CSCI235 Database Systems

Stored PL/pgSQL

Dr Janusz R. Getta

School of Computing and Information Technology -
University of Wollongong

Stored PL/pgSQL

Outline

Stored PL/pgSQL ? What is it

Stored PL/pgSQL means PL/pgSQL procedures and PL/pgSQL 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 called and processed with `CALL` statement

Stored functions can be called and processed within SQL statement wherever a function can be used, for example as a row function 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 (reuseability)

Stored PL/pgSQL ? What is it

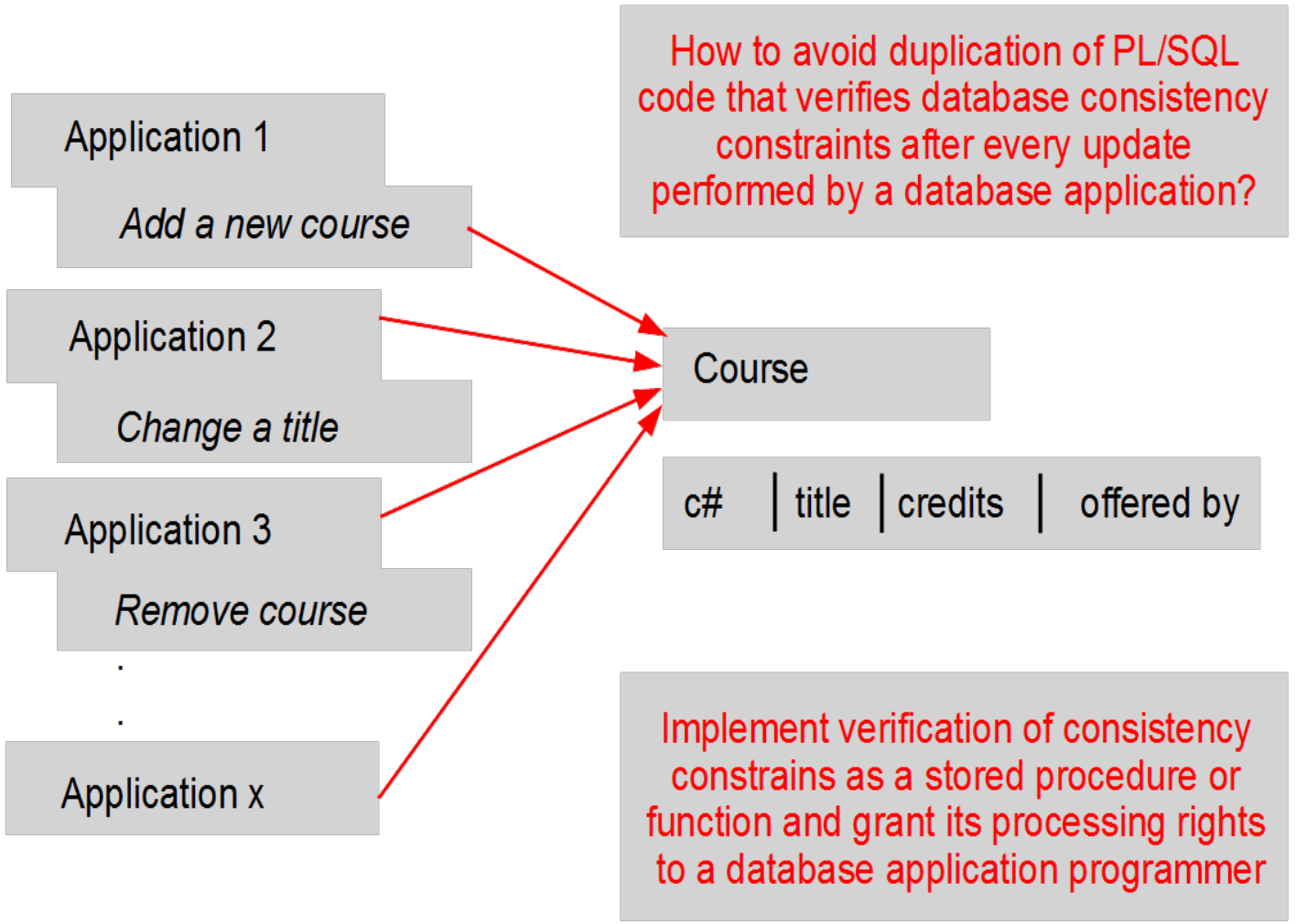
Stored procedures and functions can be created with SQL statements:

- `CREATE OR REPLACE PROCEDURE`
- `CREATE OR REPLACE FUNCTION`

Stored PL/pgSQL

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/pgSQL

Outline

CREATE OR REPLACE PROCEDURE statement

CREATE OR REPLACE PROCEDURE statement compiles and stores PL/pgSQL 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 CHAR,
                                           ctitle   IN VARCHAR,
                                           ccredits IN NUMERIC,
                                           coffer   IN VARCHAR) AS $$
BEGIN
  INSERT INTO COURSE VALUES( cnumber, UPPER(ctitle), ccredits, coffer);
END;
$$ LANGUAGE plpgsql;
```

CALL statement is used to process a procedure INSERT_COURSE

Application of stored procedure

```
BEGIN;    -- Start a transaction
CALL insert_course('CSIT115','Data Management and Security',6,'Computer Science');
COMMIT;   -- Commit insertion and end a transaction
```

Stored PL/pgSQL

Outline

CREATE OR REPLACE FUNCTION statement

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

A stored function LCOURSES returns a string of characters with the titles of courses offered by a given department

Header of stored function

```
CREATE OR REPLACE FUNCTION LCOURSES( dept_name VARCHAR )
RETURNS VARCHAR AS $$
DECLARE
  course_list VARCHAR(300);
  TITLES CURSOR FOR SELECT title FROM COURSE WHERE offered_by = dept_name;
BEGIN
  course_list := '';
  FOR course_rec IN TITLES
  LOOP
    course_list := course_list || course_rec.title || '|';
  END LOOP;
  RETURN course_list;
END;
$$ LANGUAGE plpgsql;
```

CREATE OR REPLACE FUNCTION statement

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

Application of stored function

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

The results

```
      name        |                                                    lcourses
------------------+-----------------------------------------------------------------------------------
 Computer Science | Databases|Advanced Programming|Algorithms and Data Structures|
 Physics          | Mechanics|Relativity|
 Engineering      | Materials in Design|Engineering Computing and Analysis|C++|
 Mathematics      | Calculus|
 Business         | Accounting Fundamentals|Principles of Responsible Business|
 Education        | Introductory Principles of Education|
 Arts             |
 Science          | World Archaeology|
```

Stored PL/pgSQL

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 a stored procedure INSERT_COURSE to a user janusz

Granting a processing right on a stored procedure

```
GRANT EXECUTE ON PROCEDURE INSERT_COURSE(CHAR,VARCHAR,NUMERIC,VARCHAR) TO janusz;
```

Now, a user janusz executes a stored procedure INSERT_COURSE

Application of stored procedure

```
CALL scott.INSERT_COURSE('CSIT958', 'Multimedia Databases', 6, 'Computer Science');
```

GRANT statement revisited

For example, a user `scott` grants execution rights on a stored function `LCOURSES` to a user `janusz`

Granting a processing right on a stored procedure

```
GRANT EXECUTE ON FUNCTION LCOURSES(VARCHAR) TO abc123;
```

Now, a user `janusz` executes a stored function `LCOURSES`

Application of stored procedure

```
SELECT name, scott.LCOURSES(name)
FROM DEPARTMENT;
```

References

PostgreSQL 16.1 Documentation, Chapter 43. PL/pgSQL — SQL Procedural Language

PostgreSQL Tutorial, PostgreSQL PL/pgSQL

PostgreSQL 16.1 Documentation, `CREATE PROCEDURE`

PostgreSQL 16.1 Documentation, `CREATE FUNCTION`

PostgreSQL 16.1 Documentation, `GRANT`

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