

Rozdział 3 Funkcje

funkcje znakowe, funkcje liczbowe, funkcje operujące na datach, funkcje konwersji, funkcje polimorficzne, funkcje grupowe, podział relacji na grupy, klauzule GROUP BY i HAVING



Funkcje znakowe (1)

- LOWER(wartość)
 - zamienia WIELKIE litery na *małe*
- UPPER(wartość)
 - zamienia *małe* litery na WIELKIE
- INITCAP(wartość)
 - zamienia pierwsze litery w słowie na duże
- LPAD(wartość, n [, 'ciąg']), RPAD(wartość, n [, 'ciąg'])
 - Uzupełnia kolumny z lewej (prawej) strony podanym *ciągiem* aż do długości n znaków. Jeśli ciąg nie został podany to wypełnia spacjami

```
SELECT LOWER(nazwa), INITCAP(' mieści się na '), UPPER(adres)
FROM zespoly;
```

```
SELECT LPAD(nazwa,25,'*'), LPAD(nazwa,25), RPAD(nazwa,25,'.')
FROM zespoly;
```



Funkcje znakowe (2)

- SUBSTR(wartość, n [, m])
 - z podanego łańcucha znaków wycina m znaków począwszy od pozycji n-tej
- INSTR(wartość, 'ciąg' [, m, n])
 - wskazuje miejsce pierwszego (n-tego) wystąpienia *ciągu* w łańcuchu znaków począwszy od pozycji m-tej
- LTRIM(wartość [, 'znaki']), RTRIM(wartość [, 'znaki'])
 - usuwa z lewej (prawej) strony podane znaki (spacje)
- LENGTH(wartość)
 - zwraca długość łańcucha znaków

```
SELECT SUBSTR(nazwisko,3,4), INSTR(etat,'PROF'), LENGTH(etat)
FROM pracownicy;
```

```
SELECT nazwa, LTRIM(nazwa,'ABCD')
FROM zespoly;
```



Funkcje znakowe (3)

- TRANSLATE(źródło, z, na)
 - Każde wystąpienie w *źródle* znaku z ciągu z zostanie zastąpione odpowiadającym mu znakiem z ciągu *na*
- REPLACE(źródło, wzór, nowy)
 - Każde wystąpienie w *źródle* ciągu *wzorzec* zostanie zastąpione przez ciąg *nowy*

```
SELECT etat, REPLACE(etat,'AS','**') , TRANSLATE(etat,'AS','**') ,
REPLACE(etat,'ASY','**') , TRANSLATE(etat,'ASY','**')
FROM pracownicy;
```



Funkcje liczbowe (1)

- **ROUND(wartość,n)**
 - zaokrągla wartość do n-tego dziesiętnego miejsca po przecinku
- **TRUNC(wartość,n)**
 - obcina wartość do n-tego dziesiętnego miejsca po przecinku
- **CEIL(wartość), FLOOR(wartość)**
 - najmniejsza (największa) liczba całkowita większa lub równa (mniejsza lub równa) podanej wartości

```
SELECT ROUND(123.456,1), ROUND(123.456), ROUND(123.456,-1),  
       TRUNC(123.456,1), TRUNC(123.456), TRUNC(123.456,-1)  
FROM dual;
```

```
SELECT FLOOR(1.5), CEIL(1.5)  
FROM dual;
```

Funkcje liczbowe (2)

- **POWER(wartość,n)**
 - podnosi wartość do podanej potęgi
- **SQRT(wartość)**
 - oblicza pierwiastek kwadratowy z podanej wartości
- **ABS(wartość)**
 - oblicza wartość bezwzględną wyrażenia
- **MOD(wartość1, wartość2)**
 - zwraca resztę z dzielenia
- **SIGN(wartość)**
 - zwraca -1 jeśli wartość jest ujemna, 0 dla 0 i 1 jeśli wartość jest dodatnia

```
SELECT POWER(2,16), SQRT(64),  
       ABS(-100), MOD(123456789,10)  
FROM dual;
```

Funkcje operujące na datach (1)

- Oracle przechowuje daty w polach typu DATE zawierających stulecie, rok, miesiąc, dzień, godzinę, minutę i sekundę. Zakres dat to 1 stycznia 4712 p.n.e do 31 grudnia 9999.
- Funkcje **CURRENT_DATE** i **CURRENT_TIMESTAMP** zwracają bieżącą datę i znacznik czasowy. Funkcja Oracle **SYSDATE** zwraca bieżącą datę systemową.
- Słowo kluczowe **DATE** służy do reprezentacji literałów typu DATE w domyślnym formacie.
- Wewnętrznie daty są przechowywane w postaci liczb, możliwe jest stosowanie operatorów dodawania i odejmowania.

```
SELECT CURRENT_DATE, CURRENT_TIMESTAMP  
FROM dual;
```

```
SELECT zatrudniony, SYSDATE, SYSDATE-zatrudniony  
FROM pracownicy;
```

Funkcje operujące na datach (2)

- **EXTRACT (YEAR/MONTH/DAY/HOUR/MINUTE/SECOND/ TIMEZONE_HOUR/TIMEZONE_MINUTE/TIMEZONE_REGION/ TIMEZONE_ABBR FROM data/czas/interwał)**
 - Zwraca jeden ze składników daty, znacznika czasowego lub interwału czasowego (np. dzień, rok, godzinę, ...)

```
SELECT EXTRACT (YEAR FROM DATE '2003-10-01') FROM dual;
```

```
SELECT EXTRACT (HOUR FROM CURRENT_TIMESTAMP) || ':'  
       || EXTRACT (MINUTE FROM CURRENT_TIMESTAMP) AS now  
FROM dual;
```

```
SELECT EXTRACT (YEAR FROM zatrudniony) AS rok_zatrudnienia  
FROM pracownicy
```

Funkcje operujące na datach (3)

- MONTHS_BETWEEN(data1,data2)
 - Zwraca liczbę miesięcy jakie upłynęły między datami
- ADD_MONTHS(data,n)
 - Zwraca datę plus n miesięcy kalendarzowych
- NEXT_DAY(data,dzień)
 - Zwraca następną datę po podanej przypadającą na podany dzień
- LAST_DAY(data)
 - Zwraca datę ostatniego dnia w miesiącu podanej daty

```
SELECT MONTHS_BETWEEN(SYSDATE, zatrudniony)
FROM pracownicy;
```

```
SELECT NEXT_DAY(SYSDATE,'WTOREK')
FROM dual;
```

```
SELECT LAST_DAY(DATE '1992-02-01')
FROM dual;
```

Operacje na interwałach czasowych (1)

- Interwał czasowy reprezentuje różnicę w czasie między datami i znacznikami czasowymi.
- Typy interwałów: YEAR TO MONTH, DAY TO SECOND
- Słowo kluczowe INTERVAL służy do reprezentacji literałów typu interwał czasowy.
- Przykłady literałów interwałowych:
 - INTERVAL '4 5:12' DAY TO MINUTE - 4 dni, 5 godz. i 12 minut.
 - INTERVAL '400 5' DAY(3) TO HOUR - 400 dni 5 godz.
 - INTERVAL '10' HOUR - 10 godz.
 - INTERVAL '10:22' MINUTE TO SECOND - 10 minut 22 sekundy.
 - INTERVAL '10' MINUTE - 10 minut.
 - INTERVAL '4' DAY - 4 dni.
 - INTERVAL '1-6' YEAR TO MONTH – półtora roku
 - INTERVAL '120' HOUR(3) - 120 godz.
 - INTERVAL '30.12345' SECOND(2,4) – 30.12354 sek.

Operacje na interwałach czasowych (2)

```
SQL> SELECT INTERVAL'20' DAY - INTERVAL'240' HOUR FROM dual;
```

```
INTERVAL'20'DAY-INTERVAL'240'HOUR
```

```
+0000000010 00:00:00.000000000
```

```
SQL> SELECT EXTRACT (DAY FROM
      (INTERVAL'20' DAY - INTERVAL'240' HOUR)) FROM dual;
```

```
EXTRACT(DAYFROM(INTERVAL'20'DAY-INTERVAL'240'HOUR))
```

10

Operacje na interwałach czasowych (3)

```
SELECT (sysdate - zatrudniony ) DAY TO SECOND "Pracuje dni",
      (sysdate - zatrudniony ) YEAR TO MONTH "Pracuje lat"
FROM pracownicy;
```

```
SQL> SELECT nazwisko,
      EXTRACT (DAY FROM (sysdate - zatrudniony ) DAY TO SECOND) "Pracuje dni",
      EXTRACT (YEAR FROM (sysdate - zatrudniony ) YEAR TO MONTH) "Pracuje lat"
FROM pracownicy;
```

NAZWISKO	Pracuje dni	Pracuje lat
WEGLARZ	13091	35
BLAZEWICZ	11144	30
SLOWINSKI	9560	26
BRZEZINSKI	12909	35
...		

Funkcje konwersji

CAST (wartość AS typ) – ogólna standardowa funkcja konwersji

TO_CHAR(liczba|data [, 'format'])

TO_NUMBER('tekst')

TO_DATE('tekst', 'format')

SCC	Stulecie
YYYY	Rok
BC AD	Wskaźnik ery
MM	Miesiąc
MONTH	Nazwa miesiąca
D DD DDD	Dzień
DAY	Nazwa dnia
AM PM	Wskaźnik pory dnia
HH	Godziny
HH24	Godziny (24h)
MI	Minuty
SS	Sekundy

```
SELECT nazwisko,
CAST (placa_pod AS VARCHAR2(20))
FROM pracownicy;
```

```
SELECT TO_DATE('00_01_10','YY_DD_MM')
FROM dual;
```

```
SELECT TO_CHAR(SYSDATE,
'SCC YYYY MMTH DAY HH24 MI SS')
FROM dual;
```

```
SELECT TO_NUMBER('-12345,67890')
FROM dual;
```

Funkcje polimorficzne

- NVL(wyrażenie1, wyrażenie2)
 - Jeśli wyrażenie1 ma wartość różną od NULL to funkcja zwraca wyrażenie1, w przeciwnym przypadku zwraca wyrażenie2
- NVL2(wyrażenie1, wyrażenie2, wyrażenie3)
 - Jeżeli wyrażenie1 ma wartość różną od NULL to funkcja zwraca wyrażenie2, w przeciwnym przypadku zwraca wyrażenie3
- GREATEST(w1,...) LEAST(w1,...)
 - Zwraca największą (najmniejszą) wartość z listy

```
SELECT nazwisko, NVL(placa_dod,0) AS NVL,
NVL2(placa_dod,10,0) AS NVL2
FROM pracownicy;
```

```
SELECT nazwisko,
GREATEST(placa_pod/30, NVL(placa_dod,0))
FROM pracownicy;
```

Wyrażenie CASE

```
CASE wyrażenie WHEN wartość1a THEN wartość1b
WHEN wartość2a THEN wartość2b
[ ELSE wartość3 ] END
```

```
SELECT nazwisko, etat,
( CASE etat WHEN 'DYREKTOR' THEN ' *** '
WHEN 'PROFESOR' THEN ' *** '
ELSE TO_CHAR(placa_pod) END) AS placa_pod
FROM pracownicy;
```

```
CASE WHEN warunek1 THEN wartość1
WHEN warunek2 THEN wartość2
[ ELSE wartość3 ] END
```

```
SELECT nazwisko, etat,
( CASE WHEN etat IN ('DYREKTOR','PROFESOR') THEN ' *** '
ELSE TO_CHAR(placa_pod) END) AS placa_pod
FROM pracownicy;
```

Funkcja DECODE

- DECODE(wyrażenie, S1, W1, [S2, W2, ...] domyślne)
 - Jeśli wyrażenie równa się S1 to funkcja zwraca W1, jeśli wyrażenie równa się S2 to funkcja zwraca W2, ..., w przeciwnym wypadku funkcja zwraca wartość domyślną.
- Niestandardowa funkcja w Oracle
- Częściowo pokrywa funkcjonalność CASE

```
SELECT nazwisko, DECODE(etat, 'PROFESOR', ' *** ',
'DYREKTOR', ' *** ', TO_CHAR(placa_pod) ) AS placa_pod
FROM pracownicy;
```

Funkcje grupowe

Operują na podzbiorach krotek relacji, nazywanych *grupami*, wyznaczają wartość skalarną operując na zbiorze wartości odczytanych z wielu krotek.

funkcje:

- AVG([distinct|all] wyrażenie)
- COUNT([distinct|all] wyrażenie)
- MAX([distinct|all] wyrażenie)
- MIN([distinct|all] wyrażenie)
- SUM([distinct|all] wyrażenie)
- VARIANCE([distinct|all] wyrażenie)
- STDDEV([distinct|all] wyrażenie)

	PLACA
...	200
...	340
...	null
...	456
...	120

SUM

```
SELECT COUNT(*), MAX(placa_dod)
FROM pracownicy
WHERE id_zesp=20;
```

```
SELECT AVG(placa_pod)
FROM pracownicy;
```

Podział krotek na grupy - klauzula GROUP BY (1)

NAZWISKO	...	ID_Z	PLACA	NAZWISKO	...	ID_Z	PLACA	
WEGLARZ	...	10	1730	WEGLARZ	...	10	1730	SUM
BLAZEWICZ	...	40	1350	BRZEZINSKI	...	20	960	
SLOWINSKI	...	30	1070	MORZY	...	20	830	SUM
BRZEZINSKI	...	20	960	KOSZLAJDA	...	20	590	
MORZY	...	20	830	SLOWINSKI	...	30	1070	SUM
KOSZLAJDA	...	20	590	BIALY	...	30	250	
ZAKRZEWICZ	...	30	208	ZAKRZEWICZ	...	30	208	SUM
BIALY	...	30	250	BLAZEWICZ	...	40	1350	

Podział krotek na grupy - klauzula GROUP BY (2)

- Dla każdego zespołu wylicz średnią płacę zatrudnionych w nim pracowników

```
SELECT id_zesp, AVG(placa_pod)
FROM pracownicy
GROUP BY id_zesp;
```

Podział grup na podgrupy

- W ramach każdego zespołu dla każdego etatu występującego w zespole oblicz najwyższą płacę

```
SELECT id_zesp, etat, MAX(placa_pod)
FROM pracownicy
GROUP BY id_zesp, etat;
```

Podział krotek na grupy - klauzula GROUP BY (3)

UWAGA!!!!

Na liście atrybutów w klauzuli SELECT mogą się pojawić TYLKO funkcje agregujące i atrybuty grupujące. Obecność każdego innego atrybutu spowoduje błąd.

```
SELECT etat, nazwisko, MAX(placa_pod)
FROM pracownicy
GROUP BY etat;
```

```
SELECT etat, nazwisko, MAX(placa_pod)
*
```

BŁĄD w linii 1:
ORA-00979: not a GROUP BY expression

BŁĄD!!!

Klauzula HAVING

Pozwala na wybór grup spełniających określone warunki, działa dla grup analogicznie jak klauzula WHERE dla pojedynczych krotek

- wyświetl grupy etatowe, których maksymalna płaca podstawowa przekracza 1000 złotych

```
SELECT etat, SUM(placa_pod)
FROM pracownicy
GROUP BY etat
HAVING MAX(placa_pod) > 1000;
```

- wyświetl nazwy etatów i liczbę zatrudnionych na danym etacie, uwzględnij tylko etaty, na których jest zatrudnionych co najmniej 2 pracowników otrzymujących płacę dodatkową

```
SELECT etat, COUNT(*) FROM pracownicy
WHERE placa_dod IS NOT NULL
GROUP BY etat HAVING COUNT(*) >= 2;
```