Lab 2: PL/SQL Programming

1. Implicit Cursor in PL/SQL: Use Implicit cursor to fetch ONLY one record from table.PL/SQL uses Anchored Declaration %TYPE attribute to anchor a variable’s data type. A variable gets the same data type as an existing one.

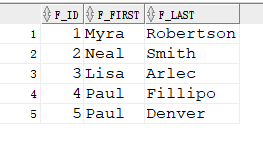
a) Program PL/SQL # 1: Edit PL/SQL program using Anchored Declaration: Very useful when you want to cross-reference the data type used in the table.

Registration Database Narrative: Students taking a set of courses, getting grades for a given Course, and each Student being supervised by one Faculty member.Execute the script Registration.sql in order to read Faculty table using PL/SQL programming

1. --1. Implicit Cursor in PL/SQL
2. **DECLARE**
3. current\_f\_last      faculty.f\_last%TYPE;
4. current\_f\_first     faculty.f\_first%TYPE;
5. **BEGIN**
6. **select** f\_last, f\_first
7. **into** current\_f\_last, current\_f\_first
8. **from** faculty
9. **where** f\_id = 2;
10. dbms\_output.put\_line('The faculty member''s name is '
11. || current\_f\_first || ' ' || current\_f\_last);
12. **END**;

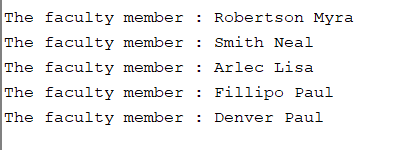


1. **select** f\_id, f\_first, f\_last **from** faculty;



2. Explicit Cursor: Retrieval of more than one row using SELECT in a given PL/SQL program.

1. --2. Explicit Cursor:
2. **DECLARE**
3. **cursor** faculty\_cursor **is**
4. **select** f\_id, f\_last, f\_first
5. **from** faculty;
6. faculty\_row faculty\_cursor%ROWTYPE;
7. **BEGIN**
8. **open** faculty\_cursor;
9. loop
10. **fetch** faculty\_cursor **into** faculty\_row;
11. exit **when** faculty\_cursor%NOTFOUND;
12. dbms\_output.put\_line('The faculty member : ' ||
13. faculty\_row.f\_last || ' ' || faculty\_row.f\_first);
14. **end** loop;
15. **close** faculty\_cursor;
16. **end**;



3. Execute the script file Registration.sql for creating tables of Registration System.a) Edit PL/SQL program using an implicit cursor to display the course number MIS 441 as shown in Figure 1. Use Anchored Declaration %TYPE.

1. **DECLARE**
2. c\_course\_name       course.course\_name%TYPE;
3. c\_credits           course.credits%TYPE;
4. c\_max\_enrollment    course.max\_enrl%TYPE;
5. **BEGIN**
6. **select** course\_name, credits, max\_enrl
7. **into** c\_course\_name, c\_credits, c\_max\_enrollment
8. **from** course
9. **where** course\_no = 'MIS 441';
10. dbms\_output.put\_line('The Course is ' || c\_course\_name
11. || ' with credits ' || c\_credits
12. || ' and max enrollment is ' || c\_max\_enrollment);
13. **END**;



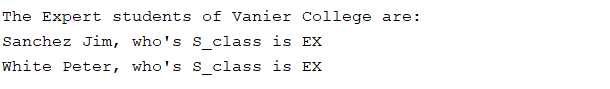
b) Edit PL/SQL program to calculate the number of records within faculty table, and display the output as shown in Figure 2.

1. **DECLARE**
2. **cursor** faculty\_cursor **is**
3. **select** f\_id
4. **from** faculty;
5. faculty\_row faculty\_cursor%ROWTYPE;
6. num     number :=0;
7. **BEGIN**
8. **open** faculty\_cursor;
9. loop
10. **fetch** faculty\_cursor **into** faculty\_row;
11. exit **when** faculty\_cursor%NOTFOUND;
12. num := num + 1;
13. **end** loop;
14. dbms\_output.put\_line('The Nember of records whithin faculty table is: '
15. || num);
16. **close** faculty\_cursor;
17. **end**;



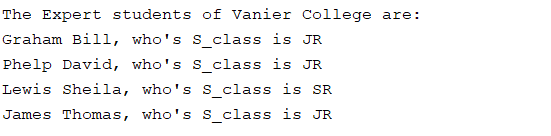
c) Write a PL/SQL program to calculate the number of records within student table, who’s considered as being registered as Expert (s\_class = ‘EX’), and display the output as shown in Figure 3. Use %ROWTYPE variable to display explicit cursor values

1. **DECLARE**
2. **cursor** student\_cursor **is**
3. **select** s\_id, s\_last, s\_first, s\_class
4. **from** student;
5. student\_row student\_cursor%ROWTYPE;
6. **BEGIN**
7. **open** student\_cursor;
8. dbms\_output.put\_line('The Expert students of Vanier College are:');
9. loop
10. **fetch** student\_cursor **into** student\_row;
11. exit **when** student\_cursor%NOTFOUND;
12. if student\_row.s\_class = 'EX' **then**
13. dbms\_output.put\_line(student\_row.s\_last || ' '
14. || student\_row.s\_first
15. || ', who''s S\_class is ' || student\_row.s\_class);
16. **end** if;
17. **end** loop;
18. **close** student\_cursor;
19. **end**;



d) Re-Write the same program of (c) taking into account to display all students registered as J**R** (Junior) and S**R** (Senior). You have to consider only the (**R**) of s\_class field as search value in WHERE clause statement within the student table. Display the output of Figure 4

1. **DECLARE**
2. **cursor** student\_cursor **is**
3. **select** s\_id, s\_last, s\_first, s\_class
4. **from** student;
5. student\_row student\_cursor%ROWTYPE;
6. **BEGIN**
7. **open** student\_cursor;
8. dbms\_output.put\_line('The Expert students of Vanier College are:');
9. loop
10. **fetch** student\_cursor **into** student\_row;
11. exit **when** student\_cursor%NOTFOUND;
12. if substr(student\_row.s\_class,-1) = 'R' **then**
13. dbms\_output.put\_line(student\_row.s\_last || ' '
14. || student\_row.s\_first
15. || ', who''s S\_class is ' || student\_row.s\_class);
16. **end** if;
17. **end** loop;
18. **close** student\_cursor;
19. **end**;



4. Answer the following questions:

1. Write the appropriate statement to declare an anchored variable which has the same data type as hos\_num\_surgery (number of surgeries / year) field in the following Hospital table.

Hospital

hos\_id NUMBER(6)

hos\_Name VARCHAR2(30)

hos\_addr VARCHAR2 (30)

hos\_num\_surgery NUMBER (8)

1. **DECLARE**
2. hos\_num\_surgery      Hospital.hos\_num\_surgery%TYPE;

2. Write the appropriate statement to declare an anchored variable which has the same data type as hos\_Name field in the above Hospital table.

1. **DECLARE**
2. hos\_department\_name      Hospital.hos\_Name%TYPE;

3. Write the appropriate statement to declare an anchored variable which has the same data type as hos\_Name field in the above Hospital table.

1. **DECLARE**
2. hos\_building\_name        Hospital.hos\_Name%TYPE;

4. Write the appropriate statement (s) that uses an implicit cursor in order to search for a given hospital where hos\_id = 555

1. **DECLARE**
2. current\_hos\_Name            Hospital.hos\_Name%TYPE;
3. current\_hos\_addr            Hospital.hos\_addr%TYPE;
4. current\_hos\_num\_surgery     Hospital.hos\_num\_surgery%TYPE;
5. **BEGIN**
6. **select** hos\_Name, hos\_addr, hos\_num\_surgery
7. **into** current\_hos\_Name, current\_hos\_addr, current\_hos\_num\_surgery
8. **from** Hospital
9. **where** hos\_id = 555;
10. dbms\_output.put\_line('The Hospital is '
11. || current\_hos\_Name || ' '
12. || current\_hos\_addr || ' '
13. || current\_hos\_num\_surgery);
14. **END**;

5. Write the appropriate statement to declare an explicit cursor which stores the fields (hos\_id, hos\_Name, hos\_num\_surgery) in the above Hospital table.

1. **DECLARE**
2. **cursor** hospital\_cursor **is**
3. **select** hos\_id, hos\_Name, hos\_addr, hos\_num\_surgery
4. **from** Hospital;
5. hospital\_row hospital\_cursor%ROWTYPE;

6. Write the appropriate statement (s) to open the declared cursor in (5) and skip through all records stored in it to display the values of fields (hos\_id, hos\_Name, hos\_num\_surgery).

1. **open** hospital\_cursor;
2. loop
3. **fetch** hospital\_cursor **into** hospital\_row;
4. exit **when** hospital\_cursor%NOTFOUND;
5. dbms\_output.put\_line(hospital\_row.hos\_id || ' '
6. || hospital\_row.hos\_Name || ' '
7. || hospital\_row.hos\_num\_surgery);
8. **end** loop;

7. Write the appropriate statement to close the declared cursor in (5).

1. **close** hospital\_cursor;

8. Give an example of two tables (Parent & Child tables) which has 1:M relationship, declare explicit cursor for each table including at least three fields for each table.

Publishers

Books

1. **CREATE** **TABLE** Publishers(
2. p\_id          number not null,
3. p\_name        varchar2(60),
4. p\_establish   **date**,
5. p\_adress      varchar2(80),
6. **CONSTRAINT** pk\_publisher\_id **PRIMARY** **KEY** (p\_id)
7. );
9. **CREATE** **TABLE** Books(
10. b\_id         number not null,
11. b\_name       varchar2(60),
12. b\_author     **date**,
13. b\_category   varchar2(80),
14. **CONSTRAINT** fk\_book\_id **FOREIGN** **KEY** (p\_id)
15. **REFERENCES** Publishers(p\_id)
16. );
18. --Declare explicit cursor publisher\_cursor to skip through all records
19. --to display the values of fields (p\_id, p\_name, p\_establish, p\_adress).
20. **DECLARE**
21. **cursor** publisher\_cursor **is**
22. **select** p\_id, p\_name, p\_establish, p\_adress
23. **from** Publishers;
24. p\_row publisher\_cursor%ROWTYPE;
25. **BEGIN**
26. **open** publisher\_cursor;
27. loop
28. **fetch** publisher\_cursor **into** p\_row;
29. exit **when** publisher\_cursor%NOTFOUND;
30. dbms\_output.put\_line(p\_row.p\_id || ' ' || p\_row.p\_name || ' '
31. || p\_row.p\_establish || ' ' ||  p\_row.p\_adress);
32. **end** loop;
33. **close** publisher\_cursor;
34. **end**;
36. --Declare explicit cursor book\_cursor to skip through all records
37. --to display the values of fields (b\_id, b\_name, b\_author, b\_category).
38. **DECLARE**
39. **cursor** book\_cursor **is**
40. **select** b\_id, b\_name, b\_author, b\_category
41. **from** Books;
42. b\_row book\_cursor%ROWTYPE;
43. **BEGIN**
44. **open** book\_cursor;
45. loop
46. **fetch** book\_cursor **into** b\_row;
47. exit **when** book\_cursor%NOTFOUND;
48. dbms\_output.put\_line(b\_row.b\_id || ' ' || b\_row.b\_name || ' '
49. || b\_row.b\_author || ' ' ||  b\_row.b\_category);
50. **end** loop;
51. **close** book\_cursor;
52. **end**;

9. True or false and why:

a. You cannot use implicit cursor to fetch multiple records from table.

**True.**

The implicit cursor fetch only one record INTO the Variables one time unless use loop to display the multiple records.

b. Explicit cursor is declared only in BEGIN block.

**False**

Explicit cursor is declared in DECLARE block before BEGIN block.

c. Explicit cursor could be used to fetch one record from table.

**True**

We could use the condition to judge records if the explicit cursor could fetch only one record which meet the condition, or only one record in the Table.

d. Implicit cursor is used to fetch only one record from table.

**True.**

Implicit cursor fetches only one record from table once.

e. Anchored variables that uses %TYPE could be used within explicit cursor.

**False.**

To declare the cursor that row variable uses %ROWTYPE.

For example:

faculty\_row faculty\_cursor%ROWTYPE;

f. The following is a valid statement

Hospital row hospital\_cursor%ROWTYPE;

**False.**

We should declare the variable name without class name.

For example:

hospital\_row hospital\_cursor%ROWTYPE;

10. True or false and why:

a. Foreign key values are unique within a given table.

**True**

Foreign key values must be unique for querying in the given table.

b. Foreign keys are referential fields linked to fields in parent tables.

**False**

A Foreign Key is a database key that is used to link two tables together. The FOREIGN KEY constraint identifies the relationships between the database tables by referencing a column, or set of columns, in the **Child table** that contains the foreign key, to the **PRIMARY KEY** column or set of columns, in the **Parent table.**

c. You need to enter table records after establishing table relationship.

**False**

We could establish table relationship as creating the tables (parent table and children table), or change the table relationship latterly by ALTER method on empty tables without enter records.

d. Enforcing Data Integrity is about to set only entity integrity.

**True**

Enforcing Data Integrity refers to the accuracy, consistency, and reliability of data that is stored in the database by database constraints. Entity Integrity in database ensures a table does not have any duplicate rows and is uniquely identified.

e. Foreign keys are fields added usually into parent tables.

**False**

A foreign key is a way to enforce referential integrity in database. A foreign key means that values in one table must also appear in another table. The referenced table is called the parent table while the table with the foreign key is called the child table. The **foreign key** in the **child table** will generally **reference** a primary key in the **parent table**.

f. Many to Many relationship is implemented as two tables within a given database

**True**

A many-to-many relationship occurs when multiple records in a table are associated with multiple records in another table. So that Many to Many relationship is implemented as two tables.

In other hand, relational database systems **usually** don't allow you to implement a direct many-to-many relationship between only two tables.

