

# ICPS'99 Participant Database

## — Principles and Practice

Ilkka Rinne

3rd August 2000

### 1 Introduction

This paper is about the participant registration system used by the organisers of the ICPS'99 conference held in Helsinki in August 1999. In the following text I assume that the reader has some experience about database systems and HTML in general.

### 2 What Did We Need?

Our main goal when we, the ICPS'99 organisers, begun to design the participant registration system, was to store the pre-registration information for later use. Most of the registration information was expected to come in using HTML pre-registration form on our homepage, but also by using email version of the form and even on printed forms by mail.

We knew that there would be a lot of changes to the participant's original information during the spring and summer before the conference. The responsibility for taking care of the participant's needs was distributed among the organising committee: each member of the committee would have their own countries to be responsible for. This meant that the participant data needed to be updated by several people from different locations, possibly simultaneously.

There was need for showing a list of all pre-registered participants on ICPS'99 homepage. As none of us had particular interest in updating the list manually regularly enough to be of any use, this information needed to be generated automatically. However, we all were anxious to follow the progress of pre-registration, so also we needed a generated list of the pre-registered participants with precise pre-registration times.

Because of the reasons mentioned above, the reliability requirements and former experience with Oracle and its WebServer in particular, we decided to use the Oracle database management system (DBMS) and PL/SQL procedures to create the registration system.

### 3 About the Environment

The Oracle 7 DBMS we used was owned by the computing center of our university and running on shared multipurpose server having unix operating system.

The programming was done using PL/SQL programming language including HTM package for easier HTML code generating. Attached to the database was Oracle's web server (OWS) with Web Request Broker (WRB) listening to HTTP requests from internet and dispatching them to the PL/SQL procedures stored in the database [1].

### 3.1 Relational Database

Relational database systems are based on relational model defined by Codd (1970). The relational model has a simple and uniform data structure: a relation. It also has solid theoretical foundation for operations on relations called relational algebra.

Information stored in a relational database is organised as *relations* containing *tuples*. A relation can be visualised as a table with tuples as its rows. A relation has one or more *attributes* which are column headers in our visualisation. [2].

References may be defined between different relations by stating that the values of certain attributes (a foreign key) in one tuple of a relation should exist also as attributes of another tuple of another relation.

If there were a relation describing a single page in a book, and also one describing the book itself for example, the page relation would have an attribute containing the same values as the book relation's key attribute. This means that the book has all and not more than the pages which have the identity of this book as their book attribute.

### 3.2 SQL and PL/SQL

Many contemporary relational databases are managed and used using a language called Structured Query Language (SQL). SQL uses the terms *table*, *row* and *column* for relation, tuple and attribute, respectively. [2]. SQL has statements operations of relational algebra like selecting, inserting, deleting and updating the rows of tables for example. Tables can also be created, modified or removed using SQL. The plain SQL is not very good as a programming language because it has no flow control statements.

Oracle DBMS has its own procedural programming language for database procedure programming called PL/SQL. PL/SQL is a quite similar to Pascal or ADA in syntax. Database information can easily be accessed with PL/SQL statements: it has cursors for browsing the result rows of a query. Procedures can be compiled and stored to the Oracle database and executed from there.

In our case we used PL/SQL procedures to fetch the wanted information from the database and write it out as HTML pages to be shown by web server. Another use was to process the data sent by web browser from HTML forms and store it into the database.

### 3.3 Web Server

Normal web server listens for HTTP requests for documents stored web server's file system. Oracle WebServer (OWS) has a special kind of listener process called Web Request Broker (WRB) listening for HTTP requests aimed for PL/SQL procedures stored in the database. WRB strips the parameters from HTTP

request's query string and forwards them to the called procedure. The HTML output of the procedure is returned to browser [1].

## 4 Interactive Web Service

In this section I describe some of fundamental design principles of an interactive web service based on HTML.

### 4.1 HTML Building Blocks

In an interactive service the server and the user go through a dialog: The user asks something from the server and (hopefully) receives an answer or the other way round.

HyperText Markup Language (HTML) initially created by Tim Berners-Lee at CERN in 1990 [4] provides means for constructing simple forms for user to provide input to the server. HTML version 4.0 provides the following user interface controls or widgets: [3]

- buttons for sending or resetting form fields
- checkboxes for selecting separate options
- radio buttons for mutually exclusive selections: only one option can be selected at a time
- inline menus or selection lists
- text input fields for a single line of text
- file select buttons for submitting a local file to the server
- hidden controls that store information between form submission, but are not shown to the user
- object controls that can be basically anything

The object controls should be used with caution because this feature has not been implemented in all browsers. A HTML link can of course be used in addition to the form widgets. All information from user must be collected using these widgets.

The forms are included in HTML pages and sent to the server by the browser when the submit button is pressed. Response from the server may be an informative HTML page or an editable page which is another form based on information sent on the previous form.

### 4.2 Statelessness and Dynamicity

It is worth noticing that the server knows nothing of HTML forms before they are submitted. It does not know if it should be waiting for an answer or submit for a form. This type of server design which processes each request as independent action is called stateless. Most interactive web services include requesting and receiving more than one HTML page however. If the showing of a page

requires a preceding request of another, the session information must generated by the server on the first request and sent back to the server on following request. HTML forms' hidden controls provides one way to do this.

There can be two types of HTML pages included in an interactive web service: static and dynamic. Static pages are stored as files in the file system of the web server and copied to browser on a request. Dynamic pages are created programmatically for each request and then sent to the browser. The resulting HTML page can be exactly the same in both cases. The reason for using slower dynamic page creation is usually the dynamic nature of the source data.

The information on a HTML page may be, and often is, stored somewhere else than the HTML file, like a database. When the page is requested the needed data is fetched from the database and only after that the requested page is created. On the other hand, if the page includes only static information that will always be the same, it makes sense to create a static HTML file for it. A simple login form could be created as a static page for example. The user name and password could then be sent to the server resulting in creating a dynamic welcome page with the users name.

### 4.3 Page Producers and Processors

When designing an interactive web service based on dynamic page creation it is often useful to create separate pieces of software for creating a page and processing the submitted data. These pieces of software can be scripts, procedures, methods or something else depending on programming language used.

The page producer fetches the data from the source, creates the page using the data and returns the page to the browser. If the created page contains a form for user input, the page producer must take care that the form will be sent to the corresponding page processor on submit, see Figure 1.

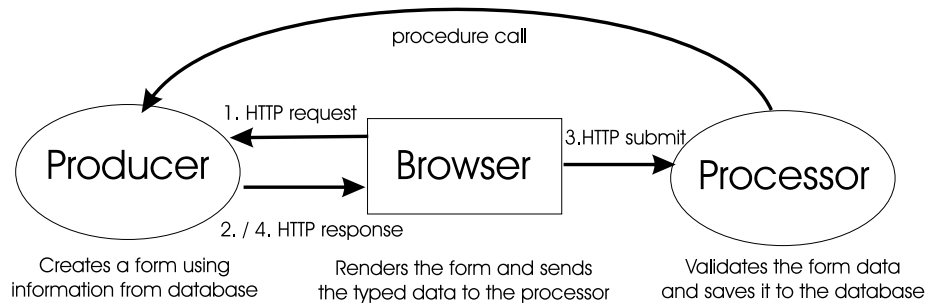


Figure 1: Page producer and processor

The page processor has to have some means to get the submitted form data, one obvious way is to use call parameters. The processor must check that the input is valid, which may require one or more data request from the data source. If everything is OK, the data may have to be stored to the data source, a result page created and sent back to the user. A good habit is to send the just saved data back to the user to confirm the operation.

If the operation cannot be completed either because of the user or system error, the reason for the failure should be told to the user and a form containing

the submitted (possibly erroneous) information be created and sent back to the user for making corrections. The original page producer can be used to create this form in most cases. On success the page processor may also call another page producer to create the result page.

## 5 ICPS'99 Database

In this section I go through the tables and an example of a web service of the ICPS'99 participant database system.

### 5.1 Tables

The ICPS'99 participant database is quite simple having only three tables: the participant information, excursion voting (visit) and homepage visit counter (visitcount).

Participant table (Table 1) reflects the pre-registration form in a quite straightforward manner. Participant number *pnumber* is fetched from an increasing integer sequence when inserting a new row to the table. Attribute *field* is the field of physics of the participant and *partfield* is the field of the participant's lecture or poster.

Extra fields *notes1* and *notes2* were used to store organisers' internal notes concerning the participant. These note attributes were later used to indicate the date the participant was accepted for the conference, the date the conference fee had arrived and if the participant had cancelled his or her registration.

The visit and the homepage counter tables had simple structure and are not very interesting.

### 5.2 Service Example

#### 5.2.1 Procedure *pedit*

The procedure declares a cursor for fetching the row from participant table (see Figure 2). The procedure tries to fetch a row having the given participant number from the participant table. If the fetch is successful, the procedure prints out a HTML form having a field for all participant's editable information. As an example a PL/SQL command printing a text field containing participant's phone number wrapped inside aHTML table cell would be:

```
http.p('<TD align=left>' || htf.formText('sphone',25,32,'123456') || '</TD>');
```

producing the following HTML code:

```
<TD align=left><INPUT TYPE=text NAME=sphone VALUE=123456 SIZE=25 LENGTH=32></TD>
```

The participant number is printed on the form as a hidden parameter. On submit the form is sent to the procedure *saveform*.

## Participant

<i>attribute</i>	<i>type</i>	<i>null</i>
pnumber	integer	not null
preregistered	date	not null
lastname	varchar2(64)	not null
firstname	varchar2(64)	not null
address	varchar2(256)	null
country	varchar2(64)	not null
university	varchar2(256)	not null
department	varchar2(256)	not null
field	varchar2(256)	not null
phone	varchar2(32)	null
fax	varchar2(32)	null
email	varchar2(64)	null
nationality	varchar2(64)	not null
sex	varchar2(1)	not null
passport	varchar2(64)	null
dateofbirth	date	not null
iapsmember	varchar2(64)	not null
committee	varchar2(256)	null
partfield	varchar2(256)	null
postertopic	varchar2(256)	null
posterdescr	varchar2(256)	null
lecturetopic	varchar2(256)	null
lecturedescr	varchar2(256)	null
hostel	varchar2(64)	null
haspaid	number(6,2) default 0.0	not null
checkedin	date	null
accomodation	varchar2(1)	not null
tosponsor1	varchar2(1) default 'N'	not null
disabled	varchar2(1) default 'N'	not null
notes1	varchar2(256)	null
notes2	varchar2(256)	null
primary key	pnumber	

Table 1: Participant table

```

cursor pcursor is
  SELECT *
  FROM ops$iorinne.participant
  WHERE pnumber = to_number(particip);

```

Figure 2: Cursor for getting a participant row

### 5.2.2 Procedure *saveform*

The procedure receives the form data created by procedure *pedit* as call parameters: one parameter for each input field of the form. The field values are trimmed from any leading or trailing space characters and cut to the maximum length of the corresponding attributes in the participant table. The form values are validated and, if everything is OK, saved to the database and confirmation message printed. In case of any errors or missing mandatory fields an error message is printed containing all noticed errors.

## 6 Conclusions

The registration system build up from scratch in couple of weeks filled all expectations we had for it. There was occasionally need for special kind of participant listings for organisers' purposes. These were coded as tailored utility procedures with PL/SQL on demand.

The notes fields proved to be a very good thing, because they could be used by organisers to store important data about the participant. These field were used for both free format text information like special accommodation arrangements and encoded data like whether this participant had been accepted for conference.

However the ICPS'99 registration system was a quite simple database application and the Oracle DBMS would have been much too heavy-weight for it if we would have had to install it for this purpose. Oracle is also quite expensive, so it is not a recommended choice for this kind of system if it is not already available. The PL/SQL code produced is not portable to other database management systems, which reduces possibilities for code reuse.

The complete PL/SQL source code and the HTML pre-registration form can be found from my homepage at <http://www.iki.fi/ior/icps99db/>

## References

- [1] M. Gruber, K. Rossi: *Oracle WebServer User's Guide, 2.0.2*, Oracle Corporation 1996 (PDF)
- [2] R. Elmasri, S. Navathe: *Fundamentals of Database Systems*, 2nd edition, Addison-Wesley 1994
- [3] D. Ragget, A. Le Hors, I. Jacobs: *HTML 4.0 Specification*, W3C Recommendation 18-Dec-1997, <http://www.w3.org/TR/REC-html40-971218>
- [4] T. Berners-Lee: *The World Wide Web: Past, Present and Future*, Lecture paper, August 1996, <http://www.w3.org/People/Berners-Lee/1996/ppf.html>