Kurze Einführung in PL/SQL für Oracle

1. Aufbau eines PL/SQL-Blockes

PL/SQL ist blockorientiert, wobei sich ein Block in die folgenden drei Teile gliedert:

- Deklaration der Variablen, eingeleitet durch das Schlüsselwort **DECLARE** (dieser Block ist optional)
- Anweisungsblock, eingeleitet durch das Schlüsselwort **BEGIN**, das Blockende wird gekennzeichnet durch die Anweisung **END;**
- Fehler-Behandlung (Exception-Handling) innerhalb des Anweisungsblocks, eingeleitet durch das Schlüsselwort **EXCEPTION** (dieser Block ist optional)

PL/SQL-Blöcke werden als Skript-Datei entwickelt und können mit START in SQL*PLUS aufgerufen werden. Das Ende eines PL/SQL-Blocks wird im Skript mit einem Slash (/) gekennzeichnet. Jede Anweisung muss mit einem Semikolon abgeschlossen sein. Zwischen Groß- und Kleinschreibung wird nicht unterschieden. Ein Block sollte nur bis zu 200 Zeilen enthalten.

```
DECLARE
     <Datendeklaration>
BEGIN
     <SQL- und PL/SQL-Code>
EXCEPTION
     <PL/SQL-Code>
END;
/
```

Wird eine stored procedure durch eine **`create or replace`**-Anweisung deklariert, so wird das Schlüsselwort `DECLARE` *nicht* benötigt!

2. Datendeklaration

Jede Variable, die in einem PL/SQL-Block benutzt wird, muß zuvor deklariert werden. Die folgenden Datentypen stehen u.a. zur Verfügung: VARCHAR2(n), NUMBER, INTEGER, DATE, ROWID und BOOLEAN.

Beispiel:
```
DECLARE
        s varchar2(5);
        n dec(6,2);
        x boolean;
```

Diese Variablen haben den Anfangswert NULL, wenn sie nicht initialisiert werden.

Initialisierung von Variablen durch **Zuweisungsoperator :=**

Beispiele

Textvariablen: `s varchar2(5) := 'ABCDE';`

Numerische Variable: `n1 dec(6,2) not null := 0;`

(hier mit der optionalen Ergänzung, dass der Wert nie NULL sein darf.)

Boolesche Variable: `x boolean := TRUE;`
Arithmetischer Ausdruck: `n2 dec(6,2) := n1+100.00;`
Mit dem reservierten Wort `CONSTANT` kann man Konstanten definieren:

```
n CONSTANT := 100;
```

Statt einer Zuweisung kann auch das Schlüsselwort `DEFAULT` benutzt werden:

```
n3 dec(10,2) DEFAULT 0;
```

Die Spaltennamen der Datenbank-Tabelle müssen nicht deklariert werden.

Mit dem Zusatz `%TYPE` kann man den Datentyp einer Tabellespalte casten:

```
n e_nr%TYPE;
```

3. Geschachtelte PL/SQL-Blöcke

Schachtelungen sind im Anweisungsblock und Exception-Teil möglich. Eine Variable ist lokal bzgl. eines Blocks, wenn sie in ihm deklariert ist. Eine Variable ist global bzgl. eines Blocks, wenn sie nicht in ihm, aber in einem übergeordneten Block deklariert ist.

4. Zuweisungen

Zuweisungsoperator: `:=`

Zuweisung mittels select: Hierbei werden Werte über Spaltennamen aus der Datenbank ermittelt, die mit Hilfe des Schlüsselwortes `INTO` der entsprechenden Variablen zugewiesen werden.

Beispiel: `select count(*) into anzahl from eteil;`

5. Arithmetische Operatoren und Vergleichsoperatoren
stehen entsprechend der jeweiligen SQL-Operatoren zur Verfügung.

6. Kontrollstrukturen: Verzweigungen

```
if    <Bedingung>
then  <Anweisungsblock 1>
else  <Anweisungsblock 2>
end if;
```

7. Kontrollstrukturen: Schleifen

Es gibt drei verschieden Schleifenarten:

```
loop ... end loop;                                  bedingungslose Schleife
for  <Zählbereich> loop ...      end loop;          Zählschleife
while <Bedingung > loop ...      end loop;          bedingte Schleife
```

Anweisung `EXIT`: Die bedingungslose Schleife ist eine unendliche Schleife, die man nur mit der EXIT-Anweisung verlassen kann. Soll von einer inneren Schleife in eine äußere Schleife terminiert werden, muss zusätzlich noch eine Marke gesetzt werden. Sie muss unmittelbar vor der zu verlassenden Schleife stehen. Die Marke wird mit doppelten spitzen Klammern definiert (<<marke>>) und mit GOTO angesprungen. Die Anweisung EXIT kann zusätzlich noch eine WHEN-Klausel haben, die eine Bedingung enthält.

Beispiel für eine Zählschleife:

```
for i in 1..10 loop
    Z:=z+10;
end loop;
```

8. Zusätzliche Cursor-Eigenschaften im Zusammenhang mit PL/SQL

Die Cursor-Declaration bei Oracle lautet `cursor <cname> is ...`

Attribute zur **Kontrolle des Cursors** (mit cursorname = c):

`c%isopen`	prüft, ob der Cursor geöffnet ist.
`c%found`	hat den Wert TRUE, wenn ein fetch-Befehl auf einen Datensatz zeigt.
`c%notfound`	hat den Wert TRUE, wenn ein fetch-Befehl nach dem letzten Datensatz des Cursors referenziert.
`c%rowcount`	gibt an, wieviele Zeilen mit dem open-Befehl in den Cursor-Speicherbereich geladen wurden.

Die **Strukturvariable `%ROWTYPE`** entspricht vom Datentyp der Record-Struktur, die vom jeweiligen Cursor oder von einer Tabelle der DB verwaltet wird.

Beispiel: Deklaration einer Strukturvariablen `satz c%rowtype;`

Verwendung eines Cursors *ohne* explizites `open`, `fetch` und `close`:

```
FOR <Variable> IN <Cursor> LOOP
   ... <Variable>.<Feld_i> ...
END LOOP;
```

(vgl. auch Beispielskript *esql4.sql*)

9. Bildschirmausgabe aus einem PL/SQL-Block

Die Bildschirmausgabe aus einem PL/SQL-Block erfolgt mit Hilfe des Standardpaketes `dbms_out` von Oracle-Prozeduren.

Beispiel:

```
declare
    wert number(5);
begin
   select count(*) into wert from lieferant;
   dbms_output.put_line('Textstring' || wert);
end;
/
```

Mit dem concatination-Operator || können Textstrings miteinander verknüpft bzw. Textstrings mit Zahlenwerten verknüpft werden, deren Wert dabei automatisch in Stringkonstanten umgewandelt wird.
Achtung: Die SQL*PLUS-Variable `serveroutput` muß auf `ON` gesetzt sein. Anzeige des aktuellen Wertes: `SHOW ALL`.
Setzen des Wertes `ON`: **`set serveroutput on`**

10. Hinweise zur Erstellung eines Skripts unter SQL*PLUS

Der **&-Parameter** bewirkt, dass bei Skriptaufruf ohne Parameterübergabe bei jedem Aufruf eines &-Parameters ein aktueller Parameterwert vom Benutzer interaktiv erfragt wird.

Kommentare setzt man

- bei einzeiligen Kommentaren durch zwei Bindestriche : **--**
- bei Kommentarblöcken durch **`/*`** zur Kennzeichnung des Kommentaranfangs und **`*/`** zur Kennzeichnung des Kommentarendes.

11. Stored Procedure & Trigger in Oracle

➢ Stored Procedure (Oracle)

Syntax Deklaration

```
create or replace procedure <proc_name> (<Parameter 1> <Datentyp>, ...)
as
<PL/SQL-Block>
end <proc_name>;
/
```

Enthält die procedure einen *Deklarationsteil*, so wird dieser in der Oracle -Version bei stored procedures **nicht** durch das Schlüsselwort `declare` eingeleitet!

Beispiel (Gehaltserhöhung um Betrag sal_incr für Angestellten emp_id)

```
create or replace procedure sal_raise (emp_id number,sal_incr number)
as
begin
  update angestellter
  set gehalt = gehalt + sal_incr
  where per_nr = emp_id;

  if sql%notfound then
     raise_application_error
          (-20011,'ungueltige Persnr. ' || to_char(emp_id));
  end if;
end sal_raise;
/
```

Der Slash muss die Deklaration der Procedure abschließen, wenn die Procedure über ein SQL-Skript, z.B. aus SQL*Plus, deklariert wird.

Syntax Aufruf – im interaktiven SQL *(z.B. in SQL*PLUS), ohne Parameterklammern, falls keine Parameter übergeben werden!*

```
exec <proc_name> (<aktueller Parameter 1, ...>);
```

Syntax Aufruf – im PL/SQL-Block*, ohne Parameterklammern, falls keine Parameter übergeben werden! :*

```
<proc_name> (<aktueller Parameter 1, ...>);
```

Syntax Löschen *einer Stored Procedure*

```
drop procedure <proc_name>;
```

➢ Trigger (Oracle) – vgl. Skript

Syntax Löschen *eines Trigger*

```
drop trigger <trigger_name>;
```

„Stored procedure / Trigger mit Kompilierungsfehler generiert.": Diese wenig aussagekräftige Meldung erscheint ggf. bei der create procedure- bzw. create trigger-Anweisung.

Mit **show errors** `procedure <procname>;` bzw. `show errors trigger <triggername>`; werden die detaillierten Fehlermeldungen angezeigt.