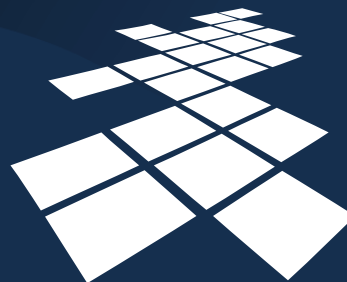


## Ćwiczenie 3 – funkcje agregujące

Funkcje agregujące,  
klauzule GROUP BY,  
HAVING



UCZELNIA  
ONLINE

Ćwiczenie 3 – funkcje agregujące

Celem ćwiczenia jest zaprezentowanie zagadnień dotyczących stosowania w zapytaniach języka SQL predefiniowanych funkcji agregujących.

Wymagania:

Umiejętność konstrukcji prostych zapytań w języku SQL, użycie funkcji wierszowych w zapytaniach.



## Plan ćwiczenia

- Charakterystyka funkcji agregujących.
- Przegląd dostępnych funkcji agregujących, sposób użycia.
- Konstruowanie zapytań z jedną grupą.
- Konstruowanie zapytań z wieloma grupami.
- Podział grup na podgrupy.
- Filtrowanie grup.
- Najczęstsze błędy przy konstrukcji zapytań z funkcjami agregującymi.
- Przegląd zaawansowanych konstrukcji zapytań z funkcjami agregującymi.

Ćwiczenie 3 – funkcje agregujące (2)

Na początku ćwiczenia zostanie omówiona charakterystyka funkcji agregujących, przegląd dostępnych funkcji agregujących i sposób ich użycia w zapytaniach. Następnie przyjrzymy się konstrukcji zapytań z jedną grupą, wykorzystujących funkcje agregujące. Kolejne zagadnienie to zapytania z wieloma grupami i podział grup na podgrupy. Dalej omówimy filtrowanie grup w zapytaniach oraz scharakteryzujemy najczęściej popełniane błędy. Ćwiczenie zakończymy przeglądem zaawansowanych konstrukcji zapytań z funkcjami agregującymi.



## Funkcje agregujące

- Działają na zbiorach rekordów, nazywanych grupami.
- Rekordy należą do tej samej grupy jeśli posiadają tę samą wartość wyrażenia grupującego.
- Funkcja agregująca dla każdej grupy wylicza pojedynczą wartość na podstawie wyrażenia, będącego jej parametrem.

Ćwiczenie 3 – funkcje agregujące (3)

Funkcje agregujące działają na zbiorach rekordów, nazywanych grupami (w przeciwieństwie do funkcji wierszowych, które zawsze działają na jednym rekordzie). Przed zastosowaniem funkcji agregującej konieczne jest podzielenie rekordów na grupy, tzw. grupowanie. Do jednej grupy należą te rekordy relacji, dla których tzw. wyrażenie grupujące zwraca tę samą wartość. Wyrażeniem grupującym jest najczęściej pojedynczy atrybut relacji. Po podziale rekordów na grupy w każdej z grup zostaje zastosowana funkcja agregująca, która wylicza pojedynczą wartość dla grupy. Stąd w wyniku zapytania otrzymujemy tyle rekordów, ile grup zostało utworzonych w wyniku operacji grupowania.



## Funkcje agregujące

- Problem: znajdź średnią płacę pracowników dla każdej grupy etatowej.

ETAT	PLACA_POD	wyrażenie grupujące: etat	
ADIUNKT	2610,2	grupa dla etat = 'ADIUNKT'	ADIUNKT
ADIUNKT	2845,5		
ASYSTENT	1839,7	grupa dla etat = 'ASYSTENT'	ASYSTENT
ASYSTENT	1850		
ASYSTENT	1889		
ASYSTENT	1971		
PROFESOR	3070	grupa dla etat = 'PROFESOR'	PROFESOR
PROFESOR	3230		
PROFESOR	3350		
PROFESOR	3960		

ETAT	SREDNIA
ADIUNKT	2727,85
ASYSTENT	1887,425
PROFESOR	3402,5

Ćwiczenie 3 – funkcje agregujące (4)

Na bieżącym slajdzie zaprezentowano przykładowe zapytanie z grupowaniem i funkcją agregującą: „znajdź średnią płacę pracowników dla każdej grupy etatowej”. Wyrażeniem grupującym, które dostarcza wartości dzielące zbiór rekordów relacji PRACOWNICY na grupy, jest atrybut ETAT. Przykładowy zbiór rekordów został podzielony na trzy grupy: pierwszą dla wartości ADIUNKT, znajdują się w niej dwa rekordy, następną dla wartości ASYSTENT, należą do niej cztery rekordy, wreszcie ostatnią dla wartości PROFESOR, również z czterema rekordami. Następnie w każdej z grup wartości atrybutu PLACA\_POD zostają poddane działaniu funkcji agregującej, wyliczającej średnią. W wyniku zapytania otrzymujemy po jednym rekordzie dla każdej grupy etatowej: rekord zawiera wartość wyrażenia grupującego, a więc atrybutu ETAT i wyliczoną średnią płacą pracowników w tej grupie.



## Rodzaje funkcji agregujących

- Funkcje:
  - **AVG** – średnia,
  - **MIN** – minimum,
  - **COUNT** – liczba wystąpień,
  - **SUM** – suma.
  - **MAX** – maksimum,
- Składnia: **nazwa\_funkcji**(all | distinct *wyrażenie*)
- Szczególny przypadek – funkcja **COUNT**:
  - **COUNT**(\*) – liczba rekordów,
  - **COUNT**(all | distinct *wyrażenie*) – liczba niepustych wartości wyrażenia.

Ćwiczenie 3 – funkcje agregujące (5)

Omówimy teraz poszczególne funkcje agregujące. Do wyliczenia średniej służy funkcja AVG. Funkcja COUNT pozwala na znalezienie liczby wystąpień. Funkcje MAX i MIN umożliwiają znalezienie, odpowiednio, wielkości maksymalnej i minimalnej w zbiorze. Funkcja SUM umożliwia wyliczenie sumy elementów.

Każda z funkcji posiada jeden parametr, będący wyrażeniem dostarczającym wartości do obliczeń. W przypadku funkcji AVG i SUM wyrażenie musi dostarczać wartości liczbowe, pozostałe funkcje agregujące przyjmują wartości dowolnego typu podstawowego. Przed wyrażeniem można umieścić słowo kluczowe DISTINCT, wówczas do obliczeń zostaną wzięte wartości wyrażenia po eliminacji powtórzeń. Umieszczenie w wywołaniu funkcji samego wyrażenia lub poprzedzenie wyrażenia słowem ALL powoduje, że do obliczeń będą brane wszystkie wartości wyrażenia.

Dodatkowego komentarza wymaga użycie funkcji COUNT. Funkcja zwróci liczbę niepustych wartości wyrażenia w grupie (wartości różnych od null). Dodanie słowa DISTINCT spowoduje policzenie różnych niepustych wystąpień wartości wyrażenia w grupie. Z kolei jeśli w wywołaniu funkcji wyrażenie zastąpimy gwiazdką (symbol \*), wówczas zostanie policzona liczba rekordów należących do grupy.

Pozostałe funkcje agregujące również pomijają przy obliczeniach wartości puste, stąd nie ma konieczności stosowania mechanizmów eliminujących wartości puste.



## Zapytania z jedną grupą

- Wszystkie rekordy, przetwarzane przez zapytanie, tworzą jedną grupę.
- Wynikiem zapytania jest jeden rekord.
- Przykład: znajdź minimalną i maksymalną wartość płacy podstawowej pracowników zespołu o identyfikatorze 20.

```
SELECT MIN(placa_pod), MAX(placa_pod)
FROM pracownicy WHERE id_zesp = 20;
```

MIN(PLACA_POD)	MAX(PLACA_POD)
1590	3960

Ćwiczenie 3 – funkcje agregujące (6)

Przejdziemy teraz do omawiania zasad konstruowania zapytań z funkcjami agregującymi. Rozpocznemy od zapytań z jedną grupą. Są to zapytania bez wyrażenia grupującego, stąd wszystkie rekordy, odczytane przez zapytanie, trafiają do tej samej jednej grupy, a wynikiem zapytania będzie zawsze jeden rekord z wartościami wyliczonymi przez umieszczone w klauzuli SELECT zapytania funkcje agregujące. Przykładowe zapytanie wylicza dwie wartości: minimalną pensję podstawową i maksymalną pensję podstawową wśród pracowników zespołu o numerze 20. Wynikiem zapytania jest jeden rekord z dwiema wartościami.



## Zapytania z wieloma grupami

- Klauzula **GROUP BY** *wyrażenie* – definiuje wyrażenie grupujące.
- Wynikiem zapytania jest jeden rekord dla każdej grupy.
- Przykład: znajdź średnią wartość płacy podstawowej wśród pracowników każdego zespołu, posortuj wynik wg identyfikatora zespołu.

<code>SELECT id_zesp, AVG(placa_pod)</code>	<code>ID_ZESP</code>	<code>AVG(PLACA_POD)</code>
<code>FROM pracownicy</code>	10	3670,1
<code>GROUP BY id_zesp</code>	20	2475,02857
<code>ORDER BY id_zesp;</code>	30	1623,33333
	40	3350
	null	1850

Ćwiczenie 3 – funkcje agregujące (7)

Konstruując zapytania z wieloma grupami konieczne jest zdefiniowanie wyrażenia grupujące. Wyrażenie grupujące umieszcza się w klauzuli **GROUP BY**. W przykładzie zbiór rekordów relacji **PRACOWNICY** zostaje podzielony na grupy ze względu na wartość atrybutu **ID\_ZESP**, następnie w każdej z grup zostaje wyliczona wartość średniej płacy pracowników, zatrudnionych w danym zespole. W wyniku zapytania otrzymujemy tyle rekordów, w ilu zespołach pracują pracownicy (rekord z pustą wartością **ID\_ZESP** jest tworzony przez rekordy pracowników, którzy nie należą do żadnego zespołu). Wynik zapytania zostaje posortowany ze względu na wartość wyrażenia grupującego, a więc atrybutu **ID\_ZESP**.



## Podział grup na podgrupy

- Dla każdego zespołu znajdź liczbę pracowników pracujących na poszczególnych etatach.

```
SELECT id_zesp, etat,
       COUNT(*)
FROM pracownicy
GROUP BY id_zesp, etat;
```

ID_ZESP	ETAT	COUNT(*)
10	ADIUNKT	1
10	DYREKTOR	1
20	ADIUNKT	1
20	ASYSTENT	3
20	PROFESOR	2
20	SEKRETARKA	1
30	DOKTORANT	2
30	PROFESOR	1
40	PROFESOR	1
	ASYSTENT	1

Ćwiczenie 3 – funkcje agregujące (8)

Istnieje możliwość podziału grup, odczytywanych przez zapytanie, na podgrupy. Realizuje się to umieszczając w klauzuli GROUP BY kilka wyrażeń grupujących. W przykładzie zbiór rekordów relacji PRACOWNICY zostaje podzielony na grupy ze względu na wartość atrybutu ID\_ZESP (pierwsze wyrażenie grupujące), następnie rekordy w każdej z grup zostają podzielone na podgrupy ze względu na wartość atrybutu ETAT (drugie wyrażenie grupujące). Funkcja agregująca zostaje wykonana w każdej z podgrup. Interpretacja wyniku przykładowego zapytania jest następująca: otrzymujemy dla każdego zespołu informację o liczbie pracowników tego zespołu zatrudnionych na poszczególnych etatach.





## Filtrowanie grup

- Klauzula **HAVING** *warunek\_logiczny* – umożliwia wybór grup, spełniających warunek logiczny.
- Warunek logiczny może być skonstruowany jedynie z funkcji agregujących i/lub wyrażeń grupujących.
- Przykład: podaj identyfikatory i średnie płace podstawowe w zespołach zatrudniających nie mniej niż trzech pracowników.

```
SELECT id_zesp, AVG(placa_pod)
FROM pracownicy
GROUP BY id_zesp
HAVING COUNT(*) >= 3;
```

ID_ZESP	AVG(PLACA_POD)
20	2475,02857
30	1623,33333

Ćwiczenie 3 – funkcje agregujące (9)

Zapytanie z grupowaniem można dodatkowo wyposażyć w mechanizm filtrowania grup. Realizuje się to umieszczając warunek logiczny w dodatkowej klauzuli HAVING. Należy pamiętać, że w warunku tym można użyć jedynie wyrażeń grupujących i/lub funkcji agregujących. Wartościowanie warunku następuje po utworzeniu grup. Grupy, dla których warunek nie jest spełniony, zostają odrzucone (nie pojawiają się w wyniku zapytania). W przykładzie zbiór rekordów relacji PRACOWNICY zostaje pogrupowany ze względu na wartość atrybutu ID\_ZESP, jednak odrzucone zostają te grupy, w których jest mniej niż trzy rekordy. W każdej z pozostałych grup zostaje wyliczona średnia płaca pracowników, należących do zespołów.



## Najczęściej popełniane błędy (1)

- Umieszczenie w klauzuli **SELECT** zapytania z jedną grupą wyrażenia nie będącego funkcją agregującą.

```
SELECT etat, SUM(placa_pod)
FROM pracownicy WHERE etat = 'PROFESOR';
```

- Umieszczenie w klauzuli **SELECT** zapytania z wieloma grupami wyrażenia nie będącego wyrażeniem grupującym lub funkcją agregującą.

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

Ćwiczenie 3 – funkcje agregujące (10)

Omówione teraz zostaną najczęściej popełniane błędy przy konstrukcji zapytań z funkcjami agregującymi.

Pierwszym błędem, często pojawiającym się przy zapytaniach z jedną grupą, jest umieszczenie w klauzuli **SELECT** wyrażenia nie będącego funkcją agregującą (w przykładzie umieszczono atrybut **ETAT**). W zapytaniach z jedną grupą w klauzuli **SELECT** mogą być umieszczone jedynie funkcje agregujące.

Kolejny błąd dotyczy zapytań z wieloma grupami i polega na umieszczeniu w klauzuli **SELECT** wyrażenia, nie będącego wyrażeniem grupującym (a więc nie występującym w klauzuli **GROUP BY**) lub funkcją agregującą. W przykładzie w klauzuli **SELECT** umieszczono atrybut **NAZWISKO**, tymczasem wyrażeniem grupującym jest atrybut **ID\_ZESP**.



## Najczęściej popełniane błędy (2)

- Umieszczenie funkcji agregującej w warunku w klauzuli **WHERE**.

```
SELECT id_zesp FROM pracownicy  
WHERE COUNT(*) > 3 GROUP BY id_zesp;
```

- Umieszczenie w warunku w klauzuli **HAVING** wyrażenia nie będącego funkcją agregującą lub wyrażeniem grupującym.

```
SELECT id_zesp, COUNT(distinct placa_dod)  
FROM pracownicy GROUP BY id_zesp  
HAVING etat = 'PROFESOR';
```

Ćwiczenie 3 – funkcje agregujące (11)

Następnym częstym błędem jest umieszczenie warunku, wykorzystującego funkcję agregującą, w klauzuli **WHERE**. Taki warunek zawsze powinien być umieszczony w klauzuli **HAVING**.

Kolejny błąd polega na konstrukcji warunku w klauzuli **HAVING** z wyrażeniem innym niż wyrażenie grupujące lub funkcja agregująca. Taki warunek powinien zostać umieszczony w klauzuli **WHERE**.



## Najczęściej popełniane błędy (3)

- Porządkowanie wyników zapytania z wieloma grupami według wartości wyrażenia nie będącego wyrażeniem grupującym lub funkcją agregującą.

```
SELECT id_zesp, COUNT(distinct placa_dod)
FROM pracownicy GROUP BY id_zesp
ORDER BY nazwisko;
```

Ćwiczenie 3 – funkcje agregujące (12)

Wreszcie ostatni błąd, polegający na użyciu do sortowania wyniku zapytania z grupowaniem wyrażenia nie będącego wyrażeniem grupującym bądź funkcją agregującą. W zaprezentowanym przykładzie wynik zapytania może zostać posortowany jedynie ze względu na wartość atrybutu ID\_ZESP (wyrażenie grupujące) lub wartość wyliczaną przez dowolną funkcję agregującą.



## Konstrukcje zaawansowane (1)

- Użycie funkcji agregującej jako parametru innej funkcji agregującej; przykład: znajdź maksymalną sumę płac pracowników w poszczególnych zespołach.

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

- Zapytanie z jedną grupą i klauzulą **HAVING**; przykład: podaj wartość średniej płacy pracowników, ale tylko wtedy, jeśli liczba pracowników jest większa od 12.

```
SELECT MAX(placa_pod) FROM pracownicy  
WHERE id_zesp in (10,20) HAVING COUNT(*) > 12;
```

Ćwiczenie 3 – funkcje agregujące (13)

Dokonyamy teraz przeglądu zaawansowanych konstrukcji wykorzystujących funkcje agregujące.

Pierwsza konstrukcja to użycie funkcji agregującej jako parametru innej funkcji agregującej. Przykładowe zapytanie należy wykonywać dwuetapowo. Pierwszy etap to wykonanie zapytania w postaci „SELECT SUM(placa\_pod) FROM pracownicy GROUP BY id\_zesp”. Wynik tego zapytania to zbiór sum płac podstawowych pracowników w poszczególnych zespołach. Drugi etap to wybór spośród wyliczonych wartości wielkości maksymalnej. W wyniku otrzymujemy jeden rekord (zapytanie wykonywane w drugim etapie jest w istocie zapytaniem działającym na jednej grupie).

Kolejny przykład prezentuje zapytanie z jedną grupą (a więc bez zdefiniowanego wyrażenia grupującego), w którym zastosowano klauzulę **HAVING**. W takim przypadku warunek filtrujący zostaje zastosowany do jedynej grupy zapytania, jeśli warunek nie jest spełniony, zapytanie zwraca wynik pusty. W przykładzie otrzymamy wynik, maksymalną płacę podstawową pracowników z zespołów 10 i 20, pod warunkiem, że w obu zespołach zatrudniono w sumie ponad 12 osób.



## Konstrukcje zaawansowane (2)

- Zapytanie z klauzulą **WHERE** i **HAVING**; przykład: dla każdego zespołu, w którym średnia płaca przekracza 1000, podaj liczbę zatrudnionych pracowników, pominiętych pracowników na etacie PROFESOR, wynik uporządkuj ze względu na sumę płac podstawowych w zespole.

```
SELECT id_zesp, COUNT (*)  
FROM pracownicy  
WHERE etat <>'PROFESOR'  
GROUP BY id_zesp  
HAVING AVG(placa_pod) > 1000  
ORDER BY SUM(placa_pod);
```

Ćwiczenie 3 – funkcje agregujące (14)

Wreszcie zapytanie, w którym użyto wszystkich zaprezentowanych dotąd klauzul. Należy pamiętać o kolejności wykonywania klauzul. Jako pierwszy zostaje przetworzony warunek w klauzuli **WHERE**, dokonujący filtrowania rekordów relacji **PRACOWNICY** ze względu na wartość atrybutu **ETAT**. Do dalszego przetwarzania zostaną wzięte tylko te rekordy, gdzie **ETAT** różni się od ciągu znaków „PROFESOR”. Następnie realizowane jest grupowanie, wyrażeniem grupującym, umieszczonym w klauzuli **GROUP BY**, jest atrybut **ID\_ZESP**. Powstałe grupy są filtrowane ze względu na warunek logiczny w klauzuli **HAVING**. Wreszcie wyliczana jest wartość funkcji agregującej, umieszczonej w klauzuli **SELECT** a wynik zostaje posortowany ze względu na wyrażenie umieszczone w klauzuli **ORDER BY**.



## Zadania

1. Wyświetl najniższą i najwyższą pensję oraz różnicę dzielącą najlepiej i najgorzej zarabiających pracowników.
2. Wyświetl średnie pensje dla wszystkich etatów. Wyniki uporządkuj wg malejącej średniej pensji.
3. Wyświetl liczbę zatrudnionych profesorów.
4. Znajdź sumaryczne miesięczne płace dla każdego zespołu. Nie zapomnij o płacach dodatkowych!
5. Wyświetl numery zespołów, które zatrudniają więcej niż dwóch pracowników. Pomiń pracowników bez przydziału do zespołów. Wyniki uporządkuj wg malejącej liczby pracowników.

Ćwiczenie 3 – funkcje agregujące (15)

Bieżący slajd rozpoczyna zbiór zadań, których celem jest utrwalenie wiadomości o konstrukcji zapytań wykorzystujących funkcje agregujące.



## Zadania

6. Wyświetl średnie pensje wypłacane w ramach poszczególnych etatów i liczbę pracowników zatrudnionych na danym etacie. Pomiń pracowników zatrudnionych po 1990 roku.
7. Dla każdego pracownika wyświetl pensję najgorzej zarabiającego podwładnego. Wyniki uporządkuj wg malejącej pensji.
8. Sprawdź, czy identyfikatory pracowników są unikalne.





- 1 **SELECT** MIN(placa\_pod) as minimum, MAX(placa\_pod) as maksimum, MAX(placa\_pod) – MIN(placa\_pod) as różnica **FROM** pracownicy;
- 2 **SELECT** etat, AVG(placa\_pod) as średnia **FROM** pracownicy **GROUP BY** etat **ORDER BY** AVG(placa\_pod) desc;
- 3 **SELECT** COUNT(\*) as profesorowie **FROM** pracownicy **WHERE** etat = 'PROFESOR';
- 4 **SELECT** id\_zesp, SUM(placa\_pod + nvl(placa\_dod, 0)) as suma\_plac **FROM** pracownicy **GROUP BY** id\_zesp;
- 5 **SELECT** id\_zesp, COUNT(\*) as ilu\_pracuje **FROM** pracownicy **WHERE** id\_zesp is not null **GROUP BY** id\_zesp **HAVING** COUNT(\*) > 2 **ORDER BY** COUNT(\*) desc;

Bieżący slajd przedstawia rozwiązania zadań (1), (2), (3), (4) i (5), których treść zacytowano poniżej.

- (1) Wyświetl najniższą i najwyższą pensję oraz różnicę dzielącą najlepiej i najgorzej zarabiających pracowników.
- (2) Wyświetl średnie pensje dla wszystkich etatów. Wyniki uporządkuj wg malejącej średniej pensji.
- (3) Wyświetl liczbę zatrudnionych profesorów.
- (4) Znajdź sumaryczne miesięczne płace dla każdego zespołu. Nie zapomnij o płacach dodatkowych!
- (5) Wyświetl numery zespołów, które zatrudniają więcej niż dwóch pracowników. Pomiń pracowników bez przydziału do zespołów. Wyniki uporządkuj wg malejącej liczby pracowników.



```
6 SELECT etat, AVG (placa_pod) as średnia, COUNT (*) as liczba  
FROM pracownicy  
WHERE extract (year from zatrudniony) <= '1990'  
GROUP BY etat;
```

```
7 SELECT id_szefa, MIN(placa_pod) as minimalna  
FROM pracownicy  
GROUP BY id_szefa  
ORDER BY MIN(placa_pod) desc;
```

```
8 SELECT id_prac FROM pracownicy GROUP BY id_prac  
HAVING COUNT(*) > 1;
```

Bieżący slajd przedstawia rozwiązania zadań (6), (7) i (8), których treść zacytowano poniżej.

- (6) Wyświetl średnie pensje wypłacane w ramach poszczególnych etatów i liczbę pracowników zatrudnionych na danym etacie. Pomiń pracowników zatrudnionych po 1990 roku.
- (7) Dla każdego pracownika wyświetl pensję najgorzej zarabiającego podwładnego. Wyniki uporządkuj wg malejącej pensji.
- (8) Sprawdź, czy identyfikatory pracowników są unikalne.



## Podsumowanie

- Funkcje agregujące działają na zbiorach rekordów, nazywanych grupami.
- Funkcja agregująca wylicza pojedynczą wartość dla każdej grupy rekordów zapytania.
- Słowo kluczowe GROUP BY umożliwia podanie wyrażenia, którego wartości posłużą do podziału zbioru rekordów zapytania na grupy.
- Do eliminacji grup, nie spełniających określonych kryteriów, służy klauzula HAVING.

Ćwiczenie 3 – funkcje agregujące (19)

W zakończonym ćwiczeniu zaprezentowano funkcje agregujące. Są to funkcje, działające na grupach rekordów, wyliczające dla każdej z grup dokładnie jedną wartość. Do podziału zbioru rekordów na grupy służy klauzula GROUP BY zawierająca wyrażenie, wokół którego wartości tworzone są grupy rekordów. Do eliminacji grup z wyniku zapytania służy klauzula HAVING, w której podaje się warunek filtrujący.

Każde z omówionych zagadnień zostało utrwalone przez serię zadań.