian GNU/Linux: <http://www.debian.org>
- ii Ubuntu: <http://www.ubuntu.com>
- iii MySQL: <http://www.mysql.com>
- iv PostgreSQL: <http://www.postgresql.org>
- v phpMyAdmin: http://www.phpmyadmin.net/home_page/index.php
- vi phpPgAdmin: <http://phppgadmin.sourceforge.net/>
- vii Apache Tomcat: <http://tomcat.apache.org/>
- viii Tomcat Connector: <http://tomcat.apache.org/connectors-doc/>
- ix Apache Webserver: <http://httpd.apache.org/>
- x Apache Cocoon: <http://cocoon.apache.org/>
- xi Sun Java JRE: <http://www.sun.com/java/>