

---

# Datenbank und Informationssysteme

---

DI. Hilbe Klaus, MBA

# Operatoren

---

## □ **Vergleiche**

=, >, <, >=, <=, <>\*

## □ **Logische**

AND, OR, NOT

## □ **Arithmetik**

+, -, \*, /

\*\* hoch bzw. Potenz, bspw. 2\*\*3 → 8

sqrt(a) Wurzel

\* Für ungleich gibt es auch folgende Operatoren: ~=, !=, ^=. Diese sind jedoch weniger bekannt und sollten daher nicht verwendet werden!

# Verschachtelte Blöcke und Scope von Variablen (1/2)

---

- Auf äußere Variablen kann im inneren Block zugegriffen werden

```
DECLARE -- äußerer Block
    v_test VARCHAR2(30) := 'Auesserer Block';
BEGIN
    BEGIN -- innerer Block
        dbms_output.Put_line('Text im inneren Block: '
            || v_test);
    END;
    dbms_output.Put_line('Text im auesseren Block: '
        || v_test);
END;
```

# Verschachtelte Blöcke und Scope von Variablen (2/2)

---

- Lokal definierte Variablen „ziehen“ vor „globalen“ Variablen

```
DECLARE -- äußerer Block
    v_test VARCHAR2(30) := 'Auesserer Block';
BEGIN
    DECLARE -- innerer Block
        v_test VARCHAR2(30) := 'Innerer Block';
    BEGIN
        dbms_output.Put_line('Text im inneren Block: '
            || v_test);
    END;
    dbms_output.Put_line('Text im auesseren Block: '
        || v_test);
END;
```

# PL/SQL und SQL

---

- ❑ DDL (Data Definition Language) Statements können nicht in PL/SQL ausgeführt werden, da diese nicht vorkompiliert werden können (werden zur Laufzeit kompiliert und ausgeführt)
- ❑ Select und DML (Data Manipulation Language) Befehle können hingegen in PL/SQL Programme eingebunden und die Ergebnisse weiterverarbeitet werden

# Zuweisung von Ergebnissen von SQL Statements an Variablen

---

Syntax:

```
SELECT Spaltenname [, Spaltenname...]  
INTO   Variablenname [, Variablenname...]  
FROM   Tabellename [WHERE Bedingung];
```

- ❑ Es können mehrere Variablen mit einem SQL-Statement befüllt werden
- ❑ Variablen können auch im SQL Teil verwendet werden
- ❑ Die Anzahl und Reihenfolge in den Spaltennamen muss jener der Variablen entsprechen
- ❑ Die Abfrage muss **GENAU EINE** Zeile zurück liefern, ansonsten wird ein Fehler generiert

# Zuweisung von Ergebnissen von SQL Statements an Variablen - Beispiel

---

Gehalt und Einstellungsdatum von Hr. „Kochhar“

DECLARE

```
v_gehalt          INTEGER;  
v_einstellungsdatum DATE;
```

BEGIN

```
SELECT salary, hire_date  
INTO    v_gehalt, v_einstellungsdatum  
FROM    employees  
WHERE   last_name = 'Kochhar';  
dbms_output.Put_line('Gehalt: '  
    || To_char(v_gehalt));  
dbms_output.Put_line('Einstellungsdatum: '  
    || To_char(v_einstellungsdatum));
```

END;

# Zuweisung von Ergebnissen von SQL Statements an Variablen - Beispiel

---

Man kann den Datentyp der Variable an den Spaltentyp der Tabelle binden.

Syntax:

*Variablenname* *Tabellenname*.*Spaltenname***%TYPE**

DECLARE

```
v_gehalt          employees.salary%TYPE;  
v_einstellungsdatum employees.hire_date%TYPE;
```

BEGIN

```
SELECT salary, hire_date  
INTO      ...
```

# DML Statements mit Variablen – Beispiel

---

Gehalt von Hr. „Kochhar“ um 10 erhöhen

```
DECLARE
```

```
    v_gehaltserhoehung INTEGER := 10;
```

```
    v_nachname VARCHAR2(100) := 'Kochhar';
```

```
BEGIN
```

```
    UPDATE employees
```

```
    SET    salary = salary + v_gehaltserhoehung
```

```
    WHERE last_name = v_nachname;
```

```
END;
```

# Auflösung von Variablen in SQL Befehlen

---

```
DECLARE
```

```
    last_name VARCHAR2(30) := 'Kochhar';
```

```
BEGIN
```

```
    SELECT last_name INTO last_name
```

```
    FROM employees
```

```
    WHERE last_name = last_name;
```

```
    dbms_output.Put_line('Nachname: ' || last_name);
```

```
END;
```

- ❑ Variablen können gleich lauten wie Tabellen- oder Spaltennamen → dies führt zu Mehrdeutigkeit!
- ❑ Die Auflösung erfolgt in folgender Reihenfolge:  
Spaltenname → Variablenname → Tabellename
- ❑ Durch einführen einer Namenskonvention sollte Mehrdeutigkeit verhindert werden!

# Bsp. für eine Namenskonvention für Variablen

---

Prefix	Beschreibung
v_	Lokale Variable
g_	Globale Variable
k_	Konstante
c_	Cursor

## Beispiele:

```
v_Mitarbeitername  -- lokale Variable für Mitarbeitername  
k_pi                -- Konstante für PI  
c_laender           -- Cursor für die Ländertabelle
```

# Aufgaben – Schreiben Sie das Ergebnis immer in eine Variable und geben Sie diese aus!

---

- Welcher Mitarbeiter wurde am 13.1.2000 eingestellt (Variable Einstellungsdatum, Mitarbeitername)
- Geben Sie die Regionsbezeichnung von „Argentina“ aus (Variable für „Country“ und Region)
- Löschen Sie das Land mit der Bezeichnung „HongKong“ (Variable für Land)
- Erhöhen Sie das Gehalt aller direkten Mitarbeiter des Managers „Steven King“ um 10% (Variable Name des Managers)
- Bonus:
  - Welcher Mitarbeiter ist am längsten in der Firma beschäftigt