Databases and the Web – Exercise 10

This week we'll enhance the data entry pages built in exercises 5-9 to allow editing data.

Outcomes:

When you've finished this exercise you should:

- Edit data from a single table and update it.
- Edit data drawn from two tables, updating a single table maintaining foreign keys.

Debugging: Because PHP is *server-side* you **must** check your code in the browser through the web server "http://..." *not* by loading the page as "file://"! It is a good idea to:

i. Open the PHP page in the browser (http://localhost/… or http://studentnet/~k*xxxxxxx*/...)

ii. Make small changes to code.

iii. Save the file (upload to StudentNet if necessary).

iv. Refresh the page in the browser.

v. Fix any reported syntax errors.

vi. Test the changes.

Repeat *ad nauseum*. (Get used to using Alt-Tab to cycle through windows ☺)

Task 1: Edit single-table data.

When editing data you are looking to do an UPDATE query, so the information needed is the row's primary key (composite or not) and the updated information – in general it's simplest to make the whole row available for editing (*except* the PK!) and just UPDATE the whole row in the query.

NB: This could be a time-consuming exercise so read all the way through as there are suggestions throughout the text below…

1. In exercise 4 you created pages to list *countries* by *continent*. Let's create a page to allow you to *edit* a country's details.

   a) Modify your *countries* list (**`week4task2.php`**) so that for each country it includes a *link* to a page, *e.g.* called **`countryedit.php`**, that passes to that page using the query string the country's **`Code`** primary key (*e.g.* like this example) so that the key ends up in **`$_GET`**.

      (i) *E.g.* the link that would select New Zealand for editing might be

```
<a href="countryedit.php?code=NZL">New Zealand</a>
```

      or

```
<a href="countryedit.php?code=NZL"
   title="New Zealand">(edit)</a>
```

2. Create **countryedit.php** with login access control from Task 2.

   a) Listing countries need not be secure but *editing* needs to be secured by the login from last week. Make sure the query string gets passed through your successful login redirection.

   b) **countryedit.php** needs to do the following:

      (i) check that the required **Code** key has been supplied;

      - *E.g.* using **preg_match** and a regular expression matching 3 upper-case letters.
      - Redirect or give an error message if not.

      (ii) query the database using that key to retrieve all of the fields from **Country** for the chosen country;

      - For example:

```
$sql = 'SELECT Name,Continent FROM Country WHERE
Code="' . $_GET['Code'] . '"';
$result = mysql_query($sql);
$row = mysql_fetch_assoc($result)
```

      (iii) present the data to the user in an HTML form so that it can be edited;

      - It's easy to do this using use things like

```
<input type="text" value="<?php echo
$row['Name']; ?> name="Name" />
```

      (iv) check for submission and validate edited data (just like the City insert page from exercise 5), presenting unacceptable data for re-editing;

      - Validating the input is more complex – I'd suggest you get editing working first, then consider validating afterwards.
      - Re-presenting the submitted data for editing requires that, for example, you replace **$row['Name']** with **$_GET['Name']** in the form elements.

      (v) submit the updated data to the database when it's OK.

      - Don't show the form, but *do* show the successfully changed data.
      - Remember the format for an UPDATE query! For example:

```
UPDATE Country  SET  Name='Banana',
                     Continent='Pineapple'
                WHERE Code='AMP';
```

   c) The PK should not be editable (this is an *edit* page.)

      - How would you allow the PK to be passed with the form (obviously it needs to go with the edited data!) but prevent the user from editing it?

- If you want to save time you could also make most of the other fields uneditable in the same way, just make sure that the "interesting" but straightforward fields like **`Name`** and **`Continent`** are editable.

d) **`Continent`** is interesting as it's an **`ENUM`** field that needs special treatment to pull out the possible options.

(i) You could populate a drop-down **`SELECT`** box using a "**`SELECT DISTINCT`**" query. (What would happen if a continent that was specified in the **`ENUM`** had no countries?)

(ii) Alternatively, look at the discussion board or MySQL ENUM manual page for some PHP that'll extract the **`ENUM`** names into an array…

e) **`Capital`** would also be an interesting field to manage as it takes a *numeric* index from the **`City`** table … you'd need to list the city names in a list and associate the HTML element with the city ID *but* there are so many cities (over 4000?) that this would be very inefficient. Leave this one uneditable for now!

Hopefully it works – test it!

Task 2: Multi-table editing (advanced!)

Once you have a working edit page (with or without validation) try to expand it so that the Capital city can be edited too. What you need to do is to arrange for a subset of the cities in the **`City`** table to be presented for selection as the **`Capital`**. This is best done with a drop-down box in response to the ***continent*** being updated.

1. Add code to the page so that when it loads it populates a drop-down menu for **`Capital`** using cities from the current **`Continent`**. It's easy enough to do:

   a) Query the database to retrieve all of the city names from the continent.

   - The **`Continent`** corresponds to the **`City`** whose **`Code`** you know (**`$_GET['Code']`**).
   - This can be done using a JOIN based on **`$_GET['Code']`** *or* the **`Continent`** that you know from 2 (ii) above.

   b) Use a loop to populate a **`<select>`** element with **`<option>`** elements that contain the city names and whose values are the city code numbers.

This is currently not dynamic – if the user changes the continent she cannot update the capital ☹

2. To make the page update the list of cities you could use client-side scripting to select from a set of options based on the continent … this provides a nice user experience but does mean you *always* send the full list of cities (all 4000+) whenever the page loads ☹ Instead do it *server-side*:

   a) Your page *already* populates the **`Capital`** drop-down with the current **`Continent`**'s cities (step 1 above) so all you need to do this automatically is:

(i) Arrange for the form to submit whenever the user changes the continent … this requires client-side scripting and the "`onchange`" event from the continent box together with the "`submit`" *method* from the form element.

(ii) How can you prevent the submitted form from being treated as a "finished editing" submission? Both of the following suggestions need more JavaScript:

- Hidden field that is removed when the submit button is pressed?
- Invalidate the `Capital` with a special code from the select's `onchange`?

b) In order to present the same functionality when JavaScript is disabled you need a "submit" button that is specially labelled to update the choice of city when it is used to submit the form rather than the submission indicating that editing is finished … (*e.g.* by examining the *value* of $_POST['submit']

3. That's (probably) it! Now whenever the continent changes:

a) the page self-submits;

b) the server-side code repopulates the form fields from the submitted data;

c) the server-side code retrieves a new cities list for the capital drop-down.

d) Pressing the "submit" button should trigger the data validation process, if present, and then UPDATE the database <phew>