Exercices for Chapter 11: Introduction to Cursors

Try It Yourself

The projects in this section are meant to have you use all of the skills that you have acquired throughout this chapter. Here are some exercises that will help you test the depth of your understanding.

In this chapter, you learned how to process data with a cursor. Additionally, you learned how to simplify the code by using a cursor `FOR LOOP`. You also encountered the more complex example of nesting cursors within cursors.

1) Write a nested cursor where the parent cursor calls information about each section of a course. The child cursor counts the enrollment. The only output is one line for each course with the Course Name and Section Number and the total enrollment.

Answer: The script should look similar to the following:

```
SET SERVEROUTPUT ON
DECLARE
   CURSOR c_course IS
      SELECT course_no, description
        FROM course
       WHERE course_no < 120;

   CURSOR c_enrollment(p_course_no IN course.course_no%TYPE)
   IS
      SELECT s.section_no section_no, count(*) count
        FROM section s, enrollment e
       WHERE s.course_no = p_course_no
         AND s.section_id = e.section_id
       GROUP BY s.section_no;
BEGIN
   FOR r_course IN c_course LOOP
      DBMS_OUTPUT.PUT_LINE
         (r_course.course_no||' '|| r_course.description);

      FOR r_enroll IN c_enrollment(r_course.course_no) LOOP
         DBMS_OUTPUT.PUT_LINE
```

```
           (Chr(9)||'Section:  '||r_enroll.section_no||
            ' has an enrollment of: '||r_enroll.count);
        END LOOP;

     END LOOP;
  END;
```

2) Write an anonymous PL/SQL block that finds all the courses that have at least one section that is at its maximum enrollment. If there are no courses that meet that criterion, then pick two courses and create that situation for each.
   a. For each of those courses, add another section. The instructor for the new section should be taken from the existing records in the instruct table. Use the instructor who is signed up to teach the least number of courses. Handle the fact that, during the execution of your program, the instructor teaching the most courses may change.
   b. Use any exception-handling techniques you think are useful to capture error conditions.

3) In order to calculate the area of a circle, the circle's radius must be squared and then multiplied by π. Write a program that calculates the area of a circle. The value for the radius should be provided with the help of a substitution variable. Use 3.14 for the value of π. Once the area of the circle is calculated, display it on the screen.

Answer: The script should look similar to the following:

```
SET SERVEROUTPUT ON
DECLARE
   v_instid_min     instructor.instructor_id%TYPE;
   v_section_id_new section.section_id%TYPE;
   v_snumber_recent section.section_no%TYPE := 0;

   -- This cursor determines the courses that have at least
   -- one section filled to capacity.
   CURSOR c_filled IS
      SELECT DISTINCT s.course_no
        FROM section s
       WHERE s.capacity = (SELECT COUNT(section_id)
                             FROM enrollment e
                            WHERE e.section_id = s.section_id);
BEGIN
   FOR r_filled IN c_filled LOOP
      -- For each course in this list, add another section.
      -- First, determine the instructor who is teaching
      -- the least number of courses. If there are more
      -- than one instructor teaching the same number of
      -- minimum courses (e.g. if there are three
      -- instructors teaching 1 course) use any of those
      -- instructors.
      SELECT instructor_id
        INTO v_instid_min
        FROM instructor
       WHERE EXISTS (SELECT NULL
                       FROM section
                      WHERE section.instructor_id =
                            instructor.instructor_id
                     GROUP BY instructor_id
```

```
                    HAVING COUNT(*) =
                      (SELECT MIN(COUNT(*))
                         FROM section
                        WHERE instructor_id IS NOT NULL
                       GROUP BY instructor_id)
                   )
            AND ROWNUM = 1;

         -- Determine the section_id for the new section
         -- Note that this method would not work in a multi-user
         -- environment. A sequence should be used instead.
         SELECT MAX(section_id) + 1
           INTO v_section_id_new
           FROM section;

         -- Determine the section number for the new section
         -- This only needs to be done in the real world if
         -- the system specification calls for a sequence in
         -- a parent. The sequence in parent here refers to
         -- the section_no incrementing within the course_no,
         -- and not the section_no incrementing within
         -- the section_id.
         DECLARE
            CURSOR c_snumber_in_parent IS
               SELECT section_no
                 FROM section
                WHERE course_no = r_filled.course_no
               ORDER BY section_no;
         BEGIN
            -- Go from the lowest to the highest section_no
            -- and find any gaps. If there are no gaps make
            -- the new section_no equal to the highest
            -- current section_no + 1.

            FOR r_snumber_in_parent IN c_snumber_in_parent LOOP
               EXIT WHEN
                  r_snumber_in_parent.section_no > v_snumber_recent + 1;
                  v_snumber_recent := r_snumber_in_parent.section_no + 1;
            END LOOP;

            -- At this point, v_snumber_recent will be equal
            -- either to the value preceeding the gap or to
            -- the highest section_no for that course.
         END;
         -- Do the insert.
         INSERT INTO section
           (section_id, course_no, section_no, instructor_id)
         VALUES
           (v_section_id_new, r_filled.course_no, v_snumber_recent,
            v_instid_min);
         COMMIT;
      END LOOP;
   EXCEPTION
      WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE ('An error has occurred');
END;
```