Wykład 7

Implementacja języka SQL w systemach baz danych Oracle – sortowanie, funkcje agregujące i podzapytania.

Przykładowa RBD o schematach relacji (tzw. płaska postać RBD):

N(PRACOWNICY) = {**ID_P**, IMIĘ, NAZWISKO, PENSJA, ID_BP, **ID_D**}
N(DZIAŁY) = {**ID_D**, NAZWA}

ID_P - identyfikator pracownika,
ID_BP - identyfikator bezpośredniego przełożonego,
ID_D - identyfikator działu, w którym pracownik jest zatrudniony

PRACOWNICY

ID_P	IMIĘ	NAZWISKO	PENSJA	ID_BP	ID_D
1001	Roman	Kowalski	2000	1002	10
1002	Jan	Kowalski	3000		10
1003	Piotr	Nowak	3000	1005	11
1004	Piotr	Sosna	4000	1005	11
1005	Anna	Koral	5000		11
1006					12

DZIAŁY

ID_D	NAZWA
10	Administracja
11	Produkcja
12	Zarząd

I. Sortowanie wyniku zapytania

W klauzuli ORDER BY można podać atrybuty (wyrażenia), względem wartości których ma zostać posortowany wynik zapytania:

- sortowanie po identyfikatorze działu:

 - rosnące dla liczb lub zgodne z kolejnością liter alfabetu dla łańcuchów:

```
select ID_P, IMIĘ, NAZWISKO, ID_D from pracownicy
order by ID_D;
```

To samo daje zastosowanie parametru sortowania ASC (ascending)

```
select ID_P, IMIĘ, NAZWISKO, ID_D from pracownicy
order by ID_D asc;
```

- malejące dla liczb lub przeciwnie niż kolejność liter w alfabecie dla łańcuchów:

Poprzez zastosowanie parametru sortowania DESC (descending)

```
select ID_P, IMIĘ, NAZWISKO, ID_D from pracownicy
order by ID_D desc;
```

Kolejność atrybutów lub wyrażeń w klauzuli ORDER BY decyduje o hierarchii sortowania:

```
select ID_P, IMIĘ, NAZWISKO, ID_D from pracownicy
order by ID_D, NAZWISKO, IMIĘ;
```

Wynik powyższego zapytania zostanie posortowany najpierw względem ID_D, później w ramach działu wg NAZWISK, a następnie gdy nazwisko będzie się powtarzać to wg IMION.

Parametry sortowania nadaje się oddzielnie każdemu atrybutowi, względem którego sortujemy!

Zatem wynik powyższego zapytania będzie sortowany rosnąco lub zgodnie z kolejnością liter w alfabecie względem każdego atrybutu sortowania.

Możemy to zmienić, np.

```
select ID_P, IMIĘ, NAZWISKO, ID_D from pracownicy
order by ID_D desc, NAZWISKO, IMIĘ desc;
```

co jest równoważne zapytaniu:

```
select ID_P, IMIĘ, NAZWISKO, ID_D from pracownicy
order by ID_D desc, NAZWISKO asc, IMIĘ desc;
```

II. Funkcje agregujące (grupujące)

COUNT - zliczanie (niepustych) wierszy
SUM - suma
AVG - średnia (niepustych wartości)
MIN - znajduje minimalną wartość
MAX - znajduje maksymalną wartość

II.1. Proste zastosowania funkcji agregujących

- COUNT

```
select COUNT(*) from pracownicy; => 6

select COUNT(1) from pracownicy; => 6

select COUNT(ID_P) from pracownicy; => 6

select COUNT(ID_BP) from pracownicy; => 3
```

- SUM

```
select SUM(PENSJA) from pracownicy; => 17000
```

- AVG

```
select AVG(PENSJA) from pracownicy; => 3400
```

- równoważny wynik daje:

select SUM(PENSJA)/COUNT(PENSJA) from pracownicy; => 3400

- nie daje równoważnego wyniku, ale traktuje puste (NULL-owe) wartości jakby były równe 0, co bywa pożądane:

select SUM(PENSJA)/COUNT(ID_P) from pracownicy; => 2833.33

- MIN

select MIN(PENSJA) from pracownicy; => 2000

- MAX

select MAX(PENSJA) from pracownicy; => 5000

II.2. Zastosowanie klauzuli GROUP BY - znajdowanie wartości funkcji agregującej dla grupy zdefiniowanej w klauzuli GROUP BY

- Znajdowanie w średniej pensji w działach

```
select AVG(PENSJA) from pracownicy
group by ID_D;                    =>    2500
                                        4000
```

Czy zastosowanie klauzuli GROUP BY jest konieczne, gdy chcemy znaleźć minimalną pensję w jednym wskazanym dziale?
- równoważne zapytania:

```
select AVG(PENSJA) from PRACOWNICY
where ID_D=10;                    => 2500

select AVG(PENSJA) from PRACOWNICY
where ID_D=10
group by ID_D;                    => 2500
```

jednak tak jest tylko w przypadku jednego działu

- dla więcej niż jednego działu to już nie są równoważne zapytania:

```
select AVG(PENSJA) from PRACOWNICY
where ID_D=10 or ID_D=11;                 => 3400

select AVG(PENSJA) from PRACOWNICY
where ID_D=10 or ID_D=11
group by ID_D;                            =>   2500
                                               4000
```

II. 3. Ważna reguła dotycząca zastosowania klauzuli GROUP BY
- atrybuty czy wyrażenia pojawiające się w klauzuli SELECT oprócz wyrażeń z funkcjami agregującymi muszą pojawić się w klauzuli GROUP BY

- przykładowo znajdźmy minimalne pensje w działach, wyświetlając także odpowiedni identyfikator działu:

```
select ID_D, MIN(PENSJA) from pracownicy
group by ID_D;
```

Błędne zapytanie ze względu na klauzulę GROUP BY:

```
select ID_D, MIN(PENSJA) from pracownicy;
```

- inny, trudniejszy przykład: znajdźmy minimalne pensje w działach, wyświetlając także odpowiedni identyfikator działu i nazwę działu:

```
select P.ID_D, NAZWA, MIN(PENSJA)
from pracownicy P, działy D
where D.ID_D=P.ID_D
group by P.ID_D,NAZWA;
```

Błędne zapytania ze względu na klauzulę GROUP BY:

```
select P.ID_D, NAZWA, MIN(PENSJA)
from pracownicy P, działy D
where D.ID_D=P.ID_D;

select P.ID_D, NAZWA, MIN(PENSJA)
from pracownicy P, działy D
where D.ID_D=P.ID_D
group by NAZWA;
```

```
select P.ID_D, NAZWA, MIN(PENSJA)
from pracownicy P, działy D
where D.ID_D=P.ID_D
group by P.ID_D;

select P.ID_D, NAZWA, MIN(PENSJA)
from pracownicy P, działy D
where D.ID_D=P.ID_D
group by D.ID_D,NAZWA;
```

- Przykład poprawnego składniowo, ale dysfunkcjonalnego zapytania z klauzulą GROUP BY:

```
select ID_P, AVG(PENSJA) from pracownicy
group by ID_P;
```

Powyższe zapytanie daje ten sam wynik, jakbyśmy nie używali funkcji agregującej, tzn.:

```
select ID_P, PENSJA from pracownicy;
```

II. 4. Zastosowanie (pod)klauzuli HAVING

Głównie jest stosowana dla podania warunków, jakie mają spełniać wartości funkcji agregujących. Zwykle wymaga klauzuli GROUP BY.

- przykład typowego zastosowania klauzuli HAVING:

```
select ID_D, AVG(PENSJA) from pracownicy
group by ID_D
having AVG(PENSJA)>=3000;
```

Błędne zastosowanie klauzuli HAVING:

```
select ID_D, AVG(PENSJA) from pracownicy
group by ID_D
having PENSJA>=3000;
```

- dozwolone, ale najczęściej niezbyt funkcjonalne, jest zastosowanie klauzuli HAVING bez klauzuli GROUP BY; jest to możliwe tylko wtedy, gdy nie jest wymagana klauzula GROUP BY dla funkcji agregującej:

```
select AVG(PENSJA) from pracownicy
having AVG(PENSJA)>=3000;

select AVG(PENSJA) from pracownicy
having MIN(PENSJA)>=3000;

select AVG(PENSJA) from pracownicy
having COUNT(PENSJA)<4;
```

- wyjątek pozwalający stosować klauzule HAVING zamiast klauzuli WHERE do nakładania warunków na wartości atrybutów, względem których grupujemy, tzn. które znajdują się w klauzuli GROUP BY:

```
select ID_D, AVG(PENSJA) from pracownicy
having ID_D=10
group by ID_D;

zamiast

select ID_D, AVG(PENSJA) from pracownicy
where ID_D=10
group by ID_D;

-----------------------------------------------------

select ID_D, AVG(PENSJA) from pracownicy
having ID_D=10 or ID_D=12
group by ID_D;

zamiast

select ID_D, AVG(PENSJA) from pracownicy
where ID_D=10 or ID_D=12
group by ID_D;
```

III. Wymagana kolejność klauzul zapytań SQL

SELECT
FROM
WHERE - opcjonalna
GROUP BY - opcjonalna
HAVING - opcjonalna
ORDER BY - opcjonalna

Tylko kolejność klauzul GROUP BY i HAVING można zmienić !!!

IV. Podzapytania

Podzapytania to zapytania konstruowane w klauzulach WHERE lub HAVING. Oracle umożliwia także wstawianie podzapytań zamiast wyrażeń lub atrybutów w klauzuli SELECT.

IV. 1. Typowe podzapytanie powinno zwracać jedną wartość (ogólnie jedną krotkę):

- Znajdźmy maksymalną pensję w firmie oraz identyfikator, imię i nazwisko pracowników, którzy ją otrzymują:

```
select ID_P, IMIĘ, NAZWISKO, PENSJA from pracownicy
where PENSJA = (select MAX(PENSJA) from pracownicy);
```

- Znajdź maksymalne pensje oraz identyfikatory działów, w których średnia pensja jest większa niż minimalna pensja w dziale 11:

```
select ID_D, MAX(PENSJA) from pracownicy
group by ID_D
having AVG(PENSJA) > (select MIN(PENSJA)
                      from pracownicy where ID_D = 11);
```

IV. 2. Konstrukcja dozwalająca na zwracanie przez podzapytanie więcej niż jednej wartości (ogólnie jednej krotki).

- Operator IN odpowiadający wieloskładnikowej alternatywie:

Zapytanie:

```
select ID_P, IMIĘ, NAZWISKO, ID_D from pracownicy
where ID_D in (10,11,14);
```

jest równoważne zapytaniu:

```
select ID_P, IMIĘ, NAZWISKO, ID_D from pracownicy
where ID_D = 10 or ID_D = 11 or ID_D = 14;
```

- Zastosowanie operatora IN z podzapytaniami:

  - Znajdź pracowników, których pensje są równe minimalnej pensji w dowolnym dziale:

```
select ID_P from pracownicy
where PENSJA in (select MIN(PENSJA) from pracownicy
                 group by ID_D);
```

IV. 3. Złącznie relacji z podzapytania z relacją z zapytania nadrzędnego (głównym). Jest to często używana konstrukcja, najczęściej wymagająca odpowiednich aliasów.

- Znajdź indentyfikator, imię, nazwisko i imię pracowników, których pensja jest mniejsza od średniej pensji w ich dziale:

```
select ID_P, IMIĘ, NAZWISKO, PENSJA from pracownicy P1
where PENSJA < (select AVG(PENSJA) from pracownicy P2
                where  P2.ID_D=P1.ID_D);
```

Ten sam, poprawny wynik daje zapytanie:

```
select ID_P, IMIĘ, NAZWISKO, PENSJA from pracownicy P1
where PENSJA < (select AVG(PENSJA) from pracownicy
                where  ID_D=P1.ID_D);
```

Natomiast zrezygnowanie z aliasu w zapytaniu nadrzędnym:

```
select ID_P, IMIĘ, NAZWISKO, PENSJA from pracownicy
where PENSJA < (select AVG(PENSJA) from pracownicy P2
                where  P2.ID_D=ID_D);
```

jest przyjmowane przez serwer, ale daje błędny wynik, ponieważ nieobdarzone aliasem ID_P w klauzuli WHERE podzapytania jest traktowane jako pochodzące z tabeli pracownicy, do której odwołuje się podzapytanie.

- Znajdź pracowników, których pensje są równe minimalnej pensji w ich dziale:

  - Rozwiązanie przez *podzapytanie złączone z podzapytaniem nadrzędnym*

```
select ID_P, IMIĘ, NAZWISKO, PENSJA from pracownicy P1
where PENSJA = (select MIN(PENSJA) from pracownicy P2
                where P2.ID_D = P1.ID_D);
```

  - Rozwiązanie przez operator IN zastosowany z podzapytaniem, które wówczas nie wymaga *złączenia z zapytaniem nadrzędnym*

```
select ID_P, IMIĘ, NAZWISKO, PENSJA from pracownicy
where (ID_D, PENSJA) in (select ID_D, MIN(PENSJA)
                         from pracownicy group by ID_D);
```

IV. 4. Zastosowanie optymalizującej kombinacji operatorów ALL i DISTINCT oraz ANY i DISTINCT zamiast funkcji agregujących MAX oraz MIN w bazach Oracle

- Jak działa operator DISTINCT - dotyczy całych krotek i tylko raz wyświetla powtarzające się krotki:

```
select PENSJA from pracownicy;          => 6 wierszy wyświetli

select DISTINCT PENSJA from pracownicy; => 5 wierszy wyświetli

select ID_P, PENSJA from pracownicy;    => 6 wierszy wyświetli

select DISTINCT ID_P, PENSJA from pracownicy;
                                        => także wyświetli 6 wierszy
```

- Problem do rozwiązania: znaleźć pracowników, którzy zarabiają WIĘCEJ niż wynosi NAJWIĘKSZA (NAJMNIEJSZA) pensja w dziale o identyfikatorze 10:

WIĘCEJ niż NAJWIĘKSZA pensja:

Funkcja MAX

```
select ID_P, IMIĘ, NAZWISKO from pracownicy
where PENSJA > (select MAX(PENSJA) from pracownicy
                    where ID_D = 10);
```

Operatory ALL i DISTINCT:

```
select ID_P, IMIĘ, NAZWISKO from pracownicy
where PENSJA > ALL(select DISTINCT PENSJA from pracownicy
                        where ID_D = 10);
```

WIĘCEJ niż NAJMNIEJSZA pensja:

Funkcja MAX

```
select ID_P, IMIĘ, NAZWISKO from pracownicy
where PENSJA > (select MIN(PENSJA) from pracownicy
                    where ID_D = 10);
```

Operatory ALL i DISTINCT:

```
select ID_P, IMIĘ, NAZWISKO from pracownicy
where PENSJA > ANY(select DISTINCT PENSJA from pracownicy
                        where ID_D = 10);
```

IV. 5. Zastosowanie podzapytania w klauzuli SELECT

Wówczas również najczęściej stosujemy złączenie relacji z takiego podzapytania z relacją występującą w klauzuli FROM głównego zapytania

Na przykład, gdy chcemy wyświetlić identyfikator pracownika, jego pensję, identyfikator działu, w którym pracuje oraz średnią pensję w tym dziale możemy zadać zapytanie:

```
select ID_P, PENSJA, ID_D,
(select AVG(P2.PENSJA) from pracownicy P2 where
P2.ID_D=P1.ID_D) srednia_pensja_w_dziale from pracownicy P1;
```