CS 328 - Week 10 Lab Exercise - 2025-04-03/04

Deadline

Due by the end of lab.

Purpose

To practice using OCI from PHP to request data from Oracle.

How to submit

Submit your files using **`~st10/328submit`** on nrs-projects, each time entering a lab number of **90**.

Requirements

- To make sure EACH class member can successfully connect from PHP on nrs-projects to the Oracle student database, this is a rare **individual** lab exercise!
- When you are done, before you leave lab, be sure to submit appropriate versions of these files using **`~st10/328submit`** on nrs-projects, with a lab number of **90**.

Problem 1 - Start your file `328lab10.php`

I have made an **adapted** version of yesterday's **`try-oracle.php`** in a new file, **`328lab10-start.php`**, to serve as the starting point for today's **`328lab10.php`**.

- Go to your desired directory under **`public_html`**, and start your **`328lab10.php`**:

  ```
  cp ~st10/328lab10-start.php 328lab10.php
  ```

- Fill in the opening comment block, putting in your **name**, the last modified **date**, and the **URL** that can be used to run your document.
  - (You *will* lose some credit if this URL does not work when I or the grader paste it into a browser!)
- **Do not include any inline or internal CSS rules in your `328lab10.php`.**
  - (If you are sufficiently annoyed by the default formatting, you may **optionally** add a **`lab10.css`** file to further format your **`328lab10.php`**.)
- **Within** the **`footer`** element near the end of the **`body`** element, add a **`p`** element whose content includes your name.
- **BEFORE YOU GO ON**, run your **`328lab10.php`** and make sure it works! (You should see today's date displayed, and your name in the footer!)

LET ME KNOW if it is not working!

Problem 2 - looping through the results of a query that returns multiple rows

Consider your **`328lab10.php`**.

It connects to Humboldt's Oracle student database using OCI (in a no-end-user-login approach) and queries for today's date.

But, this particular query happened to just return exactly one row.

So, after requesting a database connection (which returns a connection object if successful):

```
$conn = oci_connect(username: $conn_username,
                    password: $ora_php_password,
                    connection_string: null,
                    encoding: 'AL32UTF8',
                    session_mode: OCI_DEFAULT);
```

then requesting the set-up for a desired query statement (which returns a statement object if successful):

```
$date_query_str = "select sysdate
                   from dual";
$date_stmt = oci_parse($conn, $date_query_str);
```

then requesting that the statement object's statement be executed:

```
oci_execute($date_stmt, OCI_DEFAULT);
```

we could request that the first row in that executed query statement be fetched:

```
oci_fetch($date_stmt);
```

and obtain the 1st value in the fetched/current row:

```
<p> Today's date is <?= oci_result($date_stmt, 1) ?> </p>
```

and then, since we have fetched all of the desired rows (since this particular query always returns exactly one row), we are done with this statement, so we request that this statement be freed:

```
oci_free_statement($date_stmt);
```

and, when THIS particular PHP response is finished using the database connection, it explicitly closes that connection using:

```
oci_close($conn);
```

CLASS STYLE STANDARD: a PHP using OCI to connect to the Oracle student database is expected to always explicitly close that connection, using **`oci_close`**, before completing its response.

It was also mentioned that **`oci_fetch`** -- while it does NOT *return* the fetched row -- *does* return a value that is considered "truthy" if there was another row to fetch, and returns a value that is considered "falsey" if there were no more rows in the query result to be fetched.

This means that a **`while`** loop works very well for handling the rows resulting from a query statement:

```
while (oci_fetch($query_stmt))
{
        ...
        ... oci_result($query_stmt,
                       desired_val_from_current_row) ...
}
```

- While **`oci_fetch`** returns a "truthy" value, there has been a row fetched that can have its values obtained using **`oci_result`** in the body of the loop --

- ...and when **`oci_fetch`** returns a "falsey" value, that fetch attempt failed (there were no more rows to fetch), and so exiting the loop at that point is just what we want.

(And you would follow this loop with the appropriate **`oci_free_statement`** call, proceed with any other database actions this PHP is to take to complete its response, and then call **`oci_close`** to close the database connection object before its completes its response.)

Now, do the following:

- Consider the **`empl`** table (from **`set-up-ex-tbls.sql`**) and write a **`select`** statement that selects **at least THREE rows** projecting **at least TWO different columns**.

In your **`328lab10.php`**, **BETWEEN** the statements:

```
                oci_free_statement($date_stmt);
/* here --> */
                oci_close($conn);
```

...ADD the following:

- Declare a string variable **`$empl_select`** that contains a **`select`** statement that selects **at least THREE rows** projecting **at least TWO different columns** from the **`empl`** table.
- Call **`oci_parse`** for that query string, storing the resulting statement object into a variable **`$empl_stmt`**.
- Call **`oci_execute`** to execute this query (to execute this statement).
- Start an HTML **`table`** element, with an appropriate **`caption`** element and a row of appropriate table header elements (with **`scope="col"`**).
- Now LOOP through your query's results, using **`oci_fetch`** and **`oci_result`**, outputting a table row of the resulting employee information for each row in the queried result.
- AFTER the loop:
  - Appropriately end your HTML **`table`** element
  - THEN call **`oci_free_statement`** to free your **`$empl_stmt`**.

(And your **`oci_close($conn);`** statement should be **AFTER** all of the above.)

Run your **`328lab10.php`** -- you should see a table of your query's results.

Strict-validate your **`328lab10.php`**'s result by running it from a browser, viewing its source, copying and pasting that source into a file named **`328lab10.xhtml`**, and put the URL of your **`328lab10..xhtml`** into the validator.

Submit your resulting **`328lab10.php`** and **`328lab10.xhtml`** with a lab number of **90**. (If you created the optional **`lab10.css`**, submit it, also.)