WYBRANE FUNKCJE ORACLE 10*g*

Funkcje numeryczne

Składnia	Opis działania	Przykład

ABS(n)	Zwraca wartość bezwzględną liczby *n*.	`SELECT ABS(-26)`
		`FROM DUAL;`
		
		`ABS(-26)`
		`----------`
		`26`

CEIL(n)	Zwraca najmniejszą liczbę całkowitą większą bądź równą *n*.	`SELECT CEIL(14.3), CEIL(16)`
		`FROM DUAL;`
		
		`CEIL(14.3)  CEIL(16)`
		`---------- ----------`
		`15  16`

COS(n)	Zwraca cosinus kąta *n* podanego w radianach.	`SELECT`
		`COS(180 * 3.14159265359/180)`
		`"Cosinus 180 stopni"`
		`FROM DUAL;`
		
		`Cosinus 180 stopni`
		`------------------`
		`-1`

EXP(n)	Zwraca *n*-tą potęgę liczby e.	`SELECT EXP(2), EXP(0)`
		`FROM DUAL;`
		
		`EXP(2)  EXP(0)`
		`---------- ----------`
		`7,3890561  1`

FLOOR(n)	Zwraca największą liczbę całkowitą mniejszą lub równą *n*.	`SELECT FLOOR(11.7), FLOOR(2)`
		`FROM DUAL;`
		
		`FLOOR(11.7)  FLOOR(2)`
		`---------- ----------`
		`11  2`

LN(n)	Zwraca logarytm naturalny liczby *n* (*n* dodatnie).	`SELECT LN(3), LN(1)`
		`FROM DUAL;`
		
		`LN(3)  LN(1)`
		`---------- ----------`
		`1,09861229  0`

LOG(m, n)	Zwraca logarytm o podstawie *m* z liczby *n* ($m > 0, m \neq 1, n > 0$).	`SELECT LOG(10,100)`
		`FROM DUAL;`
		
		`LOG(10,100)`
		`----------`
		`2`

MOD(m, n)	Zwraca resztę z dzielenia liczby *m* przez liczbę *n*. Zwraca *m*, jeżeli *n* jest równe 0.	`SELECT MOD(15,7)`
		`FROM DUAL;`
		
		`MOD(15,7)`
		`----------`
		`1`

POWER(m, n)	Zwraca liczbę *m* podniesioną do potęgi *n* (*m*, *n* -dowolne). Jeżeli *m* jest ujemne, to *n* musi być liczbą całkowitą.	`SELECT POWER(4,2)`
		`FROM DUAL;`
		
		`POWER(4,2)`
		`----------`
		`16`

ROUND(n [, m])	Zwraca liczbę *n* zaokrągloną do *m* miejsc po przecinku. Jeśli *m* jest pominięte, przyjmuje się *m*=0. Gdy *m* jest ujemne, funkcja zaokrągla *n* do podanej liczby miejsc przed przecinkiem.	`SELECT`
		`ROUND(10.294,1), ROUND(10.294,-1)`
		`FROM DUAL;`
		
		`ROUND(10.294,1) ROUND(10.294,-1)`
		`--------------- ----------------`
		`10,3 10`

SIGN(n)	Zwraca znak liczby n (1 - gdy *n* jest dodatnie, 0 - gdy *n*=0 i -1 - gdy *n* jest ujemne).	`SELECT SIGN(-23), SIGN(14)`
		`FROM DUAL;`
		
		`SIGN(-23) SIGN(14)`
		`---------- ----------`
		`-1 1`

SIN(n)	Zwraca sinus kąta *n* podanego w radianach.	`SELECT`
		`SIN(30 * 3.14159265359/180)`
		`"Sinus 30 stopni"`
		`FROM DUAL;`
		
		`Sinus 30 stopni`
		`---------------`
		`,5`

SQRT(n)	Zwraca pierwiastek kwadratowy z liczby *n* (*n* nieujemne).	`SELECT SQRT(121), SQRT(35)`
		`FROM DUAL;`
		
		`SQRT(121) SQRT(35)`
		`---------- ----------`
		`11 5,91607978`

TAN(n)	Zwraca tangens kąta *n* podanego w radianach.	`SELECT`
		`TAN(135 * 3.14159265359/180)`
		`"Tangens 135 stopni"`
		`FROM DUAL;`
		
		`Tangens 135 stopni`
		`----------`
		`-1`

TRUNC(n [, m])	Zwraca *n* obcięte do *m* miejsc po przecinku. Jeżeli *m* nie jest podane, przyjmuje się *m*=0. Jeśli *m* jest ujemne, to obcinane są liczby przed przecinkiem.	`SELECT`
		`TRUNC(15.79,1), TRUNC(15.79,-1)`
		`FROM DUAL;`
		
		`TRUNC(15.79,1) TRUNC(15.79,-1)`
		`---------- ----------`
		`15,7 10`

Funkcje znakowe zwracające wartości znakowe

Składnia	Opis działania	Przykład

CONCAT (str1, str2)
lub
str1* || *str2
	Dokonuje złączenia (konkatenacji) łańcucha znaków *str1* z łańcuchem znaków *str2*.	`SELECT`
		`CONCAT(`
		`CONCAT(imie,`
		`' pracuje w firmie od '`
		`),`
		`data_zatrudnienia`
		`)`
		`FROM pracownicy`
		`WHERE id_pracownika = 16;`
		
		`CONCAT(CONCAT(IMIE,'PRACUJEWFIRMIEOD'),DA`
		`TA_ZATRUDNIENIA)`
		`----------`
		`KRYSTYNA pracuje w firmie od 00/06/01`
		
		`SELECT`
		`imie`
		`|| ' pracuje w firmie od ' ||`
		`data_zatrudnienia`
		`FROM pracownicy`
		`WHERE id_pracownika = 16;`
		
		`IMIE||'PRACUJEWFIRMIEOD'||`
		`DATA_ZATRUDNIENIA`
		`----------`
		`KRYSTYNA pracuje w firmie od 00/06/01`

INITCAP(str)	Zamienia pierwsze litery wyrazów występujących w łańcuchu *str* na wielkie litery, a pozostałe na małe.	`SELECT INITCAP('Małe cO nieCo')`
		`FROM DUAL;`
		`INITCAP('MAŁECONIECO')`
		`----------`
		`Małe Co Nieco`

LOWER(str)	Zamienia wszystkie litery zawarte w łańcuchu *str* na małe.	`SELECT LOWER('Małe cO nieCo') FROM DUAL;`
		`LOWER('MAŁECONIECO')`
		`--------------------`
		`małe co nieco`

LPAD (str1, n [, str2])	Zwraca łańcuch *str1* uzupełniony lewostronnie do długości *n* ciągami znaków z łańcucha *str2*. Jeśli nie określimy *str2*, domyślnie przyjęta zostanie pojedyncza spacja. Jeżeli długość *str1* jest większa niż *n*, wówczas funkcja zwróci pierwsze *n* ze znaków *str1*.	`SELECT LPAD('Gwiazdki',15,'*.') FROM DUAL;`
		`LPAD('GWIAZDKI',15,'*.')`
		`------------------------`
		`*.*.*.*Gwiazdki`
		`SELECT LPAD('Gwiazdki',5,'*.') FROM DUAL;`
		`LPAD('GWIAZDKI'`
		`---------------`
		`Gwiaz`

LTRIM(str [, set])	Usuwa, począwszy od lewego końca łańcucha *str*, znaki zawarte w zbiorze *set*, dokąd nie napotka znaku spoza tego zbioru. Jeżeli pominiemy argument *set*, domyślnie przyjęta zostanie pojedyncza spacja.	`SELECT LTRIM('xyxXxy','xy') FROM DUAL;`
		`LTRIM('XY`
		`---------`
		`Xxy`

REPLACE (str, szukaj_str [, zastąp_str])	Zwraca *str*, w którym każde wystąpienie łańcucha *szukaj_str* jest zastąpione przez *zastąp_str*. Jeżeli *zastąp_str* zostanie pominięty, to wszystkie wystąpienia *szukaj_str* są usuwane. Jeżeli *szukaj_str* jest równy NULL, to zwracany jest niezmieniony łańcuch *str*.
	(Funkcja ta jest podobna do funkcji TRANSLATE. TRANSLATE zamienia jednak pojedyncze znaki, a REPLACE pozwala zamieniać lub usuwać całe łańcuchy)
		`SELECT REPLACE('JACK and JUE','J','BL') FROM DUAL;`
		`REPLACE('JACKANDJUE','J','BL')`
		`------------------------------`
		`BLACK and BLUE`

RPAD (str1, n [, str2])	Zwraca łańcuch str1 uzupełniony prawostronnie do długości *n* ciągami znaków z łańcucha *str2*. Jeśli nie określimy *str2*, to domyślnie przyjęta zostanie pojedyncza spacja. Jeżeli długość wyrażenia *str1* jest większa niż *n*, to funkcja zwróci pierwsze *n* znaków *str1*.	`SELECT RPAD('Gwiazdki',15,'*.') FROM DUAL;`
		`RPAD('GWIAZDKI',15,'*.')`
		`------------------------`
		`Gwiazdki*.*.*.*`
		`SELECT RPAD('Gwiazdki',5,'*.') FROM DUAL;`
		`RPAD('GWIAZDKI'`
		`---------------`
		`Gwiaz`

RTRIM (str [, set])	Usuwa, począwszy od prawego końca łańcucha *str*, znaki zawarte w zbiorze set, dokąd nie napotka znaku spoza tego zbioru. Jeżeli pominiemy argument set, domyślnie przyjęta zostanie pojedyncza spacja.	`SELECT RTRIM ('INFORMATYKA*.*.*.*.', '*.') FROM DUAL;`
		
		`RTRIM('INFORMATYKA*.*.*.*.','*.')`
		`--------------------------------`
		`INFORMATYKA`

SOUNDEX(str)	Zwraca łańcuch zawierający fonetyczną reprezentację łańcucha *str*. Funkcja pozwala porównywać słowa, które są zapisywane w różny sposób, ale wymawiane tak samo.	`SELECT distinct nazwisko FROM pracownicy WHERE SOUNDEX(nazwisko) = SOUNDEX('Kowalsky');`
		
		`NAZWISKO`
		`---------`
		`KOWALSKI`
		`KOWALSKA`

SUBSTR (str, poz, [, długość])	Zwraca podciąg łańcucha *str* rozpoczynający się od pozycji *poz* o długości *długość*. Jeżeli *długość* nie jest podana, to zwrócony zostanie fragment łańcucha *str* od znaku na pozycji *poz* do końca łańcucha. Jeżeli argument *poz* ma wartość dodatnią, to pozycja liczona jest od początku łańcucha (czyli od 1), jeżeli wartość ta jest ujemna, pozycja liczona jest od końca łańcucha, *poz*=0, traktuje się jak *poz*=1. Jeżeli *długość*<1, to funkcja zwróci wartość NULL.	`SELECT SUBSTR('komputer',3,4) FROM DUAL;`
		
		`SUBSTR('KOMP`
		`------------`
		`mput`
		
		`SELECT SUBSTR('komputer',-5,4) FROM DUAL;`
		
		`SUBSTR('KOMP`
		`------------`
		`pute`

TRANSLATE (str, from_str, to_str);	Zwraca łańcuch *str*, w którym wystąpienia każdego znaku z *from_str* są zastąpione odpowiednim znakiem z *to_str*. Znaki, których nie ma w *from_str* nie są zastępowane. Argument *from_str* może zawierać więcej znaków niż *to_str*. W takim wypadku dodatkowe znaki w *from_str* nie mają odpowiadających znaków w *to_str*. Jeżeli te dodatkowe znaki pojawiają się w łańcuchu *str*, to są usuwane. Argument *to_str* nie może być pustym łańcuchem.
	
	Funkcja TRANSLATE ma działanie podobne do funkcji REPLACE. REPLACE umożliwia zamianę jednego łańcucha znaków na inny oraz usunięcie pewnego łańcucha. TRANSLATE umożliwia dokonanie jednocześnie kilku zamian pojedynczych znaków.
		`SELECT TRANSLATE ('SQL*Plus User''s Guide', ' */''', '___') FROM DUAL;`
		
		`TRANSLATE('SQL*PLUSUSER''SGUIDE','*/''','___')`
		`----------------------------------------`
		`SQL_Plus_Users_Guide`
		
		`SELECT distinct TRANSLATE (nazwisko,'KOLS' ,'abc') FROM pracownicy WHERE nazwisko='KOWALSKI';`
		
		`TRANSLATE(NAZWISKO,'KOLS','ABC')`
		`--------------------------------`
		`abWAcaI`

TRIM (set FROM str)
 	
TRIM (LEADING set FROM str)
 	
TRIM (TRAILING set FROM str)
	Usuwa wystąpienia dowolnego znaku z podanego zbioru *set* z obu stron łańcuchów *str*. Klauzule *leading* i *trailing* powodują, że funkcja TRIM działa odpowiednio jak LTRIM lub RTRIM.	`SELECT DISTINCT zawod, TRIM( 'K' FROM zawod) FROM pracownicy WHERE zawod<>'SPECJALISTA DS. REKLAMY';`
		
		`ZAWOD TRIM('K'FROMZAWOD)`
		`---------- ----------`
		`KASJER ASJER`
		`SPRZĄTACZKA SPRZĄTACZKA`
		`EKONOMISTA EKONOMISTA`
		`INFORMATYK INFORMATY`
		`MAGAZYNIER MAGAZYNIER`
		`PRAWNIK PRAWNI`
		`SEKRETARKA SEKRETARKA`
		`SPRZEDAWCA SPRZEDAWCA`
		`KIEROWCA IEROWCA`
		`PROJEKTANT PROJEKTANT`
		`PLASTYK PLASTY`
		`ANALITYK RYNKU ANALITYK RYNKU`
		`KSIĘGOWA SIĘGOWA`

UPPER(str)	Zamienia wszystkie litery zawarte w łańcuchu *str* na wielkie.	`SELECT LOWER('Małe cO nieCo') FROM DUAL;`
		
		`LOWER('MAŁECONIECO')`
		`--------------------`
		`małe co nieco`

Funkcje znakowe zwracające wartości liczbowe

Składnia	Opis działania	Przykład

ASCII(str)	Zwraca kod ASCII pierwszej litery podanego ciągu znaków *str*.	`SELECT ASCII('Z') FROM DUAL;`
		
		`ASCII('Z')`
		`----------`
		`90`

INSTR (str, substr [, poz [, m]])	Przeszukuje, począwszy od pozycji *poz*, łańcuch *str* w poszukiwaniu podłańcucha *substr*. Zwraca pozycję znaku w łańcuchu *str*, który jest pierwszym znakiem *m*-tego wystąpienia w nim łańcucha *substr*. Gdy *poz* jest liczbą ujemną, pozycja liczona jest od końca łańcucha *str*. Argument *m* musi być liczbą dodatnią. Domyślnie *m* i *poz* mają wartość 1.	`SELECT INSTR('CORPORATE FLOOR','OR',3,2) FROM DUAL;`
		
		`INSTR('CORPORATEFLOOR','OR',3,2)`
		`--------------------------------`
		`14`
		
		`SELECT INSTR('CORPORATE FLOOR','OR',-3,2) FROM DUAL;`
		
		`INSTR('CORPORATEFLOOR','OR',-3,2)`
		`--------------------------------`
		`2`

LENGTH(str)	Zwraca długość łańcucha *str*.	`SELECT LENGTH('INFORMATYKA') FROM DUAL;`
		
		`LENGTH('INFORMATYKA')`
		`--------------------`
		`11`

Funkcje operujące na datach

Składnia	Opis działania	Przykład

ADD_MONTHS (data, n)	Zwraca podaną datę powiększoną (pomniejszoną) o *n* miesięcy (*n* - dowolna liczba całkowita). Jeżeli data odpowiada ostatniemu dniu miesiąca lub gdy miesiąc w dacie zwracanej jest krótszy od tego w dacie wejściowej i nie ma w nim dnia odpowiadającego dniowi z daty wejściowej, to zwracany jest ostatni dzień miesiąca.	`SELECT TO_CHAR (data_zatrudnienia,'DD-MM-YYYY') teraz, TO_CHAR (ADD_MONTHS(data_zatrudnienia,1), 'DD-MM-YYYY') "Miesiąc później" FROM pracownicy WHERE nazwisko = 'ŁUCZAK';`
		
		`TERAZ Miesiąc później`
		`--------- ----------`
		`02-06-2003 02-07-2003`

EXTRACT (par from data)	Zwraca wybraną część daty. Parametrami mogą być m. in.: year, month, day, hour minute, second.	`SELECT EXTRACT( YEAR FROM DATE '1998-03-07') FROM DUAL;`
		
		`EXTRACT(YEARFROMDATE'1998-03-07')`
		`--------------------------------`
		`1998`

LAST_DAY(data)	Zwraca datę odpowiadającą ostatniemu dniu miesiąca zawartego w dacie wejściowej.	`SELECT SYSDATE, LAST_DAY(SYSDATE) FROM DUAL;`
		
		`SYSDATE LAST_DAY`
		`-------- --------`
		`05/10/06 05/10/31`

MONTHS_ BETWEEN (data1,data2)	Zwraca liczbę miesięcy pomiędzy datami *data1* i *data2*. Jeżeli *data1* jest późniejsza niż *data2*, wynik jest dodatni. Jeżeli *data1* jest wcześniejsza niż *data2*, wynik jest ujemny. Jeżeli obie daty dotyczą tego samego dnia miesiąca lub ostatnich dni miesiąca, to wynik będzie liczbą całkowitą. W przeciwnym przypadku część ułamkową oblicza się w stosunku do miesiąca zawierającego 31 dni.	`SELECT MONTHS_BETWEEN( TO_DATE( '02-02-1995','MM-DD-YYYY'), TO_DATE( '01-01-1995','MM-DD-YYYY')) "Liczba miesięcy pomiędzy" FROM DUAL;`
		
		`liczba miesięcy pomiędzy`
		`------------------------`
		`1,03225806`

NEXT_DAY
(data, dzień_tyg)
	Zwraca datę pierwszego dnia zgodnego z parametrem *dzień_tyg*, późniejszego niż podana data. Argument *dzień_tyg* musi być poprawną nazwą dnia tygodnia. Zwracana data posiada taką samą godzinę jak wejściowa.	`SELECT`
		`NEXT_DAY(`
		`  TO_DATE(`
		`  '02-10-2005', 'DD-MM-YYYY'`
		`  ),`
		`  'SOBOTA') "NEXT DAY"`
		`FROM DUAL;`
		
		`NEXT DAY`
		`--------`
		`05/10/08`

ROUND
(data [, fmt])
	Zwraca datę *data* zaokrągloną do jednostki wskazanej przez *fmt*. Jeżeli argument *fmt* zostanie pominięty, data zostanie zaokrąglona do najbliższego dnia.	`SELECT`
		`ROUND (`
		`  TO_DATE(`
		`  '27-10-2005','DD-MM-YYYY'),`
		`'YEAR')`
		`FROM DUAL;`
		
		`ROUND(TO`
		`--------`
		`06/01/01`

SYSDATE	Zwraca bieżącą datę i czas.
	Nie można używać tej funkcji w deklaracji więzu CHECK.
		`SELECT`
		`TO_CHAR(`
		`SYSDATE, 'MM-DD-YYYY HH24:MI:SS')`
		`TERAZ`
		`FROM DUAL;`
		
		`TERAZ`
		`--------------------`
		`10-06-2005 13:37:36`

TO_CHAR **(data [, fmt** **[, nisparam']])**	Konwertuje datę na tekst (typu VARCHAR2) używając podanego formatu *fmt*. Parametr *nlsparam* odpowiada za język, w którym zwracane są nazwy dni i miesięcy. Domyślnie jest to domyślny język sesji.	

TRUNC
(data [, fmt])
	Zwraca datę obciętą do jednostki podanej w *fmt*.	`SELECT`
		`TRUNC(`
		`  TO_DATE('27-10-05','DD-MM-YY'),`
		`  'MM')`
		`FROM DUAL;`
		
		`TRUNC(TO`
		`--------`
		`05/10/01`

W odniesieniu do dat funkcje ROUND i TRUNC mogą używać m. in. formatów opisanych w poniższej tabeli. Domyślny format 'DD' powoduje zaokrąglenie lub obcięcie daty do północy tego samego dnia (czyli usunięcie informacji o czasie).

Format	Opis
CC	wiek
D	liczba dni w tygodniu (od 1 do 7)
DAY	nazwa dnia tygodnia w pełnym brzmieniu, uzupełniona do 9 znaków
DD	numer dnia w miesiącu: od 1 do 31
DDD	numer dnia w roku (od 1 stycznia): od 1 do 31
DL	data w lokalnym formacie długim
DS	data w lokalnym formacie krótkim
DY	trzyliterowy skrót dnia
HH lub HH12	godzina dnia (od 1 do 12)
HH24	godzina dnia, zegar 24-godzinny
IW	numer tygodnia w roku według standardu ISO (1 do 53)
MI	minuta (wartości od 0 do 59)
MM	numer miesiąca
MON	trzyliterowy skrót miesiąca
MONTH	pełna nazwa miesiąca
Q	kwartał
RM	miesiąc w postaci liczby rzymskiej
RR	ostatnie dwie cyfry roku dla bieżącej daty
SS	sekunda
SSSSS	liczba sekund od północy
W	ten sam dzień tygodnia, co w pierwszy dzień roku
WW	ten sam dzień tygodnia, co pierwszy dzień miesiąca
YEAR	rok słownie
SYEAR	rok słownie, daty przed naszą erą oznaczone są wiodącym znakiem minus
YYYY	pełny czterocyfrowy rok
SYYYY	pełny czterocyfrowy rok, daty przed naszą erą oznaczone są wiodącym znakiem minus
Y	ostatnia cyfra roku
YY	ostatnie dwie cyfry roku
YYY	ostatnie trzy cyfry roku

Przy zaokrąglaniu: w przypadku roku data jest zaokrąglana do góry od 1 lipca, w przypadku kwartału - zaokrąglana w górę od 16 dnia drugiego miesiąca kwartału, w przypadku miesiąca - zaokrąglana w górę od 16 dnia miesiąca.

Znaki przestankowe wykorzystywane do wyświetlania danych zwracanych przez funkcję TO_CHAR oraz ignorowane w formacie argumentu funkcji TO_DATE:
/ , - : . ;

Funkcje konwertujące

Składnia	Opis działania	Przykład

TO_CHAR (n [, fmt])	Zamienia liczbę *n* na znak (VARCHAR2) używając opcjonalnego ciągu formatującego *fmt*.	`SELECT LENGTH(TO_CHAR(-10000)) FROM DUAL;`
		
		`LENGTH(TO_CHAR(-10000))`
		`----------`
		`6`

TO_DATE (str [, fmt])	Przekształca ciąg znaków *str* w datę (DATE), zgodnie z formatem *fmt*. Jeżeli argument *fmt* jest opuszczony *str* musi być w domyślnym formacie daty.	`SELECT TO_DATE( 'Grudzień 24, 1989, 11:00', 'MONTH DD YYYY HH:MI') FROM DUAL;`
		
		`TO_DATE(`
		`--------`
		`89/12/24`

TO_NUMBER (str [, fmt])	Zamienia łańcuch znaków *str* na liczbę (NUMBER).	`SELECT TO_NUMBER('100') FROM DUAL;`
		`TO_NUMBER('100')`
		`----------`
		`100`

Inne

Składnia	Opis działania	Przykład
COALESCE (wyr1, wyr2, ...)	Zwraca pierwsze z wyrażeń *wyr1*, *wyr2*, ..., które ma wartość różną od NULL. Jeżeli wszystkie wyrażenia mają wartość NULL, funkcja zwraca wartość NULL. Wyrażenie COALESCE (expr1, expr2) jest równoważne: CASE WHEN expr1 IS NOT NULL THEN expr1 ELSE expr2 END. Analogicznie, COALESCE (expr1, expr2, ..., expr*n*), gdzie *n*>=3 jest równoważne: CASE WHEN expr1 IS NOT NULL THEN expr1 ELSE COALESCE (expr2, ..., expr*n*) END	

DECODE (wyr, s1, r1, s2, r2,...[, domyślna])

Porównuje wyrażenie *wyr* do kolejnych *s1*, *s2*, ... Jeżeli *wyr* jest równe któremuś z tych wyrażeń, w wyniku zwracana jest odpowiadająca jemu wartość typu *r*. Jeżeli żadna z wartości *s1*, *s2*, ... nie jest równa wejściowemu wyrażeniu funkcja zwraca wartość domyślną. Jeśli jest ona pominięta, funkcja zwraca wartość NULL.
Funkcja DECODE uważa dwie wartości NULL za równe, jeśli więc wejściowe wyrażenie ma wartość NULL, to funkcja zwróci pierwsze z wyrażeń typu *r* odpowiadające wyrażeniu typu *s* o wartości NULL.

Uwaga! Oracle automatycznie konwertuje wyrażenia r2, r3,... tak, aby były tego samego typu co wyrażenie r1. Gdy konwersja jest niemożliwa (np. r1 jest liczbą, a r2 jest łańcuchem znakowym) zostanie zgłoszony błąd.

```
SELECT nazwisko, pensja,
DECODE (sign(pensja-1500), 1,
':)', -1, ':(', 'ok')
FROM pracownicy
WHERE id_pracownika in (16, 30,
51);

NAZWISKO     PENSJA   DECODE
--------   ---------- ------
RACZKOWSKA      1600  :)
LENART          1500  ok
NOWAK           1300  :(
```

CASE
WHEN war1 THEN r1
WHEN war2 THEN r2
...
[ELSE p]
END

Sprawdza kolejno warunki *war1, war2*, ...Jeśli któryś z warunków jest spełniony zwrócona zostaje odpowiednia wartość typu *r*. Jeśli nie zachodzi żaden z warunków, a użyta jest klauzula ELSE, to w wyniku zwrócona zostanie wartość *p* (w przypadku braku klauzuli ELSE zwrócona zostanie wartość NULL).

```
SELECT nazwisko, pensja,
CASE
WHEN pensja>1500 THEN  ':)'
WHEN pensja<1500 THEN  ':('
ELSE 'ok'
END
FROM pracownicy
WHERE id_pracownika in (16, 30,
51);

NAZWISKO    PENSJA   CASEWH
----------- -------- ------
RACZKOWSKA   1600     :)
LENART       1500     ok
NOWAK        1300     :(
```

CASE wyr
WHEN s1 THEN r1
WHEN s2 THEN r2
...
[ELSE p]
END

Porównuje wyrażenie wyr z kolejnymi wartościami *s1, s2*,... Jeśli wyrażenie jest równe któremuś z *s1*, *s2*,... to w wyniku zwracana jest odpowiadająca mu wartość typu *r*. Jeśli żadna z wartości *s1, s2*, ... nie jest równa wyjściowemu wyrażeniu, a jest użyta klauzula ELSE, to w wyniku zwrócona zostanie wartość *p* (w przypadku braku klauzuli ELSE zwrócona zostanie wartość NULL).

W obu przypadkach wyrażenia *r1, r2, ..., r* muszą być tego samego typu.

```
SELECT nazwisko, pensja,  CASE
                  WHEN pensja>1500 THEN  pensja
                  WHEN pensja<1500 THEN  ':('
                  ELSE 'ok'
                  END
FROM pracownicy
WHERE id_pracownika in (16, 30, 51);

BŁĄD w linii 3:
ORA-00932: niespójne typy danych: oczekiwano NUMBER, uzyskano CHAR
```

```
SELECT nazwisko, pensja,
CASE pensja
WHEN 1500 THEN  ':)'
WHEN 1600 THEN  ':]'
ELSE '?'
END
FROM pracownicy
WHERE id_pracownika in (16, 30,
51);

NAZWISKO    PENSJA    CASEPE
----------  --------- ------
RACZKOWSKA    1600      :]
LENART        1500      :)
NOWAK         1300       ?
```

GREATEST (wyr1, wyr2, ...)	Zwraca największą wartość z listy podanych wyrażeń.	`SELECT GREATEST ('ALA', 'ALICJA','ANNA') "LEAST" FROM DUAL;`
		
		`LEAST`
		`------------`
		`ANNA`

LEAST (wyr1, wyr2, ...)	Zwraca najmniejszą wartość z listy podanych wyrażeń.	`SELECT LEAST('ALA', 'ALICJA', 'ANNA') "LEAST" FROM DUAL;`
		
		`LEAST`
		`---------`
		`ALA`

NVL (wyr1,wyr2)	Jeżeli *wyr1* ma wartość NULL, to funkcja zwraca *wyr2*. W przeciwnym przypadku funkcja zwraca *wyr1*. Funkcja ta pozwala zastępować wartości NULL odpowiednimi łańcuchami.	`SELECT nazwisko, NVL(dodatek,0) FROM pracownicy WHERE nazwisko LIKE 'E%';`
		
		`NAZWISKO  NVL(DODATEK,0)`
		`--------  --------------`
		`EKIERT  0`

NVL2 (wyr1,wyr2,wyr3)	Jeżeli *wyr1* ma wartość różną od NULL, to funkcja zwraca *wyr2*. Jeżeli *wyr1* ma wartość NULL, to funkcja zwraca *wyr3*.	

Opracowane na podstawie:

Oracle® Database
SQL Reference
10*g* Release 2 (10.2)
B14200-01
June 2005

Kevin Loney
Oracle Database 10 g
Kompendium administratora
Helion 2005