Eine kurze Geschichte der Oracle Zeit

Martin Hoermann
ORDIX AG
Münster

Schlüsselworte:

Zeit, date, timestamp, timestamp with time zone, timestamp with local time zone, interval, Zeitkonvertierung

Einleitung

Oracle bietet drei Arten von Datentypen an. Der sicherlich bekannteste Datentyp ist `DATE` zum Speichern von sekundengenauen Datumswerten im Bereich vom ersten Januar 4713 vor Christus um 00:00:00 Uhr (4712 B.C. lt. Oracle Dokumentation entspricht korrekterweise dem Jahr 4713 B.C) bis zum 31. Dezember 9.999 nach Christus um 23:59:59 Uhr.
Die zweite Art existiert seit der Version 9. Es ist der Datentyp `TIMESTAMP` zur Speicherung von Datumswerten bis zu milliardsten Sekundenbruchteilen, oder einfacher bis zu neun Nachkommastellen. Als Varianten des Datentyps gibt es die beiden Ausprägungen `TIMESTAMP WITH TIME ZONE` und `TIMESTAMP WITH LOCAL TIME ZONE`, welche neben der Zeitinformation auch die Zeitzone speichern und darstellen. Als dritte Art bietet der Datentyp `INTERVAL` die Möglichkeit Zeitspannen zu speichern. Hierbei stehen die beiden Varianten `YEAR TO MONTH` und `DAY TO SECOND` zur Verfügung.

Eine Referenz für Datentypen und Zeitfunktionen findet sich in der Oracle SQL-Referenz. Einen guten Einblick in die Besonderheiten beim Umgang mit Zeiten erhält der interessierte Leser im Database Globalization Support Guide.

Funktionen und Zeitarithmetik

Die Funktion `sysdate` liefert das aktuelle Datum im Datenformat `DATE` zurück. Die Funktion `systimestamp` liefert dagegen das aktuelle Datum im Datenformat `TIMESTAMP`. Die Uhrzeit liefert das Betriebssystem, welches auch die Einstellungen bezüglich der Zeitzone und der optionalen Verwendung der Sommerzeit berücksichtigt.

Änderungen der Uhrzeit oder der Zeitzone auf Betriebssystem-Ebene haben unmittelbare Auswirkung auf die Datenbank, d. h. nach der Verwendung des Befehls `date` unter Windows ist die neue Zeit ohne Neustart der Datenbank in der aktuellen Session sichtbar.

Die Genauigkeit der Zeit ist abhängig vom Zeitgeber des Betriebssystems und der verwendeten Hardware. So liefert beispielsweise Sun Solaris beim Aufruf von `systimestamp` die aktuelle Uhrzeit bis auf sechs Nachkommastellen hinter der Sekunde. Auf einem Windows Betriebssystem werden nur drei Nachkommastellen mit Werten ungleich null geliefert. Dieser Test ist abhängig von der Version des Betriebssystems und der eingesetzten Hardware.

Die Funktionen `current_date`, `current_timestamp` und `localtimestamp` liefern das Datum unter Berücksichtigung der Zeitzone der Session bzw. der Datenbank. Die Rückgabe erfolgt als

Datentyp DATE, TIMESTAMP WITH TIME ZONE respektive als TIMESTAMP. Den Unterschied zwischen den beiden Funktionen systimestamp und current_timestamp illustriert folgendes Beispiel:

```
SELECT systimestamp, current_timestamp FROM dual;

SYSTIMESTAMP                       CURRENT_TIMESTAMP
---------------------------------- ----------------------------------
27.05.04 13:32:10,621088 +00:00  27.05.04 13:32:10,621134 +00:00

ALTER SESSION SET TIME_ZONE = '+2:0';

SELECT systimestamp, current_timestamp FROM dual;

SYSTIMESTAMP                       CURRENT_TIMESTAMP
---------------------------------- ----------------------------------
27.05.04 13:32:10,692982 +00:00 27.05.04 15:32:10,693020 +02:00
```

Wie dem Beispiel zu entnehmen ist, hat die Zeitzone einen doppelten Einfluss: Erstens wird die Zeit selbst verändert und zweitens wird der Offset mit ausgegeben. Dies ist sinnvoll, da nur mit diesen beiden Änderungen die Uhrzeit bezüglich der Zeitzone der Datenbank ermittelt werden kann. Die Funktionen dbtimezone und sessiontimezone liefern die Zeitzone der Datenbank respektive die Zeitzone der Session zurück. Mit Hilfe der Funktion tz_offset lässt sich die Differenz zwischen einer beliebigen Zeitzone und UTC (Universal Time Coordinated) ermitteln.

Ein Datum vom Datentyp DATE lässt sich über Subtraktion oder Addition mit einer Zahl in ein anderes Datum verwandeln. Dabei stellt eine ganze Zahl eine ganze Anzahl von Tagen dar und ein Bruchteil von eins entsprechend einen Bruchteil eines Tages. Diese Arithmetik führt zu einem sehr einfachen und intuitiven Gebrauch von Daten. So ist sysdate + 1 beispielsweise morgen, sysdate – 1 gestern und sysdate + 1/24 das aktuelle Datum plus eine Stunde.

Sollen ein oder mehrere Monate zu einem Datum addiert oder von einem Datum subtrahiert werden, so lässt sich dies über die Funktion ADD_MONTHS erreichen. Die Anzahl der Monate zwischen zwei Daten liefert die Funktion MONTHS_BETWEEN. Daten vom Typ TIMESTAMP können ausschließlich mit Intervall Typen manipuliert werden. So lässt sich übermorgen, bezogen auf die aktuelle Uhrzeit, beispielsweise mit dem Ausdruck systimestamp + TO_DSINTERVAL('2 00:00:00') ermitteln oder heute in einem Jahr und 3 Monaten mit systimestamp + to_yminterval('01-03'). Eine Subtraktion oder Addition mit einem numerischen Wert von einem Datum im Format TIMESTAMP führt zu einer impliziten Konvertierung in den Datentyp DATE und sollte daher nur sehr bewusst verwendet werden.

Die Funktion LAST_DAY liefert den letzten Tag des Monats, der als Argument mitgeliefert wird. Ist dagegen das Datum des nächsten Sonntags – oder eines anderen Wochentags – gefordert, so liefert die Funktion NEXT_DAY das gewünschte Ergebnis. Mit der Funktion EXTRACT lässt sich aus einem Datum eine beliebige Komponente extrahieren. So liefert der Ausdruck EXTRACT(YEAR FROM DATE '1998-03-07') das Jahr 1998.

Darstellung von Zeiten

Die sicherlich wichtigste Unterscheidung bei der Verwendung von Datumswerten ist die zwischen dem Wert selbst und dessen Darstellung. Ein Datum wird in der Regel in der interpretierten Version

seines Wertes am Bildschirm als Zeichenkette ausgegeben. Wie diese Darstellung erfolgt, ist abhängig von den National Language Support (NLS) Einstellungen. Dies veranschaulicht folgendes Skript:

```
ALTER SESSION SET NLS_DATE_FORMAT = 'YYYY-MM-DD HH24:MI:SS';
SELECT sysdate FROM dual;
```

welches am 12. Mai 2004 um 22:37:21 Uhr zu folgender Ausgabe führte: 2004-05-12 22:37:21.

Neben der Möglichkeit, die Darstellung über die NLS Einstellungen zu manipulieren, lässt sich ein Datumswert auch in ein Character-Format umwandeln, wobei das erwünschte Format anzugeben ist. Hierzu ist die `to_char` Funktion mit einem Format-String zu verwenden. So lässt sich beispielsweise die in Deutschland übliche Darstellung mit dem Befehl `to_char(sysdate, 'DD.MM.YYYY HH24:MI:SS')` erzielen. Beeindruckend ist die Vielzahl der Formatierungsmöglichkeiten, die Oracle anbietet.

Interne Darstellung von Datentypen

Die interne Darstellung des Datentyps `DATE` ist interessanterweise davon abhängig, ob sich die Variable im Speicher oder auf der Platte befindet. Exemplarisch stellt diese Ausarbeitung die Struktur für den im Speicher befindlichen Datentyp vor. Hilfreich für die folgende Untersuchung ist die Funktion `DUMP`, die zu einer Variablen den Typ, die Länge in Bytes sowie den Inhalt der einzelnen Bytes ausgibt.

```
DUMP( TO_DATE('02.04.1804 15:30:10', 'DD.MM.YYYY HH24:MI:SS' ))
-------------------------------------------------------------
Typ=13 Len=8: 7,12, 4, 2,15,30,10, 0
     |      |   | |  |  |  |  |  |  |
     |      |   | |  |  |  |  |  |  |-- immer 0
     |      |   | |  |  |  |  |  |-----  Sekunden
     |      |   | |  |  |  |  |--------  Minuten
     |      |   | |  |  |  |-----------  Stunden
     |      |   | |  |  |--------------  Tag
     |      |   | |  |-----------------  Monat
     |      |   |-|--------------------  Jahr als Tupel
     |      |--------------------------  Länge in Byte
     |-----------------------------  Datentyp
```

Der Typ `Typ=13` verweist auf den Datentyp `DATE` im Hauptspeicher. Die Länge `Len=8` zeigt an, wie viele Bytes der Datentyp im Hauptspeicher belegt. Die folgenden Abschnitte führen die Belegung der einzelnen Bytes noch einmal etwas detaillierter auf.

Die ersten beiden Byte (x,y) geben das Jahr an. Dabei werden die Jahre von 0001 bis 9999 nach Christus wie folgt berechnet: `x*256+y`, so entspricht das Krönungsjahr Napoleons 1804 der Bytekombination (7,12). Die Jahre vor Christi Geburt sind etwas umständlicher zu berechnen. Sie beginnen mit dem Tuple (237,152) für das älteste, darstellbare Jahr 4712 und laufen bis (255,255) für das Jahr 1 vor Christi Geburt. Oracle lässt eine Bandbreite von über 50.800 weiteren Jahren ungenutzt. Das entspricht den Werten zwischen (39,15) und (237,152). Ein Grund hierfür ist sicherlich, dass die Funktion „Julian Day“ ab dem ersten Januar 4712 vor Christus definiert ist, was folgende Abfrage verdeutlicht.

```
SELECT TO_CHAR(TO_DATE('01.01.-4712', 'DD.MM.SYYYY'),'J') JD
FROM DUAL;
```

```
JD
-------
0000001
```

Der Zeithorizont bis zum Jahre 9999 dürfte hingegen auch für die Besatzung des Raumschiffs Enterprise vollkommen ausreichen und birgt weiterhin den Vorteil, ebenfalls vierstellig zu sein. Die interne Darstellung des Jahres ist plattformabhängig. Nähere Erläuterungen hierzu finden sich bei Steve Adams (www.ixora.com.au). Das dritte Byte gibt den Monat an, das vierte Byte den Tag und die folgenden Bytes die Stunde, Minute und Sekunde. Auch hier *verschenkt* Oracle eine Menge an Bits auf Kosten der Strukturen. Vielleicht ist das der Grund dafür, dass sich durch diese Speicherform einige Funktionen, wie z. B. `TO_CHAR` und `ADD_MONTHS`, sehr effizient implementieren lassen. Das achte und letzte Byte ist interessanterweise immer 0x00. Ein Datum aus einer Tabelle gelesen liefert den Typ 12 zurück, der im Gegensatz zum Typ im Hauptspeicher nur 7 Byte lang ist und für die Stunde, Minute und Sekunde einen Offset von +1 hat, um 0x00 Bytes zu vermeiden. Ein solches Offset gibt es auch bei anderen Datentypen. Das achte 0 Byte entfällt.

Kalender

Die Zuordnung eines Tages zu einem Datum ist abhängig vom verwendeten Kalender. So bestimmt der Kalender beispielsweise, ob Sonntag oder Montag der erste Tag der Woche ist. Dies hat auch Einfluss darauf, ob die erste Woche im neuen Jahr die Kalenderwoche (KW) 53 oder die KW 1 ist. Liegen vier oder mehr Tage der Woche im alten Jahr, so ist dies die KW 53. Liegen aber vier oder mehr Tage der Woche im neuen Jahr, so ist dies die KW 1.

Weiterhin bestimmt der Kalender, welcher Monat wie viele Tage hat. So haben im Persischen Kalender beispielsweise die ersten sechs Monate jeweils 31 Tage, die nächsten fünf Monate 30 Tage und der Dezember hat entweder 29 oder 30 Tage. Auch die Anzahl der Tage pro Woche ist nicht zwingend sieben, wie wir es gewohnt sind. Während der Säkularisierung in Frankreich führte die Abschaffung des Sonntags beispielsweise zu einer Woche mit 10 Tagen.

Der in Deutschland übliche Kalender ist der Gregorianische, der 1582 den Julianischen Kalender ablöste. Die in Oracle zur Verfügung stehenden Kalender sind:

- Arabic Hijrah
- English Hijrah
- Gregorian
- Japanese Imperial
- Persian
- ROC Official (Republic of China)
- Thai Buddha

Der Kalender wird über die Variable `NLS_CALENDAR` eingestellt. Ist die Variable nicht explizit gesetzt, so leitet sich der Wert von der Variablen `NLS_TERRITORY` ab.

Über die Zeitrechnung lassen sich viele, interessante Dinge berichten. Insbesondere die Berechnung der Schaltjahre führte im Julianischen Kalender zu fehlerhaften Berechnungen, die mit der Einführung des Gregorianischen Kalenders behoben wurden. Diese Bereinigung lässt sich recht anschaulich mit folgender Abfrage verdeutlichen. Sie listet die ersten acht Tage des Oktobers 1582 auf.

```
SELECT to_char( to_date( '01.10.1582', 'DD.MM.YYYY' )
       + rownum - 1, 'DD.MM.YYYY') Datum
FROM   user_objects
WHERE  rownum < 7;

DATUM
----------
01.10.1582
02.10.1582
03.10.1582
04.10.1582 <<<<
15.10.1582 <<<<
16.10.1582
```

Zeitzone

Die Zeitzone der Datenbank wird beim Anlegen der Datenbank über das Kommando `CREATE DATABASE` festgelegt.

```
CREATE DATABASE
...
SET TIME_ZONE='GMT';
```

Wird keine Zeitzone angegeben, so wird die Zeitzone des Betriebssystems gewählt. Die Zeitzone lässt sich nachträglich durch das folgende Kommando ändern. Die Änderung ist allerdings erst nach einem Neustart der Datenbank aktiv. Siehe hierzu auch Metalink 227334.1.

```
ALTER DATABASE SET TIME_ZONE = '-05:00';
```

Die Zeitzone der Datenbank hat folgende Auswirkungen auf die Funktionen und Datenbankfelder:

- `sysdate/systimestamp`: Keine. Die Funktionen liefern immer die Uhrzeit und die Zeitzone des entsprechenden Betriebssystems.
- `current_date`: Die Funktion addiert die Zeitzone der Session zu der aktuellen Uhrzeit der Datenbank. Ist die Zeitzone der Session nicht gesetzt, so leitet sich diese aus der Zeitzone der Datenbank ab. Somit hat eine Veränderung der Zeitzone der Datenbank einen entsprechenden Einfluss.
- Spalten vom Datentyp `timestamp with time zone` sind unabhängig von der Datenbank-Zeitzone. Lediglich bei der Befüllung über die Funktion `current_timestamp` wird die Zeitzone entsprechend berücksichtigt. Dies liegt aber am Einfluss der Zeitzone auf die Funktion.
- Eine Spalte vom Datentyp `timestamp with local time zone` erhält immer einen entsprechend der Zeitzone angepassten Wert. Dies ist sogar unabhängig davon, ob die Spalte mit der Funktion `sysdate`, `systimestamp`, `current_date` oder `current_timestamp` befüllt wird.

Die Koordinierte Weltzeit UTC ist die Referenzzeit, von der die Zeiten in den verschiedenen Zeitzonen der Erde abgeleitet werden. Die UTC ist die Nachfolgerin der mittleren Greenwichzeit (GMT). Die deutsche Standardzeit ist die Mitteleuropäische Zeit (MEZ), die gleich der UTC plus einer Stunde ist. Während der Sommerzeit entspricht die Mitteleuropäische Sommerzeit (MESZ) der UTC plus zwei Stunden. Die mehreren hundert zur Verfügung stehenden Zeitzonen lassen sich aus der

Datenbankview `v$timezone_names` ermitteln. Folgende Abfrage zeigt z. B. alle Zeitzonen mit Europa im Namensanfang.

```
SELECT * FROM v$timezone_names WHERE tzname LIKE 'Eur%'

TZNAME               TZABBREV
-------------------- --------------------
Europe/Dublin        IST
Europe/Dublin        GMT
Europe/Dublin        BST
Europe/Istanbul      LMT
...
```

Soll eine Zeit von einer beliebigen Zeitzone in UTC (a) oder von UTC in eine beliebige Zeitzone (b) umgewandelt werden, so lässt sich dies mit folgenden Kommandos bewerkstelligen.

```
a) FROM_TZ(<timestamp> ,<tzname>) AT TIME ZONE <offset>
b) FROM_TZ(<timestamp> ,<offset>) AT TIME ZONE <tzname>
```

Die in diesem Artikel aufgeführten Sachverhalte umreißen das komplexe Thema Zeit nur rudimentär. Eine komplette Abhandlung der Oracle-Zeiten dürfte annähernd ein ganzes Buch füllen. Der Vortrag referenziert zahlreiche Quellen die zur Vertiefung des Themas geeignet sind.

Kontaktadresse:

Martin Hoermann
ORDIX AG
Westernmauer 12-16
D-33098 Paderborn

Telefon:	+49 (0) 5251-10630
Fax:	+49 (0) 180-1673490
E-Mail	info@ordix.de
Internet:	www.ordix.de