

# Vertiefung

Abfragesprache SQL in ORACLE

# Literatur

## Oracle Online-Dokumentation – Database Administration

[http://docs.oracle.com/database/121/nav/portal\\_4.htm](http://docs.oracle.com/database/121/nav/portal_4.htm)

- [Database SQL Language Reference](#)
- [Database SQL Language Quick Reference](#)

# AGENDA

- 1 Wiederholung**
- 2 Transaktionen in SQL**
- 3 DML – Vertiefung**
  - 2.1 Join und Union
  - 2.2 Unterabfragen
  - 2.3. Funktionen

# Wiederholungs-Übungen

Im folgenden Skript werden die Beispiel-Tabellen Vertreter, Verkauf und Artikel verwendet.

1. Führe das SQL-Skript dbVertreter.sql in SQLPlus aus, um die Tabellen anzulegen und mit Beispieldaten zu füllen.
2. Mache dich mit den Tabellen vertraut bzgl.:
  - Spalten und Datentypen
  - Beziehung der Tabellen zueinander
  - Datensätzen in den Tabellen und was diese bedeuten
3. Zeige alle Vertreter mit Namen und VNR an, die eine Provision von weniger als 7% erhalten.
4. Bei welchen Artikeln (Name und Artikelnummer) liegt der Preis über 100?

# Wiederholungs-Übung

5. Zeige alle Vertreter an, die vor dem 01.01.1980 geboren sind und deren Name ein i beinhaltet.
6. Füge dich als Vertreter in die Tabelle Vertreter ein mit einer Provision von 6% und der Mitarbeiternummer 7777.
7. Füge für deinen Vertreter einen Verkauf mit UNR 1014 hinzu.  
Der neue Datensatz soll abbilden, dass heute 22 Wintermäntel vom Vertreter mit VNR 7777 verkauft wurden.
8. Ändere den Preis eines Stiefels auf 88,90.
9. Füge der Tabelle Vertreter eine Spalte bonus hinzu, in die ein bis zu vierstelliger, ganzzahliger Wert eingetragen werden soll.
10. Setze den Bonus für alle Vertreter auf 500.
11. Ändere den Datentyp für die Spalte vname in der Tabelle vertreter auf VARCHAR2(20);
12. An welchen Tagen wurden Verkäufe getätigt? Erzeuge folgende Ausgabe:  
DATUM  
-----  
27.06.15  
28.06.15



# 1. Transaktionen

Eine Transaktion umfasst eine Reihe von SQL-Statements (Update, Insert..), die als Einheit ausgeführt werden (ganz oder gar nicht)

## Transaktion besitzt **ACID-Eigenschaften**

- A Atomicity** → alles oder nichts
- C Consistency** → Transaktion überführt einen konsistenten DB- Zustand in einen ebenfalls konsistenten DB-Zustand
- I Isolation** → Transaktion läuft isoliert ab
- D Durability** → Ergebnisse einer Transaktion werden dauerhaft in DB gespeichert

Transaktionen können in SQL mit einem COMMIT/ROLLBACK-Befehl gesteuert werden

## **COMMIT**

- Alle Operationen (Update, Insert..) werden in der Datenbank dauerhaft gespeichert

## **ROLLBACK**

- Alle Operationen werden rückgängig gemacht und nicht in der Datenbank gespeichert



## 2. DML-Vertiefung

Join, Union und Aggregatfunktionen

# 2.1 JOIN und UNION

- Daten werden oft auf mehrere Tabellen verteilt, um Redundanzen zu vermeiden (siehe Normalisierung)
- Um diese verteilten Daten in einer SELECT-Abfrage wieder vereinigt anzuzeigen, werden die Tabellen über einen so genannten JOIN-Befehl verbunden
- Man unterscheidet unter anderem zwischen
  - INNER JOIN
  - OUTER JOIN
  - LEFT JOIN
  - RIGHT JOIN

# Tabellen-Join

- Bisher wurden aus den Select-Statements in der Übung nur Daten aus der Tabelle Emp ausgewählt.
- Ziel ist es jetzt, zu einem Mitarbeiter Namen, Empno, Dname und Loc anzuzeigen
- Die Verbindung der Tabelle EMP zur Tabelle DEPT kann über DEPTNO hergestellt werden.

EMPNO	ENAME	....	DEPTNO
7369	SMITH	..-	20
7499	ALLEN	...	30

DEPTNO	DNAME	LOC
20	RESEARCH	DALLAS
30	SALES	CHICAGO



# INNER JOIN

```
SELECT *  
FROM emp  
INNER JOIN dept ON emp.deptno = dept.deptno;
```

**EMP**

EMPNO	ENAME	....	DEPTNO
7782	CLARK	...	10
7839	KING	...	10
7369	SMITH	..-	20
7499	ALLEN	...	30
<del>7902</del>	<del>FORD</del>	<del>...</del>	<del>NULL</del>

**DEPT**

DEPTNO	DNAME	LOC
10	ACCOUNTING	DALLAS
20	RESEARCH	DALLAS
30	SALES	CHICAGO
<del>40</del>	<del>OPERATIONS</del>	<del>BOSTON</del>



**JOIN Ergebnis**

EMPNO	ENAME	....	DEPTNO	DEPTNO	DNAME	LOC
7782	CLARK	...	10	10	ACCOUNTING	DALLAS
7839	KING	...	10	10	ACCOUNTING	DALLAS
7369	SMITH	..-	20	20	RESEARCH	DALLAS
7499	ALLEN	...	30	30	SALES	CHICAGO

INNER JOIN liefert nur Datensätze, bei denen Deptno links einer Deptno rechts zugeordnet werden kann und fügt diese zusammen.

# LEFT JOIN

```
SELECT *  
FROM emp  
LEFT OUTER JOIN dept ON emp.deptno = dept.deptno;
```

**EMP**

EMPNO	ENAME	....	DEPTNO
7782	CLARK	...	10
7839	KING	...	10
7369	SMITH	..-	20
7499	ALLEN	...	30
7902	FORD	...	NULL

**DEPT**

DEPTNO	DNAME	LOC
10	ACCOUNTING	DALLAS
20	RESEARCH	DALLAS
30	SALES	CHICAGO
<del>40</del>	<del>OPERATIONS</del>	<del>BOSTON</del>

**JOIN Ergebnis**

EMPNO	ENAME	....	DEPTNO	DEPTNO	DNAME	LOC
7782	CLARK	...	10	10	ACCOUNTING	DALLAS
7839	KING	...	10	10	ACCOUNTING	DALLAS
7369	SMITH	..-	20	20	RESEARCH	DALLAS
7499	ALLEN	...	30	30	SALES	CHICAGO
7902	FORD	...	NULL	NULL	NULL	NULL

LEFT JOIN liefert alle Datensätze der linken Tabelle und nur Datensätze der rechten Tabelle, wenn eine DEPTNO links einer DEPTNO rechts entspricht.

# RIGHT JOIN

```
SELECT *  
FROM emp  
RIGHT OUTER JOIN dept ON emp.deptno = dept.deptno;
```

**EMP**

EMPNO	ENAME	....	DEPTNO
7782	CLARK	...	10
7839	KING	...	10
7369	SMITH	..-	20
7499	ALLEN	...	30
<del>7902</del>	<del>FORD</del>	<del>...</del>	<del>NULL</del>

**DEPT**

DEPTNO	DNAME	LOC
10	ACCOUNTING	DALLAS
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

**JOIN Ergebnis**

EMPNO	ENAME	....	DEPTNO	DEPTNO	DNAME	LOC
7782	CLARK	...	10	10	ACCOUNTING	DALLAS
7839	KING	...	10	10	ACCOUNTING	DALLAS
7369	SMITH	..-	20	20	RESEARCH	DALLAS
7499	ALLEN	...	30	30	SALES	CHICAGO
NULL	NULL	...	NULL	40	OPERATIONS	BOSTON

RIGHT JOIN liefert alle Datensätze der rechten Tabelle und nur Datensätze der linken Tabelle, wenn eine DEPTNO rechts einer DEPTNO links entspricht.

# OUTER JOIN

```

SELECT *
FROM emp
FULL OUTER JOIN dept ON emp.deptno = dept.deptno;
  
```

**EMP**

EMPNO	ENAME	...	DEPTNO
7782	CLARK	...	10
7839	KING	...	10
7369	SMITH	..-	20
7499	ALLEN	...	30
7902	FORD	...	NULL

**DEPT**

DEPTNO	DNAME	LOC
10	ACCOUNTING	DALLAS
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON



**JOIN Ergebnis**

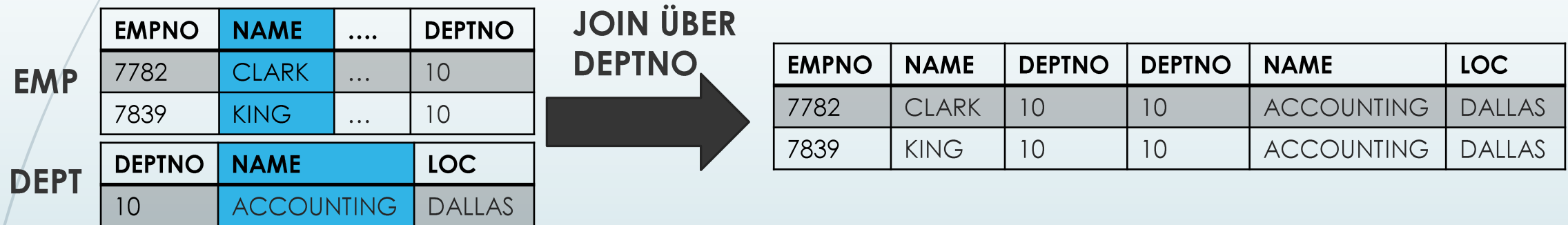
EMPNO	ENAME	...	DEPTNO	DEPTNO	DNAME	LOC
7782	CLARK	...	10	10	ACCOUNTING	DALLAS
7839	KING	...	10	10	ACCOUNTING	DALLAS
7369	SMITH	...	20	20	RESEARCH	DALLAS
7499	ALLEN	...	30	30	SALES	CHICAGO
7902	FORD	...	NULL	NULL	NULL	NULL
NULL	NULL	...	NULL	40	OPERATIONS	BOSTON

Outer JOIN liefert alle Datensätze beider Tabellen und verbindet Datensätze, wenn Deptno links einer Deptno rechts zugeordnet werden kann.



Enthalten zwei verbundene Tabellen gleiche Spaltennamen, muss bei der Auswahl der Spalten im SELECT-Statement die Tabelle angegeben werden, in der sich die Spalte befindet

**Beispiel:** Spalten NAME und DEPTNO sind in beiden Tabellen vorhanden



Angabe des Tabellennamens  
vor der Spalte

```
SELECT emp.name, dept.name  
FROM emp  
INNER JOIN dept ON emp.deptno = dept.deptno;
```

Alternativ kann für die Tabellen  
ein Alias vergeben werden

```
SELECT e.name, d.name  
FROM emp e  
INNER JOIN dept d ON e.deptno=d.deptno;
```

# UNION

- Über den UNION-Befehl können Datensätze mehrerer Abfragen in einem Ergebnis zusammengefasst werden
- Dabei werden mehrfach vorhandene Datensätze entfernt (alternativ UNION ALL: ohne Unterdrückung mehrfach vorkommender Datensätze)
- Dazu müssen Datentypen und Anzahl der Spalten beider Abfragen gleich sein

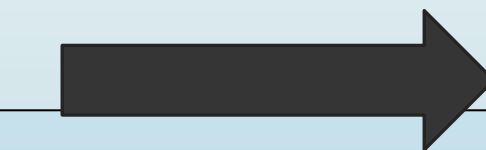
**Beispiel:** Es liegen zwei Tabellen EMPA und EMPB vor, in denen Mitarbeiterdaten für zwei Unternehmensbereiche separat erfasst werden

**EMPA**

EMPNO	ENAME	HIREDATE	SAL
7782	CLARK	03.03.1980	2300
7839	KING	01.01.1980	1500
7369	SMITH	02.02.1981	1700

**EMPB**

EMPNO	NAME	HDATE	SALARY
7499	ALLEN	15.03.1983	3500
7902	FORD	01.05.1991	2800



```
SELECT ename, sal  
FROM empa  
UNION  
SELECT name, salary  
FROM empb
```

**Ergebnis**

ENAME	SAL
CLARK	2300
KING	1500
SMITH	1700
ALLEN	3500
FORD	2800

Löse folgende Aufgaben durch JOIN der Tabellen

1. Erzeuge eine Ausgabe, in der jeder Vertreter (VNr und VName) mit all seinen Verkäufen (Datum, Anzahl und ANR) gelistet wird.
2. Reduziere die Ausgabe aus Aufgabe 1 auf Vertreter und ihre Verkäufe, bei denen mehr als 10 Artikel verkauft wurden
3. Ergänze die Ausgabe aus Aufgabe um den Namen und den Preis der jeweils verkaufen Artikel.

Verwende dazu zwei JOIN-Befehle wie unten dargestellt

```
SELECT ... FROM tabA  
INNER JOIN tabB ON (JOIN-Bedingung)  
INNER JOIN tabC ON (JOIN-Bedingung);
```

4. Zeige für den Verkäufer mit VNR 1010 alle Verkäufe (Anzahl, Datum) an, die sich auf Stiefel oder Wintermäntel beziehen und am 27.06.2015 getätigt wurden.



## 2.2 Unterabfragen

# Unterabfragen – Rückgabe eines Wertes

In einer Abfrage kann eine weitere Abfrage geschachtelt werden → **Unterabfrage**

## Anwendungsbsp:

Ausgabe aller Mitarbeiter, die  
in CLARKS Abteilung sind

EMPNO	ENAME	....	DEPTNO
7782	CLARK	...	10
7839	KING	...	10
...	...	...	...

### Option A

#### 1. In welcher Abteilung ist CLARK?

```
SELECT DEPTNO  
FROM emp  
WHERE ENAME = 'CLARK';  
→ DEPTNO 10
```

#### 2. Welche Mitarbeiter sind in Abteilung 10?

```
SELECT EMPNO, ENAME  
FROM emp  
WHERE deptno = 10;
```

### Option B

#### 1. Welche Mitarbeiter sind in CLARKS Abteilung?

Im WHERE-Bereich wird **10** durch eine Abfrage ersetzt, die die DEPTNO für CLARK zurückliefert

```
SELECT empno, ename  
FROM emp  
WHERE deptno= ( SELECT DEPTNO FROM emp  
WHERE ename = 'CLARK' ) ;
```

Unterabfrage muss wie hier **genau einen Wert** zurückliefern, dann können  
**Vergleichsoperatoren =, <, >, <=, >=** verwendet werden

# Unterabfragen – Rückgabe mehrerer Werte

Liefert die Unterabfrage eine Liste von Werten zurück, kann mit dem IN-Operator die Gleichheit mit einem Wert aus der Liste geprüft werden

## Anwendungsbsp:

Ausgabe aller Mitarbeiter, die in SMITHS oder CLARKS Abteilung sind

EMPNO	ENAME	...	DEPTNO
7782	CLARK	...	10
7839	KING	...	10
7369	SMITH	...	20
7788	SCOTT	...	20

### Option A

#### 1. In welcher Abteilung sind Smith und CLARK?

```
SELECT DEPTNO  
FROM emp  
WHERE ENAME IN ('CLARK', 'SMITH');  
→ DEPTNO 10, 20
```

#### 2. Welche Mitarbeiter sind in Abteilung 10?

```
SELECT EMPNO, ENAME  
FROM emp  
WHERE deptno IN (10,20);
```

### Option B

#### 1. Welche Mitarbeiter sind in SCOTTS Abteilung?

```
SELECT empno, ename  
FROM emp  
WHERE deptno IN  
( SELECT DEPTNO FROM emp  
  WHERE ename IN ('CLARK', 'SMITH') );
```

## Löse folgende Aufgaben mit Hilfe von Unterabfragen.

1. Zeige alle Vertreter (VNR, VNAME) an, die bereits Artikel mit der ANR 13 verkauft haben.
2. Zeige alle Artikel (ANR, ANAME) an, die der Vertreter mit der VNR 4321 am 27.06.2015 verkauft hat.
3. Zeige alle Vertreter (VNR, VNAME) an, die bereits Artikel verkauft haben, deren Preis über 100 € liegt.
4. Zeige alle Vertreter, die noch nie den Artikel mit der ANR 22 verkauft haben
5. Welche Vertreter (vnr) haben am 27.06.2015 mehr Artikel mit ANR 12 verkauft als der Vertreter mit VNR 4321?



## 2.3 Funktionen



# Aggregatfunktionen

Aggregatfunktionen werden auf eine Spalte angewendet.  
Sie fassen mehrere Werte zusammen.

- **Sum(spalte)**  
Summiert alle Werte einer Spalte auf
- **AVG(spalte)**  
Bildet den Durchschnitt über alle Spaltenwerte
- **COUNT(spalte) / COUNT(\*)**  
liefert Anzahl der Werte in der Spalte
- **MIN(spalte)/MAX(spalte)**  
gibt den kleinsten/größten Wert in der Spalte zurück

# Aggregatfunktion - SUM()

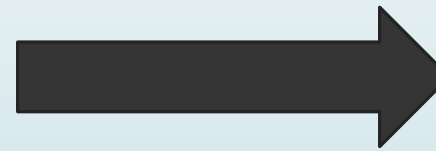
SUM(spalte) bildet die Summe über alle ausgewählten Werte in der Spalte

## Anwendungsbsp:

Wie hoch ist das Gesamteinkommen aller Mitarbeiter in Abteilung 10?

EMPNO	ENAME	SAL	DEPTNO
7782	CLARK	2450	10
7839	KING	5000	10
7369	SMITH	800	20

```
SELECT SUM(sal)
FROM emp
WHERE deptno = 10;
```

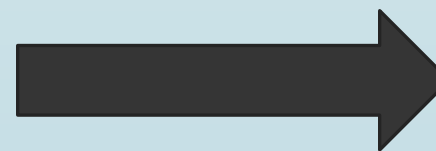


## Ergebnis

SUM(SAL)
7450

## Vergabe eines Alias für die Sum-Spalte

```
SELECT SUM(sal) summe
FROM emp
WHERE deptno = 10;
```



## Ergebnis

summe
7450

# Aggregatfunktion - AVG()

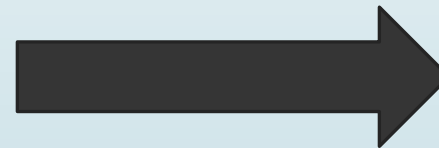
AVG(spalte) bildet den Durchschnitt aller ausgewählten Werte in der Spalte

## Anwendungsbsp:

Wie hoch ist das Durchschnittsgehalt aller Mitarbeiter?

EMPNO	ENAME	SAL	DEPTNO
7782	CLARK	2450	10
7839	KING	5000	10
7369	SMITH	800	20

```
SELECT AVG(sal) avgSAL  
FROM emp;
```



## Ergebnis

avgSAL
2750

# Aggregatfunktion - COUNT()

COUNT(spalte) gibt die Anzahl aller ausgewählten Werte in der Spalte zurück

COUNT(\*) gibt die Anzahl aller ausgewählten Datensätze zurück

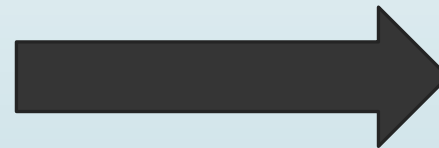
## Anwendungsbsp:

Wie viele Mitarbeiter arbeiten in Abteilung 10?

EMPNO	ENAME	SAL	DEPTNO
7782	CLARK	2450	10
7839	NULL	5000	10
7369	SMITH	800	20

## Anzahl Datensätze

```
SELECT COUNT (*)  
FROM emp  
WHERE deptno = 10;
```

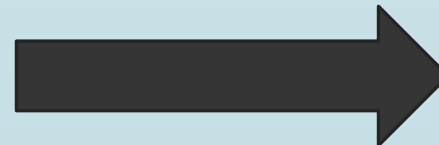


## Ergebnis

COUNT(*)
2

## Anzahl Werte für ENAME

```
SELECT COUNT (ename)  
FROM emp  
WHERE deptno = 10;
```



## Ergebnis

COUNT(*)
1

# Aggregatfunktion - MAX()/MIN()

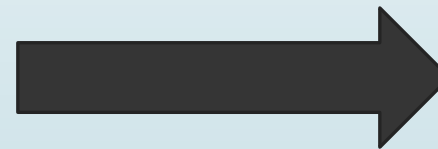
MIN(spalte)/MAX(spalte) gibt den kleinsten/größten Wert in dieser Spalte zurück

## Anwendungsbsp:

Zeige das höchste/niedrigste Gehalt?

EMPNO	ENAME	SAL	DEPTNO
7782	CLARK	2450	10
7839	KING	5000	10
7369	SMITH	800	20

```
SELECT MIN(sal) minSAL, MAX(sal) maxSAL  
FROM emp;
```



## Ergebnis

minSAL	maxSAL
800	5000

Für folgende Aufgaben werden die Tabellen Vertreter, Verkauf und Artikel verwendet.

1. Wie viel kostet der teuerste Artikel?
2. Wie hoch ist die durchschnittliche Provision aller Vertreter?
3. Wie viele Artikel hat der Vertreter Mueller insgesamt verkauft?
4. Wie viele Wintermäntel hat der Vertreter Jahred insgesamt verkauft?
5. Wessen Provision liegt über der durchschnittlichen Provision aller Vertreter?

**Tipp:** Nutze hier eine Unterabfrage, um den Durchschnitt aller Vertreter zu ermitteln

# Single Row Funktionen

- Single Row Funktionen können auf alle Werte in einer Spalte angewendet werden.
- können unter anderem in der SELECT-Liste oder in der WHERE-Bedingung verwendet werden
- Verändern Anzeige der einzelnen Werte in dieser Spalte und nicht die gespeicherten Daten
- **Funktionsgruppen**
  - Numerische Funktionen
  - Character Funktionen
  - Datumsfunktionen
  - NULL-bezogene Funktionen
  - usw. ( [vollständige Liste](#) )

# Numerische Funktionen

**FLOOR(n)**

-- rundet n auf die nächste Ganzzahl ab

**CEIL(n)**

-- rundet n auf die nächste Ganzzahl auf

**ROUND(n,p)**

-- rundet n auf p Stellen hinter dem Komma ab

**SQRT(n)**

-- gibt die Quadratwurzel aus n zurück



# Numerische Funktionen

## ➤ Numerische Funktion in der SELECT-Liste

EMP - Tabelle

EMPNO	SAL
7782	2450.33
7839	5000.70
7369	800.99



```
SELECT empno, FLOOR(sal), CEIL(sal)
FROM emp;
```

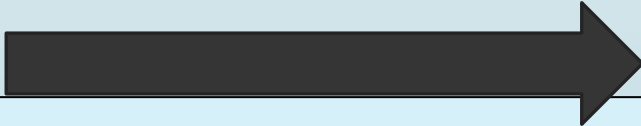
Abfrageergebnis

EMPNO	FLOOR(SAL)	CEIL(SAL)
7782	2450	2451
7839	5000	5001
7369	800	801

## ➤ Numerische Funktion in der WHERE-Clause

EMP - Tabelle

EMPNO	ENAME	SAL
7782	CLARK	2450.33
7839	KING	5000.70
7369	SMITH	800.99



```
SELECT empno, ename, sal
FROM emp
WHERE floor(sal) = 800;
```

Abfrageergebnis

EMPNO	ENAME	SAL
7369	SMITH	800.99

# Character-Funktionen

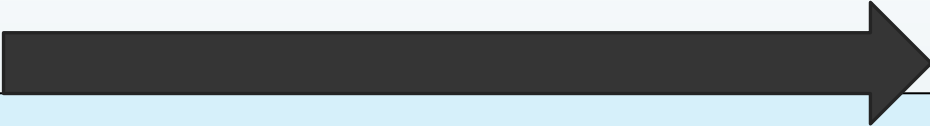
- UPPER(s)** -- wandelt alle Buchstaben der Zeichenkette s in Großbuchstaben um
- LOWER(s)** -- wandelt alle Buchstaben der Zeichenkette s in Kleinbuchstaben um
- LENGTH(s)** -- gibt die Länge der Zeichenkette s zurück
- SUBSTR(s,p,l)** -- Gibt einen Teil der Zeichenkette s zurück: beginnend ab dem Zeichen an Position p werden l Zeichen zurückgegeben

# Character-Funktionen

## ➤ Character-Funktion in der SELECT-Liste

EMP - Tabelle

EMPNO	ENAME
7782	CLARK
7839	KING
7369	SMITH



```
SELECT ename, LOWER(ename) low,  
SUBSTR(ename,2,3) sub,  
LENGTH(ename) len  
FROM emp;
```

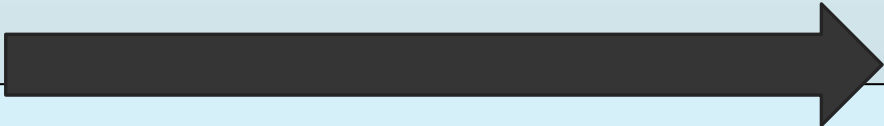
Abfrageergebnis

ENAME	low	sub	len
CLARK	clark	lar	5
KING	king	ing	4
SMITH	smith	mit	5

## ➤ Character-Funktion in der WHERE-Clause

EMP - Tabelle

EMPNO	ENAME
7782	CLARK
7839	KING
7369	SMITH



```
SELECT empno, ename  
FROM emp  
WHERE SUBSTR(ename,1,1) = 'S';
```

Abfrageergebnis

EMPNO	ENAME
7369	SMITH

## NULL-Wert bezogene Funktionen

NVL(a1, a2)                    -- wenn a1 NULL ist, wird a2 zurückgegeben,  
                                      wenn a1 nicht NULL ist, wird a1 zurückgegeben

## Datumsfunktionen

LAST\_DAY(d)                    -- gibt das Datum für den letzten Tag des Monats  
                                      zurück der das Datum d beinhaltet

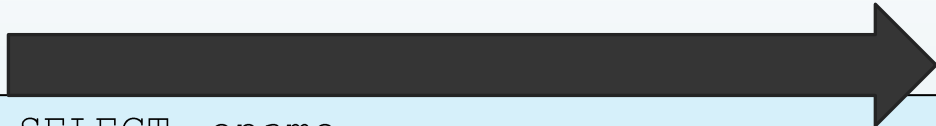
TO\_CHAR(d, FORMAT)            -- wandelt Datum d in Zeichenkette um  
                                      entsprechend der [Formatangabe](#).

# Sonstige Funktionen

## ➔ Funktionen in der SELECT-Liste

EMP - Tabelle

ENAME	HIREDATE
CLARK	09.06.81
KING	17.11.81
SMITH	NULL



```
SELECT  ename,  
        TO_CHAR(hiredate, 'mm.yy') dat,  
        LAST_DAY(hiredate) last,  
        NVL(hiredate, '01.01.1900') nulld  
FROM emp;
```

Abfrageergebnis

ENAME	dat	last	nulld
CLARK	06.81	30.06.81	09.06.81
KING	11.81	30.11.81	17.11.81
SMITH	NULL	NULL	01.01.00

## ➔ Funktionen in der WHERE-Clause

EMP - Tabelle

ENAME	HIREDATE
CLARK	09.06.81
KING	17.11.81
7369	SMITH



```
SELECT ename, hiredate  
FROM emp  
WHERE TO_CHAR(hiredate, 'mm') = '06';
```

Abfrageergebnis

ENAME	HIREDATE
CLARK	09.06.81

1. Zeige alle Vertreter mit Namen und VNR an, deren Name länger als 6 Zeichen ist
2. Zeige alle Vertreter, die im Juli oder im Mai geboren sind

3. Erzeuge folgende Ausgabe.:

VNR	VNAME_UP	PROV_GRD
1010	MUELLER	,1
3401	SCHMITZ	,1
5337	RITSCH	,1
4321	JAHRED	0

4. Zeige alle Artikel an, die mit 'Wi' beginnen. Löse diese Aufgabe mit der substr()-Funktion.

5. Erzeuge folgende Ausgabe:

VNR	VNAME	DAT
-----	-----	-----
1010	Mueller	03-05
3401	Schmitz	15-08
5337	Ritsch	23-07
4321	Jahred	27-12

6. Zeige alle Vertreter (VNR, Vname) an, die im selben Jahr geboren sind wie der Vertreter Jahred.
7. Erhöhe bei den Vertretern den Bonus um 300, die in einem Monat geboren sind, der 31 Tage hat.