Tablespace: sprawdzenie, powiększenie, utworzenie, usunięcie

Poziom: operacyjny **Dotyczy: Oracle Database 12c–19c** **Czas: ~25 min**

Tablespace kończy się miejscem częściej niż cokolwiek innego w Oracle — te fiszki to komplet: od sprawdzenia zajętości, przez powiększenie w locie, po utworzenie nowego i bezpieczne usunięcie starego.

Skróty użyte na tej stronie (1)

Skrót	Co znaczy
WMS	Warehouse Management System: system zarządzania magazynem

Fiszka

TYLKO ODCZYT

Wylistowanie wszystkich tablespace wraz z rozmiarem plików

```
SELECT tablespace_name, file_name, ROUND(bytes/1024/1024,2) AS size_mb FROM
dba_data_files;
```

Sprawdzamy, jakie tablespace istnieją w bazie i ile miejsca zajmuje każdy plik z osobna — pierwsze zapytanie, kiedy chcemy się rozeznać w rozkładzie danych na dysku.

Przykład:

```
TABLESPACE_NAME  FILE_NAME                                                  SIZE_MB
---------------- ---------------------------------------------------------- ----------
```

```
SYSTEM          /oracle/db/wmsprd/oradata/system01.dbf                  1600
SYSAUX          /oracle/db/wmsprd/oradata/sysaux01.dbf                   3290
UNDOTBS1        /oracle/db/wmsprd/oradata/undotbs01.dbf                  6600
USERS           /oracle/db/wmsprd/oradata/users01.dbf                     520
INTG            /oracle/db/wmsprd/oradata/intg07.dbf                    32000
INTG            /oracle/db/wmsprd/oradata/intg21.dbf                    32000
INTG            /oracle/db/wmsprd/oradata/intg22.dbf                    32000
```

□□ TYLKO ODCZYT

Sprawdzenie przestrzeni w tablespace: podział na tabele, indeksy i wolne miejsce

```
SELECT ts.tablespace_name,
       ROUND(df.total_size_mb, 2) AS total_tablespace_mb,
       ROUND(fs.free_space_mb, 2) AS free_space_mb,
       ROUND((df.total_size_mb - fs.free_space_mb), 2) AS used_space_mb,
       ROUND(tbl.tables_size_mb, 2) AS tables_size_mb,
       ROUND(idx.indexes_size_mb, 2) AS indexes_size_mb,
       ROUND(oth.other_size_mb, 2) AS other_segments_mb
FROM (SELECT tablespace_name FROM dba_tablespaces) ts
LEFT JOIN (SELECT tablespace_name, SUM(bytes)/1024/1024 total_size_mb FROM
dba_data_files GROUP BY tablespace_name) df
  ON ts.tablespace_name = df.tablespace_name
LEFT JOIN (SELECT tablespace_name, SUM(bytes)/1024/1024 free_space_mb FROM
dba_free_space GROUP BY tablespace_name) fs
  ON ts.tablespace_name = fs.tablespace_name
LEFT JOIN (SELECT tablespace_name, SUM(bytes)/1024/1024 tables_size_mb FROM dba_segments
WHERE segment_type = 'TABLE' GROUP BY tablespace_name) tbl
  ON ts.tablespace_name = tbl.tablespace_name
LEFT JOIN (SELECT tablespace_name, SUM(bytes)/1024/1024 indexes_size_mb FROM
dba_segments WHERE segment_type = 'INDEX' GROUP BY tablespace_name) idx
  ON ts.tablespace_name = idx.tablespace_name
LEFT JOIN (SELECT tablespace_name, SUM(bytes)/1024/1024 other_size_mb FROM dba_segments
WHERE segment_type NOT IN ('TABLE','INDEX') GROUP BY tablespace_name) oth
  ON ts.tablespace_name = oth.tablespace_name
WHERE ts.tablespace_name = 'NazwaTablespace';
```

Rozbijamy zajętość jednego tablespace na to, ile w nim faktycznie waży ile leży w tabelach, ile w indeksach, a ile w pozostałych segmentach — przydaje się, kiedy tablespace nagle urósł i szukamy, co dokładnie go zapełniło.

Przykład:

```
TABLESPACE  TOTAL_TABLESPACE_MB  FREE_SPACE_MB  USED_SPACE_MB  TABLES_SIZE_MB
INDEXES_SIZE_MB  OTHER_SEGMENTS_MB
---------- -------------------- ------------- -------------- ----------------
---------------- ------------------
INTG                     96000           17024          78976
51193.44         15814.56               8290.12
```

TYLKO ODCZYT

Zajętość przestrzeni we wszystkich tablespace naraz, z pilnowaniem TEMP

```
SELECT a.tablespace_name,
       ROUND(a.bytes_alloc/1024/1024,2) megs_alloc,
       ROUND(NVL(b.bytes_free,0)/1024/1024,2) megs_free,
       ROUND((a.bytes_alloc - NVL(b.bytes_free,0))/1024/1024,2) megs_used,
       ROUND((NVL(b.bytes_free,0)/a.bytes_alloc)*100,2) pct_free
FROM (SELECT tablespace_name, SUM(bytes) bytes_alloc FROM dba_data_files GROUP BY
tablespace_name) a,
     (SELECT tablespace_name, SUM(bytes) bytes_free FROM dba_free_space GROUP BY
tablespace_name) b
WHERE a.tablespace_name = b.tablespace_name (+)
ORDER BY 1;
```

Liczymy zajętość każdego tablespace jednym zapytaniem — dla zwykłych tablespace'ów sięgamy do `dba_free_space`, natomiast dla TEMP to zapytanie nic nie pokaże, bo TEMP nie zwalnia miejsca w ten sam sposób co tablespace danych — jego rzeczywistą zajętość liczymy przez `sys.v_$temp_space_header` w kolejnej fiszce.

Przykład:

```
TABLESPACE_NAME  MEGS_ALLOC  MEGS_FREE  MEGS_USED  PCT_FREE
---------------- ----------- ---------- ---------- ----------
SYSAUX                  3290     1354.69    1935.31      41.18
```

```
SYSTEM                 1600      142.5    1457.5        8.91
UNDOTBS1                770      519       251          67.4
USERS                   438       26.69     411.31       6.09
```

🔒 TYLKO ODCZYT

Zajętość TEMP na podstawie plików tymczasowych

```
SELECT tablespace_name,
       ROUND(SUM(h.bytes_used + h.bytes_free)/1048576,2) alloc_mb,
       ROUND(SUM(NVL(p.bytes_used,0))/1048576,2) used_mb,
       ROUND(SUM((h.bytes_free + h.bytes_used) - NVL(p.bytes_used,0))/1048576,2) free_mb
FROM sys.v_$temp_space_header h, sys.v_$temp_extent_pool p
WHERE p.file_id (+) = h.file_id
GROUP BY tablespace_name;
```

TEMP działa inaczej niż zwykły tablespace — miejsce zajęte przez sortowanie wraca do puli w ramach istniejącego pliku zamiast trwale go pomniejszać, więc `dba_free_space` go nie widzi. Do sprawdzenia realnego zużycia sięgamy do `v$temp_space_header` (rozmiar alokowany w pliku) i `v$temp_extent_pool` (co z tego jest faktycznie w użyciu). Szerzej o TEMP i sesjach, które go zajmują, mówi strona UNDO i TEMP.

Przykład:

```
TABLESPACE_NAME  ALLOC_MB  USED_MB  FREE_MB
---------------- --------- -------- --------
TEMP                 3000     235     2765
```

⚠️ ZMIENIA SYSTEM

Powiększenie datafile do wskazanego rozmiaru: ALTER DATABASE DATAFILE RESIZE

```
ALTER DATABASE DATAFILE 'SciezkaDoPliku' RESIZE RozmiarMB;
```

Powiększamy istniejący plik danych do konkretnej wielkości — najprostszy sposób na dołożenie miejsca w tablespace, kiedy plik jeszcze nie doszedł do limitu na dysku ani do `MAXSIZE`.

Przykład:

```
ALTER DATABASE DATAFILE '/oracle/db/wmsprd/oradata/data02.dbf' RESIZE 20000M;
```

```
Database altered.
```

□□ ZMIENIA SYSTEM

Włączenie i skonfigurowanie automatycznego powiększania datafile: AUTOEXTEND

```
ALTER DATABASE DATAFILE 'SciezkaDoPliku' AUTOEXTEND ON NEXT RozmiarMB MAXSIZE RozmiarMB;
```

> **Pojęcie: autoextend.** Mechanizm, który pozwala plikowi rosnąć samodzielnie, kiedy zapełni się do końca — zamiast czekać, aż baza zgłosi błąd braku miejsca. `NEXT` określa, o ile MB plik urośnie za jednym razem, `MAXSIZE` to twardy sufit, powyżej którego plik i tak przestanie rosnąć, nawet z włączonym autoextend.

Przykład:

```
ALTER DATABASE DATAFILE '/oracle/db/oradata/archive01.dbf' AUTOEXTEND ON NEXT 500M
MAXSIZE 8192M;
```

```
Database altered.
```

Sam `MAXSIZE` da się też zmienić osobno, bez ruszania trybu autoextend:

```
ALTER DATABASE DATAFILE 'SciezkaDoPliku' MAXSIZE RozmiarMB;
```

□□ **NEXT** = Near-End Crosstalk: przesłuch zbliżny, czyli ile sygnału z jednej pary przecieka do sąsiedniej pary na tym samym końcu kabla

□□ ZMIENIA SYSTEM

Powiększenie tablespace przez dodanie nowego datafile

```
ALTER TABLESPACE NazwaTablespace ADD DATAFILE 'SciezkaDoPliku' SIZE RozmiarMB AUTOEXTEND
ON NEXT RozmiarMB MAXSIZE RozmiarMB;
```

Zamiast rozciągać istniejący plik, dokładamy do tablespace kolejny — sensowne rozwiązanie, kiedy dysk pod dotychczasowy datafile się skończył albo chcemy rozłożyć operacje wejścia-wyjścia na więcej niż jeden plik fizyczny. To polecenie od razu tworzy nowy plik na dysku, w podanej ścieżce.

Przykład:

```
ALTER TABLESPACE INTG ADD DATAFILE '/oracle/db/wmsprd/oradata/data04.dbf' SIZE 4096M
AUTOEXTEND ON NEXT 500M MAXSIZE 8192M;
```

```
Tablespace altered.
```

NEXT = Near-End Crosstalk: przesłuch zbliżny, czyli ile sygnału z jednej pary przecieka do sąsiedniej pary na tym samym końcu kabla

TYLKO ODCZYT

Sprawdzenie konfiguracji datafile przed powiększeniem: rozmiar, MAXSIZE, autoextend

```
SELECT file_id, file_name, bytes/1024/1024 "FILESIZE_MB",
       maxbytes/1024/1024 "MAX_FILESIZE_MB",
       autoextensible,
       increment_by*8/1024 "AUTOEXTEND_VALUE_IN_MB"
FROM dba_data_files
WHERE tablespace_name = 'NazwaTablespace';
```

Sprawdzamy bieżący rozmiar, limit `MAXSIZE` i krok autoextend, zanim zdecydujemy, czy powiększać istniejący plik, czy dokładać nowy — to samo zapytanie sprawdza, czy plik w ogóle ma jeszcze zapas do `MAXSIZE`.

Przykład:

```
   FILE_ID FILE_NAME                                      FILESIZE_MB MAX_FILESIZE_MB AUT
AUTOEXTEND_VALUE_IN_MB
---------- ---------------------------------------------- ----------- --------------- ---
-----------------------
        18 /oracle/db/wmsprd/oradata/intg07.dbf               32000           32000
YES                    100
        21 /oracle/db/wmsprd/oradata/intg21.dbf               32000           32000
YES                    100
```

Formatowanie kolumn pod czytelny output ustawiamy przy okazji:

```
col FILE_NAME for a52
set linesize 500
```

□□ ZMIENIA SYSTEM

Utworzenie nowego tablespace: CREATE TABLESPACE

```
CREATE TABLESPACE NazwaTablespace DATAFILE 'SciezkaDoPliku' SIZE RozmiarMB ONLINE;
```

Zakładamy nowy tablespace razem z jego pierwszym datafile w jednym poleceniu — dopiero potem, w miarę potrzeb, dokładamy kolejne pliki poprzednią fiszką `ALTER TABLESPACE ... ADD DATAFILE`.

Przykład:

```
CREATE TABLESPACE prognozy DATAFILE '/oracle/db/wmsprd/oradata/prognozy.dbf' SIZE 500M
ONLINE;
```

```
Tablespace created.
```

Domyślną lokalizację, w której Oracle zakłada nowe pliki bez podawania pełnej ścieżki, ustawiamy raz na poziomie instancji:

```
ALTER SYSTEM SET db_create_file_dest = 'SciezkaDoKatalogu';
```

□□ **SET** = Switch Embedded Teaming: zespół kart sieciowych zbudowany w przełączniku wirtualnym Hyper-V; łączy dwie karty w jedno łącze bez ustawień po stronie przełącznika fizycznego, więc bez LACP

□□ TYLKO ODCZYT

Weryfikacja plików powiązanych z tablespace przed usunięciem

```
SELECT file_name FROM dba_data_files WHERE tablespace_name = 'NazwaTablespace';
```

Sprawdzamy, jakie pliki fizyczne wisiałyby po tym tablespace na dysku, gdybyśmy usuwali go bez opcji kasującej pliki — wykonujemy to zawsze przed właściwym `DROP TABLESPACE`, żeby wiedzieć, czego szukać na dysku w razie pomyłki.

Przykład:

```
FILE_NAME
----------------------------------------------------
/oracle/db/oradata/gis01.dbf
/oracle/db/oradata/gis02.dbf
```

Sprawdzamy też, jakie obiekty w ogóle leżą w danym tablespace, zanim go usuniemy:

```
SELECT owner, segment_name, segment_type FROM dba_segments WHERE tablespace_name =
'NazwaTablespace';
```

Zmaterializowane widoki wymagają osobnego dropnięcia przed dropnięciem tablespace — `DROP TABLESPACE` sam z siebie nie usunie ich definicji z katalogu, tylko dane pod spodem:

```
DROP MATERIALIZED VIEW NazwaSchematu.NazwaWidoku;
```

□□ OSTROŻNIE · NIEODWRACALNE

Usunięcie tablespace wraz z zawartością i plikami: DROP TABLESPACE INCLUDING CONTENTS AND DATAFILES

```
DROP TABLESPACE NazwaTablespace INCLUDING CONTENTS AND DATAFILES;
```

Usuwamy tablespace razem ze wszystkimi obiektami, które w nim leżały, i razem z plikami fizycznymi na dysku — bez `INCLUDING CONTENTS` polecenie odmówi wykonania, jeśli w tablespace jest cokolwiek, a bez `AND DATAFILES` sam wpis w bazie zniknie, ale pliki `.dbf` zostaną na dysku jako martwe śmieci.

Przykład:

```
DROP TABLESPACE GIS INCLUDING CONTENTS AND DATAFILES;
```

```
Tablespace dropped.
```

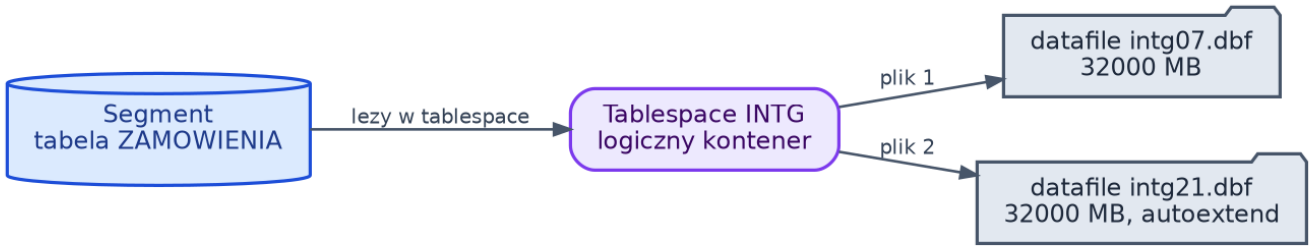
Z życia wzięte

Z ŻYCIA WZIĘTE

O 15:50 w piątek Marek zgłasza, że WMS przestał przyjmować nowe zamówienia. Tomek sprawdza zajętość tablespace INTG na WMSPRD i widzi, że plik `intg22.dbf` doszedł do swojego `MAXSIZE`, mimo że dysk pod nim ma jeszcze zapas. Po dodaniu kolejnego datafile poleceniem `ALTER TABLESPACE ADD DATAFILE` zamówienia znowu zapisują się bez błędu.

Czym jest tablespace

Pojęcie: tablespace. Logiczna przestrzeń dyskowa, w której Oracle trzyma dane — tabele, indeksy, segmenty wycofania, dane tymczasowe. Tablespace sam w sobie nie jest plikiem — jest kontenerem, pod którym leży jeden lub więcej plików fizycznych na dysku.



Segment tabeli leży w tablespace INTG, a tablespace rozkłada się fizycznie na dwa pliki datafile na dysku (kliknięcie otwiera pełny rozmiar)

Pojęcie: datafile. Fizyczny plik na dysku, w którym faktycznie leżą bajty danych. Jeden tablespace może mieć wiele datafile — dokładamy kolejny, kiedy potrzebujemy więcej miejsca, a dysk pod istniejący plik się skończył albo wolimy rozłożyć dane na kilka woluminów. Konfigurację i przenoszenie datafile rozwija strona Pliki danych: konfiguracja, przenoszenie, zmiana nazwy.

Pojęcie: segment. Obiekt bazy, który faktycznie zajmuje miejsce w tablespace — tabela, indeks, partycja, segment wycofania. Segment żyje wewnątrz tablespace i rozrasta się o kolejne extenty, dopóki starcza miejsca w datafile'ach, które ten tablespace tworzą. Zależność jest więc trójwarstwowa: tablespace (logiczna przestrzeń) → datafile (fizyczny plik) → segment (obiekt zajmujący w nim miejsce). Rozmiary segmentów i tabel liczymy na

> stronie Rozmiary tabel, segmentów i indeksów.

Dwa tablespace mają specjalną rolę i własne strony: TEMP, na sortowania i operacje tymczasowe, oraz UNDO, na dane wycofania transakcji — obydwa opisane na stronie UNDO i TEMP.

> **Pojęcie: BIGFILE i SMALLFILE.** Tablespace typu SMALLFILE (domyślny) może mieć wiele datafile, każdy do około 4 mln bloków. Tablespace typu BIGFILE ma dokładnie jeden, bardzo duży plik — upraszcza zarządzanie przy naprawdę dużych tablespace'ach kosztem utraty możliwości rozłożenia danych na kilka plików fizycznych. W naszych bazach produkcyjnych korzystamy z klasycznego SMALLFILE i dokładamy kolejne datafile w miarę potrzeb — to podejście widać we wszystkich fiszkach niżej.

Częste błędy

Dropnięcie tablespace bez wcześniejszego sprawdzenia zmaterializowanych widoków. `DROP TABLESPACE ... INCLUDING CONTENTS` potrafi usunąć dane, ale definicja zmaterializowanego widoku bywa zależna od segmentu, który zniknął — bezpieczniej zawsze najpierw przejrzeć `dba_segments` pod kątem `MATERIALIZED VIEW` i dropnąć je jawnie.

Ustawienie `MAXSIZE` mniejszego niż aktualny rozmiar pliku. Baza przyjmie polecenie, ale plik po prostu przestanie rosnąć w chwili, gdy dojdzie do tego limitu — a to zwykle wychodzi dopiero przy najbliższym powiadomieniu o braku miejsca, nie przy samej zmianie parametru.

Liczenie zajętości TEMP tak jak zwykłego tablespace. Zapytanie po `dba_free_space` dla TEMP zwykle nie zwróci nic sensownego — właściwe dane dają `v$temp_space_header` i `v$temp_extent_pool`, pokazane w osobnej fiszce wyżej.

Powiązane: Pliki danych: konfiguracja, przenoszenie, zmiana nazwy · UNDO i TEMP · Rozmiary tabel, segmentów i indeksów · Rozmiar bazy: pliki, tablespace'y, autoextend

Źródła: notatki własne, dokumentacja Oracle Database (Administrator's Guide — Managing Tablespaces)

Revision #43
Created 2026-09-24 23:34:25 UTC by Tomasz Konieczny
Updated 2026-10-26 14:08:56 UTC by Tomasz Konieczny