



Procedural Language (PL/SQL)

Νικόλαος Μήτρου
Καθ. ΕΜΠ



Procedural Language (PL/SQL)

- Τι είναι – εισαγωγή
- Υλοποιήσεις:
 - PL/PostgreSQL – PL/pgSQL
 - Oracle PL/SQL
 - ...
- Δομή – Δηλώσεις PL/pgSQL
- Εκτελέσιμες εντολές και έλεγχος ροής PL/pgSQL
- Τύποι δεδομένων και παράμετροι συναρτήσεων PL/pgSQL



PL/SQL

Τι είναι - εισαγωγή

- Γλώσσα προγραμματισμού για SQL servers
- Εμπλουτίζει την SQL με δομές ελέγχου
- Ορίζει συναρτήσεις (functions) ή/και ρουτίνες (procedures)
- Εκτελεί σύνθετους υπολογισμούς
- Κληρονομεί τύπους, συναρτήσεις και τελεστές που ορίζονται από το χρήστη
- **Με τη χρήση της, αποφεύγονται οι πολλές χρονοβόρες δοσοληψίες με τον server**



PL/SQL (συνέχεια)

Υλοποιήσεις

- **ORACLE PL/SQL**

- Oracle Database 12c, PL/SQL

<http://www.oracle.com/technetwork/database/features/plsql/index.html>

- Using Oracle PL/SQL, <http://infolab.stanford.edu/~ullman/fcdb/oracle/or-plsql.html>

- **PL/PostgreSQL**

- PL/pgSQL Documentation

<http://www.postgresql.org/docs/10/static/plpgsql.html>

- PL/pgSQL Tutorial, <http://www.w3resource.com/PostgreSQL/pl-pgsql-tutorial.php>



PL/pgSQL

Δομή

- Η βασική δομή είναι το block, που αποτελεί και την εμβέλεια των μεταβλητών
- Το όνομα (label) χρησιμοποιείται για αναφορά
- Μπορούμε να έχουμε εμφωλευμένα block
- **messages**

```
[ <<label>> ]  
[ DECLARE  
  declarations ]  
BEGIN  
  statements  
END [label];
```

Παράδειγμα

```
CREATE FUNCTION [schema.]somefunc() RETURNS INTEGER AS $$  
<< outerblock >>  
DECLARE  
  quantity INTEGER := 30;  
BEGIN  
  RAISE NOTICE 'Quantity here is %', quantity; -- Prints 30  
  quantity := 50;  
  -- this is a single-line comment  
  -- multiline comments are included within /* ...*/  
  -- Create a subblock  
  DECLARE  
    quantity INTEGER := 80;  
  BEGIN  
    RAISE NOTICE 'Quantity here is %', quantity; -- Prints 80  
    RAISE NOTICE 'Outer quantity here is %',  
      outerblock.quantity; -- Prints 50  
  END;  
  RAISE NOTICE 'Quantity here is %', quantity; -- Prints 50  
  RETURN quantity;  
END;  
$$ LANGUAGE plpgsql;
```

\$\$: Οριοθετούν το σώμα της συνάρτησης



PL/pgSQL (συνέχεια)

Δηλώσεις (version 9.5)⁽¹⁾

- Όλες οι μεταβλητές πρέπει να δηλώνονται (με εξαίρεση, ακέραια μεταβλητή σε βρόχο FOR)
- Παραδείγματα δηλώσεων:
 - `myint1 INTEGER;`
 - `myint2 INTEGER DEFAULT 1;`
 - `myint3 CONSTANT INTEGER := 20;`
 - `mystring VARCHAR;`
 - `myrow tablename%ROWTYPE;`
 - `myfield tablename.columnname%TYPE;`
- Η γενική δομή μιας δηλωτικής πρότασης είναι:

```
name [ CONSTANT ] type [ COLLATE(2) collation_name ]  
[ NOT NULL ] [ { DEFAULT | := } expression ];
```

⁽¹⁾ Την τελευταία έκδοση θα την δείτε εδώ: <http://www.postgresql.org/docs/10/static/plpgsql.html>

⁽²⁾ COLLATE: δηλώνει τον τρόπο παράθεσης/ταξινόμησης αλφαριθμητικών (strings) με βάση π.χ. το character set



PL/pgSQL (συνέχεια)

Παράμετροι συναρτήσεων & τύποι δεδομένων επιστροφής

- Οι συναρτήσεις μπορούν να δέχονται και να επιστρέφουν ως παραμέτρους όλους τους επιτρεπούς από τον server τύπους δεδομένων (δεδομένα βαθμωτά, πίνακες ή σύνθετα - **%rowtype**)
- Μπορούν, επίσης, να οριστούν να επιστρέφουν ένα “set” (ή πίνακα) από οποιαδήποτε δεδομένα μπορούν να επιστραφούν ως μεμονωμένα
- Μπορούν απλά να εκτελούν ένα σύνολο λειτουργιών και να μην επιστρέφουν κάτι (**RETURNS void**)



PL/pgSQL (συνέχεια)

Παράδειγμα συνάρτησης, με μία παράμετρο εισόδου και μία εξόδου

schema (εφόσον δεν είναι το public)

```
DROP FUNCTION IF EXISTS example1.find_student(integer);
CREATE FUNCTION example1.find_student(persID INTEGER) RETURNS text AS $$
-- Instead of the above two statements: CREATE OR REPLACE FUNCTION ...
DECLARE
    last_name VARCHAR;
    first_name VARCHAR;
BEGIN
    SELECT name, givenname INTO STRICT last_name, first_name --(*)
    FROM example1.student WHERE IDstudent = persID;
    RETURN first_name || '====' || last_name; --(**)
END;
$$ LANGUAGE plpgsql;
```

```
SELECT example1.find_student(202002);
```

find_student
text
ΟΔΥΣΣΕΑΣ====ΑΡΑΠΕΛΛΗΣ

(*) STRICT is optional. If present, the query must return exactly one row

(**) String concatenation in postgresQL: string || string



PL/pgSQL (συνέχεια)

Δομές ελέγχου

```
IF boolean-expression THEN
    statements
[ ELSIF boolean-expression THEN
    statements
[ ELSIF boolean-expression THEN
    statements
    ... ] ]
[ ELSE
    statements ]
END IF;
```

```
CASE
    WHEN boolean-expression THEN
        statements
[ WHEN boolean-expression THEN
    statements
    ... ]
[ ELSE
    statements ]
END CASE;
```

```
[ <<label>> ]
    FOR identifier IN [ REVERSE ]
expression1 .. expression2 LOOP
    statement;
    [...]
END LOOP;
```

```
[ <<label>> ]
    WHILE condition LOOP
    statement;
    [...]
END LOOP;
```



PL/pgSQL (συνέχεια)

Παράδειγμα 3:

Να εισαχθούν στον πίνακα `room` του σπουδαστολογίου τόσες νέες εγγραφές, όσα τα διαφορετικά Ιδρύματα προέλευσης των σπουδαστών, με τύπο χρήσης 'office' και label 'ΓΕΩΠ-2024.x', x=1,2,....

Προετοιμασία, με δύο αλλαγές στο σπουδαστολόγιο:

- Το κλειδί του πίνακα `room` δηλώνεται ως **SERIAL** για να εισάγεται αυτόματα
- Δημιουργείται βοηθητικός πίνακας **roomuniversity** (υπόδειξη)

```
-- 1. Δημιουργούμε αντίγραφο του 'room'  
CREATE TABLE room_temp AS TABLE exercisel.room
```

```
-- 2. Δημιουργούμε νέο 'room' με SERIAL key
```

```
DROP TABLE exercisel.room  
CREATE TABLE exercisel.room (  
  IDroom SERIAL PRIMARY KEY,  
  label VARCHAR(50) DEFAULT NULL,  
  use room_use,  
  seats INTEGER );
```

```
-- 3. Εισάγουμε τις παλαιές εγγραφές από το αντίγραφο
```

```
INSERT INTO exercisel.room (IDroom, label, seats)  
  SELECT IDroom, label, seats FROM room_temp;
```

```
-- 4. Αλλάζουμε την τιμή εκκίνησης της ακολουθίας του κλειδιού
```

```
ALTER SEQUENCE exercisel.room_IDroom_seq RESTART WITH 202410;
```

```
-- 5. Δημιουργούμε πίνακα
```

```
-- roomuniversity
```

```
CREATE TABLE exercisel.roomuniversity (  
  roomID INTEGER NOT NULL,  
  univ VARCHAR(50),  
  PRIMARY KEY (roomID, univ),  
  FOREIGN KEY (roomID)  
    REFERENCES exercisel.room(IDroom)  
  ON DELETE CASCADE  
);
```



PL/pgSQL (συνέχεια)

Παράδειγμα 3 (συνέχεια)

```
CREATE OR REPLACE FUNCTION exercisel.more_rooms() RETURNS void AS $$
DECLARE
    cnt INTEGER := 1;
    u exercisel.person.university%TYPE;
    s INTEGER;
    str VARCHAR := 'ΓΕΩΠ-2024-';
BEGIN
-- Βρόχος σάρωσης στοιχείων πίνακα (Ίδρυμα και αντίστοιχος αριθμός σπουδαστών)
FOR u,s IN (SELECT university, COUNT(university) student_num
            FROM exercisel.person WHERE category='student' AND ak_year = 2024
            GROUP BY university
            ORDER BY student_num DESC)
LOOP
    INSERT INTO exercisel.room (label, use, seats) VALUES
        (CONCAT(str,cnt), 'office' ,s);
        -- το πρωτεύον κλειδί, IDroom, παράγεται αυτόματα
-- Ενημερώνεται και ο πίνακας roomuniversity, με το roomID να λαμβάνεται
-- από τη μόλις καταχωρηθείσα νέα εγγραφή στον πίνακα room
    INSERT INTO exercisel.roomuniversity (roomID, univ) VALUES
        ((SELECT IDroom FROM exercisel.room WHERE label=CONCAT(str,cnt)), u);
        cnt = cnt + 1;
END LOOP;
END;
$$ LANGUAGE plpgsql;

SELECT exercisel.more_rooms();
```



PL/pgSQL (συνέχεια)

Οι συναρτήσεις PL/pgSQL στη διεπαφή PgAdmin

The screenshot displays the PgAdmin interface. On the left, the Object Explorer shows a tree structure of database objects. The 'exercise1' schema is expanded, and the 'Functions (7)' folder is selected. The 'more_rooms()' function is highlighted with a red box. A red arrow points from this function to the 'Source' property in the Properties pane. Another red arrow points from the 'Source' property to the 'Code' tab of the function's properties dialog. The 'Code' tab shows the following SQL code:

```
1
2
3 DECLARE
4     cnt INTEGER := 1;
5     u exercisel.person.university%TYPE;
6     s INTEGER;
7     str VARCHAR := 'ΓΕΩΠ-2024-';
8
9 BEGIN
10  -- Βρόχος σάρωσης στοιχείων πίνακα (Ιδρυμα και αντίστοιχος αριθμός σπουδαστών)
11  FOR u,s IN (SELECT university, COUNT(university) student_num
12             FROM exercisel.person WHERE category='student' AND ak_year = 2024
13             GROUP BY university
14             ORDER BY student_num DESC)
15  LOOP
16     INSERT INTO exercisel.room (label, use, seats) VALUES
17     (CONCAT(str,cnt), 'office' ,s);
18     -- το πρωτεύον κλειδί, IDroom, παράγεται αυτόματα
19     -- Ενημερώνεται και ο πίνακας roomuniversity, με το roomID να λαμβάνεται
20     -- από τη μόλις καταχωρηθείσα νέα εγγραφή στον πίνακα room
21     INSERT INTO exercisel.roomuniversity (roomID, univ) VALUES
22     ((SELECT IDroom FROM exercisel.room WHERE label=CONCAT(str,cnt)), u);
23     cnt = cnt + 1;
24  END LOOP;
```



PL/pgSQL (συνέχεια)

Παράδειγμα (συνέχεια): οι πίνακες **room** και **roomuniversity**

	idroom integer	label character varying(50)	use example2.ro	seats integer
1	1	ΣΑΤΜ-ΜΑ	class	80
2	2	ΣΑΤΜ-ΑΙΘΟΥΣΑ 19	class	40
3	202401	ΓΕΩΠ-2024ΕΜΠ-1	office	3
4	202402	ΓΕΩΠ-2024ΕΜΠ-2	office	3
5	202403	ΓΕΩΠ-2024ΕΜΠ-3	office	3
6	202404	ΓΕΩΠ-2024ΕΜΠ-4	office	3
7	202405	ΓΕΩΠ-2024ΕΜΠ-5	office	1
8	202406	ΓΕΩΠ-2024ΧΑΡΟΚ. ΠΑΝ.-1	office	3
9	202407	ΓΕΩΠ-2024ΧΑΡΟΚ. ΠΑΝ.-2	office	3
10	202408	ΓΕΩΠ-2024ΧΑΡΟΚ. ΠΑΝ.-3	office	1
11	202409	ΓΕΩΠ-2024ΑΠΘ-1	office	2
12	202410	ΓΕΩΠ-2024ΠΑΝ. ΑΙΓΑΙΟΥ-1	office	1
13	202411	ΓΕΩΠ-2024ΑΣΠΑΙΤΕ-1	office	1
14	202412	ΓΕΩΠ-2024ΣΧ. ΙΚΑΡΩΝ-1	office	1
15	202413	ΓΕΩΠ-2024 ΕΜΠ-1	office	1
16	202414	ΓΕΩΠ-2024ΣΤΡΑΤ. ΣΧ. ΕΥΕ	office	1
17	202415	ΓΕΩΠ-2024ΠΑΝ. ΠΕΙΡΑΙΩΣ-	office	1

	roomid integer	univ character varying(50)
1	202401	ΕΜΠ
2	202402	ΕΜΠ
3	202403	ΕΜΠ
4	202404	ΕΜΠ
5	202405	ΕΜΠ
6	202406	ΧΑΡΟΚ. ΠΑΝ.
7	202407	ΧΑΡΟΚ. ΠΑΝ.
8	202408	ΧΑΡΟΚ. ΠΑΝ.
9	202409	ΑΠΘ
10	202410	ΠΑΝ. ΑΙΓΑΙΟΥ
11	202411	ΑΣΠΑΙΤΕ
12	202412	ΣΧ. ΙΚΑΡΩΝ
13	202413	ΕΜΠ
14	202414	ΣΤΡΑΤ. ΣΧ. ΕΥΕΛΠΙΔΩΝ
15	202415	ΠΑΝ. ΠΕΙΡΑΙΩΣ