CS 328 - Week 6 Lab Exercise - 2024-02-23

Deadline

Due by the end of lab.

Purpose

To practice writing a PL/SQL trigger.

How to submit

Submit your files for this lab using `~st10/328submit` on nrs-projects, each time entering a lab number of **86**.

Requirements

- You are required to work in **pairs** for this lab exercise.
  - This means **two** people working at **ONE** computer, one typing ("driving"), one saying what to type ("navigating"),

    while **BOTH** are looking at the **shared** computer screen and **discussing** issues along the way.
- Make sure **BOTH** of your names appear in each file submitted.
- When you are done, before you leave lab, somehow e-mail or copy the lab exercise files so that **BOTH** of you have copies, and **BOTH** of you should submit these files using `~st10/328submit` on nrs-projects, with a lab number of **86**.
- You are expected to follow the style standards from the posted "CS 328 SQL and PL/SQL Coding Standards so far" (at https://nrs-projects.humboldt.edu/~st10/s24cs328/328-sql-plsql-coding-standards.pdf).

Lab set-up

- On nrs-projects, if the driver has not previously executed **`set-up-ex-tbls.sql`** in their Oracle account, they should do so, so that they have the tables `empl`, `dept`, and `customer` in their database.
  - If needed, they can get a copy of this script using:

    ```
    cp ~st10/set-up-ex-tbls.sql .   # don't forget the blank and period!
    ```
- In a SQL script **`lab6.sql`**:
  - In opening comment(s), FIRST put the script name, both of your names, and today's date/last modified date.
  - Put in the SQL*Plus command:

    ```
    set serveroutput on
    ```

    ...JUST in case you decide to use `dbms_output.put_line` statements in debugging your trigger.
  - Start spooling to a file **`lab6-out.txt`**:

`spool lab6-out.txt`

...(and make sure you **`spool off`** at the script's end!)

- Put both of your names in a **`prompt`** command.

Problem 1 - create a new table `dept_changes`

An organization has decided it would like to keep track, over time, of its trends in changing department names.

In your script `lab6.sql`, drop and create a table named `dept_changes` that has four attributes:

- `dept_num`, of type `char(3)`
- `change_date`, of type `date`
- `prev_dept_name`, of type `varchar2(15)`
- `next_dept_name`, of type `varchar2(15)`

Also:

- make the pair of attributes `dept_num` and `change_date` its primary key
- define `dept_num` as a foreign key referencing `dept`

(The first time you run this, the `drop table` command should give an error complaining that there is not a table with this name to drop. As long as you do not get this error on subsequent runs of your script, that's fine!)

Problem 2 - trigger `log_name_changes`

In your script `lab6.sql`, now write a PL/SQL trigger `log_name_changes` that meets the following requirements:

- It should fire after each update to the `dept` table, for each row updated.
- *If* the firing update changed a department's name, it should insert a new row into the table `dept_changes` containing:
  - the updated department's department number,
  - the current date (use `sysdate` for this),
  - the previous name of the updated department, and
  - the department name after the update.
  - (If the update changed something else -- for example, the department location -- then this trigger will *not* make any changes to the table `dept_changes`. That is, it will simply do nothing in that case.)
- Look in the posted SQL script **`328lect06-1.sql`** at the version of the trigger **`empl_trig`** that we created during class on Monday.
  - Create an opening comment block for your trigger that has a **`trigger:`** part and **`purpose:`** part in the same style that you see here. (Note that this trigger's purpose is considerably simpler that `empl_trig`'s!)

  - Follow that with the PL/SQL code creating your trigger.
- Remember to follow your PL/SQL trigger with:

  `/`

  `show errors`
- Then put a comment saying you are about to **test** your trigger `log_name_changes`.
- Follow that with:
  - A `commit;` statement to commit the pre-testing version of your database.
  - A `prompt` command noting that these are the pre-test contents of `dept` and `dept_changes`, followed by two `select` statements showing their contents.
  - A `prompt` command noting which department is getting which new name, followed by an `update` statement making that change.
  - A `prompt` command noting which second/different department is getting which new name, followed by an `update` statement making that change.
  - A `prompt` command noting what third/different department is getting a new **location**, followed by an `update` statement making that change.
  - (You may add additional tests if you would like -- precede each with an appropriate descriptive `prompt` command.)
  - A `prompt` command noting that these are the post-test contents of `dept` and `dept_changes`, describing what the changes should be, followed by two `select` statements showing their contents.
  - Finish with a `rollback;` statement to UNDO these changes that were just made for testing purposes.

- Make sure that your `lab6.sql` ends with:

  `spool off`

If successful, your resulting **`lab6-out.txt`** should show that you created the desired new table, that your trigger successfully compiled, and that its tests passed.

BEFORE you leave lab:

Make sure that you **both** have copies of the files:

- **`lab6.sql`** and **`lab6-out.txt`**

...and you BOTH submit these two files using `~st10/328submit` on nrs-projects, with a lab number of **86**.

How the navigator can get files `lab6.sql` and `lab6-out.txt`:

These may be in a directory that is harder for the navigator to make a copy from than `public_html`.

For example -- they might be in a directory `328lab6` that is **not** a sub-directory of `public_html`.

Here is an approach for this:

- The **driver** should *temporarily* make the directory with these files world-readable and -executable, and these files world-readable -- assuming the driver is currently in their directory `328lab6`:

  ```
  chmod 755 .   # notice the space and the period!
  chmod 644 lab6.sql lab6-out.txt
  ```

- Now the **navigator** can copy these into a directory of their choice -- assuming the driver's username is `ab12` and the navigator is within the directory they want to copy into:

  ```
  cp ~ab12/328lab6/* .  # notice the space and the period!
  ```

- The **driver** and **navigator** should **BOTH** then **protect** these files:

  ```
  chmod 600 lab6.sql lab6-out.txt
  ```

  ...and both can protect the directory containing them:

  ```
  chmod 700 .   # notice the space and the period!
  ```

  ...and **both** can now submit these using `~st10/328submit` from the directory containing these files.