



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 $$
DECLARE
    last_name VARCHAR;
    first_name VARCHAR;
BEGIN
    SELECT name, givenname INTO STRICT last_name, first_name --(*)
    FROM example1.student WHERE IDstudent = persID;
    RETURN first_name || '=====' || last_name; --(**)
END;
$$ 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 'ΣΑΤΜ-Λ2.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 room
-- 2. Δημιουργούμε νέο 'room' με SERIAL key
DROP TABLE room
CREATE TABLE 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 room (IDroom, label, seats)
  SELECT IDroom, label, seats FROM room_temp;
-- 4. Αλλάζουμε την τιμή εκκίνησης της ακολουθίας του κλειδιού
ALTER SEQUENCE room_IDroom_seq RESTART WITH 201910;
```

```
-- 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)

```
DROP FUNCTION IF EXISTS exercisel.more_rooms();
CREATE FUNCTION exercisel.more_rooms() RETURNS void AS $$
    DECLARE
        cnt INTEGER := 1;
        u exercisel.person.university%TYPE;
        s INTEGER;
        str VARCHAR := 'ΣΑΤΜ-Λ2.';
    BEGIN
        -- Βρόχος σάρωσης στοιχείων πίνακα (Ίδρυμα και αντίστοιχος αριθμός σπουδαστών)
        FOR u,s IN (SELECT university, COUNT(university) student_num
                    FROM exercisel.person WHERE category='student' AND idperson > 202000
                    GROUP BY university
                    ORDER BY student_num DESC)
        LOOP
            INSERT INTO exercisel.room (label, use, seats) VALUES
                (CONCAT(str,cnt), 'office', s);
            -- το πρωτεύον κλειδί, IDroom, παράγεται αυτόματα
            -- Ενημερώνεται και ο πίνακας roomuniversity, με το roomID να λαμβάνεται
            -- από τη μόλις καταχωρηθείσα νέα εγγραφή στον πίνακα room
            INSERT INTO exercisel.roomuniversity (roomID, univ) VALUES
                ((SELECT IDroom FROM exercisel.room WHERE label=CONCAT(str,cnt)), u);
            cnt = cnt + 1;
        END LOOP;
    END;
$$ LANGUAGE plpgsql;
SELECT exercisel.more_rooms();
```



PL/pgSQL (συνέχεια)

Οι συναρτήσεις PL/pgSQL στη διεπαφή PgAdmin

The screenshot displays the PgAdmin interface with the following components:

- Left Panel (Object Explorer):** Shows a tree view of the database structure. The 'exercise1' schema is expanded, and the 'Functions (7)' folder is selected. The 'more_rooms()' function is highlighted with a red box.
- Properties Panel:** Shows the 'Source' property of the selected function, which is also highlighted with a red box. A red arrow points from this box to the code editor.
- Code Editor:** Displays the source code of the 'more_rooms' function. The code is as follows:

```
1
2
3 DECLARE
4     cnt INTEGER := 1;
5     u exercisel.person.university%TYPE;
6     s INTEGER;
7     str VARCHAR := 'ΣΑΤΜ-Α2.';
8 BEGIN
9     -- Βρόχος σάρωσης στοιχείων πίνακα (Ιδρυμα και αντίστοιχος αριθμός σπουδαστών)
10    FOR u,s IN (SELECT university, COUNT(university) student_num
11               FROM exercisel.person WHERE category='student' AND idperson > 202000
12               GROUP BY university
13               ORDER BY student_num DESC)
14    LOOP
15        INSERT INTO exercisel.room (label, use, seats) VALUES
16            (CONCAT(str,cnt), 'office', s);
17        -- το πρωτεύον κλειδί, IDroom, παράγεται αυτόματα
18        -- Ενημερώνεται και ο πίνακας roomuniversity, με το roomID να λαμβάνεται
19        -- από τη μόλις καταχωρηθείσα νέα εγγραφή στον πίνακα room
20        INSERT INTO exercisel.roomuniversity (roomID, univ) VALUES
21            ((SELECT IDroom FROM exercisel.room WHERE label=CONCAT(str,cnt)), u);
22        cnt = cnt + 1;
23    END LOOP;
```



PL/pgSQL (συνέχεια)

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

idroom [PK] serial	label character varying(50)	use exercise1.room_use	seats integer
201901	ΣΑΤΜ-ΜΑ	class	80
201902	ΣΑΤΜ-Λ20	class	40
201903	ΣΑΤΜ-Λ1.10	office	1
201904	ΣΗΜΜΥ-Β.2.12	office	1
202010	ΣΑΤΜ-Λ2.1	office	9
202011	ΣΑΤΜ-Λ2.2	office	3
202012	ΣΑΤΜ-Λ2.3	office	3
202013	ΣΑΤΜ-Λ2.4	office	2
202014	ΣΑΤΜ-Λ2.5	office	2
202015	ΣΑΤΜ-Λ2.6	office	2
202016	ΣΑΤΜ-Λ2.7	office	2
202017	ΣΑΤΜ-Λ2.8	office	1
202018	ΣΑΤΜ-Λ2.9	office	1
202019	ΣΑΤΜ-Λ2.10	office	1
202020	ΣΑΤΜ-Λ2.11	office	1

roomid [PK] integer	univ [PK] character varying(50)
202010	ΕΜΠ
202011	ΣΧΟΛΗ ΙΚΑΡΩΝ
202012	ΔΠΘ
202013	ΠΑΝ. ΑΙΓΑΙΟΥ
202014	ΕΜΠ ΚΑΙ ΣΣΕ
202015	ΓΕΩΠΟΝΙΚΟ ΠΑΝ. ΑΘΗΝΩΝ
202016	ΠΑΝ. ΘΕΣΣΑΛΙΑΣ
202017	ΟΙΚΟΝ. ΠΑΝ. ΑΘΗΝ.
202018	ΔΠΘ
202019	ΕΥΕΛΠΙΔΩΝ
202020	ΧΑΡΟΚΟΠΕΙΟ ΠΑΝ.