Database Applications
Lecture 6: Stored Procedures

Santha Sumanasekara
August 2022

RMIT UNIVERSITY

1

1

SQL not enough?

- SQL is a declarative query language.
- Using a declarative database query language may also result in better code than what can be created manually.
- Declarative query languages are also easier to use as they simply focus on what must be retrieved and do so quickly.
  However, declarative languages have their own trade-offs.
- Users have little to no control over how inputs are dealt with.
- If the user wants to use a functionality that the query language doesn't support, they are often at a loss.

RMIT UNIVERSITY

2

2

```
-- User Inputs a name of a director
-- Run SQL query to find movieID
and title directed by this director
```

Output from one SQL cannot be passed as input to the next SQL.

```
-- Find the stars in
each of these movies
```

RMIT UNIVERSITY

3

3

DIRNAME	MVTITLE
Allen, Woody	Interiors
	Manhattan
	Annie Hall
Hitchcock, Alfred	North by Northwest
	Rope
	Psycho
	Rear Window
	The Birds
	Vertigo
De Mille, Cecil B	Samson and Delilah
	The Ten Commandments
Kramer, Stanley	Inherit the Wind
	Judgment at Nuremberg

No control over the formatting of the output.

RMIT UNIVERSITY

4

4

```
SELECT m.mvtitle, awrd_success(m.mvtitle)
    FROM movie m;
```

Adding new (user-defined) functions .

RMIT UNIVERSITY

5

5

Stored Procedures

Extending the capabilities of SQL!

RMIT UNIVERSITY

6

Stored Procedures

- PL/SQL is Oracle's procedural language extension to SQL.
- TransactSQL (T-SQL) is SQL Server's procedural language extension to SQL.
- It provides a server-side, stored procedural language that is easy-to-use, seamless with SQL, robust, portable, and secure.
- PL/SQL enables you to mix SQL statements with procedural constructs, such as loops, conditions, and exceptions.

RMIT UNIVERSITY

7

7

Stored Procedures

```
CREATE OR REPLACE PROCEDURE proc_name (parameter_list)
AS
    local declarations;
    ....
BEGIN
    statements to execute;
    ....
EXCEPTION
    rules to handle exceptions;
    ....
END;
```

Optional, but most procedures do have it

Optional. But, it is a good programming practice to include exception handling.

RMIT UNIVERSITY

8

8

Stored Functions

```
CREATE OR REPLACE FUNCTION function_name (parameter_list)
RETURN return_type
AS
    local declarations;
    ....
BEGIN
    statements to execute;
    ....
    RETURN return_value;
EXCEPTION
    rules to handle exceptions;
    ....
END;
```

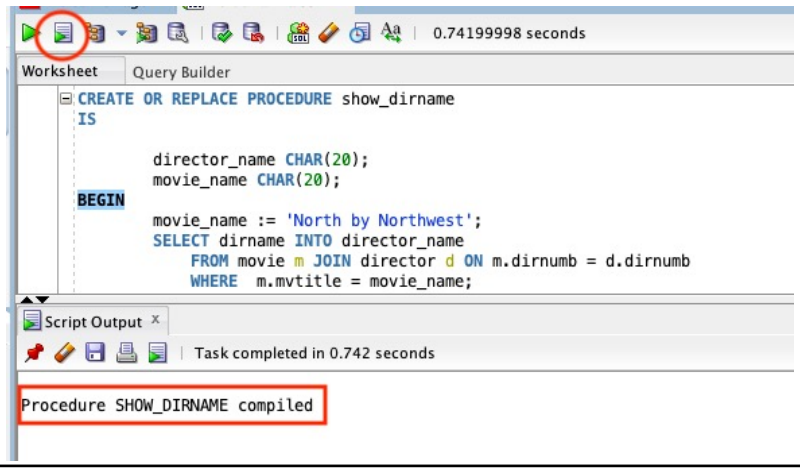
Must have a return value.

9

9

Compiling Procedures (or functions)

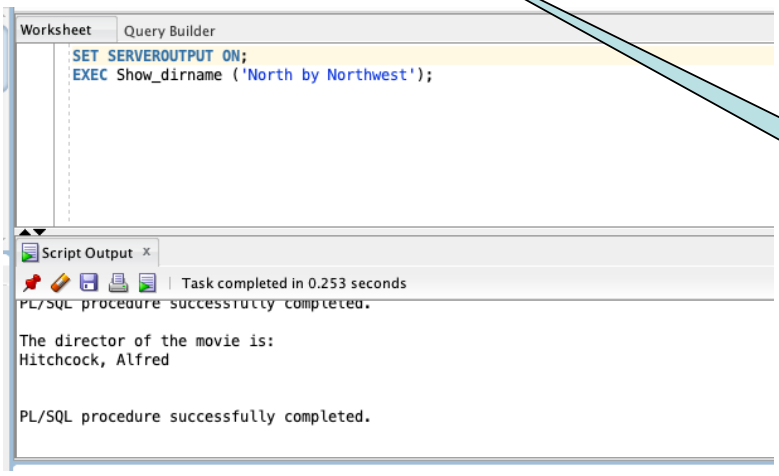
Edit on SQL Developer and run it as a script to compile And store!



10

Calling Procedures

```
SET SERVEROUTPUT ON;
EXEC procedure_name (parameter_values);
```



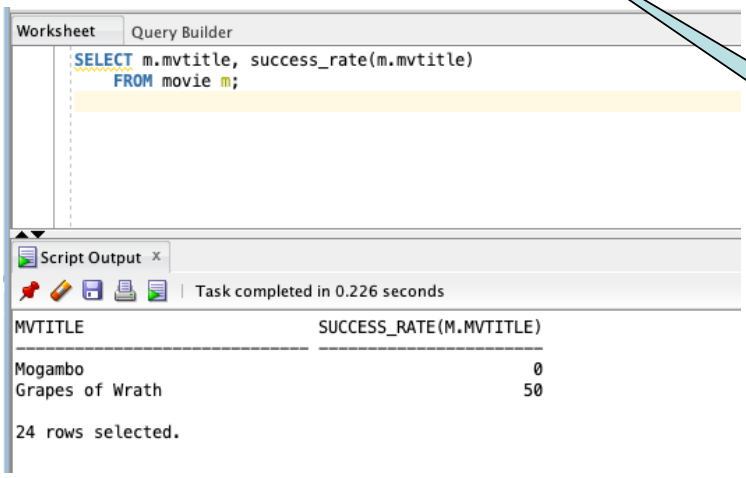
Directs output of the procedure back to the console..

11

11

Calling Functions

```
SELECT m.mvtitle, success_rate(m.mvtitle)
    FROM movie m;
```



Always called within another SQL.

12

Stored Procedures

Mastering PL/SQL procedures and functions

RMIT UNIVERSITY

13

Your first PL/SQL SP (stored Procedure)

- Let's suppose we retrieve information about a known movie.
- In this procedure, we do not pass any parameters (it is a "known" movie).
- Some formatting is applied to the output.

RMIT UNIVERSITY

14

14

Stored Functions

Local Variables

Further formatting is possible here

```
CREATE OR REPLACE PROCEDURE show_dirname
AS
        director_name CHAR(20);
        movie_name CHAR(20);
BEGIN
        movie_name := 'North by Northwest';
        SELECT dirname INTO director_name
            FROM movie m JOIN director d
                         ON m.dirnumb = d.dirnumb
            WHERE  m.mvtitle = movie_name;
        DBMS_OUTPUT.put_line('The director of the movie is:');
        DBMS_OUTPUT.put_line(director_name);
END;
```

15

15

Let's improve it

- Let's allow the user to enter a movie title as an input and pass it as an input parameter.
- This input is used in the filtering condition within the query.
- Some formatting is also applied to the output.

RMIT UNIVERSITY

16

16

```
CREATE OR REPLACE PROCEDURE Show_dirname
                  (movie_name IN movie.mvtitle%type)
AS
    director_name director.dirname%type;

BEGIN
    SELECT dirname INTO director_name
        FROM movie m JOIN director d ON
                     m.dirnumb = d.dirnumb
        WHERE lower(m.mvtitle) = lower(movie_name);
    DBMS_OUTPUT.put_line('The director of the movie is:');
    DBMS_OUTPUT.put_line(director_name);
END;
```

Input parameters

Used in the query

RMIT UNIVERSITY

17

17

Let's improve it – Exception Handling

Worksheet | Query Builder

```
set SERVEROUTPUT ON;
exec Show_dirname('North by North');
```

No such movie!

Script Output

Task completed in 0.166 seconds

```
Error report -
ORA-01403: no data found
ORA-06512: at "E09686.SHOW_DIRNAME", line 7
ORA-06512: at line 1
01403. 00000 -  "no data found"
*Cause:    No data was found from the objects.
*Action:   There was no data from the objects which may be due to end of fetch.
```

RMIT UNIVERSITY

18

18

```
CREATE OR REPLACE PROCEDURE Show_dirname
                  (movie_name IN movie.mvtitle%type)
AS
    director_name director.dirname%type;

BEGIN
    SELECT dirname INTO director_name
        FROM movie m JOIN director d ON
                     m.dirnumb = d.dirnumb
        WHERE lower(m.mvtitle) = lower(movie_name);
    DBMS_OUTPUT.put_line('The director of the movie is:');
    DBMS_OUTPUT.put_line(director_name);

EXCEPTION
      WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE ('The query did not return a result
set');
END;
```

Tells what to do if query didn't produce any rows

19

Let's try this:

- Let's suppose we input a director name and find out the movie titles he/ she directed.
- So, how does that different to the previous example?

RMIT UNIVERSITY

20

20

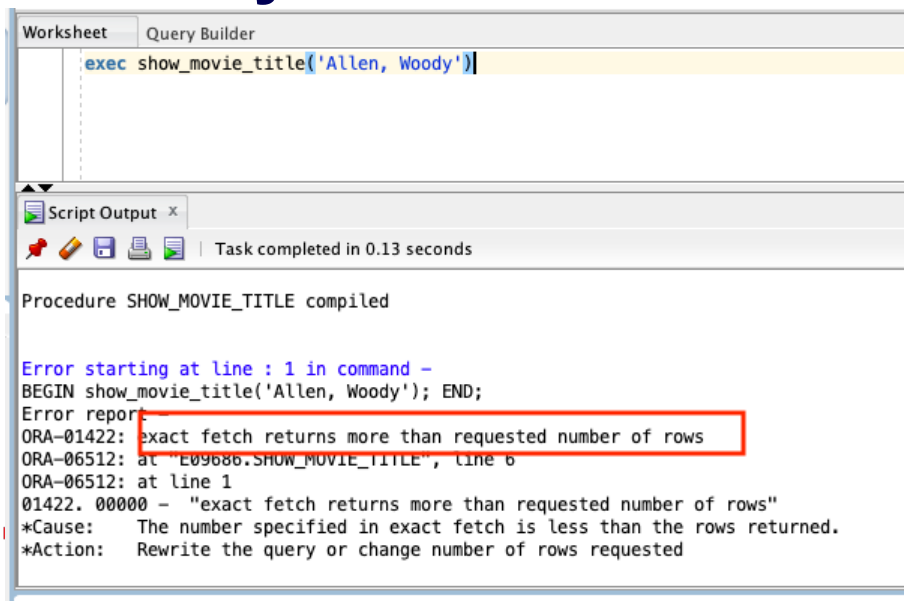
```
CREATE OR REPLACE PROCEDURE show_movie_title
                    (director_name IN director.dirname%type)
AS
    l_movie_title movie.mvtitle%type;
BEGIN
    SELECT mvtitle INTO l_movie_title
        FROM movie m JOIN director d
                     ON m.dirnumb = d.dirnumb
        WHERE lower(d.dirname) = lower(director_name);
    DBMS_OUTPUT.put_line('The title of the movie is:');
    DBMS_OUTPUT.put_line(l_movie_title);

EXCEPTION
    WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE ('The query did not return a result
set');

END;
```

21

Let's try this:



22

Solution: SQL Cursors

- Similar to SQL cursors you defined in the assignment (in PHP), we can define cursors in PL/SQL.
- Same principle:
  - Define a cursor using an SQL statement;
  - Open it;
  - Fetch data in a loop;
  - Close it.
- We have to use a local variable to fetch data (row at a time) out of the cursor.

RMIT UNIVERSITY

23

23

```
CREATE OR REPLACE PROCEDURE show_movie_title
          (director_name IN director.dirname%type)
AS
    CURSOR mv_cursor IS SELECT *
    -- there is no INTO clause1
         FROM movie m JOIN director d ON
                      m.dirnumb = d.dirnumb
         WHERE lower(d.dirname) = lower(director_name);

    l_movie mv_cursor%ROWTYPE;
BEGIN
    OPEN mv_cursor;
    LOOP
        FETCH mv_cursor INTO l_movie;
        EXIT WHEN mv_cursor%NOTFOUND;

        DBMS_OUTPUT.put_line('The title of the movie is:');
        DBMS_OUTPUT.put_line(l_movie.mvtitle);
    END LOOP;
    CLOSE mv_cursor;
```

24

More SQL Cursors

- It is possible to have multiple nested cursors.
- Inner cursors can use results from outer cursors.
- Exercise: Find the movies directed by a given director and actors in each of them.
- A simple (nested) query can produce this result, however, output would be ugly.
- Use nested cursors!

RMIT UNIVERSITY

25

25

```
CREATE OR REPLACE PROCEDURE show_movie_title_and_stars (director_name IN director.dirname%type)
AS
        CURSOR mv_cursor IS SELECT m.mvnumb, m.mvtitle
            FROM movie m JOIN director d ON m.dirnumb = d.dirnumb
            WHERE lower(d.dirname) = lower(director_name);

        l_movie mv_cursor%ROWTYPE;

        CURSOR star_cursor IS SELECT *
            FROM movstar ms JOIN star s ON ms.starnumb = s.starnumb
            WHERE ms.mvnumb = l_movie.mvnumb;
        l_star star_cursor%ROWTYPE;
        BEGIN

        OPEN mv_cursor;
        LOOP
            FETCH mv_cursor INTO l_movie;
            EXIT WHEN mv_cursor%NOTFOUND;

            -- DBMS_OUTPUT.put_line('The title of the movie is:');
            DBMS_OUTPUT.put_line(l_movie.mvtitle);
            DBMS_OUTPUT.put_line('Stars');

            OPEN star_cursor;
            LOOP
                FETCH star_cursor INTO l_star;
                EXIT WHEN star_cursor%NOTFOUND;
                      DBMS_OUTPUT.put_line('    ' || l_star.starname);

            END LOOP;
            CLOSE star_cursor;

        END LOOP;
        CLOSE mv_cursor;

EXCEPTION

      WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE ('The query did not return a result set');

END;
```

26

```
CURSOR mv_cursor IS SELECT m.mvnumb, m.mvtitle
            FROM movie m JOIN director d ON m.dirnumb = d.dirnumb
            WHERE lower(d.dirname) = lower(director_name);

        l_movie mv_cursor%ROWTYPE;

CURSOR star_cursor IS SELECT *
            FROM movstar ms JOIN star s ON ms.starnumb = s.starnumb
            WHERE ms.mvnumb = l_movie.mvnumb;

        l_star star_cursor%ROWTYPE;
```

L_movie.mvnumb is coming from outer query

RMIT UNIVERSITY

27

27

```
OPEN mv_cursor;
LOOP
    FETCH mv_cursor INTO l_movie;
    EXIT WHEN mv_cursor%NOTFOUND;

    DBMS_OUTPUT.put_line('The title of the movie is:');
    DBMS_OUTPUT.put_line(l_movie.mvtitle);
    DBMS_OUTPUT.put_line('Stars');

    OPEN star_cursor;
    LOOP
        FETCH star_cursor INTO l_star;
        EXIT WHEN star_cursor%NOTFOUND;
        DBMS_OUTPUT.put_line('    ' || l_star.starname);

    END LOOP;
    CLOSE star_cursor;

END LOOP;
CLOSE mv_cursor;
```

Outer cursor

Inner cursor

28

PL/SQL Functions!

Users can define their own functions and store with other database objects

RMIT UNIVERSITY

29

PL/SQL Functions

- In our movies table, we store the number of award nominations each movie received and how many awards actually won.
- Award_success_percentage = $\frac{awards}{Nominations}$ X 100 %
- Let's write a function to compute it and return as a decimal value.
- A PL/SQL function will always return a result.

RMIT UNIVERSITY

30

30

```
CREATE OR REPLACE FUNCTION success_rate
       (movie_name IN movie.mvtitle%TYPE) RETURN NUMBER
AS
    l_success DECIMAL (6,2);

BEGIN
    SELECT awrd/noms*100.0 INTO l_success
        FROM movie
        WHERE lower(mvtitle) = lower(movie_name);
RETURN l_success;
END;
```

L_success is the return value of the function.

RMIT UNIVERSITY

31

31

PL/SQL Functions

- Calling PL/SQL functions – always called by another SQL statement.

```
SELECT success_rate('Laura')
    FROM dual;
```

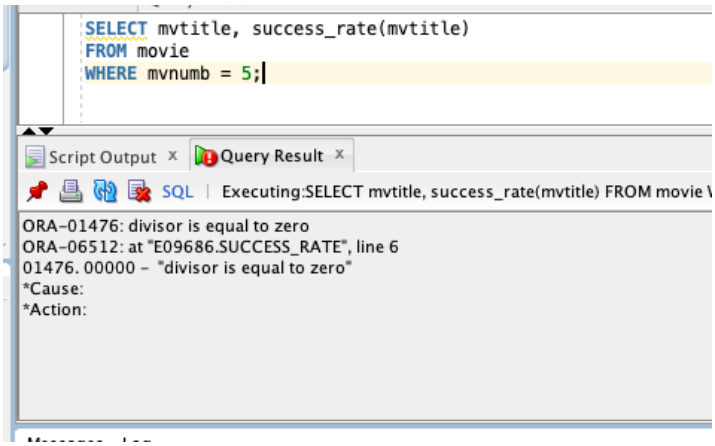
```
SELECT mvtitle, success_rate(mvtitle)
    FROM movie
    WHERE mvnumb = 18;
```

```
SELECT mvtitle, success_rate(mvtitle)
    FROM movie
    WHERE mvnumb = 5;
```

32

32

PL/SQL Functions




```
SELECT mvtitle, success_rate(mvtitle)
    FROM movie
    WHERE mvnumb = 5;
```

33

33

PL/SQL Functions -- Exceptions

```
CREATE OR REPLACE FUNCTION success_rate
       (movie_name IN movie.mvtitle%TYPE) RETURN NUMBER
AS
    l_success DECIMAL (6,2);

BEGIN
    SELECT awrd/noms*100.0 INTO l_success
        FROM movie
        WHERE lower(mvtitle) = lower(movie_name);
RETURN l_success;

EXCEPTION
      WHEN ZERO_DIVIDE THEN
      RETURN 0;
END;
```

34

34

PL/SQL Functions

- Functions can be called from anywhere you can call built-in functions.
- So, it is possible to call them from other PL/SQL procedures (or functions).
- Revisit the first procedure to add award success percentage to the output.

RMIT UNIVERSITY

35

35

```
BEGIN
        OPEN mv_cursor;
        LOOP
            FETCH mv_cursor INTO l_movie;
            EXIT WHEN mv_cursor%NOTFOUND;

            l_success := success_rate(l_movie.mvtitle);

            DBMS_OUTPUT.put_line('The title of
                 the movie is: ' ||l_movie.mvtitle);
            DBMS_OUTPUT.put_line('Award Success Rate: '
                                 || l_success || '%');
        END LOOP;
        CLOSE mv_cursor;
```

RMIT UNIVERSITY

36

36

Local Functions

- In previous examples, functions were created and saved as stand-alone objects. They are available from anywhere (within your schema).
- Sometimes, it is more desirable to define local functions (and procedures), especially, to improve the modularity of your programs.
- They are defined within the declaration section of your procedure(or function).
- Such functions are not visible outside its home procedure (or, function).

RMIT UNIVERSITY

37

37

```
create or replace PROCEDURE show_movie_title_and_success (director_name IN director.dirname%type)
AS
       CURSOR mv_cursor IS SELECT *
        -- there is no INTO clause
            FROM movie m JOIN director d ON m.dirnumb = d.dirnumb
            WHERE lower(d.dirname) = lower(director_name);

        l_movie mv_cursor%ROWTYPE;
        l_success DECIMAL(6,2);

        FUNCTION my_success_rate (movie_name IN movie.mvtitle%TYPE) RETURN NUMBER
        IS
        l_success DECIMAL (6,2);
        BEGIN
            SELECT awrd/noms*100.0 INTO l_success
                FROM movie
                WHERE lower(mvtitle) = lower(movie_name);
            RETURN l_success;
        EXCEPTION

            WHEN ZERO_DIVIDE THEN
            RETURN 0;
        END my_success_rate;

BEGIN
        OPEN mv_cursor;
        LOOP
            FETCH mv_cursor INTO l_movie;
            EXIT WHEN mv_cursor%NOTFOUND;
            l_success := my_success_rate(l_movie.mvtitle);
            DBMS_OUTPUT.put_line('The title of the movie is: ' ||l_movie.mvtitle);
            DBMS_OUTPUT.put_line('Award Success Rate: ' || l_success || '%');
        END LOOP;
        CLOSE mv_cursor;

EXCEPTION

      WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE ('The query did not return a result set');

END;
```

This function's scope is limited to this 'show_movie_title_and_success' procedure

38

Further Reading

- PL/SQL is a very comprehensive programming language
- We only covered the key features – mainly how we can incorporate SQL and SQL cursors to fetch data.
- There are so many other functionalities.
- Further readings:
  - Oracle PL/SQL Programming by Steven Feuerstein and Bill Pribyl -- http://shop.oreilly.com/product/9780596514464.do
  - Oracle PL/SQL homepage: https://www.oracle.com/database/technologies/appdev/plsql.html
  - Handling PL/SQL Errors: https://docs.oracle.com/cd/B28359_01/appdev.111/b28370/errors.htm#LNPLS007

RMIT UNIVERSITY

39

39

Menti-time!

Menti.com
Code: to be supplied

RMIT UNIVERSITY

40

Questions?

RMIT UNIVERSITY

41