Der Umgang mit Datumswerten ist alltäglich bei der Arbeit mit der Oracle-Datenbank. Dennoch treten gerade in diesem Bereich immer wieder Fragen auf, beispielsweise bei der Fehlermeldung, wenn man zum 31. Januar einen Monat hinzurechnet, oder wann man DATE und wann TIMESTAMP einsetzt. Der Artikel stellt die Unterschiede zwischen DATE und TIMESTAMP vor und geht im Detail auf den Umgang mit Datums-Intervallen ein.

DATE und TIMESTAMP in der Praxis

Carsten Czarski, ORACLE Deutschland B.V. & Co KG

TIMESTAMP ist der jüngere und mächtigere Datentyp. Andererseits kommt es immer noch vor, dass Werkzeuge nicht richtig mit TIMESTAMP umgehen. Aus diesem Grund wird das ältere DATE selbst auf 10*g*- oder 11*g*-Datenbanken immer noch recht häufig genutzt. Das ist kein Problem, denn solange man keine speziellen Features des TIMESTAMP benötigt, sind beide Typen gleichwertig:

- Die kleinste Einheit eines DATE ist die Sekunde, TIMESTAMP kann dagegen mit Millisekunden umgehen.
- TIMESTAMP unterstützt Zeitzonen, allerdings muss man dann den speziellen Datentyp TIMESTAMP WITH TIME ZONE verwenden. SQL-Funktionen wie FROM_TZ sind recht hilfreich beim Konvertieren zwischen Zeitzonen.

```
select sysdate, dump(sysdate) bytes from dual;

SYSDATE             BYTES
------------------- ----------------------------------------
21.09.2009 12:13:50 Typ=13 Len=8: 217,7,9,21,12,13,50,0
select systimestamp, dump(systimestamp) bytes from dual;
SYSTIMESTAMP
---------------------------------
BYTES
--------------------------------------------------------------------------
21.09.2009 12:15:21,413578 +02:00
Typ=188 Len=20: 217,7,9,21,10,15,21,0,16,179,166,24,2,0,5,0,0,0,0,0
```

Listing 1

```
select
  to_date('2009-09-21 14:45', 'YYYY-MM-DD HH24:MI') -
  to_date('2009-07-21 07:30', 'YYYY-MM-DD HH24:MI') INTERVAL
from dual;

INTERVAL
-----------------
     62,302083333
```

Listing 2

```
select
  to_timestamp('2009-09-21 14:45', 'YYYY-MM-DD HH24:MI') -
  to_timestamp('2009-07-21 07:30', 'YYYY-MM-DD HH24:MI') INTVAL
from dual;

INTVAL
---------------------------------------------
+000000062 07:15:00.000000000
```

Listing 3

Weder DATE noch TIMESTAMP speichern das Datum intern als Zahl, wie beispielsweise Unix es macht. Das kann man mit der DUMP-Funktion auch sehr einfach selbst nachvollziehen. In Listing 1 ist allerdings gut erkennbar, dass TIMESTAMP intern mit UTC arbeitet. „217,7“ wird hexadezimal als „D9 07“ ausgedrückt. Als 16-bit-Zahl wäre das „07D9“, was wiederum in dezimaler Schreibweise „2009“ ist. Man sieht also, dass die Information zu Jahr, Monat, Tag, Stunde, Minute und Sekunde explizit in den Bytes enthalten sind.

Datums-Intervalle ermitteln

DATE und TIMESTAMP erlauben Datums-Arithmetik: Ein Datums-Intervall zwischen zwei DATE-Werten wird als numerischer Wert, der zwischen zwei TIMESTAMP-Werten liegt, als Intervall-Datentyp ermittelt. Das numerische Intervall kann in einen Intervall-Datentyp konvertiert werden. Meist wird (schon aus historischen Gründen) mit der numerischen Variante gearbeitet: Dabei ist die Einheit, mit der man arbeitet, 1 Tag. Eine Stunde ist demnach 1/24 davon, eine Minute ist 1/1440 und so weiter. Möchte man also wissen, welche Zeitspanne zwischen zwei DATE-Werten liegt, so zieht man sie voneinander ab.

Die Ausgabe der Abfrage in Listing 2 ist, wie gesagt, in Tagen zu interpretieren. Es kommen also 62 und ein knapper Drittel-Tag heraus. Verwendet man dagegen TIMESTAMP, so bekommt man als Ergebnis einen Intervall-Datentyp (siehe Listing 3).

Der bei der Arbeit mit DATE zurückgegebene numerische Wert kann, wie schon erwähnt, in einen solchen Intervall-Datentyp konvertiert werden (siehe Listing 4). Es zeigt sich allerdings ein Rundungsproblem, deshalb wäre es besser, sich das Ergebnis der Datums-Arithmetik auch bei Verwendung von DATE direkt als Intervallwert zurückgeben zu lassen und den Umweg über NUMBER zu vermeiden (siehe Listing 5).

Zieht man zwei Datumswerte voneinander ab, so kann man also sowohl bei der Arbeit mit DATE als auch mit TIMESTAMP zu einem Intervall-Datentyp kommen.

Neben der schöneren Anzeige in SQL*Plus bietet dieser noch einige handfeste Vorteile: Die einzelnen Elemente eines Datums-Intervalls wie Tage, Minuten oder Stunden können mit der EXTRACT-Funktion ausgelesen werden (siehe Listing 6). Die Extraktion der Minuten funktioniert analog (siehe Listing 7).

Intervalle hinzufügen oder abziehen

Geht es bei der Datumsarithmetik um das Addieren oder Subtrahieren eines Intervalls von einem vorhandenen Zeitstempel, so bieten sich wiederum sowohl numerische als auch Intervall-Datentypen an. Listing 8 zeigt, wie man zwölf Stunden zu einem Datum addiert.

Beide Varianten funktionieren sowohl mit DATE als auch mit TIMESTAMP. Bei Verwendung der numerischen Variante mit TIMESTAMP sollte man aber aufpassen, denn die Datenbank konvertiert dann implizit in ein DATE. Listing 9 zeigt das anhand der Ausgabe in SQL*Plus: Der SYSTIMESTAMP wird mitsamt Millisekunden und Zeitzone ausgegeben; wird die Zahl „1“ (für einen Tag) addiert, fehlen diese – es ist nun ein DATE.

Wie eingangs gezeigt, können sich durch das Hin- und Herkonvertieren auch Rundungsdifferenzen ergeben. Prinzipiell sind die Intervall-Datentypen also vorzuziehen – mit nachfolgender Besonderheit.

Monate abziehen oder hinzufügen

Geht es ans Addieren oder Subtrahieren von Monaten, so können sich in der Praxis Probleme ergeben, wie Lis-

```
select
  numtodsinterval(
    to_date('2009-09-21 14:45', 'YYYY-MM-DD HH24:MI') -
    to_date('2009-07-21 07:30', 'YYYY-MM-DD HH24:MI')
   ,'DAY'
  ) DATETIME_INTERVAL
from dual;

DATETIME_INTERVAL
---------------------------------------------
+000000062 07:14:59.999999999
```

Listing 4

```
select
 (
   to_date('2009-09-21 14:45', 'YYYY-MM-DD HH24:MI') -
   to_date('2009-07-21 07:30', 'YYYY-MM-DD HH24:MI')
 ) day(9) to second DATETIME_INTERVAL
from dual;

DATETIME_INTERVAL
-----------------------------------------
+000000062 07:15:00.000000000
```

Listing 5

```
select
  extract(
    HOUR from
    (
      to_date('2009-09-21 14:45', 'YYYY-MM-DD HH24:MI') -
      to_date('2009-07-21 07:30', 'YYYY-MM-DD HH24:MI')
    ) day(9) to second
  ) INTERVAL_HOURS
from dual;

INTERVAL_HOURS
--------------
             7
```

Listing 6

```
select
  extract(
    MINUTE from
    (
      to_date('2009-09-21 14:45', 'YYYY-MM-DD HH24:MI') -
      to_date('2009-07-21 07:30', 'YYYY-MM-DD HH24:MI')
    ) day(9) to second
  ) INTERVAL_MINUTES
from dual;

INTERVAL_MINUTES
----------------
              15
```

Listing 7

```
select
  to_date('2009-09-21 14:45', 'YYYY-MM-DD HH24:MI') + (12/24) NUME-
RIC
  to_date('2009-09-21 14:45', 'YYYY-MM-DD HH24:MI') + interval '12'
    hour INTERVAL
from dual;

NUMERIC              INTERVAL
-------------------  -------------------
22.09.2009 02:45:00  22.09.2009 02:45:00
```

Listing 8

```
SQL> select systimestamp ts_val, systimestamp + 1 date_val from dual;

TS_VAL                                DATE_VAL
------------------------------------ -------------------
05.10.2009 09:59:28,213998 +02:00    06.10.2009 09:59:28

1 Zeile wurde ausgewählt.
```

Listing 9

```
select
  to_date('2012-01-31', 'YYYY-MM-DD') + interval '1'
month
from dual;

FEHLER in Zeile 2:
ORA-01839: Datum für angegebenen Monat nicht gültig
```

Listing 10

```
select
  add_months(to_date('2012-01-31', 'YYYY-MM-DD'), 1)  plusmonat,
  add_months(to_date('2012-03-31', 'YYYY-MM-DD'), -1) minusmonat
from dual;

PLUSMONAT           MINUSMONAT
------------------- -------------------
29.02.2012 00:00:00 29.02.2012 00:00:00
```

Listing 11

```
declare
  datum  date;
  "TAGE" number := 1;
begin
  datum := sysdate + interval "TAGE" day;
end;

FEHLER in Zeile 5:
ORA-06550: Zeile 5, Spalte 31:
PLS-00103: Fand das Symbol "TAGE" als eines der ...
```

Listing 12

ting 10 zeigt. Addiert man einen Monat zum 31. Januar, kommt der 31. Februar heraus. Dieses Datum gibt es nicht, also wird eine Fehlermeldung ausgelöst. Dieses Problem tritt nur bei der Addition oder Subtraktion von Monaten zu einem Datum auf und auch nur dann, wenn es den Tag des alten Monats im neuen Monat nicht gibt. In der Praxis sind diese Fälle aber gar nicht so selten. Speziell, wenn es ans Addieren oder Subtrahieren von Monaten geht, sollte man also eine der folgenden Möglichkeiten verwenden:

- Wer mit DATE arbeitet, kann die Funktion ADD_MONTHS verwenden (siehe Listing 11).
- Nutzt man dagegen TIMESTAMP, so funktioniert ADD_MONTHS zwar auch, die Datenbank wandelt implizit aber wieder in ein DATE um. Informationen wie Millisekunden oder die Zeitzone gehen also verloren. Eine Alternative ist es, mit dem ersten Tag des nächsten Monats zu arbeiten, anschließend einen Monat hinzuzufügen und dann wieder einen Tag abzuziehen.

Datums-Intervalle

Auch in PL/SQL kann man mit Intervall-Datentypen arbeiten – allerdings sind ein paar Besonderheiten zu beachten. Die erste ist, dass die „INTERVAL ‚XX'"-Syntax nicht mit Variablen, also dynamisch verwendet werden kann. Listing 12 löst einen Fehler aus, denn in PL/SQL braucht es eine Variable, die als Intervall-Datentyp definiert ist (siehe Listing 13).

Intervalle können auch aus Zahlenwerten heraus berechnet werden. Möchte man wissen, wie viel 2,65 Tage genau sind, dann lässt sich das einfach ermitteln (siehe Listing 14). Für größere Zeit-Intervalle gibt es neben dem Intervall „Day-To-Second" auch „Year-To-Month" (siehe Listing 15).

In der Praxis besteht oft der Bedarf nach einer Kombination. Werden zwei Datumswerte voneinander abgezogen, so können durchaus sowohl „Year-To-Month" als auch „Day-To-Second" erforderlich sein. Zunächst ermittelt man dann das Intervall „Year-To-Month",

zieht dieses vom späteren End-Datum ab und ermittelt dann vom Rest das Intervall „Day-To-Second“. Listing 16 zeigt die Vorgehensweise.

Fazit

Die Oracle-Datenbank erlaubt beim Umgang mit Datumswerten wesentlich mehr, als man im ersten Moment vermuten würde. Spezielle Kenntnis über die Intervall-Datentypen ist bei der Umsetzung von Datums-Arithmetik sehr wichtig, denn die richtige Anwendung der vorhandenen Funktionen kann in einigen Fällen sehr viel Aufwand für umständliche Programmierung sparen.

Eine weitere, sehr wichtige Besonderheit des TIMESTAMP-Datentyps ist die Unterstützung von Zeitzonen. Das Umrechnen eines Datums von einer Zeitzone in eine andere ist mit TIMESTAMP völlig unproblematisch. Eine ausführliche Betrachtung würde jedoch den Rahmen dieses Artikels sprengen, deshalb sei auf das entsprechende Blog-Posting des Autors verwiesen.

Weitere Informationen

1. Blog-Posting des Autors: http://sql-plsql-de.blogspot.co.uk/2009/11/date-time-stamp-und-zeitzonen-in-sql-und.html
2. Oracle Database SQL Language Reference: http://docs.oracle.com/cd/E11882_01/server.112/e26088/toc.htm

Carsten Czarski
carsten.czarski@oracle.com
http://twitter.com/cczarski
http://sql-plsql-de.blogspot.com

```
declare
  v_zeit timestamp;
  v_hrs  interval day(9) to second;
begin
  -- 0 days, 12 hours, 0 minutes, 0 seconds
  v_hrs := '0 12:00:00';
  v_zeit := to_timestamp(
   '2009-09-21 14:45', 'YYYY-MM-DD HH24:MI:SS'
  ) + v_hrs;
  dbms_output.put_line(v_zeit);
end;
```

Listing 13

```
Select numtodsinterval(2.65, 'DAY') interval from dual;

INTERVAL
---------------------------------------------------
+000000002 15:36:00.000000000
```

Listing 14

```
select numtoyminterval(4.55, 'YEAR') intervall from dual;

INTERVALL
---------------------------------------------------
+000000004-06
```

Listing 15

```
declare
  v_enddate   date :=
    to_date('2009-09-21 14:45', 'YYYY-MM-DD HH24:MI:SS');
  v_startdate date :=
    to_date('2007-07-15 07:30', 'YYYY-MM-DD HH24:MI:SS');
  v_ym        interval year(9) to month;
  v_ds        interval day(9) to second;
begin
  -- Jahre und Monate ermitteln
  v_ym := (v_enddate - v_startdate) year to month;
  -- Jahr und Monate vom Enddatum abziehen
  v_enddate := v_enddate - v_ym;
  -- Tage, Stunden, Minuten und Sekunden ermitteln
  v_ds := (v_enddate - v_startdate) day(9) to second;

  dbms_output.put_line('Years:   '||extract(YEAR from v_ym));
  dbms_output.put_line('Months:  '||extract(MONTH from v_ym));
  dbms_output.put_line('Days:    '||extract(DAY from v_ds));
  dbms_output.put_line('Hours:   '||extract(HOUR from v_ds));
  dbms_output.put_line('Minutes: '||extract(MINUTE from v_ds));
  dbms_output.put_line('Seconds: '||extract(SECOND from v_ds));
end;
/

Years:   2
Months:  2
Days:    6
Hours:   7
Minutes: 15
Seconds: 0
```

Listing 16