

# Funktionen

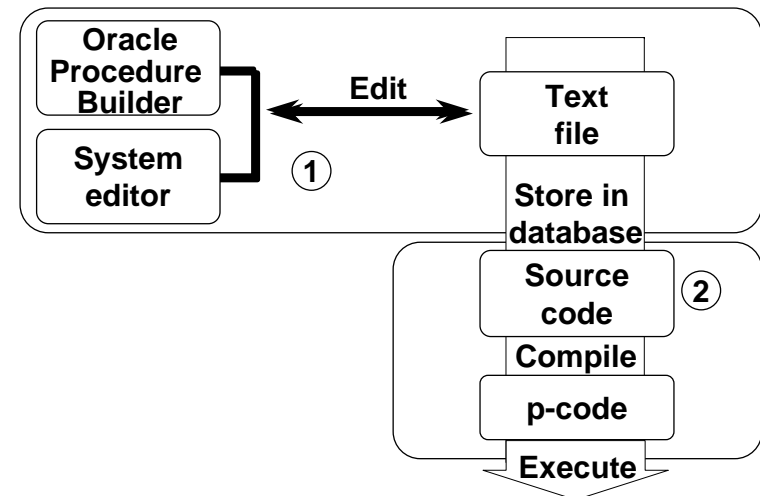
# Überblick über Stored Functions

- Eine Funktion ist ein benannter PL/SQL-Block, der einen Wert zurückgibt.
- Eine Funktion kann in der Datenbank als Objekt zur wiederholbaren Ausführung gespeichert werden.
- Eine Funktion kann als Teil eines Ausdruckes aufgerufen werden.

# Syntax zum Schreiben einer Funktion

```
CREATE [OR REPLACE] FUNCTION function_name
(argument1 [mode1] datatype1,
 argument2 [mode2] datatype2,
 . . .
RETURN datatype
IS | AS
PL/SQL Block;
```

# Schreiben einer Funktion



## Schreiben einer Stored Function mit SQL\*Plus

1. Text in einen Editor eingeben und speichern als Skriptfile mit FUNCTION (.sql/ extension).
2. Skriptfile ausführen zur Übersetzung in p-code und beide Files in der Datenbank speichern.
3. Funktion in einer Oracle – Umgebung aufrufen und ausführen.

SQL5 basierend auf OAI-Kurs Copyright © Oracle Corporation, 1998. All rights reserved.

ORACLE®

## Schreiben einer Stored Function mit SQL\*Plus: Beispiel

```
SQL> CREATE OR REPLACE FUNCTION get_sal
 2  (v_id IN emp.empno%TYPE)
 3  RETURN NUMBER
 4  IS
 5  v_salary emp.sal%TYPE :=0;
 6  BEGIN
 7  SELECT sal
 8  INTO v_salary
 9  FROM emp
10  WHERE empno = v_id;
11  RETURN (v_salary);
12 END get_sal;
13 /
```

SQL6 basierend auf OAI-Kurs Copyright © Oracle Corporation, 1998. All rights reserved.

ORACLE®

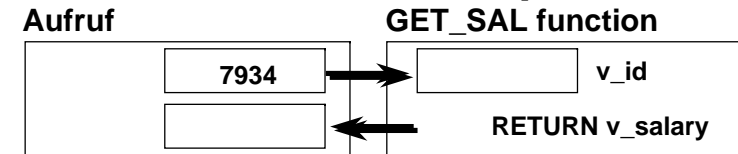
## Funktion ausführen

- Aufrufen einer Funktion als Teil eines PL/SQL Ausdruckes.
- Erzeugen einer Host-Variablen zum Speichern des Rückgabe-Wertes.
- Ausführen der Funktion. Die Hostvariable wird mit dem RETURN-Wert belegt sein.

SQL7 basierend auf OAI-Kurs Copyright © Oracle Corporation, 1998. All rights reserved.

ORACLE®

## Ausführen einer Funktion in SQL\*Plus: Beispiel



```
SQL> START get_salary.sql
Procedure created.
```

```
SQL> VARIABLE g_salary number
```

```
SQL> EXECUTE :g_salary := get_sal(7934)
PL/SQL procedure successfully completed.
```

```
SQL> PRINT g_salary
          G_SALARY
-----
                1300
```

SQL8 basierend auf OAI-Kurs Copyright © Oracle Corporation, 1998. All rights reserved.

ORACLE®

## Vorteile von benutzerdefinierten Funktionen in SQL Ausdrücken

- **Anfrageeffizienz:** Funktionen in der WHERE Klausel können Daten filtern
- **Zeichenketten manipulieren**
- **Liefert parallele Anfrageausführung**

SQL9 basierend auf OAI-Kurs Copyright © Oracle Corporation, 1998. All rights reserved.

ORACLE®

## Wo können benutzerdefinierte Funktionen aufgerufen werden?

- **In der Selectliste eines SELECT Kommandos**
- **Als Bedingung in der WHERE und HAVING-Klausel**
- **In CONNECT BY, START WITH, ORDER BY und GROUP BY Klauseln**
- **In VALUES Klauseln im INSERT Kommando**
- **In der SET Klausel im UPDATE Kommando**

SQL10 basierend auf OAI-Kurs Copyright © Oracle Corporation, 1998. All rights reserved.

ORACLE®

## Funktionsaufruf aus SQL-Ausdrücken: Restriktionen

- **Eine benutzerdefinierte Funktion muss eine stored function sein.**
- **Eine benutzerdefinierte Funktion muss eine ROW Funktion sein, nicht eine GROUP Funktion.**
- **Eine benutzerdefinierte Funktion hat nur IN Parameter, nicht OUT oder IN OUT.**
- **Datentypen müssen CHAR, DATE oder NUMBER sein, nicht die PL/SQL Typen wie BOOLEAN, RECORD oder TABLE.**
- **Rückgabotyp muss ein Oracle Server-interner Typ sein.**

SQL11 basierend auf OAI-Kurs Copyright © Oracle Corporation, 1998. All rights reserved.

ORACLE®

## Funktionsaufruf aus SQL-Ausdrücken: Restriktionen

- **INSERT, UPDATE oder DELETE Kommandos sind nicht erlaubt.**
- **Unterprogrammaufrufe, die die oben genannten Bedingungen nicht erfüllen, sind nicht erlaubt.**

SQL12 basierend auf OAI-Kurs Copyright © Oracle Corporation, 1998. All rights reserved.

ORACLE®

# Löschen von Funktionen

## Mit SQL\*Plus

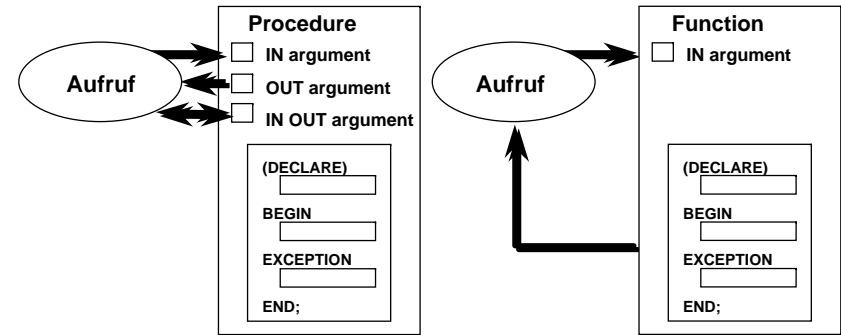
- **Syntax**

```
DROP FUNCTION function_name
```

- **Beispiel**

```
SQL> DROP FUNCTION get_salary;
Function dropped.
```

# Prozedur oder Funktion?



# Vergleich von Prozeduren und Funktionen

Prozedur	Funktion
Ausführung als PL/SQL Statement	Aufruf als Teil von einem Ausdruck
Kein RETURN Datentyp	Muss einen RETURN Datentype enthalten
Kann einen oder mehrere Werte zurückgeben	Kann einen Wert zurückgeben.

# Testen mit der DBMS\_OUTPUT Package

## Die DBMS\_OUTPUT Package:

- **Sammelt Informationen in einem Puffer**
- **Erlaubt Abfragen auf die Informationen aus dem Puffer**

# Benutzung von DBMS\_OUTPUT

## 1. Freigabe von SERVEROUTPUT

```
SQL> SET SERVEROUTPUT ON
```

## 2. Text in den Puffer schreiben

```
SQL> EXECUTE DBMS_OUTPUT.PUT ('testing 1 2 3')  
PL/SQL procedure successfully completed.
```

## 3. Ausgabe aus dem Puffer

```
SQL> EXECUTE DBMS_OUTPUT.NEW_LINE  
testing 1 2 3  
PL/SQL procedure successfully completed.
```

oder

```
SQL> EXECUTE DBMS_OUTPUT.PUT_LINE  
('testing 1 2 3 testing 4 5 6')  
testing 1 2 3 testing 4 5 6  
PL/SQL procedure successfully completed.
```

SQL17 basierend auf OAI-Kurs Copyright © Oracle Corporation, 1998. All rights reserved.

ORACLE®

# Testen einer Stored Procedure mit DBMS\_OUTPUT

```
CREATE OR REPLACE PROCEDURE remove_and_process_emp  
(v_id IN s_emp.id%TYPE)  
IS  
  v_no_of_employees NUMBER; v_dept_id s_emp.dept_id%TYPE;  
BEGIN  
  ...../* SELECT statement to count the number of employees */ .....  
  dbms_output.put_line('no of employees in dept ' || v_dept_id || ' is '  
  || v_no_of_employees);  
  IF v_no_of_employees = 1 THEN  
    dbms_output.put_line('As there is only 1 person left in the dept.');    dbms_output.put_line('fire_emp then remove department');    fire_emp (v_id);  
    remove_dept (v_dept_id);  
  ELSE  
    dbms_output.put_line('As there is more than 1 person left in the dept, ');  
    dbms_output.put_line('fire_emp only, will be fired');    fire_emp (v_id);  
  END IF;  
  COMMIT;  
EXCEPTION  
  WHEN NO_DATA_FOUND THEN  
    RAISE_APPLICATION_ERROR (-20200, 'Employee does not exist.');END remove_and_process_emp;
```

SQL18 basierend auf OAI-Kurs Copyright © Oracle Corporation, 1998. All rights reserved.

ORACLE®

## Output testen

Die Ausgabe bei Ausführung der Prozedur von der vorigen Folie

```
SQL> SET SERVEROUTPUT ON  
SQL> EXECUTE remove_and_process_emp (22)  
no of employees in dept 44 is 2  
as there is more than 1 person left in the dept,  
fire_emp only, will be fired.  
  
PL/SQL procedure successfully completed.
```

SQL19 basierend auf OAI-Kurs Copyright © Oracle Corporation, 1998. All rights reserved.

ORACLE®

## Ausgabe einer Fehlermeldung RAISE\_APPLICATION\_ERROR

- Vordefinierte Fehlermeldung, die einen Nicht-Standard-Fehlercode und eine Fehlermeldung aus einem im Data-Dictionary gespeicherten Unterprogramm zurückgibt.
- Kann nur ein Aufruf aus einem solchen Unterprogramm sein.

SQL20 basierend auf OAI-Kurs Copyright © Oracle Corporation, 1998. All rights reserved.

ORACLE®

# Ausgabe einer Fehlermeldung

## RAISE\_APPLICATION\_ERROR

### Syntax

```
raise_application_error (error_number,  
                        message[, {TRUE | FALSE}]);
```

- **error\_number** benutzerspezifische Fehlernummer für die Ausnahme zwischen -20000 und -20999.
- **message** benutzerspezifische Fehlermeldung für die Ausnahme - Zeichenkette bis zu 2048 Bytes lang.
- TRUE | FALSE optionaler Boolean Parameter. TRUE – der Fehler wird an die Fehlerliste angefügt  
FALSE (Standard) der Fehler überschreibt die **aktuelle Liste**

SQL21 basierend auf OAI-Kurs Copyright © Oracle Corporation, 1998. All rights reserved.

ORACLE®

# RAISE\_APPLICATION\_ERROR

## Beispiel

```
create or replace procedure new_emp  
(name emp.ename%type , ...)  
is  
begin  
if valid_deptno(abteilung) then  
insert into emp  
values(seq_empno.nextval, name, beruf, vorgesetzter,  
       sysdate, gehalt, provision, abteilung);  
else  
raise_application_error(-20002,'keine Abteilung');  
end if;  
commit;  
end new_emp;  
/
```

SQL22 basierend auf OAI-Kurs Copyright © Oracle Corporation, 1998. All rights reserved.

ORACLE®