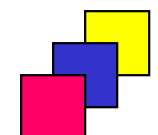


Język SQL. Rozdział 7.

Zaawansowane mechanizmy w zapytaniach

**Ograniczanie rozmiaru zbioru wynikowego,
klauzula WITH, zapytania hierarchiczne.**



Ograniczanie liczności zbioru wynikowego (1)

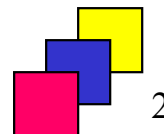
- Element standardu języka SQL.
- Odczytuje n rekordów od początku zbioru wynikowego (tzw. zapytanie „*top-n*”):

```
SELECT nazwisko, placa_pod FROM pracownicy  
ORDER BY placa_pod DESC FETCH FIRST 3 ROWS ONLY;
```

```
SELECT nazwisko, placa_pod FROM pracownicy  
ORDER BY placa_pod DESC FETCH FIRST 25 PERCENT ROWS ONLY;
```

- *FIRST* może zostać zastąpione przez *NEXT*, *ROWS* przez *ROW*,
- zwracanych jest dokładnie n rekordów: jeśli na pozycjach n , $n+1$, ... są rekordy o tej samej wartości kryterium porządkującego zbiór, rekordy na pozycji $n+1$, ... nie jest zwracany, chyba że:

```
SELECT nazwisko, placa_pod FROM pracownicy  
ORDER BY placa_pod DESC FETCH FIRST 3 ROWS WITH TIES;
```



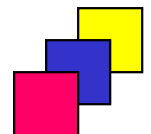
Ograniczanie liczności zbioru wynikowego (2)

- Odczytuje n rekordów od zadanej pozycji (przesunięcia) zbioru wynikowego:

```
SELECT nazwisko, placa_pod FROM pracownicy  
ORDER BY placa_pod DESC  
OFFSET 5 ROWS FETCH NEXT 5 ROWS ONLY;
```

```
SELECT nazwisko, placa_pod FROM pracownicy  
ORDER BY placa_pod DESC  
OFFSET 5 ROWS FETCH NEXT 10 PERCENT ROWS WITH TIES;
```

- przesunięcie (klauzula *OFFSET*) musi być wyrażone liczbą rekordów.
- zaimplementowane w:
 - Oracle12c,
 - Microsoft SQL Server (klauzula *SELECT TOP n*),
 - DB2,
 - PostgreSQL,
 - MySQL (klauzula *LIMIT*).



Ograniczanie liczności zbioru wynikowego (3)

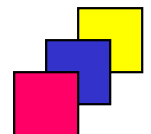
- Rozwiązanie charakterystyczne dla Oracle – wykorzystanie pseudokolumny *ROWNUM*.
- *ROWNUM* – numer rekordu w zbiorze wynikowym, przyznawany w momencie odczytu rekordu z dysku (ale przed sortowaniem!):

```
SELECT ROWNUM, nazwisko, etat, placa_pod FROM pracownicy;
```

```
SELECT ROWNUM, nazwisko, etat, placa_pod FROM pracownicy  
ORDER BY placa_pod DESC;
```

- Wykorzystanie w zapytaniu „*top-n*”:

```
SELECT ROWNUM AS pozycja, T.nazwisko, T.etat, T.pensja  
FROM (SELECT nazwisko, etat, placa_pod AS pensja  
FROM pracownicy  
ORDER BY pensja DESC) T  
WHERE ROWNUM <= 3;
```

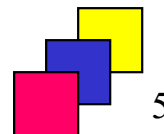


Adres rekordu

- Określa dokładnie lokalizację rekordu.
- Składa się z:
 - numeru obiektu (tabeli lub perspektywy), do której należy rekord,
 - numeru pliku,
 - numeru bloku dyskowego wewnątrz pliku,
 - numeru krotki wewnątrz bloku.
- Dostępny za pomocą pseudokolumny *ROWID*:

```
SELECT ROWID, id_zesp FROM pracownicy WHERE nazwisko = 'HAPKE';
```

- Zastosowania:
 - jednoznaczna identyfikacja rekordu w całej bazie danych,
 - jako referencja do rekordu w poleceniu SQL.

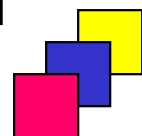


Klauzula WITH (1)

- Upraszcza konstruowanie skomplikowanych zapytań przez możliwość definicji tymczasowego zbioru (-ów) rekordów:

```
WITH  
prac_zesp AS (SELECT nazwa, nazwisko, etat,  
                placa_pod + NVL(placa_dod,0) AS placa  
                FROM pracownicy JOIN zespoly USING (id_zesp))  
SELECT * FROM prac_zesp  
WHERE placa > 1200;
```

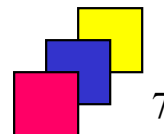
```
WITH  
prac_zesp(nazwa, nazwisko, etat, placa)  
    AS (SELECT nazwa, nazwisko, etat,  
            placa_pod + NVL(placa_dod,0)  
            FROM pracownicy JOIN zespoly USING (id_zesp))  
SELECT * FROM prac_zesp  
WHERE placa > 1200;
```



Klauzula WITH (2)

- **Możliwa definicja wielu zbiorów:**

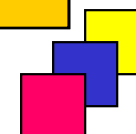
```
WITH
  profesorowie AS (SELECT * FROM pracownicy
                   WHERE etat = 'PROFESOR'),
  asystenci AS (SELECT * FROM pracownicy
               WHERE etat = 'ASYSTENT')
SELECT *
FROM profesorowie pr
WHERE EXISTS
  (SELECT *
   FROM asystenci
   WHERE id_szefa = pr.id_prac);
```



Klauzula WITH – zastosowania (1)


```
SELECT nazwa, SUM(placa_pod) AS suma_plac
FROM pracownicy p JOIN zespoly USING (id_zesp)
GROUP BY nazwa
HAVING 2 >=
    (SELECT COUNT(SUM(placa_pod))
     FROM pracownicy JOIN zespoly USING (id_zesp)
     GROUP BY nazwa HAVING SUM(placa_pod) > SUM(p.placa_pod))
ORDER BY suma_plac DESC;
```

```
WITH
    zespoly_stat AS
    (SELECT nazwa, SUM(placa_pod) AS suma_plac
     FROM pracownicy JOIN zespoly USING (id_zesp)
     GROUP by nazwa)
SELECT * FROM zespoly_stat s
WHERE 2 >=
    (SELECT COUNT(*)
     FROM zespoly_stat
     WHERE suma_plac > s.suma_plac)
ORDER BY suma_plac DESC;
```



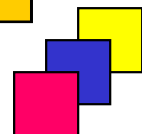
Klauzula WITH – zastosowania (2)

```
SELECT nazwisko, nazwa  
FROM pracownicy JOIN zespoly USING (id_zesp)  
WHERE 3 <= (SELECT COUNT(*)  
            FROM pracownicy  
            WHERE id_zesp = ???);
```



```
WITH  
  zespoly_prac AS  
  (SELECT nazwisko, nazwa, id_zesp  
   FROM pracownicy JOIN zespoly USING (id_zesp))  
SELECT nazwisko, nazwa  
FROM zespoly_prac zp  
WHERE 3 <= (SELECT COUNT(*) FROM pracownicy  
            WHERE id_zesp = zp.id_zesp);
```

```
SELECT nazwisko, nazwa FROM  
  (SELECT nazwa, nazwisko, id_zesp  
   FROM pracownicy JOIN zespoly USING(id_zesp)) zp  
WHERE 3 <= (SELECT COUNT(*)  
            FROM pracownicy WHERE id_zesp = zp.id_zesp);
```



Rekurencja

- **Relacja ZNA:**
- **Problem: znajdź możliwości kontaktu przy wykorzystaniu relacji "osoba zna osobę".**
- **Reguły:**
 - `kontakt (X, Y) <= zna (X, Y)`
 - `kontakt (X, Y) <= zna (X, Z) i kontakt (Z, Y)`

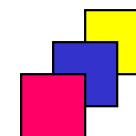
KTO	KOGO
Jaś	Małgosia
Małgosia	Piotrek
Jaś	Adaś
Adaś	Ola

```
WITH kontakt(kto, kogo) AS
(SELECT kto, kogo
FROM zna
UNION ALL
SELECT z.kto, k.kogo
FROM zna z JOIN kontakt k ON z.kogo = k.kto)
SELECT kto, kogo FROM kontakt;
```

KTO	KOGO
Jaś	Małgosia
Małgosia	Piotrek
Jaś	Adaś
Adaś	Ola
Jaś	Piotrek
Jaś	Ola

Zapytania hierarchiczne

- Zapytania hierarchiczne pozwalają na rekurencję w relacjach zawierających dane hierarchiczne (np. szef – podwładny – podwładny podwładnego – ...).
- Konieczna identyfikacja korzenia hierarchii (może być wiele korzeni – wiele hierarchii) oraz warunku wyszukującego rekordy-dzieci danego rekordu-rodzica.
- Realizacja w standardzie SQL – rekurencyjna klauzula *WITH*.
- Realizacja w Oracle:
 - od Oracle11gR2 – rekurencyjna klauzula *WITH*,
 - przed Oracle11gR2 – klauzule *CONNECT BY* i *START WITH*.



Zapytania hierarchiczne w standardzie SQL

- **Rekurencyjna klauzula *WITH*:**

```
WITH
  podwladni (id_prac, id_szefa, nazwisko, poziom) AS
  -- definicja korzenia hierarchii
  (SELECT id_prac, id_szefa, nazwisko, 1
   FROM pracownicy
   WHERE id_prac = 100
   UNION ALL
   -- rekurencyjna definicja niższych poziomów
   SELECT p.id_prac, p.id_szefa, p.nazwisko, poziom+1
   FROM podwladni s JOIN pracownicy p ON s.id_prac = p.id_szefa)
  -- wskazanie sposobu przeszukiwania hierarchii i sortowania rekordów-dzieci
  SEARCH DEPTH FIRST BY nazwisko SET porzadek_potomkow
  SELECT id_prac, id_szefa, nazwisko, poziom
  FROM podwladni
  ORDER BY porzadek_potomkow;
```

- **musi wykonywać operację zbiorową *UNION ALL*,**
- **musi mieć listę aliasów kolumn.**

Zapytania hierarchiczne w Oracle

- Pseudokolumna *LEVEL* reprezentuje poziom rekurencji w drzewie hierarchii.
- Operator *PRIOR* służy do odwoływania się do rodzica danego węzła.
- Klauzula *START WITH* definiuje korzeń drzewa.
- Klauzula *ORDER SIBLINGS BY* określa sposób uporządkowania rekordów-dzieci.

```
SELECT id_prac, id_szefa, nazwisko, LEVEL AS poziom  
FROM pracownicy  
CONNECT BY id_szefa = PRIOR id_prac  
START WITH nazwisko = 'WEGLARZ'  
ORDER SIBLINGS BY nazwisko;
```

```
SELECT id_prac, id_szefa, nazwisko, LEVEL AS poziom  
FROM pracownicy  
CONNECT BY id_szefa = PRIOR id_prac  
START WITH etat = 'PROFESOR';
```



Hierarchia pracowników

