

PL/SQL



# Βασική Δομή

---

**DECLARE**

*/\* Μεταβλητές και τύποι \*/*

**BEGIN**

*/\* Διαδικασίες και εντολές PL/SQL \*/*

*/\* Είναι το μόνο τμήμα που απαιτείται \*/*

**EXCEPTION**

*/\* Εντολές χειρισμού λαθών \*/*

**END;**

# Βασική Δομή (2)

---

- Για να εκτελέσουμε ένα πρόγραμμα PL/SQL γράφουμε μετά τον κώδικα:
  - Μια γραμμή που περιέχει μόνο μία τελεία ( . )
  - Μια γραμμή με την λέξη `run;`
- Η εκτέλεση γίνεται:
  - Γράφοντας τις εντολές στην γραμμή εντολών της `sqlplus`
  - Γράφοντας τον κώδικα σε ένα αρχείο και καλώντας το από την `sqlplus` (`@codefile.sql`)

# Μεταβλητές και τύποι

---

- Τύποι μεταβλητών:
  - NUMBER (ακέραιοι ή πραγματικοί αριθμοί)
  - VARCHAR(n)
    - Γενικά, τύποι της sqlplus για προσδιορισμό των στηλών ενός πίνακα
  - BOOLEAN
  - Ίδιος με τον τύπο μίας στήλης ενός πίνακα
  - Εγγραφή με πολλά πεδία (record)
- Η αρχική τιμή μεταβλητών είναι NULL
- Οι μεταβλητές δεν πρέπει να έχουν ίδιο όνομα με στήλες πινάκων
- Στις μεταβλητές ανατίθεται τιμή με τον τελεστή :=

# Παραδείγματα

---

**Beers (name, price)**

**DECLARE**

amount NUMBER(7,3);

price NUMBER;

myBeer varchar(30);

purchaseDate DATE;

isGood BOOLEAN;

yourBeer Beers.name%TYPE;

beerTuple Beers%ROWTYPE;

# Παραδείγματα (2)

---

**Beers (name, price)**

**DECLARE**

price NUMBER := 3;

myBeer varchar(30);

**BEGIN**

price := price + 2;

myBeer := 'Amstel';

**END;**

.

**run;**

# Εντολές SQL στην PL/SQL

---

- Εντολές SQL που επιτρέπονται:
  - SELECT
  - INSERT
  - UPDATE
  - DELETE
  
- Εντολές SQL που δεν επιτρέπονται:
  - CREATE
  - DROP
  - ALTER

# Παράδειγμα

---

- Αρχικός πίνακας T:

col1	col2
1	3
2	4

- PL/SQL κώδικας:

```
DECLARE
```

```
  a T.col1%TYPE;
```

```
  b T.col2%TYPE;
```

```
BEGIN
```

```
  SELECT col1, col2 INTO a,b FROM T WHERE col1 > 1;
```

```
  INSERT INTO T VALUES(b, a);
```

```
END;
```

```
.
```

```
run;
```



# Παράδειγμα (2)

---

- Πίνακας T μετά την εκτέλεση του PL/SQL κώδικα:

col1	col2
1	3
2	4
4	2

# Εντολές SQL στην PL/SQL (2)

---

- Η PL/SQL μας επιτρέπει να εκτιμήσουμε τι συνέβη με την εκτέλεση μιας SQL εντολής:
  - `SQL%ROWCOUNT`: ο αριθμός των γραμμών που επεξεργάστηκε η εντολή
  - `SQL%FOUND`: TRUE αν επεξεργάστηκε τουλάχιστον μία γραμμή
  - `SQL%NOTFOUND`: TRUE αν δεν επεξεργάστηκε καμία γραμμή

# Παράδειγμα

---

Beers (name, price)  
delHistory (table\_name, delRows, date)

**DECLARE**

rowsDeleted NUMBER;

**BEGIN**

**DELETE FROM** Beers **WHERE** price < 2;

rowsDeleted := SQL%ROWCOUNT;

**INSERT INTO** delHistory

**VALUES**('Beers', rowsDeleted, SYSDATE);

**END;**

.

**run;**

# Χειρισμός λαθών

---

- Σύνταξη:

**EXCEPTION**

**WHEN** exception-identifier **THEN** actions;

- Αναγνωριστικά λαθών στην PL/SQL (παραδείγματα):
  - NO\_DATA\_FOUND
  - TOO\_MANY\_ROWS

# Παράδειγμα

---

Beers (name, price)  
Error\_Table (comments)

**DECLARE**

    b\_name Beers.name%TYPE;

    b\_price Beers.price%TYPE;

**BEGIN**

**SELECT** name, price **INTO** b\_name, b\_price

**FROM** Beers **WHERE** price **BETWEEN** 2 **AND** 5;

    .....

**EXCEPTION**

**WHEN** NO\_DATA\_FOUND **THEN**

**INSERT INTO** error\_table **VALUES**('no such price');

**END;**

.

**run;**

# Έλεγχος ροής

---

## □ Η εντολή IF

### ■ Σύνταξη:

```
IF <condition> THEN <statement-list>
```

```
ELSE <statement-list>
```

```
END IF;
```

ή

```
IF <condition1> THEN <statement-list>
```

```
ELSIF <condition_2> THEN <statement-list>
```

.....

```
ELSIF <condition_n> THEN <statement-list>
```

```
ELSE <statement-list>
```

```
END IF;
```

# Παράδειγμα

---

T (e, f)

```
DECLARE
  a NUMBER;
  b NUMBER;
BEGIN
  SELECT e,f INTO a, b FROM T WHERE e > 1;
  IF b=1 THEN
    INSERT INTO T VALUES(b, a);
  ELSE
    INSERT INTO T VALUES(b+10, a+10);
  END IF;
END;
.
run;
```

# Έλεγχος ροής (2)

---

## □ Η εντολή LOOP

- Σύνταξη:

LOOP

    <loop-body> /\* λίστα από PL/SQL εντολές \*/

END LOOP;

- Τουλάχιστον μία από τις εντολές στο loop-body πρέπει να είναι της μορφής:

    EXIT WHEN <condition>;



# Παράδειγμα

---

T (e, f)

```
DECLARE
  i NUMBER := 1;
BEGIN
  LOOP
    INSERT INTO T VALUES(i, i);
    i := i + 1;
    EXIT WHEN i > 100;
  END LOOP;
END;
.
run;
```

# Έλεγχος ροής (3)

---

## □ Η εντολή WHILE

### ■ Σύνταξη:

```
WHILE <condition> LOOP  
    <loop-body>  
END LOOP;
```

# Παράδειγμα

---

T (e, f)

```
DECLARE
  i NUMBER := 1;
BEGIN
  WHILE i <= 100 LOOP
    INSERT INTO T VALUES(i, i);
    i := i + 1;
  END LOOP;
END;
.
run;
```

# Έλεγχος ροής(4)

---

## □ Η εντολή FOR

### ■ Σύνταξη:

```
FOR <var> IN <start>..<finish> LOOP  
    <loop-body>  
END LOOP;
```

- Η μεταβλητή var είναι οποιαδήποτε μεταβλητή, είναι τοπική στο σώμα του βρόγχου και δεν χρειάζεται να δηλωθεί
- Τα <start> και <finish> είναι σταθερές

# Παράδειγμα

---

**T (e, f)**

```
DECLARE
  i NUMBER;
BEGIN
  FOR i IN 1..100 LOOP
    INSERT INTO T VALUES(i, i);
  END LOOP;
END;
.
run;
```

# Cursors

---

- Μεταβλητή όπου αποθηκεύουμε  $n$ -άδες κάποιας σχέσης (πίνακας ή αποτέλεσμα ερώτησης)
- Μπορούμε να διαβάσουμε σειριακά και να επεξεργαστούμε τις τιμές κάθε στοιχείου

# Cursors (2)

---

- Οι κέρσορες ελέγχονται μέσω τεσσάρων διαφορετικών τύπων `actions`:
  - `DECLARE`: δηλώνει των κέρσορα και την ερώτηση `SQL` που θα εκτελεστεί
  - `OPEN`: εκτελεί την ερώτηση `SQL` και «γεμίζει» τις γραμμές του κέρσορα
  - `FETCH`: φορτώνει τιμές τις τρέχουσας γραμμής του κέρσορα σε μεταβλητές και μετακινεί τον δείκτη του κέρσορα στην επόμενη γραμμή
  - `CLOSE`: κλείνει τον κέρσορα (ελευθερώνει τη μνήμη)

# Cursors (3)

---

- Δήλωση του κέρσορα - σύνταξη:  
CURSOR identifier IS query-expression;
  - identifier είναι το όνομα του κέρσορα.
  - query-expression είναι μια SELECT εντολή της SQL.
- Άνοιγμα κέρσορα - σύνταξη:  
OPEN cursor-identifier;
- Κλείσιμο κέρσορα - σύνταξη:  
CLOSE cursor-identifier;
- FETCH - σύνταξη:  
FETCH cursor-identifier INTO var, var, .....



# Παίρνοντας πληροφορίες για τους cursors

---

- Τιμές ελέγχου:
  - FOUND: TRUE αν η τελευταία εντολή FETCH βρήκε μία γραμμή στον κέρσορα
  - NOTFOUND: το αντίθετο του FOUND
  - ROWCOUNT: επιστρέφει τον αριθμό των γραμμών (records) του κέρσορα
  - ISOPEN: TRUE αν ο κέρσορας είναι ανοικτός (έχει εκτελεστεί η εντολή OPEN)

# Παράδειγμα

---

T (e, f)

```
DECLARE
  a T.e%TYPE;
  b T.f%TYPE;
  CURSOR Tcursor IS
    SELECT e, f FROM T
    WHERE e < f
    FOR UPDATE;          /*FOR UPDATE OF table*/
BEGIN
  OPEN Tcursor;
  LOOP
    FETCH Tcursor INTO a, b;
    EXIT WHEN Tcursor%NOTFOUND;
    DELETE FROM T WHERE CURRENT OF Tcursor;
    INSERT INTO T VALUES(b , a);
  END LOOP;
  CLOSE Tcursor;
END;
```

.

Run;

# Διαδικασίες (Procedures)

---

- Οι διαδικασίες είναι παρόμοιες με αυτές σε άλλες γλώσσες προγραμματισμού
- Παράδειγμα:

- SQL

```
CREATE TABLE T(a INTEGER, b VARCHAR(10));
```

- PL/SQL

```
CREATE PROCEDURE addtuple1(i NUMBER) AS
```

```
BEGIN
```

```
  INSERT INTO T VALUES(i, 'xxx');
```

```
END addtuple1;
```

```
.
```

```
run;
```

# Διαδικασίες (Procedures) (2)

---

- Σύνταξη:

```
CREATE PROCEDURE proc-name
    (<param-list>)AS
<local-var-declarations>
BEGIN
    <procedure-body>
END proc_name;
.
run;
```

# Διαδικασίες (Procedures) (3)

---

- `proc-name`: όνομα της διαδικασίας
- `<param-list>`: `param-id type, ...`
  - `param-id` είναι το όνομα παραμέτρου
  - `type` είναι ο τύπος της μεταβλητής
    - Οι τύποι των μεταβλητών δεν επιτρέπεται να έχουν όρια

# Διαδικασίες (Procedures) (4)

---

- Η εντολή run στο τέλος απλώς δημιουργεί τη διαδικασία, δεν την εκτελεί
- Η εκτέλεση γίνεται σε ένα άλλο PL/SQL block ως εξής:

```
BEGIN
```

```
    addtuple1(99);
```

```
END;
```

```
.
```

```
run;
```

# Παράδειγμα

---

T (a, b)

```
CREATE PROCEDURE addtuple2(  
    x T.a%TYPE,  
    y T.b%TYPE)
```

```
AS
```

```
BEGIN
```

```
    INSERT INTO T VALUES(x, y);
```

```
END addtuple2;
```

```
.
```

```
run;
```

```
BEGIN
```

```
    addtuple2(10, 'abc');
```

```
END;
```

```
.
```

```
run;
```

# Συναρτήσεις (Functions)

---

- Σύνταξη:

```
CREATE FUNCTION fun-name (<param-list>
                        RETURN <return-type>
                        AS
```

```
<local-var-declarations>
```

```
BEGIN
```

```
    <procedure-body>
```

```
END fun-name;
```

```
.
```

```
run;
```



# Συναρτήσεις (Functions) (2)

---

- `fun-name`: όνομα της συνάρτησης
- `<param-list>`: όπως στην διαδικασία
- `<return-type>`: τύπος μεταβλητής που επιστρέφεται
- Στο σώμα της συνάρτησης (`<procedure-body>`) πρέπει να υπάρχει η εντολή `RETURN <expression>`;

# Διαδικασίες - Συναρτήσεις

---

- Για να βρούμε ποιες διαδικασίες ή συναρτήσεις έχουμε δημιουργήσει πρέπει να γράψουμε τα εξής στην SQL:

```
SELECT object_type, object_name  
FROM user_objects  
WHERE object_type = 'PROCEDURE'  
OR      object_type = 'FUNCTION' ;
```

- Για να διαγράψουμε μια διαδικασία ή συνάρτηση:  

```
DROP procedure <procedure_name>;  
DROP function <function_name>;
```

# Ανακαλύπτοντας λάθη

---

- Η PL/SQL δεν σας λέει πάντα για τα λάθη μεταγλώττισης
- Δίνει ένα απλό μήνυμα "procedure created with compilation errors"
- Για να δούμε τα λάθη δίνουμε την εντολή:  
`show errors procedure <procedure_name>;`

# Τυπώνοντας Μεταβλητές (Παράδειγμα)

---

**DECLARE**

a NUMBER;

b NUMBER;

**BEGIN**

**SELECT** e,f **INTO** a, b **FROM** T **WHERE** e > 1;

**IF** b=1 **THEN**

**INSERT INTO** T **VALUES**(b, a);

dbms\_output.put\_line('η τιμή του b είναι: ' || b);

**ELSE**

**INSERT INTO** T **VALUES**(b+10, a+10);

dbms\_output.put\_line('η τιμή του b είναι: ' || b);

**END IF;**

**END;**

.

run;

T (e, f)

**Για ενεργοποίηση εκτύπωσης μεταβλητών:**

**set serveroutput on**