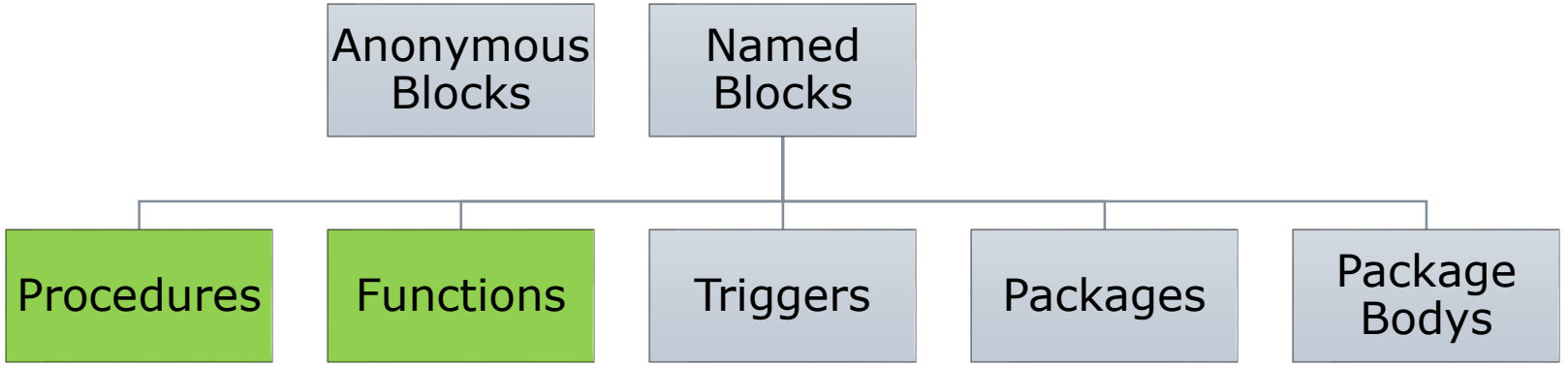
Datenbank und Informationssysteme

Procedures and Functions

Procedures and Functions



- Named PL/SQL blocks which are stored in the database
- As they are named and stored in the database, they can be reused
- Are compiled once (anonymous blocks are compiled on every execution)
- Can use parameters

Procedure – Syntax

CREATE [OR REPLACE] PROCEDURE
Procedurename
[
(parameternamе1 **{in|out|in out}** datatype1**,**
parameternamе2 **{in|out|in out}** datatype2**,**
…)
]
IS|AS
[Declaration section]
BEGIN
Execution section
END;

Same structure as for anonymous blocks

Procedure – Syntax

- **Parameter**
  - Default mode is “in”
  - A default value for each parameter can be defined,
    eg. v_inputtext varchar2 DEFAULT 'World'
  - Size of datatype may **NOT** be defined, eg. instead of 'varchar2(30)' you **MUST** write only 'varchar2'
  - **In**: value is provided by calling component, parameter value may not be changed in procedure
  - **Out**: value will be returned to calling component and may be changed in procedure
  - **In out**: value is provided by calling component and may be changed in procedure. The value will be returned to the calling component
- **IS|AS**
  - Are synonyms

Procedure – Syntax

- If there are no parameters omit brackets

```
CREATE OR REPLACE PROCEDURE
    Test()
AS..
```

→ will produce a compiler error

- Procedures with no code must have a „NULL“ statement in the execution part

```
CREATE OR REPLACE PROCEDURE DoNothing
AS
BEGIN
    NULL;
END;
```

- Syntax for dropping a procedure is:

  DROP PROCEDURE *Procedurename;*

Example: Simple Procedure

```
CREATE OR REPLACE PROCEDURE

   Hello_Varchar
       (v_inputtext IN VARCHAR2 DEFAULT 'World')

AS -- instead of declare
BEGIN

    dbms_output.Put_line('Hello ' || v_inputtext);

END;
```

Invoking a Procedure

- In a PL/SQL Block by calling the procedure name in PLSQL block

```
BEGIN
    …
    Hello_Varchar('Klaus');
    …
END;
```

- With: **EXECUTE Procedurename** in the SQL Developer

```
EXECUTE Hello_Varchar('Klaus');
-- or for short: exec Hello_Varchar('Klaus')
--> Returns "Hello Klaus"


EXECUTE Hello_Varchar();
-- Returns "Hello World"
```

Example: Out Parameter

```
CREATE OR REPLACE PROCEDURE Calc_Circle_Area
 (p_radius INTEGER, p_area OUT DECIMAL)
AS
BEGIN
  p_area := p_radius*p_radius*3.141;
END;

CREATE OR REPLACE PROCEDURE print_area
AS
  v_area DECIMAL(10,2);
BEGIN
  Calc_Circle_Area(5, v_area);
  dbms_output.Put_line(v_area);
END;

EXEC print_area -- prints 79
```

Functions – Syntax

```
CREATE [OR REPLACE] FUNCTION
   Functionname
      [(parametername1 {in|out|in out} datatype1,
        parametername2 {in|out|in out} datatype2,
       …)]
RETURN datatype                    -- additional for functions
IS|AS
        [Declaration section]
BEGIN
        Execution section
        RETURN Expression; -- additional for functions
END;
```

Functions

- A function is a named PL/SQL block that accepts parameters and must return a **SINGLE OUTPUT VALUE**
- A function can be called
  - As a parameter of another procedure/function

    … `dbms_output.put_line(My_function('Hello'));`
  - Using in PL/SQL expression, to assign the result in a variable

    … `v_myvar := My_function('Hello');`
  - In a SQL statement*

```
SELECT vorname, nachname, My_function('Hello')
FROM    …
```

* As SQL supports not the same data types as PL/SQL there are some restrictions regarding the usage of functions in SQL

Example Concat Function

```
--#################################
-- Simple Function concatenating Strings
--#################################
CREATE OR REPLACE FUNCTION

String_Concat
     (v_inputtext1 IN VARCHAR2,
      v_inputtext2 IN VARCHAR2)
RETURN VARCHAR2

AS
BEGIN

    RETURN v_inputtext1 || v_inputtext2;

END;
```

Example Calling a Function

```
-- Setting a variable to the return value
-- of the function
DECLARE
  v_concated_string VARCHAR2(50);
BEGIN
  v_concated_string :=
      String_concat('Hello ', 'World');
  dbms_output.Put_line(v_concated_string);
END;
```

Example Calling a Function

```
-- Calling as a parameter of a function
BEGIN
  dbms_output.put_line
          (String_Concat('Hello ', 'World'));
END;
-- Writes to output "Hello World"


--############################
-- Using it in a SQL Statement
--############################
SELECT first_name,
       last_name,
       String_Concat(first_name, last_name)
FROM   employees;
```

Example Function „get_salary“

```
-- Returns salary of an employee
CREATE OR REPLACE FUNCTION get_salary
   (p_first_name employees.first_name%TYPE,
    p_last_name employees.last_name%TYPE)
RETURN  employees.salary%TYPE
AS
      v_salary employees.salary%TYPE;
BEGIN
      SELECT salary
      INTO   v_salary
      FROM   employees
      WHERE  first_name = p_first_name AND
             last_name = p_last_name;
      RETURN v_salary;
END;
```

Errorhandling in Development

- Even if there are compilation errors the procedure is created and stored in the database
- To see the error messages in a SQL Worksheet run the command

  SHOW ERRORS
- Best is to use the “compile editor” as error messages are more self explaining!!!!

Tasks (1/2)

- Erstellen Sie eine Funktion, die als Parameter die Abteilungsnummer bekommt und den Namen des Managers liefert
- Erstellen Sie eine Funktion, die die MitarbeiterNr bekommt und die Anzahl seiner Jobs liefert
- Erstellen Sie einen SQL Befehl welcher Ihnen den Mitarbeiter Nachnamen und die Anzahl seiner Jobs liefert. Verwenden Sie in diesem SQL die **vorher erstellte Funktion**.
- Erstellen Sie eine Prozedur, welche als Parameter eine MitarbeiterID und einen Wert für eine Gehaltserhöhung erwartet (absolut, nicht prozentuell). Die Prozedur soll dann das Gehalt dieses Mitarbeiters um den Wert der Gehaltserhöhung erhöhen

Testen Sie Ihre entwickelten Funktionen/Prozeduren!!!

Tasks (2/2)

- Erstellen Sie eine Funktion „Lsubstr“ (für Left Substring), die als Parameter einen String und ein Trennzeichen übergeben bekommt und den linken Teil des Strings zurückliefert bis zu diesem Zeichen (exklusive diesem Zeichen).
  Beispielsweise liefert LSubstr('klaus@gmail.com', '@') → „klaus“ zurück.
  - Überprüfen Sie Ihre Funktion, in dem Sie die Vorwahl der Telefonnummer (alle Zeichen vor dem „.“) der Mitarbeitertabelle mittels SQL und der erstellen Funktion ausgeben