Bazy danych, ORACLE
Przydatne funkcje formatujące

wersja 1.1

21 marca 2006

Uwaga: Wszystkie przykłady operują na schemacie SUMMIT2. Dlatego też należy upewnić się, że skrypt *summit2.sql* wykonał się bezbłędnie.

Niniejsze opracowanie powstało w głównej mierze w oparciu o oryginalną dokumentację firmy ORACLE:

Oracle8*i* SQL Reference Release 3 (8.1.7) Part Number A85397-01

1. Przydatne funkcje formatujące

1.1. Funkcje tekstowe: konkatenacja, RPAD, LPAD, SUBSTR, LTRIM, RTRIM, LENGTH, REPLACE, UPPER, LOWER, TRIM, CHR, INSTR, ASCII

```
SELECT '-->'||first_name||'-'||last_name||'<--' „Pracownik Ngao”
FROM emp
WHERE UPPER(last_name) = UPPER('NgAo');

-- Długość etykiety to: 25 + 25 + 3 + 3 + 1 = 57
-- first_name = VARCHAR2(25), last_name = VARCHAR2(25),
-- '-->' = 3, '<--' = 3 '-' = 1

Pracownik Ngao
------------------------------------------------------------
-->LaDoris-Ngao<--
```

```
SELECT RPAD('-->'||first_name||'-'||last_name||'<--',25) "Pracownik Ngao"
FROM emp
WHERE UPPER(last_name) = UPPER('NgAo') ;

-- Długość etykiety obcięta do 25 znaków.

Pracownik Ngao
-------------------------
-->LaDoris-Ngao<--
```

```
SELECT RPAD(last_name,20,'#%') "Pracownicy"
FROM emp
WHERE UPPER(last_name) LIKE 'D%';

Pracownicy
--------------------
Dumas#%#%#%#%#%#%#%#
Dancs#%#%#%#%#%#%#%#
```

```
SELECT SUBSTR('To jest tekst',8,6) "Tekst" FROM DUAL;

Tekst
------
 tekst


SELECT SUBSTR('To jest tekst',-8,6) "Tekst" FROM DUAL; -- liczba ujemna

Tekst
```

```
------
st tek


SELECT SUBSTR('To jest tekst',3) "Tekst" -- pomijamy trzeci argument
FROM DUAL;

Tekst
-----------
 jest tekst
```

```
SELECT LTRIM('...aaa...bbbb...','.') "LTRIM example" FROM DUAL;

LTRIM example
-------------
aaa...bbbb...


SELECT RTRIM('...aaa...bbbb...','.') "RTRIM example" FROM DUAL;

RTRIM example
-------------
...aaa...bbbb


SELECT RTRIM(LTRIM('...aaa...bbbb...','.'),'.') "R(L)TRIM" FROM DUAL;

R(L)TRIM
----------
aaa...bbbb
```

```
SELECT
  first_name, LENGTH(first_name) "Długość imienia",
  last_name, LENGTH(last_name) "Długość nazwiska"
FROM emp
WHERE LENGTH(last_name) > 6
ORDER BY LENGTH(last_name) DESC;
```

FIRST_NAME	Długość imienia	LAST_NAME	Długość nazwiska
Mark	4	Quick-To-See	12
Carmen	6	Velasquez	9
Antoinette	10	Catchpole	9
Alexander	9	Markarian	9
Midori	6	Nagayama	8
Sylvie	6	Schwartz	8
Molly	5	Urguhart	8
Audry	5	Ropeburn	8
Yasmin	6	Sedeghi	7

```
SELECT REPLACE('Wygrałem !!!','!','?') "REPLACE" FROM DUAL;

REPLACE
------------
Wygrałem ???
```

```
SQL> SELECT UPPER('duży') "D", LOWER('MAŁY') "m" FROM DUAL;

D    m
```

```
---- ----
DUŻY mały
```

```
SELECT TRIM (0 FROM 000123456789000) "TRIM Example" FROM DUAL;

TRIM Exam
---------
123456789

-- Pozostałe kombinacje. Sprawdź sam!
SELECT TRIM (LEADING 0 FROM 000123456789000) "TRIM Example" FROM DUAL;

SELECT TRIM (TRAILING 0 FROM 000123456789000) "TRIM Example" FROM DUAL;

SELECT TRIM (BOTH 0 FROM 000123456789000) "TRIM Example" FROM DUAL;

SELECT TRIM (' ' FROM '  00012345678900  ') "TRIM Example" FROM DUAL;

SELECT TRIM (BOTH '*' FROM '***12345678900***') "TRIM Example" FROM DUAL;
```

```
SELECT
  TRANSLATE('123ABC456',
  '0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ',
  '9999999999XXXXXXXXXXXXXXXXXXXXXXXXXX.') "Tajne"
FROM DUAL;

Tajne
---------
999XXX999
```

```
SELECT
  TRANSLATE('Którą łapkę zwichał piesek? ',
  'ąęśćżźńółĄĘŚĆŻŹŃÓŁ',
  'aesczznolAESCZZNOL') " Tekst niby po polsku"
FROM DUAL;

Tekst niby po polsku
----------------------------
Ktora lapke zwichal piesek?
```

```
SELECT CHR(67)||CHR(65)||CHR(84) "Dog" FROM DUAL;

Dog
---
CAT


SELECT
  LOWER(CHR(67))||
  LOWER(CHR(65))||
  LOWER(CHR(84)) "Dog" FROM DUAL;

Dog
---
cat
```

```
-- Szukamy 2 wystąpienia znaku 'x' poczynając od 5 znaku.
```

```
SELECT INSTR('abcxdefxghix','x', 5, 2) "INSTR" FROM DUAL;

INSTR
----------
        12
```

```
SELECT ASCII('Q') FROM DUAL;

ASCII('Q')
----------
        81
```

1.2. Funkcje agregujące: MIN, MAX, SUM, AVG, COUNT

```
SELECT
  COUNT(salary) "COUNT",
  SUM(salary) "SUM",
  AVG(salary) "AVG",
  MIN(salary) "MIN",
  MAX(salary) "MAX"
FROM emp;

     COUNT        SUM        AVG        MIN        MAX
---------- ---------- ---------- ---------- ----------
        25      31377    1255,08        750       2500
```

1.3. Funkcje operujące na polach typu DATE: SYSDATE, ROUND, TRUNC, LAST_DAY

```
SELECT
  TO_CHAR (SYSDATE, 'MM-DD-YYYY HH24:MI:SS')"Teraz"
FROM DUAL;

Teraz
--------------------
02-18-2003 20:08:07
```

```
SELECT
  ROUND(TO_DATE('27-01-2000','DD-MM-YYYY'),'YEAR') "Nowy Rok"
FROM DUAL;

Nowy Rok
--------
00/01/01

SELECT
  TO_CHAR(ROUND(TO_DATE('27-01-2000','DD-MM-YYYY'),'YEAR'),'DD-MM-YYYY')
  "Nowy Rok"
FROM DUAL;

Nowy Rok
----------
01-01-2000

SELECT
  TO_CHAR(ROUND(TO_DATE('27-01-2000','DD-MM-YYYY'),'MONTH'),'DD-MM-YYYY')
  "Nowy Miesiąc"
FROM DUAL;
```

```
Nowy Miesiąc
------------
01-02-2000
```

```
SELECT
  TO_CHAR(TRUNC(TO_DATE('27-07-2000','DD-MM-YYYY'),'YEAR'),'DD-MM-YYYY')
  "Nowy Rok"
FROM DUAL;

Nowy Rok
----------
01-01-2000
```

```
SELECT
  TO_CHAR(
    NEXT_DAY(TO_DATE('18-02-2003','DD-MM-YYYY'),'niedziela'),'Day, DD-MM-YYYY')
    "To jest niedziela"
FROM DUAL;

To jest niedziela
------------------------
Niedziela   , 23-02-2003
```

```
SELECT
  SYSDATE "Dzisiaj",
  LAST_DAY(SYSDATE) "Ostatni dzień",
  LAST_DAY(SYSDATE) - SYSDATE "Jeszcze zostało"
FROM DUAL;

Dzisiaj    Ostatni dzień Jeszcze zostało
---------- ------------- ---------------
2003-02-18 2003-02-28    10
```

1.4. Funkcje numeryczne

```
ABS
ACOS
ADD_MONTHS
ATAN
ATAN2
BITAND
CEIL
COS
COSH
EXP
FLOOR
LN
LOG
MOD
POWER
ROUND (number function)
SIGN
SIN
SINH
SQRT
TAN
TANH
TRUNC (number function)
```

1.5. Funkcje konwertujące: TO_DATE, TO_CHAR

```
SELECT
  TO_DATE(
  '26 styczeń, 1967, 11:24:59 przed poł',
  'dd month, YYYY, HH:MI:SS A.M.',
  'NLS_DATE_LANGUAGE = Polish') "Data"
FROM DUAL;

Data
-------------------
1967-01-26 11:24:59

SELECT
  TO_DATE(
  '26 styczeń, 1967, 11:24:59 po poł',
  'dd month, YYYY, HH:MI:SS P.M.',
  'NLS_DATE_LANGUAGE = Polish') "Data"
FROM DUAL;

Data
-------------------
1967-01-26 23:24:59

SELECT
  TO_DATE(
  '26 January, 1967, 11:24:59 A.M.',
  'dd Month, YYYY, HH:MI:SS A.M.',
  'NLS_DATE_LANGUAGE = American') "Data"
FROM DUAL;

SELECT
  TO_DATE(
  '26 January, 1967, 11:24:59 P.M.',
  'dd Month, YYYY, HH:MI:SS P.M.',
  'NLS_DATE_LANGUAGE = American') "Data"
FROM DUAL;

SELECT TO_CHAR(
   TO_DATE(
  '26 styczeń, 1967, 11:24:59 po poł',
  'dd month, YYYY, HH:MI:SS P.M.',
  'NLS_DATE_LANGUAGE = Polish'),'DD month YYYY, HH24:MI:SS')
  "Data"
FROM DUAL;

Data
-----------------------------
26 styczeń     1967, 23:24:59

SELECT
  TO_CHAR(
    TO_DATE('26-01-2003, 23:45:33','DD-MM-YYYY, HH24:MI:SS'),
      'day # Month # YEAR, DD-MM-YYYY, g:HH24-->MI-->SS')
      "Dziwnie zapisana data"
FROM DUAL;

Dziwnie zapisana data
--------------------------------------------------------------------------------
niedziela    # Styczeń     # TWO THOUSAND THREE, 26-01-2003, 23-->45-->33
```

1.6. Inne przydatne funkcje: NVL

```
SELECT
  address, NVL(city,'-'), NVL(state,'?'), NVL(country,'!') "Kraj"
FROM warehouse;

NVL(CITY,'-') NVL(STATE,'?') Kraj
------------- -------------- ----------------
Seattle       WA             USA
Bratislava    ?              Czechozlovakia
Sao Paolo     ?              Brazil
Lagos         ?              Nigeria
Hong Kong     ?              !
```

Poniższy przykład pokazuje, że nieumiejętne korzystanie z kolumn numerycznych z atrybutem **NULL** może prowadzić do **bardzo trudno wykrywalnych błędów w działaniu programów** (mimo tego, że z punktu widzenia składni SQL i PL/SQL programy są całkowicie poprawne). W wyniku działania poniższego programu otrzymamy napis „Otrzymano NULL!" zamiast poprawnej sumy = 60.

Powodem błędu jest to, że sumowanie wartości kolumn **zawsze** w wyniku da wartość NULL, gdy **chociaż jedna** sumowana komórka ma wartość NULL! Usunąć ten błąd można w prosty sposób – należy zmienić definicję kursora, tak aby wartości NULL były konwertowane do liczby zero za pomocą funkcji NVL.

```
DROP TABLE liczby;
CREATE TABLE liczby (liczba NUMBER);

-- W jednej kolumnie jest wartość NULL!
INSERT INTO liczby VALUES (10);
INSERT INTO liczby VALUES (20);
INSERT INTO liczby VALUES (null);
INSERT INTO liczby VALUES (30);

DECLARE
-- Błędna definicja kursora!
CURSOR c_temp IS SELECT liczba FROM liczby;

-- Poprawna definicja kursora!
-- CURSOR c_temp IS SELECT NVL(liczba,0) FROM liczby;

uv_liczba NUMBER;
uv_sum NUMBER := 0;
BEGIN
  OPEN c_temp;
  LOOP
    FETCH c_temp INTO uv_liczba;
      -- Uwaga. EXIT musi być w tym miejscu, aby poprawnie sumowało.
      EXIT WHEN c_temp%NOTFOUND;
      uv_sum := uv_sum + uv_liczba;
  END LOOP;
  CLOSE c_temp;
  DBMS_OUTPUT.PUT_LINE(NVL(TO_CHAR(uv_sum),'Otrzymano NULL!'));
END;
```

Warto również przeanalizować poniższe zapytania:

```
SELECT 1+3+5 "Wynik dodawania" FROM DUAL;

SELECT NVL(TO_CHAR(1+3+5),'NULL') "Wynik dodawania" FROM DUAL;

SELECT 1+3+5+null "Wynik dodawania" FROM DUAL;
```

```
SELECT NVL(TO_CHAR(1+3+5+null),'NULL') "Wynik dodawania" FROM DUAL;
```