Fragen zu den Oracle Funktionen

Die Tabelle „dual“

Die Tabelle „dual“ ist eine Tabelle, die aus nur einer Spalte und einer Zeile besteht. (Der Name und Inhalt ist dabei unwichtig.) Wir verwenden diese Tabelle, um bei Ausdrücken, die gleiche Ausgaben liefen, nicht unnötig viele Wiederholungen zu bekommen.

```
1  SELECT * FROM dual;
2
3
```

DUMMY
X

Beispiel:

Wir wollen mit CONCAT die Texte „Hallo“ und „ Welt“ (vor dem „W“ ist ein Leerzeichen) verknüpfen. Das machen wir mit `CONCAT('Hallo', ' Welt')`.

Aber welche Tabelle verwenden wir für die SELECT-Abfrage?

Angenommen wir verwenden die Tabelle **DEPT** mit den vier Zeilen (Abteilungen 10, 20, 30 und 40), dann bekommen wir für jede Zeile in DEPT einmal das gleiche Ergebnis von CONCAT(‚Hallo‘, , Welt‘) angezeigt.

```
1  SELECT CONCAT('Hallo', ' Welt') FROM dept
2
```

CONCAT('HALLO','WELT')
Hallo Welt
Hallo Welt
Hallo Welt
Hallo Welt

Die Tabelle **EMP** hat sogar 14 Zeilen und liefert das gleiche Ergebnis 14-mal:

```
1  SELECT CONCAT('Hallo', ' Welt') FROM emp
2
```

CONCAT('HALLO','WELT')
Hallo Welt
Hallo Welt
Hallo Welt
Hallo Welt
Hallo Welt
Hallo Welt
Hallo Welt
Hallo Welt
Hallo Welt
Hallo Welt
Hallo Welt
Hallo Welt
Hallo Welt
Hallo Welt

Die Tabelle **DUAL** hat nur eine Zeile. Deshalb wird das Ergebnis vom Ausdruck nur 1-mal angezeigt:

```
1  SELECT CONCAT('Hallo', ' Welt') FROM dual
2
```

CONCAT('HALLO','WELT')
Hallo Welt

Funktionen

TO_CHAR bei Zahlen

Die Funktion TO_CHAR konvertiert Zahlen und Datumswerte in einen Text. Der zweite Parameter ist der Format-String, der das Ausgabeformat der Zahl oder des Datums angibt.

```
1  /* TO_CHAR macht aus Zahlen und Datumswerten einen Text */
2  SELECT sal, TO_CHAR(.5, '0D99') FROM emp
```

SAL	TO_CHAR(.5,'0D99')
5000	0.50
2850	0.50
2450	0.50

Warum ich „sal“ in diese Abfrage inklduert habe, weiß ich nicht mehr. Der wichtige Teil ist die zweite Spalte „`TO_CHAR(.5, '0D99')`“.

Der erste Parameter (`.5`) ist die Kurzschreibweise von „0.5“ (Null komma fünf).

Der zweite Parameter (`'0D99'`) gibt das Ausgabeformat an. Dabei steht 0 für eine beliebige Zahl und 9 für eine beliebige Zahl, welche, wenn sie am Anfang steht, weggelassen wird. Wir kennen das aus der Mathematik: wir schreiben „800“ und nicht „0800“. Wir lassen führende Nullen weg. Die „0“ beim Format-String verhindert das.

Einfaches Beispiel aus der Tabelle EMP:
Zeigen Sie die Gehälter der Mitarbeiter mit Beruf „CLERK“ an.

```
1  SELECT sal, TO_CHAR(sal, '9999'), TO_CHAR(sal, '0999') FROM emp WHERE job = 'CLERK'
2
```

SAL	TO_CHAR(SAL,'9999')	TO_CHAR(SAL,'0999')
800	800	0800
1100	1100	1100
950	950	0950
1300	1300	1300

Der Format-String „9999“ macht aus 800 den Text „800“ (ohne führende Null). Der Format-String „0999“ zeigt bei 3-stelligen Zahlen die führende Null mit an und macht aus 800 den Text „0800“.

Zurück zum vorherigen Beispiel:

```
1  /* TO_CHAR macht aus Zahlen und Datumswerten einen Text */
2  SELECT sal, TO_CHAR(.5, '0D99') FROM emp
```

SAL	TO_CHAR(.5,'0D99')
5000	0.50
2850	0.50
2450	0.50

Die Zahl 0,5 wird gemäß „0D99“ formatiert.

1. Die führende Null wird angezeigt
2. „D“ ist das Dezimaltrennzeichen. Achtung: Das englische Format verwendet als Dezimaltrennzeichen den Punkt (.) und als Tausendertrennzeichen das Komma (,)!
   Der Format-String „0D99“ formatiert eine einstellige Zahl in eine Zahl mit führender Null und zwei Nachkommastellen. Hier „0.50“.

```
1  SELECT TO_CHAR(0.5, '0D99'), TO_CHAR(3.4567, '0D99'), TO_CHAR(12, '0D99') FROM dual
2
```

TO_CHAR(0.5,'0D99')	TO_CHAR(3.4567,'0D99')	TO_CHAR(12,'0D99')
0.50	3.46	#####

Spalte 1:

0.5 wird zum Text „0.50“. Achten Sie ebenfalls auf die Null am Ende der Nachkommastellen. Auch diese würden wir in der Mathematik weglassen. Hier zwingt der Format-String aber diese anzuzeigen.

Spalte 2:

Die Zahl 3,4567 wird zu einer Zahl mit zwei Nachkommastellen konvertiert. Dabei gelten die üblichen Regeln fürs Runden: Bis 5 abrunden. 5 und höher aufrunden. Wir wollen auf die zweite Nachkommastelle runden. Also wird die dritte Nachkommastelle (6) fürs Runden herangezogen und die Zahl aufgerundet zu „3.46“.

Spalte 3:

Unser Format-String konvertiert nur einstellige Zahlen. 12 ist aber zweistellig. Zahlen die mehr Stellen haben als im Format-String angegeben, können nicht konvertiert werden. Deshalb zeigt und Oracle da nur „#####“ an.

TO_CHAR bei Datums-Werten

Bei Datumswerten geben wir im Format-String die Formatierungszeichen für die Tage, Monate, Jahre, Stunden, Minuten, Sekunden, usw an. Diese entnehmt bitte dem PDF „Oracle-Funktionen (Testunterlage)“

```
SELECT hiredate,
    TO_CHAR(hiredate, 'D'),
    TO_CHAR(hiredate, 'DD'),
    TO_CHAR(hiredate, 'DDD')
FROM emp
```

HIREDATE	TO_CHAR(HIREDATE,'D')	TO_CHAR(HIREDATE,'DD')	TO_CHAR(HIREDATE,'DDD')
17-NOV-81	3	17	321
01-MAY-81	6	01	121
09-JUN-81	3	09	160
02-APR-81	5	02	092
19-APR-87	1	19	109
03-DEC-81	5	03	337

- D gibt den Wochentag als Zahl zurück. 1 = Sonntag, 2 = Montag, …, 7 = Samstag. Der 17.11.1981 war anscheinend ein Dienstag (3).
- DD gibt den Tag des Monats (mit führender Null) zurück. Der 17.11.1981 war der 17. Tag des Monats November.
- DDD gibt den Tag im Jahr zurück: Der 17.11.1981 war der 321 Tag im Jahr 1981.

Anmerkung:

Der 17.11.2024 ist der 322. Tag im Jahr, weil wir heuer ein Schaltjahr haben:

```
SELECT TO_DATE('2024-11-17', 'YYYY-MM-DD'), TO_CHAR(TO_DATE('2024-11-17', 'YYYY-MM-DD'), 'DDD')
FROM dual
```

TO_DATE('2024-11-17','YYYY-MM-DD')	TO_CHAR(TO_DATE('2024-11-17','YYYY-MM-DD'),'DDD')
17-NOV-24	322

TO_DATE macht aus einem Text („2024-11-17“) und dem richtigen Format-String ein Datum.

- YYYY: vierstellige Jahreszahl
- MM: zweistellige Monatszahl (01 bis 12)
- DD: zweistelliger Tag (01 bis 31)

TO_NUMBER

Konvertiert einen Text in eine gültige Zahl. Es gelten beim Format-String die Zeichen für Zahlen (0, 9, D, G).

```
1  SELECT TO_NUMBER('15', '99'), TO_NUMBER('5', '99'), TO_NUMBER('05', '99'), TO_NUMBER('05', '09') FROM dual;
2  SELECT TO_NUMBER('555', '99')  FROM dual;
3  SELECT TO_NUMBER('5', '09')  FROM dual;
4
5
```

TO_NUMBER('15','99')	TO_NUMBER('5','99')	TO_NUMBER('05','99')	TO_NUMBER('05','09')
15	5	5	5

Download CSV

```
ORA-01722: invalid number
More Details: https://docs.oracle.com/error-help/db/ora-01722

ORA-01722: invalid number
More Details: https://docs.oracle.com/error-help/db/ora-01722
```

Die erste Anweisung liefert gültige Zahlen zurück. Der Text im 1. Parameter passt zum Format-String.

Die zweite Anweisung: Der Text „555“ ist eine dreistellige Zahl, der Format-String erlaubt aber nur zweistellige Zahlen. → FEHLER

Die dritte Anweisung: Der Text „5“ hat KEINE führende Null. Der Format-String („09“) verlangt das aber. → FEHLER

Beispiel aus dem Unterricht:

```
1  SELECT TO_NUMBER('1,000999.50', '9G999999D99') FROM dual
2
```

TO_NUMBER('1,000999.50','9G999999D99')
1000999.5

Der Text hat ein englisches Tausendertrennzeichen (,) bei der Million (aber nicht nach der 3. Stelle). Der Format-String lautet deshalb „9G999999D99“ und erlaubt alle Zahlen mit Tausendertrennzeichen nach

der 6. Stelle, ohne Tausendertrennzeichen nach der 3. Stelle einem englischen Dezimaltrennzeichen und zwei Nachkommastellen. PASST. 😊

SOUNDEX

SOUNDEX gibt die Aussprache bzw. den Laut eines Wortes (hier in der englischen Sprache) als Code zurück.

```
1  SELECT SOUNDEX('KING') FROM dual;
2  SELECT SOUNDEX('KYNG') FROM dual;
```

SOUNDEX('KING')
K520

Download CSV

SOUNDEX('KYNG')
K520

Ein Beispiel für den deutschen Sprachgebrauch: Meier, Maier, Meyer, Mayer

```
1  SELECT SOUNDEX('Maier'), SOUNDEX('Meier'), SOUNDEX('Mayer'), SOUNDEX('Maeer') FROM dual;
```

SOUNDEX('MAIER')	SOUNDEX('MEIER')	SOUNDEX('MAYER')	SOUNDEX('MAEER')
M600	M600	M600	M600

Auch in der englischen Sprache klingen die Namen identisch.

TO_CHAR(hiredate, ‚D‘) < 16

Ich habe anscheinend im Unterricht folgende Anweisung geschrieben:

```
1  SELECT * FROM emp WHERE TO_CHAR(hiredate, 'D') < 16
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT	-	17-NOV-81	5000	-	10
7698	BLAKE	MANAGER	7839	01-MAY-81	2850	-	30
7782	CLARK	MANAGER	7839	09-JUN-81	2450	-	10
7566	JONES	MANAGER	7839	02-APR-81	2975	-	20
7788	SCOTT	ANALYST	7566	19-APR-87	3000	-	20
7902	FORD	ANALYST	7566	03-DEC-81	3000	-	20
7369	SMITH	CLERK	7902	17-DEC-80	800	-	20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100	-	20
7900	JAMES	CLERK	7698	03-DEC-81	950	-	30
7934	MILLER	CLERK	7782	23-JAN-82	1300	-	10

Was macht diese Anweisung?

Ich gebe alle Mitarbeiter zurück die an einem Wochentag (als Zahl) < 16 eingestellt wurden. Die Wochentage liegen zwischen 1 und 7. Damit sind alle Wochentage kleiner als 16. Die Anweisung ist zwar korrekt, aber völlig sinnfrei.

Hier hat sich ein Tippfehler eingeschlichen: Ich wollte bestimmt alle Mitarbeiter zurückgeben, die in der ersten Hälfte des Monats eingestellt wurden. Richtig wäre also:

```
1  SELECT * FROM emp WHERE TO_CHAR(hiredate, 'DD') < 16
2
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7698	BLAKE	MANAGER	7839	01-MAY-81	2850	-	30
7782	CLARK	MANAGER	7839	09-JUN-81	2450	-	10
7566	JONES	MANAGER	7839	02-APR-81	2975	-	20
7902	FORD	ANALYST	7566	03-DEC-81	3000	-	20
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7900	JAMES	CLERK	7698	03-DEC-81	950	-	30

Das Format-Zeichen „D“ gibt den Tag der Woche (1 bis 7) zurück.

Das Format-Zeichen „DD“ gibt den Tag des Monats (01 bis 31) zurück.

Frage:

TO_CHAR(hiredate, ‚DD‘) liefert die Texte „01“ bis „31“. Wieso kann dieser Text mit der Zahl 16 vergleichen werden?

Antwort:

Weil ORACLE den Text für den Vergleich automatisch in einer Zahl umwandelt. Brave Datenbank. 😊