PHP and databases

Objectives

Upon completion of this session you should be able to:

- extract data from a database using PHP
- create formatted output pages based on the data

Background

In the last session we looked how we can extract web form data in PHP and store it in variables by using the **$_REQUEST** function. This function is *passed* the name of a form field (e.g. postcode) and *returns* the value of the field when the user clicked on the Submit button:

```
<?php
$dateofbirth = $_REQUEST['dateofbirth'] ;
$monthofbirth = $_REQUEST['monthofbirth'] ;
$yearofbirth = $_REQUEST['yearofbirth'] ;
$postcode = $_REQUEST['postcode'] ;
.
.
.
?>
```

We also used that information to create (relatively) simple HTML-formatted *reply pages* using the **echo** statement. (If you are not sure about this process, you should go back and review it now.)

We need to turn to how we use the form data we have extracted in a more effective and useful way. In particular, we will need to look at how we can use the data to 'look up' things in a database, and how we add the data that the user has sent into a database.

Doing this properly requires that we know enough about databases to access one and (eventually) to make one. This is most definitely not a database unit, so we want to limit our involvement with databases to a necessary minimum; this minimum includes

- how PHP can connect to databases
- how we retrieve data (records) from a database using PHP, and
- how we add records to a database to store form data.

We will cover this in the next two sessions.

Oracle

For any program or script to access a database requires a *driver* of some kind; a piece of software that links the program (in our case a PHP script and the interpreter) to the database. There are several ways that we can do this, depending on which platform (operating system) the web server is running, and on the database systems available on that system. Each way has a somewhat different setup, and uses slightly different PHP syntax. This makes for code that is not fully portable, but that is one of the problems that 'real world' system developers in this field have to grapple with. We will focus on how to make it work, and on understanding what is happening, rather than on different ways of doing this. Those who want to know more can obviously do some research to extend the material we are covering here.

Of the various methods that we could have used for this unit, we have chosen to use *Oracle* database management system to create and manage the databases, and the Deakin web server to deliver the PHP pages. The *Oracle* database management system is available on the University system, it is accessible by Telnet. Also the Deakin web server has support for using PHP with Oracle databases.

You will need to upload your PHP pages to your webspace on the Deakin webserver which is in the "public_html" folder of your network folder.

Tasks

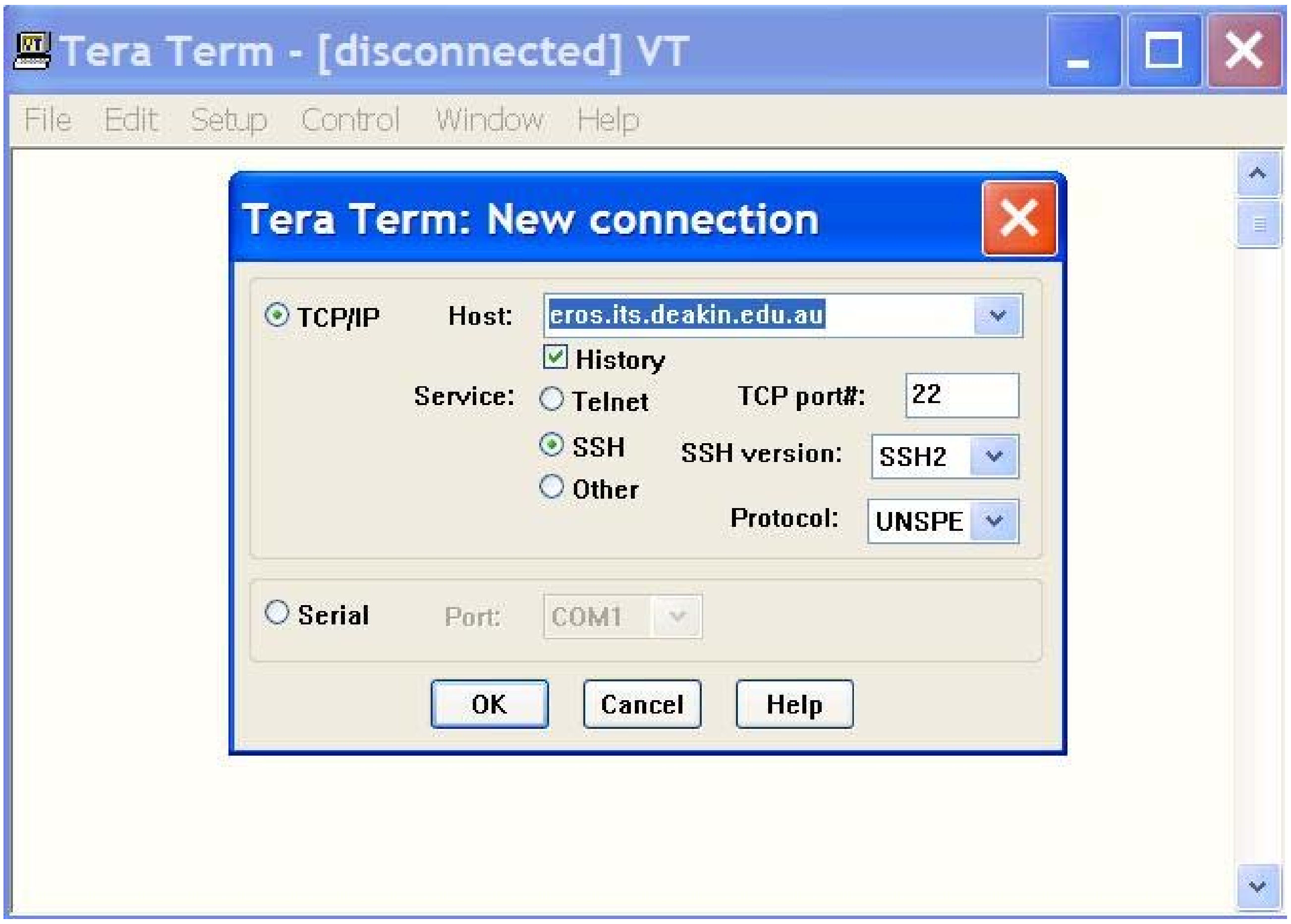
In this (and the next) session we are going to use PHP to connect to and retrieve information from an Oracle database table named "Login"; in the third session of this sequence we will add records to (and change values) in the database table. Download this login table and save it onto your Deakin home area (eg. public_html/SIT104/).

This table (called *Login*) has three fields: id, username, password. If you open this file in note pad, you would see all the sql statements used to create a table and insert records in this *Login* table:

```
DROP TABLE Login;
CREATE TABLE Login(
        ID  NUMBER(4) NOT NULL,
        USERNAME VARCHAR2(50),
        PASSWORD VARCHAR2(20),
        Primary key (ID)
);
INSERT INTO Login VALUES (1,'john','john08');
INSERT INTO Login VALUES (2,'nicole','nic');
INSERT INTO Login VALUES (3,'james','james');
INSERT INTO Login VALUES (4,'samantha','sam21');
commit;
```

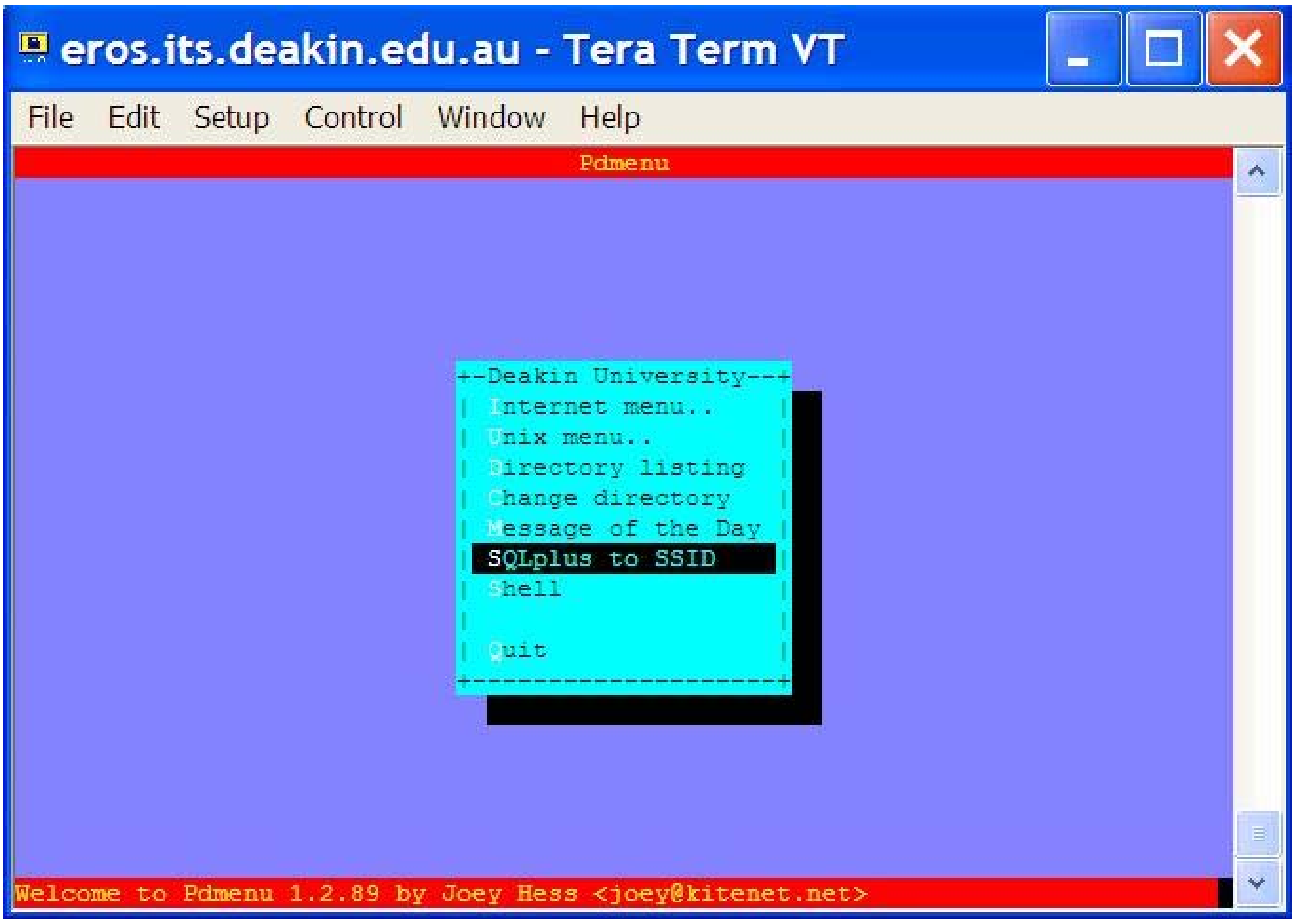
This table currently has four records (lines of data) in it. We are going to retrieve those data using PHP, and display the data in neat web pages.

Now telnet (eg. using Tera Term Pro) to Deakin web server:


Tera Term - [disconnected] VT
File Edit Setup Control Window Help
Tera Term: New connection

TCP/IP [✓] Host: eros.its.deakin.edu.au
History [✓]
Service: Telnet [✓ SSH] Other
TCP port#: 22
SSH version: SSH2
Protocol: UNSPE
Serial Port: COM1

Input Deakin user id and password, and select "SQLPlus to SSID" from the menu:


eros.its.deakin.edu.au - Tera Term VT
File Edit Setup Control Window Help

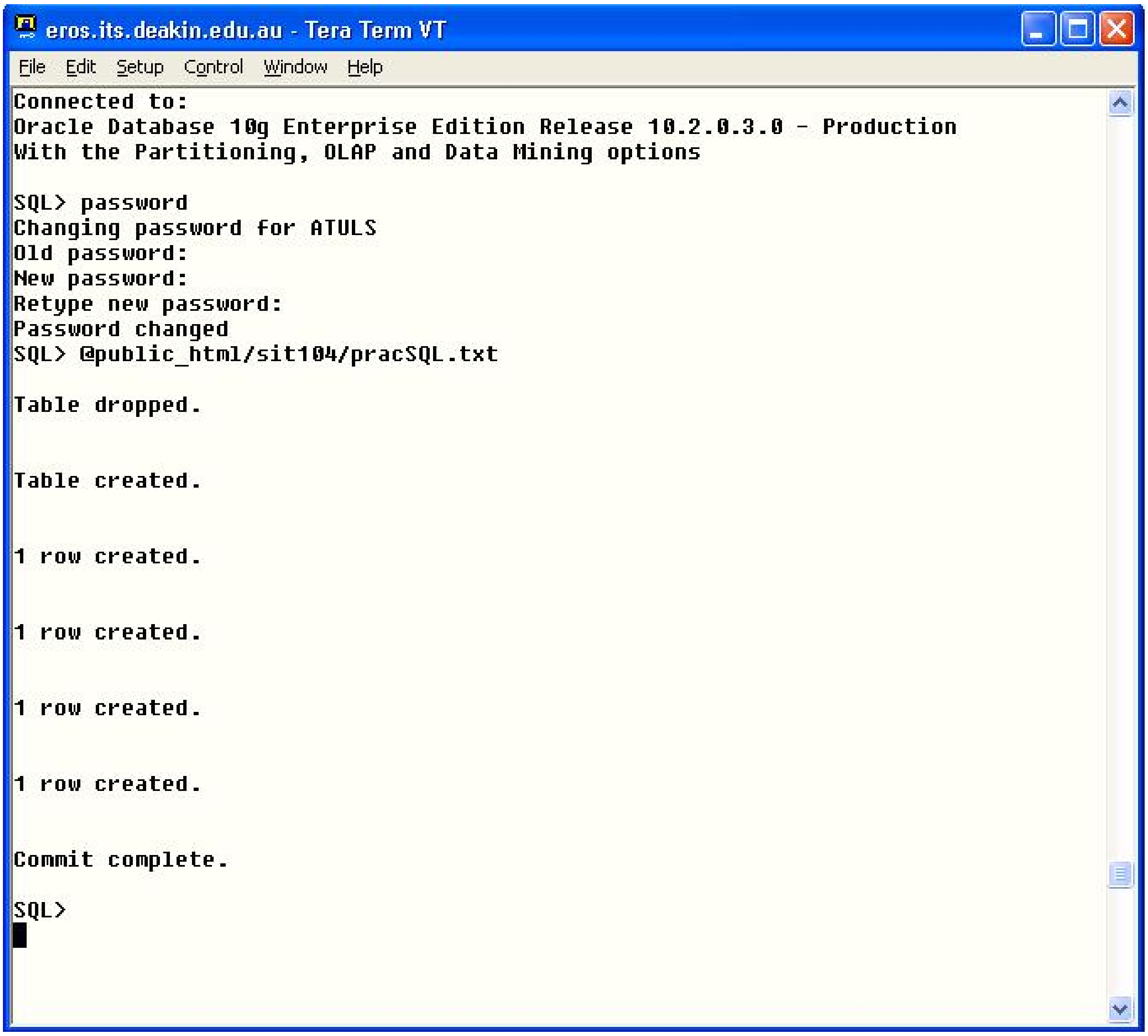
```
                              Pdmenu

              +-Deakin University---+
              | Internet menu..     |
              | Unix menu..         |
              | Directory listing   |
              | Change directory    |
              | Message of the Day  |
              | SQLplus to SSID     |
              | Shell               |
              |                     |
              | Quit                |
              +---------------------+

Welcome to Pdmenu 1.2.89 by Joey Hess <joey@kitenet.net>
```

By default, your Deakin password is also your Oracle database password. **We'd better change the Oracle**

password as we have to hardcode it in the php file later on. Under SQL> prompt, type "password" to change it and "@public_html/SIT104/pracSQL.txt" to load the login table in Oracle database. Path "public_html/SIT104" is where we save the login table pracSQL.txt:


eros.its.deakin.edu.au - Tera Term VT
File Edit Setup Control Window Help

```
Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.3.0 - Production
With the Partitioning, OLAP and Data Mining options

SQL> password
Changing password for ATULS
Old password:
New password:
Retype new password:
Password changed
SQL> @public_html/sit104/pracSQL.txt

Table dropped.


Table created.


1 row created.


1 row created.


1 row created.


1 row created.


Commit complete.

SQL>
```

Now the login table is created in the Oracle database. We need to *connect to* our database; that is, we have to tell the PHP server where the database is, and how to access it. Create a new document, and save it as ReadDB.php. You can edit it on your Deakin home area (or your hard disk if you are not in the labs), then upload it to the Deakin web server when you are ready to test it. :

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>ReadDB.php</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-
8859-1">
</head>
<body>
<?php

/* Set oracle user login and password info */
$dbuser = "xxx"; /* your deakin login */
```

```
$dbpass = "xxxxx"; /* your oracle access password */
$db = "SSID";
$connect = OCILogon($dbuser, $dbpass, $db);

if (!$connect) {
echo "An error occurred connecting to the database";
exit;
}

/* build sql statement using form data */
$query = "SELECT * FROM Login";

/* check the sql statement for errors and if errors report them */
$stmt = OCIParse($connect, $query);
//echo "SQL: $query<br>";
if(!$stmt) {
echo "An error occurred in parsing the sql string.\n";
exit;
}
OCIExecute($stmt);?>




</body>
</html>
```

Your Oracle user id (which is the same as your Deakin user name), password and the name of the Oracle database "SSID" are all we need to connect to the database .

Once we have established a connection we can work with the database in any way we want. The last statement **OCIExecute($stmt)** retrieves a collection of records ('sets of data') from the database; it does so using a particular and important method, namely the Structured Query Language, **SQL**.

An SQL primer

SQL is a clever way of structuring *queries* on databases. A *query* is any attempt to 'ask questions' of a database, and it returns one or more (or no) records that match that query. SQL was developed to provide a clear and consistent way to formulate such queries (and many others). All SQL queries are formed as pseudo-sentences, with a particular structure and keywords. Perhaps the simplest would be:

SELECT * FROM LOGIN

which means *retrieve all the fields* (SELECT *) *in the table called Login* (FROM Login). If we want only some of the records - for example, we want only those records with the username *john* - we need to add the WHERE keyword to the statement, so we would change this to :

SELECT * FROM LOGIN WHERE USERNAME='john'

This will retrieve all fields for all the relevant records (if any) ready to be displayed; the name of the field (username) must *exactly* match one of the fields in the database.

... which brings us full-circle to the *Execute* statement in our code earlier!
This means we just need to change the contents of the string inside **OCIExecute( )** to change what records we

retrieve. We can put any legal SQL statement we want into there; in the next session we will look at how we can create that statement 'on the fly' using values extracted from a form completed by a user (that is, a 'lookup form').

Using the OCIFetch

The whole point of being able to retrieve data in this way is to use that data, and the most obvious use is to display it. Start with some normal HTML to set up the page and the table (including the first row of headers); put this at the top of the body section or after command "OCIExecute($stmt);?>" :

```
<h1>PHP and Oracle databases</h1>
<h4>Table: <em>Login</em></h4>
<div align="center">
<table border="0" bgcolor="#339933" cellpadding="5" cellspacing="1">
<tr bgcolor="#006633">
<td width="150" style="color:#ffff99">ID</td>
<td width="150" style="color:#ffff99">Username</td>
<td width="150" style="color:#ffff99">Password</td>
</tr>
```

Leave plenty of blank lines, then copy the following lines and paste them just before the close of the body section to 'close out' the table and the page:

```
</table>
</div>
```

Notice that this is all 'pure' HTML, but the bulk of the table needs to be created using the retrieved data, so it has to be done inside a PHP code block (<?php . . . ?>). We need to display the records, one row of the table for each record we have retrieved - so we need to put all the code to do this between these two existing blocks. It also means that we need to *loop* through all the records; actually we should probably check first if there *are* any records, otherwise the table will be empty and strange-looking! (We'll do that in the next session.)

We need to loop through the records, but we don't know how many there are (yet), so we can't use a **for** loop; this is one of the uses of the **while** loop, and our *stopping condition* is when we have reached the 'end of the file' :

```
while(OCIFetch($stmt)) {
.
.
// do stuff with the retrieved records
.
.
}
```

We access the individual fields in our retrieved records using statements of the form *$fg1 = OCIResult($stmt,"ID"); //ID must appear uppercase* or *$fg1 = OCIResult($stmt, 1); // 1 is the column number of ID field in login table* . This places the value of the Drainage field for the current record into a variable called **$fg1** (which can be any legitimate variable name, of course), ready to be displayed using our familiar **echo** statement:

```
$fg1 = OCIResult($stmt,"ID");
echo ($fg1);
```

We can of course modify the default appearance of the resulting text by placing normal HTML tags in with it as strings:

$fg1 = OCIResult($stmt,"DRAINAGE");
echo ("<em>".$fg1."</em>");

Okay, copy and paste all of the following code between the two blocks you have already put into the file (between </tr> ... and ... </table>):

```
<?php

// Display all the values in the retrieved records, one record per
row, in a loop
while(OCIFetch($stmt)) {
// Start a row for each record
echo("<tr valign=top bgcolor=#ccffcc>");
// Output fields, one per column
// ID value in column one
$fg1 = OCIResult($stmt,"ID"); //"ID";
echo("<td width=150>");
echo ($fg1);
echo("</td>");
// USERNAME value in column two
$fg2 = OCIResult($stmt,"USERNAME"); //"USERNAME";
echo("<td width=150>");
echo ($fg2);
echo("</td>");
// ID value in column one
$fg3 = OCIResult($stmt,"PASSWORD"); //"PASSWORD";
echo("<td width=150>");
echo ($fg3);
echo("</td>");
echo("</em></td>");

// End the row
echo("</tr>");
}
// Close the connection
OCILogOff ($connect);
?>
```

You should be able to see that we put the field value into a variable ($fg1, $fg2, $fg3), create a table cell (<td>) for each field (the order that we display them as columns is up to us), write the value of the field into it with an echo statement, then close the cell (</td>).

If all goes well ... when you deliver this page from the Deakin web server [using its full URL, such as http://www.deakin.edu.au/~your_user_id/SIT104/ReadDB.php, you should see the web page.