Wykład 8

Implementacja języka SQL w systemach baz danych Oracle – specyficzne konstrukcje i funkcje Oracle SQL, funkcje numeryczne, znakowe, daty i czasu, parametryzacja zapytań.

I. Wybrane specyficzne funkcje i konstrukcje Oracle SQL

I. 1 Operatory DISTINCT i ALL - dotyczące całych wierszy

DISTINCT - w przypadku, gdy wiersz wynikowy zapytania powtarza się, powoduje wyświetlenie go jednokrotnie:

```
select DISTINCT PENSJA from PRACOWNICY;
```

ALL - jest to ustawienie domyślne i dlatego nie wymaga stosowania; powoduje wyświetlenie wszystkich wierszy wynikowych zapytania także wtedy, gdy się powtarzają:

```
select ALL PENSJA from PRACOWNICY;

select PENSJA from PRACOWNICY;
```

I. 2 Predefiniowana wartość NULL - wartość pusta (brak wartości) atrybutu lub wyrażania.

Dozwolone konstrukcje: Atrybut lub wyrażenie IS NULL
Atrybut lub wyrażenie IS NOT NULL

```
select ID_P, PENSJA from PRACOWNICY where PENSJA is not null;

select ID_P, PENSJA + 2000 from PRACOWNICY where PENSJA is not null;

select ID_P, PENSJA + 2000 from PRACOWNICY where PENSJA + 2000 is not null;
```

I. 3 Funkcja NVL - pozwala wyświetlić dowolną wartość, jeśli atrybut lub wyrażenie posiada wartości NULL.

NVL(atrybut lub wyrażenie, wartość)

wyrażenie lub atrybut jest NULL => wyświetlana jest wartość

```
select ID_P, NVL(PENSJA,0) from PRACOWNICY;
```

I. 4 Operator LIKE

atrybut lub wyrażenie LIKE wzorzec

wzorce: 'S' , 'S%' , 'S_ _'

- Wyświetlenie każdego nazwiska zaczynającego się na 'S' (także składającego się tylko z litery 'S':

  select ID_P, NAZWISKO from PRACOWNICY where NAZWISKO LIKE 'S%';

- Wyświetlenie każdego nazwiska zaczynającego się na 'S' i składającego się dokładnie z 3 znaków, z których żaden nie może być spacją:

  select ID_P, NAZWISKO from PRACOWNICY where NAZWISKO LIKE 'S_ _';

I. 5 Konstrukcja BETWEEN … AND ….

Konstrukcja:

atrybut lub wyrażenie BETWEEN wartość1 AND wartość2

jest równoważna koniunkcji:

atrybut lub wyrażenie >= wartość1
AND atrybut lub wyrażenie <= wartość2

Dlatego jest używana do sprawdzania warunku, czy wartość atrybutu lub wyrażenia należy do przedziału domkniętego [wartość1 , wartość2]

```
select ID_P, NAZWISKO from PRACOWNICY
where PENSJA between 1000 and 3000;
```

```
select ID_P, NAZWISKO from PRACOWNICY
where PENSJA >= 1000 and PENSJA <= 3000;
```

I. 6 Operator EXISTS

Konstrukcja EXISTS (podzapytanie) zwraca wartość logiczną:

Podzapytanie zwraca chociaż jeden wiersz wynikowy => TRUE

Podzapytanie nie zwraca żadnego wiersza wynikowe => FALSE

Znaleźć pracowników, którzy posiadają podwładnych

```
select ID_P, IMIĘ, NAZWISKO from PRACOWNICY P1 where
EXISTS (select ID_P from PRACOWNICY P2 where P2.ID_BP=P1.ID_P);
```

Znaleźć pracowników, którzy nie posiadają podwładnych

```
select ID_P, IMIĘ, NAZWISKO from PRACOWNICY P1 where NOT
EXISTS (select ID_P from PRACOWNICY P2 where P2.ID_BP=P1.ID_P);
```

I. 7 Drzewa zależności:

konstrukcja CONNECT BY [PRIOR] … [START WITH] …

CONNECT BY - definiuje strukturę powiązań (musi wystąpić)

PRIOR - definiuje hierarchię w drzewie zależności, tzn. znajduje wszystkie wartości (nadrzędne) dla atrybutu z PRIOR, a następnie wszystkie rekordy powiązane z nimi (podrzędne)

START WITH - definiuje warunek początku konstruowania drzewa zależności

```
select level, ID_P, IMIĘ, NAZWISKO, ID_BP from PRACOWNICY
CONNECT BY PRIOR ID_P = ID_BP
START WITH ID_BP is null;
```

```
select level, ID_P, IMIĘ, NAZWISKO, ID_BP from PRACOWNICY
CONNECT BY PRIOR ID_P = ID_BP;
```

=> drzewo zależności będzie konstruowane bez narzucenia warunku początkowego na wartość atrybutu nadrzędnego

```
select level, ID_P, IMIĘ, NAZWISKO, ID_BP from PRACOWNICY
CONNECT BY ID_P = ID_BP;
```

=> tak jak w poprzednim przykładzie, ale traktowane jako 1 poziom hierarchii.

I. 8 Instrukcje wielokrotnego wyboru w Oracle SQL

DECODE(atrybut lub wyrażenie, wartość_spr1, wartość_zwr1
[, wartość_spr2, wartość_spr2, … , wartość_zwr_domyślnie])

Kod tej funkcji można by w języku C/C++ przedstawić, używając instrukcji switch:

```
typ_wartości_zwracanej DECODE (typ_wyrażenia wyrażenie,
wartość_spr1 [, wartość_spr2, …])
{
  switch  (wyrażenie)
  {
          case wartość_spr1:
                  return wartość_zwr1;
        [
          case wartość_spr2:
                  return wartość_zwr2;
          …
         default:
                  return wartość_zwr_domyślnie;
        ]
  }
}
```

lub następującej if-elseif-else:

```
typ_wartości_zwracanej DECODE(typ wyrażenia wyrażenie,
wartość_spr1 [, wartość_spr2, …])
{
          if  (wyrażenie == wartość_spr1)
          {
                  return wartość_zwr1;
          }
        [
          elseif (wyrażenie == wartość_spr2)
          {
                  return wartość_zwr2;
          }
          …
          else
          {
                  return wartość_zwr_domyślnie;
          }
        ]
}
```

```
select ID_P, decode(ID_D,12,'Zarząd',11,'Produkcja')
from PRACOWNICY;

select ID_P, decode(ID_D, 12,'Zarząd',11,'Produkcja','Administracja)
from PRACOWNICY;
```

```
CASE atrybut lub wyrażenie
WHEN wyrażenie_spr_1 THEN wyrażenie_zwr_1
[WHEN wyrażenie_spr_2 THEN wyrażenie_zwr_2]
…
[WHEN wyrażenie_sprn THEN wyrażenie_zwr_n]
[ELSE wyrażenie_zwr]
END
```

```
select ID_P, CASE ID_D WHEN 12 THEN 'Zarząd'
                       WHEN 11 THEN 'Produkcja' END
from PRACOWNICY;
```

```
select ID_P, CASE ID_D WHEN 12 THEN 'Zarząd'
                       WHEN 11 THEN 'Produkcja'
                       ELSE 'Administracja' END
from PRACOWNICY;
```

```
CASE
WHEN kryterium_1 THEN wyrażenie_zwr_1
[WHEN kryterium_2 THEN wyrażenie_zwr_2]
…
[WHEN kryterium_sprn THEN wyrażenie_zwr_n]
[ELSE wyrażenie_zwr]
END
```

```
select
AVG(CASE WHEN PENSJA > 3000 THEN PENSJA ELSE 3000 END)
from PRACOWNICY;
```

II. Podstawowe typy danych (dziedziny atrybutów) w Oracle'u

II. 1 Numeryczne typy danych

NUMBER - liczby zmiennoprzecinkowe,
maksymalnie 38 cyfr znaczących, maksymalnie $9.99 * 10^{124}$
NUMBER(w) - NUMBER o ilości cyfr znaczących w<=38
NUMBER(w,d) - NUMBER(w), gdzie skala d to ilość cyfr znaczących
po kropce dziesiętnej spośród wszystkich
w cyfr znaczących,
d może być ujemne (-84 <= d <= 127)

NUMBER(w) jest równoważny NUMBER(w,0)

II. 2 Tekstowe typy danych

CHAR(r) - łańcuchy znaków o stałej długości r (1 <= r <= 2000 znaków
lub bajtów) uzupełniany spacjami przy porównaniach
CHAR - domyślnie r=1

VARCHAR2(r) - łańcuch znaków o zmiennej długości r (1 <= r <= 4000
znaków lub bajtów), rozmiar r musi być podany

II. 3 Typ daty i czasu

DATE - przechowuje zarówno datę, jak i czas.

II. 4 Typy LOBs (Large Objects)

CLOB (Character Large Object) – obiekt tekstowy

BLOB (Binary Large Object) – obiekt binarny

BFILE (Binary File) – plik

Ograniczenie:
Dozwolony jest jeden atrybut o typie danych z grupy LOB w tabeli.

III. Wybrane funkcje numeryczne

III. 1 Funkcje zaokrąglające:

CEIL(x) - zaokr. do najmniejszej liczby całkowitej nie mniejszej od x ("do góry")
FLOOR(x) - zaokr. do największej liczby całkowitej nie większej od x ("do dołu")
TRUNC(x[,m]) - obcięcie do x do m miejsc po przecinku (domyślnie m=0),
m może być ujemne
ROUND(x[,m]) - zaokrąglenie x do m miejsc po przecinku (domyślnie m=0),
m może być ujemne

III. 2 Funkcje znaku

ABS(x) - |x|

SIGN(x) - funkcja signum $\operatorname{sgn}(x) = \begin{cases} -1 \text{ dla } x < 0 \\ 0 \text{ dla } x = 0 \\ 1 \text{ dla } x > 0 \end{cases}$

III. 3 Matematyczne funkcje elementarne

- potęgowanie, pierwiastkowanie i logarytmowanie:

SORT(x) - $\sqrt{x}$, jeśli x < 0, to zwraca NULL
POWER(m,x) - m^x

EXP(x) - e^x
LN(x) - ln(x), x>0
LOG(m,x) - $\log_m x$

- funkcje trygonometryczne i hiperboliczne funkcje trygonometryczn (x w rad):

SIN(x) - sin(x)
COS(x) - cos(x)
TAN(x) - tg(x)

SINH(x) - sinh(x)
COSH(x) - cosh(x)
TANH(x) - tgh(x)

- reszta z dzielenia x przez y

MOD(x,y)

IV. Funkcje konwersji

Funkcja	Zwracana wartość
TO_CHAR(expr[,fmt])	skonwertuj *expr* z wartości typu **number** lub **date** do wartości znakowej według formatu *fmt*.
TO_DATE(char[,fmt])	konweruje wyrażenie typu **char** do typu **date** według formatu *fmt*
TO_NUMBER(char [,fmt])	konwertuje wyrażenie typu **char** do typu **number** wed lug formatu *fmt*

- konwersja liczby lub daty i czasu do tekstu
  TO_CHAR(wartość liczbowa lub data i czas[, format konwersji]) => data i czas
  select to_char(SYSDATE,'DD.MM.YYYY HH24:MI:SS') from dual;
- konwersja tekstu do daty i czasu
  TO_DATE(data i czas[, format konwersji]) => data
  select to_date('22.10.2006 21:45:34','DD.MM.YYYY HH24:MI:SS') from dual;
- konwersja tekstu do liczby
  TO_NUMBER(tekst[, format konwersji]) => liczba
  select to_number('1220.35','9,999,999.99');

Formaty konwersji *wspólne dla* TO_CHAR *i* TO_DATE

format	znaczenie
MM	numer miesiąca, cyframi arabskimi
RM	numer miesiąca, cyframi rzymskimi
MON	trzyliterowy skrót nazwy miesiąca, dużymi literami
Mon	j/w, ale tylko pierwsza litera duża
mon	jw, wszystkie litery małe
MONTH	pełna nazwa miesiąca, dużymi literami
Month	j/w, tylko pierwsza litera duża
month	j/w, wszystkie litery małe
DDD	numer dnia w roku
DD	numer dnia w miesiącu
D	numer dnia w tygodniu
DY	trójliterowy skrót nazwy dnia tygodnia, dużymi literami
Dy	j/w, tylko pierwsza litera duża
dy	j/w, wszystkie litery małe
DAY	pełna nazwa dnia tygodnia, wszystkie litery duże
Day	j/w, tylko pierwsza litera duża
day	j/w, wszystkie litery małe
YYYY	czterocyfrowy numer roku
SYYYY	j/w, dla lat pne - jako liczba ujemna
IYYY	j/w zgodny z normą ISO
YYY	trzy ostatnie cyfry numeru roku
IYY	j/w, ISO format

YY	dwie ostatnie cyfry roku
IY	j/w, ISO format
Y	ostatnia cyfra numeru roku
I	j/w, ISO format
RR	ostatnie dwie cyfry numeru roku, być może z innego wieku
YEAR	pełna, słowna nazwa roku, wszystkie litery duże
Year	j/w, tylko pierwsza litera duża
year	j/w, wszystkie litery małe
Q	numer kwartału w roku
WW	numer tygodnia w roku
W	numer tygodnia w miesiącu
IW	numer tygodnia w roku, ISO format
J	data Juliańska (od 31/12 4713 pne)
HH	numer godziny w dniu, 1 - 12
HH12	j/w
HH24	numer godziny, format 24 godzinny
MI	minuta w godzinie
SS	sekundy minuty
SSSSS	sekundy od północy

***Formaty konwersji używane tylko w funkcji* TO_CHAR**

format	znaczenie
"string"	wypisywany bez zmian
fm	poprzedza **Month** lub **Day**, powoduje, że nazwy miesięcy i dni będą zajmowały tylko tyle znaków, ile występuje w nazwie.**w przeciwnym razie** wszystkie nazwy będą wyrównywane do tej samej długości
TH	dodane do numeru roku, miesiąca, dnia, godziny, minuty lub sekundy powoduje wypisanie przyrostka 'th', kapitalizacja liter - jak w formacie liczby
SP	dodane do liczby powoduje wypisanie jej wartości słownie
SPTH	jak **SP** + **TH**
THSP	jak **TH** + **SP**

N/w znaki będą ignorowane przez **TO_DATE** i wypisywane bez zmian przez **TO_CHAR**:

/	,	-	:
.	**A.M.**	**P.M.**	**a.m.**
p.m.	**AM**	**am**	**PM**
pm	**CC (wiek)**	**SCC (wiek, - dla pne)**	**B.C.**
BC	**AD**	**A.D.**	**bc**
b.c.	**ad**	**a.d.**	

***Formaty używane do wypisywania liczb używane w* TO_CHAR, TO_NUMBER**

format	znaczenie
9	cyfra, ilość 9 określa szerokość pola
0	wyświetlaj wiodące zera
$	pozycja wyświetlania znaku $
.	pozycja kropki dziesiętnej
,	pozycja przecinka, oddzielającego trzycyfrowe grupy cyfr
MI	wypisuj minus **po prawej stronie** liczby ujemnej
PR	liczbe ujemne wyświetlaj w nawiasach
EEEE	wyświetlaj liczby w postaci wykładniczej
V	wyświetlaj liczbę pomnożoną przez 10^n, *n* jest ilością cyfr po **V**
B	wyświetlaj końcowe zera jako zera a nie spacje

V. Operacje na wartościach typu DATE (data i czas)

- Predefiniowany operator bieżącej daty i czasu systemowego: SYSDATE

```
select SYSDATE from dual;
```

- Wyświetlanie bieżącej daty i czasu systemowego zgodnych z zadanym formatem:

```
select to_char(SYSDATE,'DD.MM.YYYY') from dual;

select to_char(SYSDATE,'HH24:MI:SS') from dual;

select to_char(SYSDATE, 'DD.MM.YYYY  HH24:MI:SS') from dual;
```

- Zwiększanie oraz zmniejszanie wartości typu DATE

Przykładowo:

SYSDATE + x - zwiększenie bieżącej daty i czasu systemowego o x dni

SYSDATE - x - zmniejszenie bieżącej daty i czasu systemowego o x dni

Zatem zwiększenie bieżącej daty i czasu systemowego:

- o x godzin uzyskujemy poprzez SYSDATE + x/24

- o x minut uzyskujemy poprzez SYSDATE + x/(24*60)
  lub SYSDATE + x/1440

- o x sekund uzyskujemy poprzez SYSDATE + x/(24*60*60)
  lub SYSDATE + x/86400

Funkcje daty i czasu

Funkcja	Zwracana wartość
ADD_MONTHS(d,n)	data *d* plus *n* miesięcy
LAST_DAY(d)	data ostatniego dnia miesiąca do którego należy data *d*
MONTHS_BETWEEN(d,e)	ilość miesięcy o jaką data *e* poprzedza datę *d*
NEW_TIME(d,a,b)	jaka data w strefie *b* odpowiada dacie *d* w strefie *a*. *a* i *b* są znakowymi kodami stref czasowych
NEXT_DAY(d,char)	zwraca datę pierwszego dnia tygodnia o nazwie *char* która jest nie wcześniejsza niż data *d*
SYSDATE	bieżąca data i czas

VI. Operacje na łańcuchach tekstowych

VI. 1 Konkatenacja (*sklejanie*) łańcuchów tekstowych

-funkcja concat(tekst1,tekst2)

```
select concat('Studiuję ','fizykę informatyczną') from dual;
```

-operator konkatenacji ||

```
select 'Studiuję ' || 'fizykę informatyczną' from dual;
```

VI. 2 Inne wybrane funkcje manipulacji na łańcuchach

Podstawowe funkcje manipulacji na łańcuchach

Funkcja	Zwracana wartość
CHR(n)	zwraca znak o podanym kodzie
CONCAT($char_1$,$char_2$)	konkatenuje $char_1$ i $char_2$.
INITCAP(char)	zmienia wszystkie pierwsze znaki w słowach na duże litery
LOWER(char)	zmienia wszystkie litery w *char* na małe
UPPER(char)	zmień wszystkie litery w *char* na duże litery
NLS_INITCAP(char [, nls_sort])	jak ***initcap*** ale pozwala zdefiniować '*narodowe´* zasady kapitalizacji wyrazów ('*nls_sort*')
NLS_LOWER(char [, nls_sort])	jak ***lower*** ale z możliwością zdefiniowania zasad '*narodowych*' zmiany liter na małe ('*nls_sort*')
NLS_UPPER(char [, nls_sort])	jak ***upper*** ale z możliwością zdefiniowania zasad '*narodowych*' zmiany liter na duże ('*nls_sort*')
LPAD($char_1$,n[,$char_2$])	dostawia z lewej strony łańcucha $char_1$ tyle znaków $char_2$, aby uzupełnić łańcuch $char_1$ o *n* znaków. Domyślnie $char_2$ jest spacją.
RPAD($char_1$,n[,$char_2$])	dostawia z prawej strony łańcucha $char_1$ tyle znaków $char_2$, aby uzupełnić łańcuch $char_1$ o *n* znaków. Domyślnie $char_2$ jest spacją.
LTRIM(char[,set])	usuwa z początku łańcucha *char* wszystkie znaki zgodne ze znakami z *set*. Domyślnie usuwa spacje.
RTRIM(char[,set])	usuwa z końca łańcucha *char* wszystkie znaki zgodne ze znakami z *set*. Domyślnie usuwa spacje.

REPLACE(char,search_string [,replacement_string])	
	w łańcuchu *char* zamień każde wystąpienie *search_string* na *replacement_string*. Jeżeli *replacement_string* nie występuje - usuń każde wystąpienie *search_string*.
TRANSLATE(char,from,to)	zamień w *char* wszystkie znaki występujące w łańcuhcu *from* na znali występujące w łańcuchu *to*
SUBSTR(char,m[,n])	zwraca podłańcuch z *char*, zaczynający się od znaku o numerze *m* mający *n* znaków. Jeżeli *n* nie wystąpi - podłańcuch zaczynający się od znaku *m* zawierający wszystkie pozostałe znaki (do końca łańcucha *char*).
SUBSTRB(char,m[,n])	jak ***substr*** ale numery *m* oraz *n* określają bajty, nie znaki (dla niektórych platform sprzętowych to nie jest to samo)

Funkcje wspomagające operacje na łańcuchach

Funkcja	Zwracana wartość
ASCII(char)	zwraca kod ASCII znaku *char*. W systemach w których znaki mają kody wielobajtowe zwraca kod pierwszego bajtu.
INSTR($char_1$,$char_2$[,n[,m]])	zwraca pozycję m_{tego} (domyślnie: pierwszego) wystąpienia znaku $char_2$ w łańcuchu $char_1$. Zaczyna wyszukiwanie od n_{tego} (domyślnie: od pierwszego) znaku.
INSTRB(char1,char2[,n[,m]])	jak ***instr*** ale zwraca pozycję *bajtu* a nie znaku
LENGTH(char)	zwraca długość łańcucha *char* liczoną w znakach
LENGTHB(char)	zwraca długość łańcucha *char* liczoną w bajtach

Przykładowe zadanie:
Wyświetl nazwiska pracowników oraz ich odpowiedniki, w których pierwsze wystąpienie litery 'K' zamieniono na 'Z':

```
select ID_P, NAZWISKO,
translate(substr(NAZWISKO,1,instr(NAZWISKO,'K')), 'K', 'Z' ) ||
substr(NAZWISKO, instr(NAZWISKO,'K')+1 ) "Nazwisko po zamianie"
from PRACOWNICY;
```

Obecnie to samo można uzyskać za pomocą funkcji regexp_replace:

```
select ID_P, NAZWISKO,
      regexp_replace(NAZWISKO,'K','Z',2) ”Nazwisko po zmianie”
from PRACOWNICY;
```

VII. Parametryzacja zapytań

Parametr w zapytaniu definiujemy używając znaku &

```
select ID_P, IMIĘ, NAZWISKO, ID_D from PRACOWNICY
where ID_D = &NR_DZIALU;
```

Podczas wykonywania tego zapytania serwer zażąda podania wartości parametru o nazwie NR_DZIALU.