CSCI235 Database Systems

Stored PL/SQL

Dr Janusz R. Getta

School of Computing and Information Technology -
University of Wollongong

Stored PL/SQL

Outline

Stored PL/SQL ? What is it

Stored PL/SQL means PL/SQL procedures and PL/SQL functions pre-compiled and stored in a data dictionary ready to be processed

Stored procedures and functions can be referenced or called any number of times by multiple applications processing the relational tables

Stored procedures and functions can accept parameters when processed (called)

Stored procedures can be processed (called) with `EXECUTE` statement

Stored functions can be processed (called) in SQL statement wherever a function can be used, e.g. as row functions in `SELECT` statement

Stored procedures and stored functions can be used to extend the functionality of data retrieval and data manipulation statements of SQL (extensibility) and to eliminate duplication of code in the database applications (re-useability)

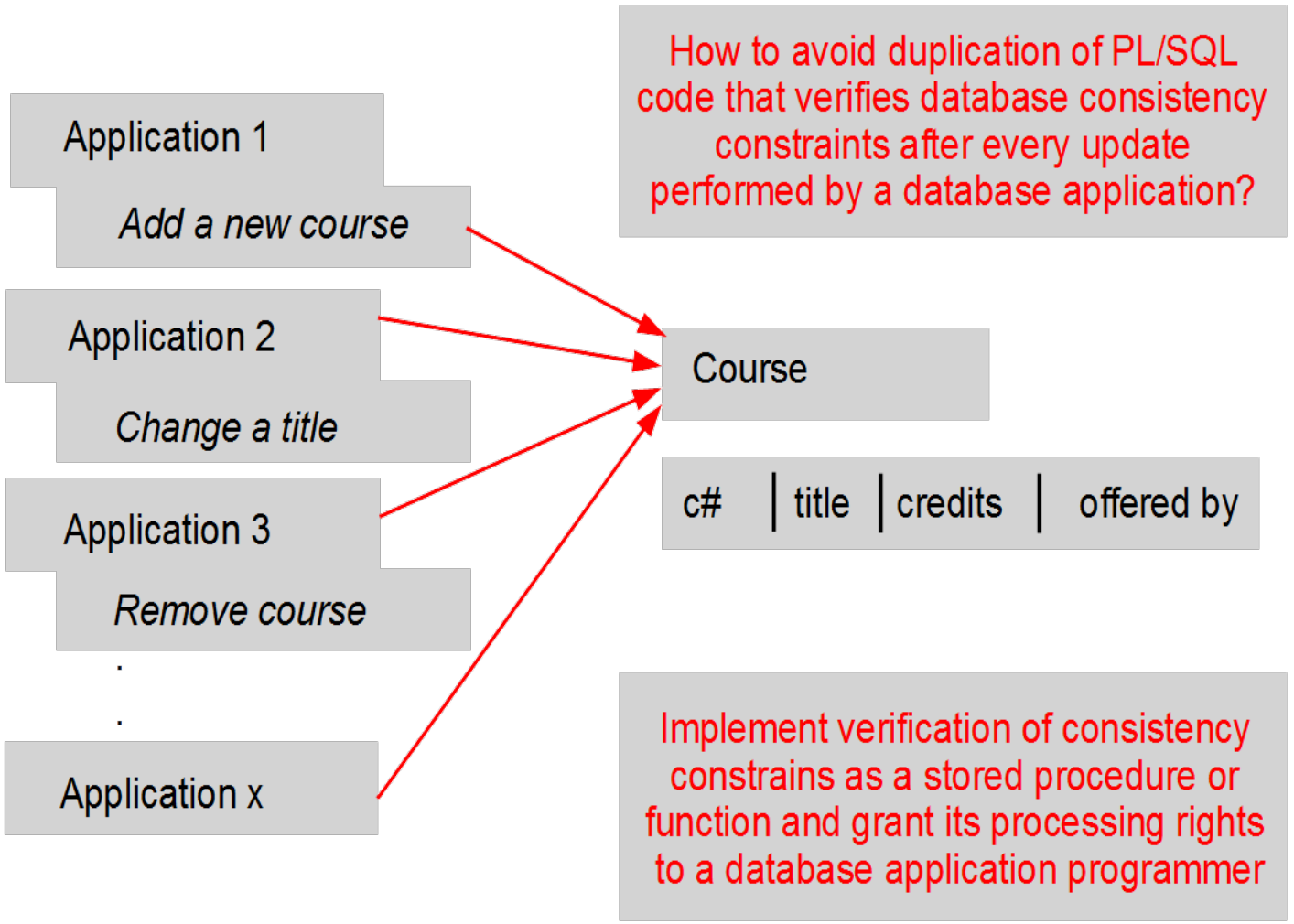
Stored PL/SQL ? What is it

Stored procedures and functions are created with `CREATE OR REPLACE PROCEDURE` and `CREATE OR REPLACE FUNCTION` SQL statements

Stored PL/SQL

Outline

Applications - reusability



Applications - extensibility

Find the names of all departments together with a list of courses offered by each department, display the results in the following form:

Sample results

```
DEPARTMENT NAME | LIST OF COURSES OFFERED
-------------------------------------------------------
Math            | Calculus Topology Logic Algebra
Comp Sci        | Python Java Databases
Biol            |
Phys            | Relativity Mechanics
Astro           | Astrology
```

Implement a function `LCOURSES( dept_name )` that returns a list of courses offered by a department whose name is a value of a parameter `dept_name`

Use a function `LCOURSES` as a row function in `SELECT` statement

Application of stored function

```
SELECT dname, LCOURSES( dname )
FROM DEPARTMENT;
```

A function `LCOURSES` is called for every row retrieved from a relational table `DEPARTMENT` like any standard row function, e.g. `UPPER` function

Stored PL/SQL

Outline

CREATE OR REPLACE PROCEDURE statement

CREATE OR REPLACE PROCEDURE statement compiles and stores PL/SQL procedure in a data dictionary

The following stored procedure INSERT_COURSE converts the values of string parameters to upper case and inserts a row into a relational table COURSE

Header of stored procedure

```
CREATE OR REPLACE PROCEDURE INSERT_COURSE( cnumber  IN NUMBER,
                                           ctitle   IN VARCHAR,
                                           ccredits IN NUMBER,
                                           coffer   IN VARCHAR) IS
```

Body of stored procedure

```
BEGIN
  INSERT INTO COURSE VALUES( cnumber, UPPER(ctitle), ccredits, UPPER(coffer) );
  COMMIT;
END INSERT_COURSE;
```

EXECUTE statement is used to process a procedure INSERT_COURSE

Application of stored procedure

```
EXECUTE INSERT_COURSE(666, 'Java for kids', 6, 'Comp Sci');
```

Stored PL/SQL

Outline

CREATE OR REPLACE FUNCTION statement

CREATE OR REPLACE FUNCTION statement compiles and stores PL/SQL function in a data dictionary

The following stored function LCOURSES lists the names of departments together with the titles of courses offered by each department

Header of stored function

```
CREATE OR REPLACE FUNCTION LCOURSES( dept_name VARCHAR ) RETURN VARCHAR IS
```

Declaration and initialization of local variable

```
  course_list VARCHAR(300);
BEGIN
  course_list := '';
```

Iterations over cursor

```
  FOR course_cur_rec IN (SELECT title FROM COURSE WHERE offered_by = dept_name);
```

Body of cursor

```
  LOOP
    course_list := course_list || course_cur_rec.title || ' ';
  END LOOP;
```

Returning a result

```
  RETURN course_list;
END LCOURSES;
```

CREATE OR REPLACE FUNCTION statement

A stored function LCOURSES is called as a row function in SELECT statement

Application of stored function

```
SELECT dname, LCOURSES( dname )
FROM COURSE;
```

Stored PL/SQL

Outline

GRANT statement revisited

In addition to read and write access rights it is possible to grant `EXECUTE` rights on stored procedures and functions

For example, a user `scott` grants execution rights on `INSERT_COURSE` to a user `janusz`

Granting a processing right on a stored procedure

```
GRANT EXECUTE ON INSERT_COURSE TO janusz;
```

Now, a user `janusz` executes a stored procedure `INSERT_COURSE`

Application of stored procedure

```
EXECUTE scott.INSERT_COURSE(958, 'Multimedia Databases', 6, 'Comp Sci');
```

References

Database PL/SQL Language Reference

Database SQL Language Reference, `CREATE PROCEDURE`

Database SQL Language Reference, `CREATE FUNCTION`

Database SQL Language Reference, `GRANT`

T. Connoly, C. Begg, Database Systems, A Practical Approach to Design, Implementation, and Management, Chapter 8 Advanced SQL, Pearson Education Ltd, 2015