# *Database Management II (420-D20-HR)*

# *Lab 2 – PL/SQL Cursors*

Date assigned: Tuesday, January 31, 2017

Date due: **Tuesday, January 31, 2017, 17:50**

**Objectives:**

At the end of this lab you will be able to:

* Define user-defined record types
* use %ROWTYPE to define a record variable;
* declare, open and close a cursor;
* use an explicit cursor to retrieve multiple rows from a table;
* use a CURSOR FOR loop
* use a while loop

**References:**

Class notes and examples (S04, S05)

**Marking and Time management:**

|  |  |  |  |
| --- | --- | --- | --- |
| Section | Question | Mark | Effort (minutes) |
| 1. Records | 1 | 18 | 0 |
|  | 2 | 14 | 0 |
| 1. Explicit Cursors | 1 | 4 | 0 |
|  | 2 | 17 | 0 |
| 1. Cursor For | 1 | 6 | 0 |
|  | 2 | 7 | 0 |
| 1. Cursor WHILE | 1 | 20 | 0 |
| English and handed in properly |  | 5 | 0 |
| Totals |  | 91 | 0 |

**To Start:**

1. Rename this document to ***username*\_D20\_L02\_Cursors.docx**
2. Start **SQL Developer** and connect to your Oracle account.

**To be handed in:**

1. ***username\_*D20\_L02\_Cursor.docx** containing your responses created for this lab should uploaded to **Moodle**. Please remember to fill in the Effort section of the Marking Rubric.

**For each of the blocks created for the following problems, include a comment block at the top of the block with the question number, your name and a brief description of what the block does. Insert comments throughout to explain the steps.**

**Use ISO/ANSI standards joins for all joins.**

**Name all identifiers according to the naming standards shown below.**

**Format all blocks using the SQL Developer Formatter.**

**Provide sample output.**

**Naming Standards:**

|  |  |
| --- | --- |
| **Identifier type** | **Prefix** |
| local variables | lv\_ |
| local constant | lc\_ |
| local record | lrec\_ |
| local cursor | lcur\_ |

# Records

## Write a PL/SQL block that displays the course section id, the term name, the course number and title, the course teacher and the number of students for a given course section.

Hint: take a look at Example 01 for slides S04 for an example of a user-defined record.

### Define a record type that contains fields for all the columns retrieved in the query. Course number and title should be combined in one field.

### Use the record type you declared to declare a locally defined record.

### Retrieve the data into your record in your select statement.

### Issue an error message if the CSID is invalid.

**Sample output – valid csid:**

Course section: 1206 Term: Spring 2003

Course: CIS265 - Systems Analysis

Teacher: Professor Jones

Number of students: 3

**Sample output – invalid csid:**

ERROR: 2222 is an invalid course section id.

**SQL:**

**DECLARE**

**lv\_csid IU\_CRSSECTION.CSID%TYPE;**

**lv\_csid\_exists number := 0;**

**lv\_term IU\_TERM.TERMDESC%TYPE;**

**lv\_courseid IU\_CRSSECTION.COURSEID%TYPE;**

**lv\_courseName IU\_COURSE.TITLE%TYPE;**

**lv\_teacher IU\_FACULTY.NAME%TYPE;**

**lv\_numStudents NUMBER;**

**BEGIN**

**lv\_csid := &courseSectionId;**

**SELECT count(\*) INTO lv\_csid\_exists**

**FROM iu\_crssection**

**WHERE csid = lv\_csid;**

**if (lv\_csid\_exists = 0)**

**THEN**

**DBMS\_OUTPUT.PUT\_LINE('The CSID does not exist');**

**ELSE**

**SELECT t.termdesc, cs.courseid, c.title, f.name**

**INTO lv\_term, lv\_courseid, lv\_courseName, lv\_teacher**

**FROM iu\_crssection cs, iu\_course c, iu\_faculty f, iu\_term t**

**WHERE cs.csid = lv\_csid --1101**

**AND cs.facultyid = f.facultyid**

**AND t.termid = cs.termid**

**AND c.courseid = cs.courseid;**

**SELECT count(studentid) INTO lv\_numStudents**

**FROM iu\_registration**

**WHERE csid = lv\_csid;**

**DBMS\_OUTPUT.PUT\_LINE('Course Sectiom: ' || lv\_csid || ' Term: ' || lv\_term);**

**DBMS\_OUTPUT.PUT\_LINE('Course: ' || lv\_courseid || ' - ' || lv\_coursename);**

**DBMS\_OUTPUT.PUT\_LINE('Teacher: Professor ' || lv\_teacher);**

**DBMS\_OUTPUT.PUT\_LINE('Number of students: ' || lv\_numStudents);**

**END IF;**

**END;**

**Sample output:**

Course Sectiom: 1206 Term: Spring 2003

Course: CIS265 - Systems Analysis

Teacher: Professor Jones

Number of students: 3

## Write a PL/SQL block that uses %ROWTYPE to read a record for a course whose courseid is entered at run-time. It should print the course number, title, pre-requisite number and the number of sections of the course that have been offered. If there is no prerequisite, state that there is no prerequisite. If the course number is invalid, it should print an error message.

**Sample output – prerequisite exists:**

Course: CIS265 Systems Analysis

Prerequisite: CIS253

Number of sections: 3

**Sample output – no prerequisite exists:**

Course: CIS253 Database Systems

No prerequisite

Number of sections: 4

**SQL:**

**DECLARE**

**lrec\_course IU\_COURSE%ROWTYPE;**

**lv\_course\_e NUMBER := 0;**

**lv\_courseid IU\_COURSE.courseid%TYPE;**

**lv\_courseName IU\_COURSE.title%TYPE;**

**lv\_prereqid IU\_COURSE.prereq%TYPE;**

**lv\_sections NUMBER;**

**BEGIN**

**lv\_courseid := '&courseid';**

**SELECT count(\*) INTO lv\_course\_e**

**FROM iu\_course**

**WHERE courseid = lv\_courseid;**

**IF lv\_course\_e = 1**

**THEN**

**SELECT \* INTO lrec\_course**

**FROM iu\_course**

**WHERE courseid = lv\_courseid;**

**SELECT count(csid) INTO lv\_sections**

**FROM iu\_crssection**

**WHERE courseid = lv\_courseid;**

**lv\_prereqid := lrec\_course.prereq;**

**lv\_coursename := lrec\_course.title;**

**DBMS\_OUTPUT.PUT\_LINE('Course: ' || lv\_courseid || ' ' ||lv\_courseName);**

**IF lv\_prereqid IS NULL**

**THEN DBMS\_OUTPUT.PUT\_LINE('No Prerequisits');**

**ELSE DBMS\_OUTPUT.PUT\_LINE('Prerequisite: ' || lv\_prereqid);**

**END IF;**

**DBMS\_OUTPUT.PUT\_LINE('Number of sections: ' || lv\_sections);**

**else**

**DBMS\_OUTPUT.PUT\_LINE('The course does not exist');**

**end if;**

**END;**

**Sample output:**

Course: EN100 Basic English

No Prerequisits

Number of sections: 0

# Explicit Cursors

***Purpose:*** Learn to declare, open, fetch and close a cursor

Learn to use an explicit cursor to retrieve multiple rows from a table in a basic loop

***To Do:***

## Consider the following block that is supposed to list the name and phone of all students living in Ontario.

**DECLARE**

lv\_student\_name **VARCHAR2** (25);

lv\_phone **CHAR** (13);

**BEGIN**

**SELECT** s.**LAST** || ', ' || s.**FIRST**,

'('

|| **SUBSTR** (s.phone, 1, 3)

|| ')'

|| **SUBSTR** (s.phone, 4, 3)

|| '-'

|| **SUBSTR** (s.phone, 7)

**INTO** lv\_student\_name,

lv\_phone

**FROM** iu\_student s

**WHERE** s.province = 'ON';

**DBMS\_OUTPUT**.put\_line (lv\_student\_name || ' ' || lv\_phone);

**END**;

What happens when you run it? Remember that you can only return one row from a SELECT statement in PL/SQL.

To retrieve more than one row, you must use a cursor. There are 4 steps to using a cursor: To use a cursor:

* 1. Declare the cursor
  2. Open the cursor
  3. Fetch the data from the cursor into local variables
  4. Close the cursor

**1. Declare the cursor**

Add the following declaration to the **declare** section to declare a cursor to hold all the rows retrieved from the table. (Notice that the select the statement is exactly the same as the one above without the INTO clause.)

**CURSOR** lcur\_student

**IS**

**SELECT LAST** || ', ' || **FIRST** student\_name,

'(' || **SUBSTR** (phone, 1, 3) || ')'

|| **SUBSTR** (phone, 4, 3) || '-'

|| **SUBSTR** (phone, 7) student\_phone

**FROM** iu\_student

**WHERE** province = 'ON';

**2. Open the cursor**

Add the following line after the begin statement:

**OPEN** lcur\_student;

**3. Fetch the data from the cursor into local variables**

Replace the SELECT statement in the block body with the following:

**FETCH** lcur\_student

**INTO** lv\_student\_name, lv\_phone;

**4. Close the cursor**

Add the following line before the end statement:

**CLOSE** lcur\_student;

Run the block now. What happens?

**Your answer:**

Tyler, Mickey (718)555-2222

To see more than one row, you have to use a loop:

### Add a loop statement after opening the cursor and before doing the fetch.

### Use the %NOTFOUND cursor attribute to check whether or not the cursor contained any more rows after doing the fetch and exit it there are no more rows:

**EXIT WHEN** lcur\_student%**NOTFOUND**;

### End the loop after displaying the student data.

Run the block now. What happens?

**Your answer:**

Tyler, Mickey (718)555-2222

Lee, Brian (212)555-5555

You have seen how the %NOTFOUND cursor attribute is used with an explicit cursor. To see how the %ROWCOUNT attribute works add the following output statements at appropriate locations.

**DBMS\_OUTPUT**.put\_line ('After open, rowcount is '

|| lcur\_student%**ROWCOUNT** );

**DBMS\_OUTPUT**.put\_line ('After fetch, rowcount is '

|| lcur\_student%**ROWCOUNT** );

**DBMS\_OUTPUT**.put\_line ('Before close, rowcount is '

|| lcur\_student%**ROWCOUNT** );

Save the block completed block below:

**SQL:**

**DECLARE**

**lv\_student\_name VARCHAR2 (25);**

**lv\_phone CHAR (13);**

**CURSOR lcur\_student**

**IS**

**SELECT LAST ||', '|| FIRST student\_name,**

**'('|| SUBSTR(phone, 1, 3) ||')'|| SUBSTR(phone, 4, 3) ||'-'|| SUBSTR(phone, 7) student\_phone**

**FROM iu\_student**

**WHERE province = 'ON';**

**BEGIN**

**OPEN lcur\_student;**

**DBMS\_OUTPUT.put\_line ('After open, rowcount is ' || lcur\_student%ROWCOUNT );**

**LOOP**

**FETCH lcur\_student**

**INTO lv\_student\_name, lv\_phone;**

**DBMS\_OUTPUT.put\_line ('After fetch, rowcount is '|| lcur\_student%ROWCOUNT );**

**EXIT WHEN lcur\_student%NOTFOUND;**

**DBMS\_OUTPUT.put\_line (lv\_student\_name || ' ' || lv\_phone);**

**END LOOP;**

**DBMS\_OUTPUT.put\_line ('Before close, rowcount is '|| lcur\_student%ROWCOUNT );**

**CLOSE lcur\_student;**

**END;**

**Sample output:**

After open, rowcount is 0

After fetch, rowcount is 1

Tyler, Mickey (718)555-2222

After fetch, rowcount is 2

Lee, Brian (212)555-5555

After fetch, rowcount is 2

Before close, rowcount is 2

## Create a PL/SQL block to declare a cursor to select last name, first name, salary, and hire date from the EMPLOYEE table. Retrieve each row from the cursor and print the employee’s information if the employee’s salary is greater than $50,000 and the hire date is before 31-Dec-1997. Use an explicit cursor with **a basic loop** (loop/ end loop). The output should look like:

John Smith was hired on 15-Apr-1960 and has a salary of $265,000

Larry Houston was hired on 19-May-1967 and has a salary of $150,000

Sandi Roberts was hired on 02-Dec-1991 and has a salary of $75,000

Alex McCall was hired on 10-May-1997 and has a salary of $66,500

Derek Dev was hired on 15-Mar-1995 and has a salary of $80,000

**SQL:**

**DECLARE**

**lv\_fname nn\_employee.fname%TYPE;**

**lv\_lname nn\_employee.lname%TYPE;**

**lv\_hire nn\_employee.hiredate%TYPE;**

**lv\_salary nn\_employee.salary%TYPE;**

**CURSOR lcur\_stuff**

**IS SELECT fname, lname, hiredate, salary**

**FROM nn\_employee;**

**BEGIN**

**OPEN lcur\_stuff;**

**LOOP**

**FETCH lcur\_stuff**

**INTO lv\_fname, lv\_lname, lv\_hire, lv\_salary;**

**EXIT WHEN lcur\_stuff%NOTFOUND;**

**IF lv\_salary > 50000 and lv\_hire < to\_date('31-Dec-1997', 'dd-Month-YYYY')**

**THEN**

**DBMS\_OUTPUT.PUT\_LINE(lv\_fname || ' ' || lv\_lname**

**|| ' was hired on ' || to\_date(lv\_hire, 'dd-Month-YY')**

**|| ' and has a salary of ' || to\_char(lv\_salary, '$999,999'));**

**END IF;**

**END LOOP;**

**CLOSE lcur\_stuff;**

**END;**

**Sample output:**

John Smith was hired on 15-APR-60 and has a salary of $265,000

Larry Houston was hired on 19-MAY-67 and has a salary of $150,000

Sandi Roberts was hired on 02-DEC-91 and has a salary of $75,000

Alex McCall was hired on 10-MAY-97 and has a salary of $66,500

Derek Dev was hired on 15-MAR-95 and has a salary of $80,000

# Cursor FOR Loop

***Purpose:*** Learn to use a cursor for loop to retrieve multiple rows from a table

***To Do:***

## Create a variant of your solution to B2 that uses an explicit cursor (i.e. declare a cursor) in a for loop.

**SQL:**

**DECLARE**

**CURSOR lcur\_stuff**

**IS SELECT fname, lname, hiredate, salary**

**FROM nn\_employee;**

**lrec\_emp lcur\_stuff%ROWTYPE;**

**BEGIN**

**FOR lrec\_emp IN lcur\_stuff**

**LOOP**

**IF lrec\_emp.salary > 50000 and lrec\_emp.hiredate < to\_date('31-Dec-1997', 'dd-Month-YYYY')**

**THEN**

**DBMS\_OUTPUT.PUT\_LINE(lrec\_emp.fname || ' ' || lrec\_emp.lname**

**|| ' was hired on ' || to\_date(lrec\_emp.hiredate, 'dd-Month-YY')**

**|| ' and has a salary of ' || to\_char(lrec\_emp.salary, '$999,999'));**

**END IF;**

**END LOOP;**

**END;**

**Sample output:**

John Smith was hired on 15-APR-60 and has a salary of $265,000

Larry Houston was hired on 19-MAY-67 and has a salary of $150,000

Sandi Roberts was hired on 02-DEC-91 and has a salary of $75,000

Alex McCall was hired on 10-MAY-97 and has a salary of $66,500

Derek Dev was hired on 15-MAR-95 and has a salary of $80,000

## Write an anonymous block that uses an explicit cursor in a for loop to list the department names with the manager name. The output should look like:

Department: 10 Finance Manager: Sandi Roberts

Department: 20 InfoSys Manager: Derek Dev

Department: 30 Sales Manager: Stanley Garner

Department: 40 Marketing Manager: Larry Houston

**SQL:**

**DECLARE**

**lv\_mfname nn\_employee.fname%TYPE;**

**lv\_mlname nn\_employee.lname%TYPE;**

**CURSOR lcur\_dept**

**IS SELECT deptid, deptname, employeeid FROM nn\_dept;**

**lrec\_dept lcur\_dept%ROWTYPE;**

**BEGIN**

**FOR lrec\_dept IN lcur\_dept**

**LOOP**

**SELECT fname, lname INTO lv\_mfname, lv\_mlname**

**FROM nn\_employee**

**WHERE employeeid = lrec\_dept.employeeid;**

**DBMS\_OUTPUT.PUT\_LINE('Department: ' || lrec\_dept.deptid**

**|| ' ' || lrec\_dept.deptname || ' Manager: '**

**|| lv\_mfname || ' ' || lv\_mlname);**

**END LOOP;**

**END;**

**Sample output:**

Department: 10 Finance Manager: Sandi Roberts

Department: 20 InfoSys Manager: Derek Dev

Department: 30 Sales Manager: Stanley Garner

Department: 40 Marketing Manager: Larry Houston

# Cursor While loop

***Purpose:***

* Learn the syntax of a WHILE loop in PL/SQL
* Learn how to use of a cursor with a WHILE loop
* Learn how to do simple math in PL/SQL

***To Do:***

## Using a basic explicit cursor (declare, open, fetch, close) and a while loop, list all employees in the NN\_EMPLOYEE table sorted by employee’s last name. For each employee, show first name, last name, and total compensation. After the list of employees is output, display a total for the company wages (total compensation).

Hint: Total compensation for an employee is salary + commission

Output should look like

Sunny Chen earns $35,000

Derek Dev earns $100,000

Stanley Garner earns $50,000

Larry Houston earns $160,000

Alex McCall earns $66,500

Sandi Roberts earns $75,000

Jinku Shaw earns $27,500

John Smith earns $300,000

---------------------------------------------------

Company-wide compensation $814,000

**SQL:**

**DECLARE**

**lv\_total number:=0;**

**CURSOR lcur\_emp**

**IS SELECT fname, lname, salary, commission**

**FROM nn\_employee**

**ORDER BY lname;**

**lrec\_emp lcur\_emp%ROWTYPE;**

**BEGIN**

**OPEN lcur\_emp;**

**FETCH lcur\_emp INTO lrec\_emp;**

**WHILE lcur\_emp%FOUND LOOP**

**lv\_total := lv\_total + lrec\_emp.salary + nvl(lrec\_emp.commission, 0);**

**DBMS\_OUTPUT.PUT\_LINE(lrec\_emp.fname || ' ' || lrec\_emp.lname || ' earns '**

**|| to\_char(lrec\_emp.salary + nvl(lrec\_emp.commission, 0), '$999,999'));**

**FETCH lcur\_emp INTO lrec\_emp;**

**END LOOP;**

**DBMS\_OUTPUT.PUT\_LINE('-----------------------------------------');**

**DBMS\_OUTPUT.PUT\_LINE('Company-Wide Compensation' || to\_char(lv\_total, '$999,999'));**

**CLOSE lcur\_emp;**

**END;**

**Sample output:**

Sunny Chen earns $35,000

Derek Dev earns $100,000

Stanley Garner earns $50,000

Larry Houston earns $160,000

Alex McCall earns $66,500

Sandi Roberts earns $75,000

Jinku Shaw earns $27,500

John Smith earns $300,000

-----------------------------------------

Company-Wide Compensation $814,000