Tipps & Tricks: Neue Datentypen

Bereich:	SQL	Erstellung:	03/2005 MP
Versionsinfo:	10.1, 10.2, 11.1	Letzte Überarbeitung:	07/2009 MA

Neue Datentypen

Die Version 9i bietet drei neue Datentypen als Erweiterung des Datentyps DATE, zwei neue Datentypen um Zeitintervalle zu speichern, drei neue vordefinierte Objekttypen, einige Erweiterungen zu LOBs und eine UNICODE-Unterstützung. Im Zusammenhang mit den neuen Datentypen wurden auch einige neue Datums- und Konvertierungsfunktionen eingeführt, die ebenfalls an dieser Stelle erwähnt werden sollen.

Neue DateTime-Datentypen

Zu den Erweiterungen von DATE gehören die Datentypen:

- TIMESTAMP[(n)]: er ermöglicht die Angabe von Sekundenbruchteilen; n liegt dabei zwischen 0...9, Default-Wert ist 6; der entsprechende Initialisierungs-Parameter lautet NLS_TIMESTAMP_FORMAT.
- TIMESTAMP[(n)] WITH TIME ZONE: er beinhaltet zusätzlich die Abweichung von UTC (Coordinated Universal Time); zwei Werte werden dabei als gleich erachtet, wenn sie der gleichen Zeitzone angehören; der entsprechende Initialisierungs-Parameter lautet NLS_TIMESTAMP_TZ_FORMAT.
- TIMESTAMP[(n)] WITH LOCAL TIME ZONE: er beinhaltet ebenfalls die Abweichung von UTC, jedoch wird diese nicht explizit in der Datenbank gespeichert; stattdessen wird der Wert auf die Zeitzone der Datenbank umgerechnet und als solche eingetragen; ein SELECT liefert die lokale Session-Zeit.

Beispiel:

```
SQL> ALTER DATABASE SET TIME_ZONE = '-5:00';

SQL> CREATE TABLE new_times (
  2         zeit1 TIMESTAMP(3),
  3         zeit2 TIMESTAMP(3) WITH TIME ZONE,
  4         zeit3 TIMESTAMP(3) WITH LOCAL TIME ZONE);

SQL> INSERT INTO new_times VALUES (
            '25.02.2002 10:34:58',
            TIMESTAMP'2002-02-25 10:34:58.456 +2:00',
            TIMESTAMP'2002-02-25 10:34:58.456 -3:00');

SQL> SELECT * FROM new_times;

ZEIT1                  ZEIT2                          ZEIT3
---------------------- ------------------------------ ----------------------
25.02.02 10:34:58,000 25.02.02 10:34:58,456 +02:00 25.02.02 14:34:58,456
```

Neue Interval-Datentypen

Zur Speicherung von Zeitintervallen gibt es nun die Datentypen:

- INTERVAL YEAR[(n)] TO MONTH: damit lassen sich Zeitintervalle in Jahren und Monaten angeben; die Anzahl an Stellen der Jahreszahl kann optional mit angegeben werden; Default-Wert ist 2;
- INTERVAL DAY[(n)] TO SECOND: damit lassen sich Zeitintervalle in Tagen, Stunden, Minuten, Sekunden und Sekundenbruchteilen angeben; die Anzahl an Stellen für den Tag kann optional mit angegeben werden; n liegt dabei zwischen 0..9, der Default-Wert ist 2.

Intervalle können zu DateTime-Werten addiert oder von ihnen abgezogen werden und es ergibt sich wieder ein DateTime-Wert. Sie können aber auch miteinander addiert, voneinander subtrahiert, mit einer Zahl multipliziert oder durch eine dividiert werden, es ergibt sich immer wieder ein Intervall.

Beispiel:

```
SQL> CREATE TABLE diff (
  2         col1 INTERVAL YEAR TO MONTH,
  3         col2 INTERVAL DAY TO SECOND);

SQL> INSERT INTO diff VALUES (
            '7-9', '18 14:29:39.98765');

SQL> SELECT * FROM diff;

COL1            COL2
--------------- --------------------
+07-09          +18 14:29:39.987650
```

Neue Datums- und Konvertierungsfunktionen

Die nachfolgende Tabelle listet die neuen Funktionen auf, die gemeinsam mit den neuen Datentypen eingeführt wurden:

CURRENT_DATE	gibt aktuelles Datum der Session als Datentyp DATE zurück (entspricht SYSDATE)
CURRENT_TIMESTAMP	gibt aktuelles Datum und Uhrzeit der Session als Datentyp TIMESTAMP WITH TIME ZONE zurück
LOCALTIMESTAMP	gibt aktuelles Datum und Uhrzeit der Session als Datentyp TIMESTAMP zurück
DBTIMEZONE	gibt die Zeitzone der Datenbank zurück
SESSIONTIMEZONE	gibt die Zeitzone der Session zurück
TZ_OFFSET (SESSIONTIMEZONE | DBTIMEZONE |Zonenname | <offset>)	gibt die Differenz der Zeitzone im Vergleich zur aktuellen Zeit zurück
	wandelt einen TIMESTAMP-Wert in einen TIMESTAMP WITH TIME

Neue vordefinierte Objekttypen

Die neuen Oracle-definierten Objekttypen lauten wie folgt:

- [SYS.]AnyType: er beinhaltet die Beschreibung eines vorhandenen Datentyps und kann vordefiniert oder vom Benutzer definiert sein.
- [SYS.]AnyData: er beinhaltet die Beschreibung plus die Instanz eines vorhandenen Datentyps und kann in Tabellenspalten oder Prozeduren eingesetzt werden.
- [SYS.]AnyDataSet: er beinhaltet die Beschreibung plus mehrere Instanzen eines vorhandenen Datentyps und kann für Parameter bei Prozeduren eingesetzt werden.

Hinweis: Für Any-Typen gibt es vordefinierte Schnittstellen für PL/SQL und OCI.

Neue LOB-Features

Ab Version 9i stehen drei neue LOB-Features zur Verfügung:

- Mit dem ALTER TABLE-Befehl können LONG-Spalten ganz einfach zu CLOB-Spalten und LONG RAW-Spalten zu BLOB-Spalten migriert werden.

Beispiel:

```
SQL> CREATE TABLE articles (id NUMBER NOT NULL,
                            description LONG NOT NULL);

SQL> ALTER TABLE articles MODIFY (description CLOB);
```

Hinweis: Da zwei LONG-Spalten innerhalb einer Tabelle nicht zulässig sind, kann eine neue LONG- bzw. LONG RAW-Spalte erst nachträglich eingefügt werden.

```
SQL> ALTER TABLE articles ADD (picture LONG RAW);
```

- CLOB-Felder können prinzipiell wie VARCHAR2-Felder behandelt werden, also mit String-Funktionen und -Operatoren.

Beispiele:

```
SQL> SELECT id, description
  2  FROM articles
  3  WHERE description LIKE '%fga%';

SQL> SELECT substr(description,3,2)
  2  FROM articles;
```

- LOB`s sind nun auch in partitionierten Index-Organized Tables zulässig.

Unterstützung von UNICODE

NCHAR, NVARCHAR und NCLOB haben nun grundsätzlich das UNICODE-Format, da für den NATIONAL CHARACTER SET nur noch UNICODE-Zeichensätze zulässig sind.

String-Funktionen können mit jeder beliebigen Kombination aus CHAR/ VARCHAR2/ NCHAR/ NVARCHAR2 umgehen, auch innerhalb ein und derselben Funktion.