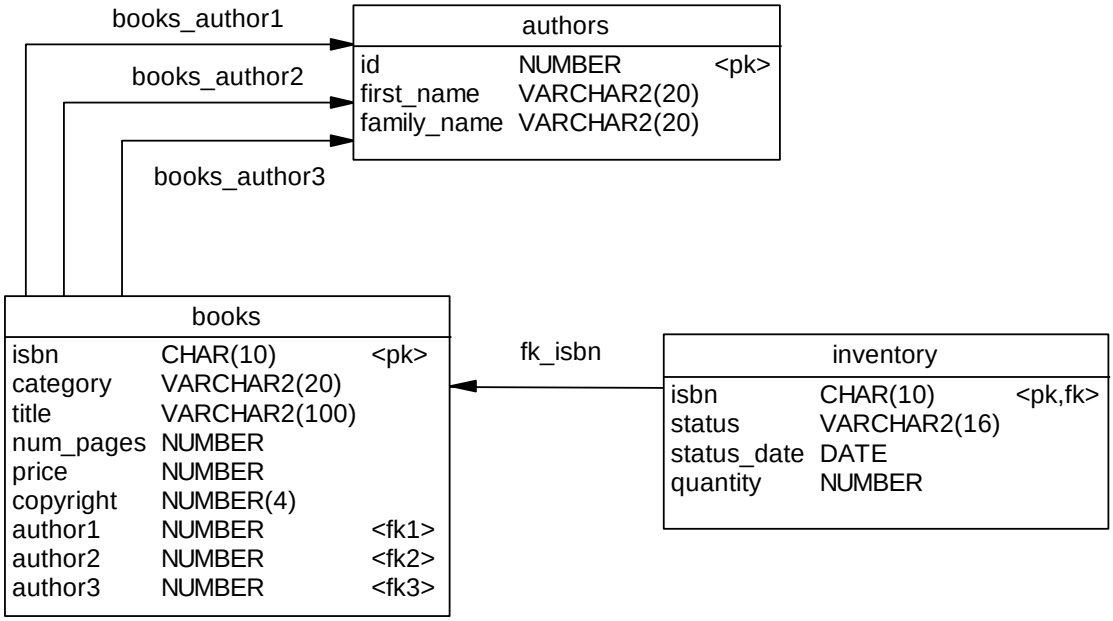
Database Systems
Lab 8
PL/SQL Programming 2

Three tables will be used in this session and the script to create and populate them is provided on the shared folder. The file is called **'author.sql'**. Before completing any of these exercises you should run this script in iSQLPlus to create the sample database

The schema used for this session is as follows:



Task (1)
Load and run the following sql script files

- ✓ while.sql
- ✓ for.sql
- ✓ loop.sql

What does each of these programs do? Make brief notes on the differences between the loops.

Task (2)
Write an anonymous block to display ONLY the first 6 titles from 'books' table.
Hint: Use a simple cursor and the cursor function %ROWCOUNT.

```
SET SERVEROUTPUT ON
DECLARE
     CURSOR book_cur IS SELECT isbn, title FROM books;
  book_record book_cur%ROWTYPE;

BEGIN
OPEN book_cur;
      LOOP
        FETCH book_cur INTO book_record;
        EXIT WHEN book_cur%ROWCOUNT > 6;
      DBMS_OUTPUT.PUT_LINE ('ISBN : ' || book_record.isbn || '   ' ||
book_record.title);
      END LOOP;
      CLOSE book_cur;
```

```
END;
/
```

Run the script create2006.sql.

Introduction to PL/SQL Procedures

A procedure is a named PL/SQL block that is created by using a CREATE PROCEDURE statement. When called, or invoked, a procedure performs a sequence of actions. The anatomy of the procedure includes:

- ✓ The header: "PROCEDURE getemp IS"
- ✓ The declaration section: With variable declarations emp_id and lname
- ✓ The executable section: Contains the PL/SQL and SQL statements used to obtain the last name of employee 100. The executable section makes use of the DBMS_OUTPUT package to print the last name of the employee.

```
SET SERVEROUTPUT ON;
CREATE OR REPLACE PROCEDURE getemp IS
   emp_id employees.employee_id%type;
    lname employees.last_name%type;
BEGIN
    emp_id := 176;
    SELECT last_name INTO lname
    FROM EMPLOYEES
    WHERE employee_id = emp_id;
    DBMS_OUTPUT.PUT_LINE('Last name: '||lname);
END;
/
SHOW ERRORS;
```

To call a procedure by using an anonymous block, use:

```
BEGIN
   getemp;
END;
/
```

The procedure called ADD_JOB to insert a new job into the JOBS table. The procedure requires that the job_id and the title of the job is provided using two parameters.

```
CREATE OR REPLACE PROCEDURE add_job (
  jobid jobs.job_id%TYPE,
  jobtitle jobs.job_title%TYPE) IS
BEGIN
  INSERT INTO jobs (job_id, job_title)
  VALUES (jobid, jobtitle);
  COMMIT;
END add_job;
/
SHOW ERRORS
```

Task(3)

Craete and run the above procedure

Introduction to PL/SQL Functions

A function is a named PL/SQL block that is created with a CREATE FUNCTION statement. Functions are used to compute a value that must be returned to the caller. PL/SQL functions follow the same block structure as procedures. However, the header starts with the keyword FUNCTION followed by the function name. The header includes the return data type after the function name (using the keyword RETURN followed by the data type).

Example

```
CREATE OR REPLACE FUNCTION avg_salary RETURN NUMBER IS
   avg_sal employees.salary%type;
BEGIN
  SELECT AVG(salary) INTO avg_sal
  FROM EMPLOYEES;
   RETURN avg_sal;
END;
/
```

The example declares a variable called avg_sal in the declaration section, and uses the RETURN statement to return the value retrieved from the SELECT statement. The value returned represents the average salary for all employees.

A function can be called from:

- Another PL/SQL block where its return value can be stored in a variable or supplied
- as a parameter to a procedure
- A SQL statement, subject to restrictions.

To call a function from an anonymous block, use the following:

```
SET SERVEROUTPUT ON
BEGIN
      dbms_output.put_line('Average Salary: ' || avg_salary);
END;
/
```

Task(4)

Create and run the above function

Task(5)

Create a function called TOTAL_SALARY to compute the sum of all employee salaries.

a. Create a function called TOTAL_SALARY that returns a NUMBER.
b. In the executable section, execute a query to store the total salary of all employees in a local variable that you declare in the declaration section. Return the value stored in the local variable. Save and compile the code.

```
CREATE OR REPLACE FUNCTION total_salary RETURN NUMBER IS
   avg_sal employees.salary%type;
BEGIN
  SELECT SUM(salary) INTO avg_sal
  FROM EMPLOYEES;
   RETURN avg_sal;
END;
/
```

c. Use an anonymous block to invoke the function. To display the result computed by the function, use the DBMS_OUTPUT.PUT_LINE procedure.

```
SET SERVEROUTPUT ON
BEGIN
dbms_output.put_line('Total Salary: ' || total_salary);
END;
/
```

Task (5)

Create a function called GET_ANNUAL_COMP to return the annual salary computed from an employee's monthly salary and commission passed as parameters.

- Develop and store the function GET_ANNUAL_COMP, accepting parameter values for monthly salary and commission. Either or both values passed can be NULL, but the function should still return a non-NULL annual salary.
- Use the following basic formula to calculate the annual salary:

(salary*12) + (commission_pct*salary*12)

```
CREATE OR REPLACE FUNCTION get_annual_comp(
  sal  IN employees.salary%TYPE,
  comm IN employees.commission_pct%TYPE)
 RETURN NUMBER IS
BEGIN
  RETURN (NVL(sal,0) * 12 + (NVL(comm,0) * nvl(sal,0) * 12));
END get_annual_comp;
/
```

- Use the function in a SELECT statement against the EMPLOYEES table for employees in department 90. This can be achieved as follows:

```
SELECT employee_id, last_name,
       get_annual_comp(salary,commission_pct) "Annual
Compensation"
FROM   employees
WHERE department_id=90
/
```