

Vertiefung DML, DDL

Funktionen, GROUP BY, HAVING
Views anlegen und verwenden



Funktionen

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

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;
```

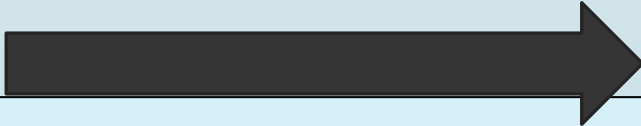
Abfrageergebnis

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

➤ Numerische Funktion in der WHERE-Bedingung

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

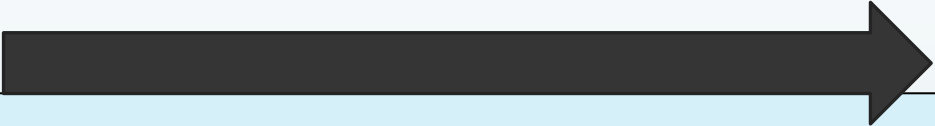
- UPPER(s)** -- wandelt alle Buchstaben der Zeichenkette s in Großbuchstaben um
- LOWER(s)** -- wandelt alle Buchstaben der Zeichenkette s in Kleinbuchstaben um
- LENGTH(s)** -- gibt die Länge der Zeichenkette s zurück
- SUBSTR(s,p,l)** -- Gibt einen Teil der Zeichenkette s zurück: beginnend ab dem Zeichen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

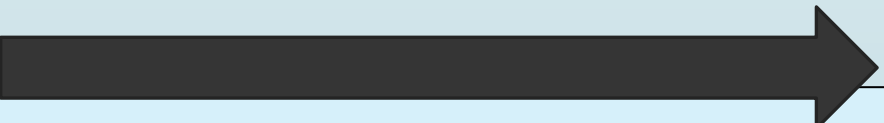
Abfrageergebnis

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

➤ Character-Funktion in der WHERE-Bedingung

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

TO_CHAR(d, FORMAT)

- wandelt Datum d in Zeichenkette um entsprechend der [Formatangabe](#).

TO_DATE(d, FORMAT)

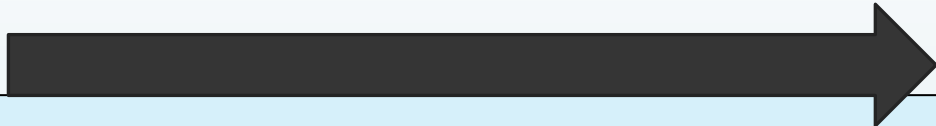
- wandelt String d in ein Datum um entsprechend der Formatangabe

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,  
        NVL(hiredate, '01.01.1900') nulld  
FROM emp;
```

Abfrageergebnis

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

➔ Funktionen in der WHERE-Clause

EMP - Tabelle

ENAME	HIREDATE
CLARK	09.06.81
KING	17.11.81
SMITH	NULL



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

Abfrageergebnis

ENAME	HIREDATE
CLARK	09.06.81

Datumsvergleiche in der Where-Bedingung

trunc() und **to_date()**

TRUNC()

- Date in Oracle beinhaltet immer eine Uhrzeit
- Um diese abzuschneiden, kann die Funktion **trunc()** verwendet werden

Beispiel: Abfrage aller Ausleihvorgänge, die am 01.01.2011 getätigt wurden

```
SELECT *  
FROM lib_rental  
WHERE trunc(lend) = '01.01.2011';
```

TO_DATE()

- Je nach Sessioneinstellungen wird ein anderes Datumsformat verwendet (z.B. erst Monat, dann Tag) und das Datum wird falsch bzw. gar nicht erkannt.
- Funktion **to_date('DATUM','FORMAT')** legt das Datumsformat fest

```
SELECT *  
FROM lib_rental  
WHERE trunc(lend) = to_date('01.01.2011','dd.mm.yyyy');
```

Aggregatfunktionen

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()

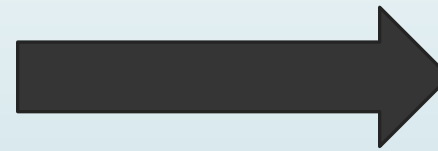
SUM(spalte) bildet die Summe über alle ausgewählten Werte in der Spalte

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;
```

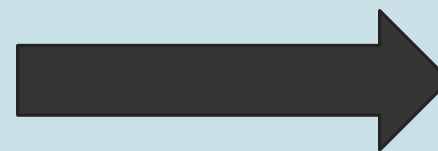


Ergebnis

SUM(SAL)
7450

Vergabe eines Alias für die Sum-Spalte

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



Ergebnis

summe
7450

Aggregatfunktion - AVG()

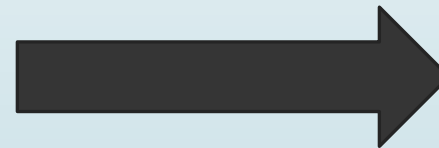
AVG(spalte) bildet den Durchschnitt aller ausgewählten Werte in der Spalte

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

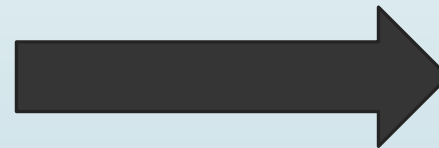
Anwendungsbsp:

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;
```

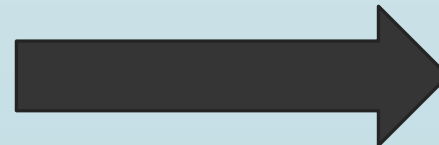


Ergebnis

COUNT(*)
2

Anzahl Werte für ENAME

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



Ergebnis

COUNT(ename)
1

Aggregatfunktion - MAX()/MIN()

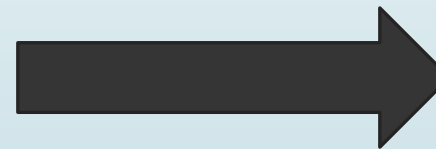
MIN(spalte)/MAX(spalte) gibt den kleinsten/größten Wert in dieser Spalte zurück

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

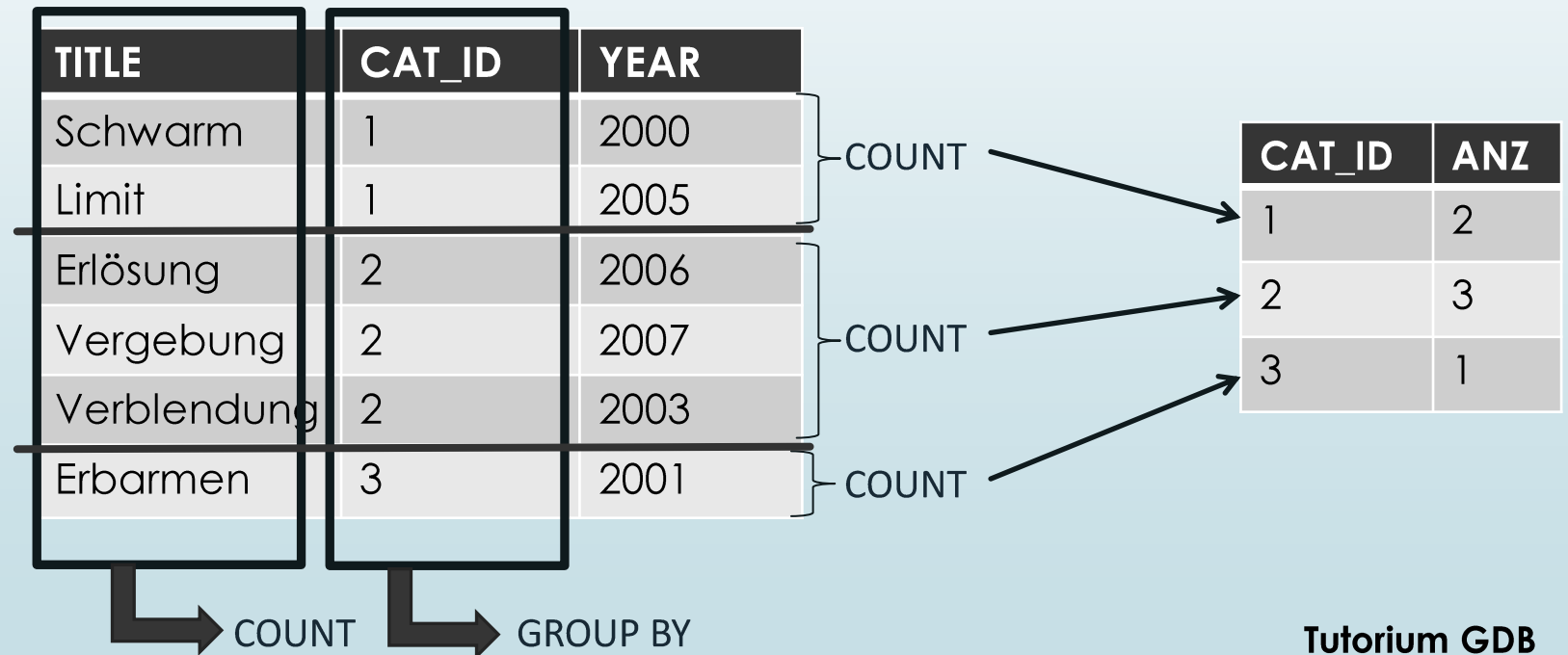
minSAL	maxSAL
800	5000

GROUP BY

- Gruppiert eine Spalte nach Spaltenwerten
- Gleiche Werte werden in einer Gruppe zusammengefasst
- Aggregatfunktion wird dann auf jede Gruppe angewendet

Beispiel: Wie viele Bücher gibt es in jeder Kategorie?

```
SELECT cat_id,  
       COUNT(title) anz  
FROM lib_book  
GROUP BY cat_id;
```



Having

- Schränkt die Ergebniswerte des Group By ein
- Wird angewendet auf jede Gruppe

Beispiel: In welchen Kategorien gibt es mehr als ein Buch?

CAT_ID	ANZ
1	2
2	3
3	1

HAVING COUNT(title) > 1

CAT_ID	ANZ
1	2
2	3

```
SELECT cat_id,  
       COUNT(title) anz  
FROM lib_book  
GROUP BY cat_id;
```

```
SELECT cat_id,  
       COUNT(title) anz  
FROM lib_book  
GROUP BY cat_id  
HAVING COUNT(title) > 1;
```



Views

SQL Views – Anlegen und Löschen

- View kann als virtuelle Tabelle angesehen werden
- Enthält keine Daten, sondern basiert auf einer gespeicherten SQL-Abfrage, die Daten aus einer oder mehreren Tabellen selektiert
- Ähnlich einer Tabelle kann eine Abfrage auf Views gemacht werden

Anlegen

```
CREATE (OR REPLACE) VIEW vname  
AS  
SELECT-STATEMENT;
```

```
CREATE OR REPLACE VIEW v_book_pub AS  
SELECT book_id, title, year, pub_name  
FROM lib_book b  
INNER JOIN lib_publisher p  
    ON (b.pub_id=p.pub_id);
```

Löschen

```
DROP VIEW viewname;
```

```
DROP VIEW v_book_pub;
```