Data Advanced

6. PL/SQL Iteratie

Koen Bloemen

Sander De Puydt

DE HOGESCHOOL MET HET NETWERK

Elfde-Liniestraat 24, 3500 Hasselt, www.pxl.be

PL/SQL Iteratie

- Overzicht
- Basic loop
- WHILE loop
- FOR loop
- CURSOR loop
- Nesting loops

Overzicht

- EXIT
  - (Vroegtijdige) beëindiging van de loop
  - Controle wordt doorgegeven aan statement onmiddellijk na de loop
- CONTINUE
  - Slaat de rest van de statements over
  - Hertest onmiddellijk de voorwaarde van de iteratie
- *(GOTO)*
  - Geeft controle door aan het gelabelde statement
  - Sterk afgeraden om te gebruiken => program flow beter programmeren!
- DBMS_OUTPUT
  - Ingebouwde package om output te tonen in het DB management systeem
  - Alle tekst moet tussen single quotes (‘) behalve variabelen
  - SQL*Plus: login.sql script uitbreiden met SET SERVEROUTPUT ON FORMAT WRAPPED

Basic LOOP – voorbeeld IF THEN

```
DECLARE
   teller NUMBER := 0;
BEGIN
   LOOP
      teller := teller + 1;
      IF teller > 5 THEN
         EXIT;
      END IF;
      dbms_output.put_line('Binnen loop: ' || teller);
   END LOOP;
   dbms_output.put_line('Einde loop: ' || teller);
END;
```

```
Binnen loop: 1
Binnen loop: 2
Binnen loop: 3
Binnen loop: 4
Binnen loop: 5
Einde loop: 6
```

Basic LOOP – voorbeeld EXIT WHEN

```
DECLARE
   teller NUMBER := 0;
BEGIN
   LOOP
      teller := teller + 1;
      IF teller > 5 THEN
         EXIT;
      END IF;
      EXIT WHEN teller > 5;
      dbms_output.put_line('Binnen loop: ' || teller);
   END LOOP;
   dbms_output.put_line('Einde loop: ' || teller);
END;
```

```
Binnen loop: 1
Binnen loop: 2
Binnen loop: 3
Binnen loop: 4
Binnen loop: 5
Einde loop: 6
```

WHILE LOOP – voorbeeld

- Een WHILE loop wordt gebruikt om de conditie te controleren VOORDAT code wordt uitgevoerd
- Loop variabele moet gedeclareerd en geïnitialiseerd zijn

```
DECLARE
   teller NUMBER := 0;
BEGIN
   WHILE teller < 5 LOOP
      teller := teller + 1;
      dbms_output.put_line('While loop: ' || teller);
   END LOOP;
END;
```

```
While loop: 1
While loop: 2
While loop: 3
While loop: 4
While loop: 5
```

FOR LOOP – voorbeeld

- Een FOR loop wordt gebruikt om een vast aantal keer code te doorlopen
- Declaratie van loop variable is NIET nodig buiten de loop
- Verhoging van loop variable gebeurt **automatisch met 1**
- Als je wil verlagen moet je het REVERSE keyword gebruiken

```
DECLARE
   teller NUMBER := 0;
BEGIN
   FOR teller IN 1..5 LOOP
      teller := teller + 1;
      dbms_output.put_line('For loop: ' || teller);
   END LOOP;
END;
```

```
For loop: 1
For loop: 2
For loop: 3
For loop: 4
For loop: 5
```

```
BEGIN
   FOR teller IN REVERSE 1..5 LOOP
      dbms_output.put_line('For loop: ' || teller);
   END LOOP;
END;
```

```
Reverse for loop: 5
Reverse for loop: 4
Reverse for loop: 3
Reverse for loop: 2
Reverse for loop: 1
```

CURSOR LOOP – voorbeeld

- Een speciale FOR loop om data uit te lezen
- Gevolgd door een SELECT statement
- Resultaat van query worden in cursor bijgehouden

```
BEGIN
   FOR mdw IN (SELECT mnr, naam, voorn FROM Medewerkers WHERE mnr < 7600) LOOP
      dbms_output.put_line('Cursor loop: ' || mdw.mnr || '-' || mdw.naam || ' ' || mdw.voorn);
   END LOOP;
END;
```

```
Cursor loop: 7369-CASPERS JANA
Cursor loop: 7499-ALLARD NELE
Cursor loop: 7521-DEFOUR THOMAS
Cursor loop: 7566-JACOBS EMMA
```

CURSOR LOOP – voorbeeld

- Je kan een CURSOR ook opslaan om de leesbaarheid van je code te verbeteren
- Syntax: *CURSOR c_variable_name IS SELECT column FROM table;*

```
DECLARE
    CURSOR c_mdw IS SELECT mnr, naam, voorn FROM medewerkers WHERE mnr < 7600;
BEGIN
    FOR mdw IN c_mdw LOOP
        dbms_output.put_line('Cursor loop: ' || mdw.mnr || '-' || mdw.naam || ' ' || mdw.voorn);
    END LOOP;
END;
```

```
Cursor loop: 7369-CASPERS JANA
Cursor loop: 7499-ALLARD NELE
Cursor loop: 7521-DEFOUR THOMAS
Cursor loop: 7566-JACOBS EMMA
```

Nesting LOOPs – voorbeeld

```
DECLARE
   teller_i NUMBER := 0;
   teller_j NUMBER := 0;
BEGIN
   <<outer_loop>>
   LOOP
      teller_i := teller_i + 1;
      EXIT outer_loop WHEN teller_i > 2;
      dbms_output.put_line('Outer loop: ' || teller_i);
      teller_j := 0;
      <<inner_loop>>
      LOOP
         teller_j := teller_j + 1;
         EXIT inner_loop WHEN teller_j > 3;
         dbms_output.put_line('Inner loop: ' || teller_j);
      END LOOP inner_loop;
   END LOOP outer_loop;
END;
```

```
Outer loop: 1
Inner loop: 1
Inner loop: 2
Inner loop: 3
Outer loop: 2
Inner loop: 1
Inner loop: 2
Inner loop: 3
```

- Een loop kan gelabeled worden
- Moet tussen << en >>
- Elk type loop kan genest worden (basic, for, while, cursor)