Przestrzeń i pliki bazy

Tablespace, datafile, UNDO, TEMP, redo logi i pliki kontrolne.

- Tablespace: sprawdzenie, powiększenie, utworzenie, usunięcie
- Pliki danych: konfiguracja, przenoszenie, zmiana nazwy
- UNDO i TEMP
- Redo logi: grupy, przełączanie, rozmiar
- Pliki kontrolne

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.
```

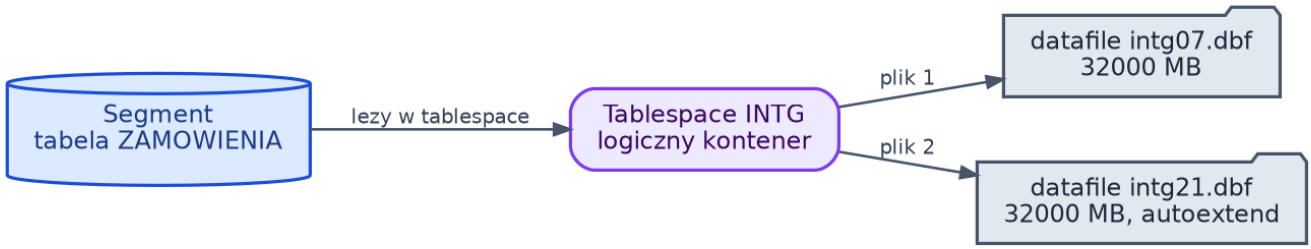
Z życia wzięte

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)

Pliki danych: konfiguracja, przenoszenie, zmiana nazwy

Poziom: operacyjny **Dotyczy: Oracle Database 12c–19c** **Czas: ~20 min**

Datafile trzeba czasem przenieść na inny dysk albo poprawić mu nazwę po migracji — te fiszki pokazują, jak sprawdzić jego bieżącą konfigurację i jak bezpiecznie zmienić mu adres, offline i online.

Skróty użyte na tej stronie (2)

Skrót	Co znaczy
SQL	Structured Query Language: język zapytań do relacyjnych baz danych
WMS	Warehouse Management System: system zarządzania magazynem

Fiszka

ZMIENIA SYSTEM

Sprawdzenie konfiguracji wszystkich datafile: rozmiar, autoextend, MAXSIZE

```
SELECT tablespace_name, file_name, bytes/1048576 file_size_mb,
       autoextensible,
       increment_by*8192/1048576 AS increment_by_in_mb,
       TRUNC((CASE WHEN bytes > maxbytes THEN bytes ELSE maxbytes END)/(1024*1024))
maxsize_of_datafile_in_mb
FROM dba_data_files
ORDER BY file_id;
```

Wyciągamy pełny obraz wszystkich plików danych w bazie naraz — rozmiar bieżący, czy autoextend jest włączony, o ile rośnie za jednym razem i jaki jest jego realny sufit. Kolumna

`maxsize_of_datafile_in_mb` bierze większą z dwóch wartości, bo `MAXSIZE` bywa ustawiony niżej niż faktyczny bieżący rozmiar pliku, jeśli plik urósł ręcznie po ustawieniu limitu.

Przykład:

```
TABLESPACE_NAME  FILE_NAME                                          FILE_SIZE_MB AUT
INCREMENT_BY_IN_MB MAXSIZE_OF_DATAFILE_IN_MB
---------------- ------------------------------------------------ ------------ ---
------------------ --------------------------
SYSTEM           /oracle/db/wmsprd/oradata/system01.dbf                1600
NO                    0                         1600
UNDOTBS1         /oracle/db/wmsprd/oradata/undotbs01.dbf               6600
YES                 100                         8192
INTG             /oracle/db/wmsprd/oradata/intg07.dbf                 32000
YES                 100                        32000
```

ZMIENIA SYSTEM

Sprawdzenie konfiguracji tempfile

```
SELECT tablespace_name, file_name, bytes/1048576 file_size_mb,
       autoextensible,
       increment_by*8192/1048576 AS increment_by_in_mb,
       TRUNC((CASE WHEN bytes > maxbytes THEN bytes ELSE maxbytes END)/(1024*1024))
maxsize_of_datafile_in_mb
FROM dba_temp_files
ORDER BY file_id;
```

To samo zapytanie, tylko po widoku `dba_temp_files` zamiast `dba_data_files` — sprawdzamy rozmiar i autoextend plików tymczasowych, na przykład zanim zdecydujemy o przebudowie tablespace TEMP opisanej na stronie UNDO i TEMP.

Przykład:

```
TABLESPACE_NAME  FILE_NAME                                          FILE_SIZE_MB AUT
INCREMENT_BY_IN_MB MAXSIZE_OF_DATAFILE_IN_MB
---------------- ------------------------------------------------ ------------ ---
------------------ --------------------------
TEMP             /oracle/db/wmsprd/oradata/temp01.dbf                  3000
```

```
YES                 500                         8192
```

ZMIENIA SYSTEM

Przeniesienie datafile offline: procedura klasyczna

Zanim zaczniemy: ta metoda działa na każdej wersji Oracle i nie wymaga zamykania całej instancji, ale wymaga zdjęcia offline tablespace, do którego przenoszony plik należy — przez czas przenoszenia dane w tym tablespace są niedostępne dla aplikacji. Dysk docelowy musi mieć wolne miejsce co najmniej na rozmiar przenoszonego pliku.

Zdejmujemy tablespace offline:

```
ALTER TABLESPACE NazwaTablespace OFFLINE;
```

Kopiujemy plik w systemie operacyjnym na nową lokalizację — z poziomu powłoki systemowej albo bezpośrednio z SQL*Plus przez `HOST`:

```
HOST cp SciezkaDoPliku SciezkaDoNowegoPliku
```

Informujemy bazę o nowej lokalizacji pliku:

```
ALTER DATABASE RENAME FILE 'SciezkaDoPliku' TO 'SciezkaDoNowegoPliku';
```

Wracamy z tablespace online:

```
ALTER TABLESPACE NazwaTablespace ONLINE;
```

Weryfikujemy, że baza widzi plik pod nową ścieżką:

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

Przykład:

```
Tablespace altered.


Database altered.


Tablespace altered.


FILE_NAME
```

```
----------------------------------------------------
/oracle/db/wmsprd/oradata_new/intg07.dbf
```

Dopiero po potwierdzeniu, że baza otworzyła plik pod nowym adresem i tablespace wrócił online bez błędów, usuwamy starą kopię pliku z dysku.

SQL = Structured Query Language: język zapytań do relacyjnych baz danych

ZMIENIA SYSTEM

Przeniesienie datafile online od wersji 12c: ALTER DATABASE MOVE DATAFILE

```
ALTER DATABASE MOVE DATAFILE 'SciezkaDoPliku' TO 'SciezkaDoNowegoPliku';
```

Od wersji 12c przenosimy plik bez zdejmowania tablespace offline i bez przerwy w dostępności danych — Oracle kopiuje plik w tle, przełącza się na nową lokalizację i, w zależności od wariantu polecenia, sam usuwa starą kopię. To krótsza i bezpieczniejsza droga niż procedura klasyczna, o ile baza jest już na 12c lub wyżej.

Przykład:

```
ALTER DATABASE MOVE DATAFILE '/oracle/db/19300/dbs/intg07' TO
'/oracle/db/wmsprd/oradata/intg07.dbf';
```

```
Database altered.
```

Weryfikujemy nową ścieżkę tym samym zapytaniem po `dba_data_files` co w klasycznej procedurze.

Z życia wzięte

Z ŻYCIA WZIĘTE

Po migracji dysków pod ora01 na nowy wolumen ścieżki kilku plików danych WMSPRD wciąż wskazują na stary punkt montowania. Tomek sprawdza `dba_data_files` i zauważa, że jeden z plików `intg07.dbf` nie doszedł do nowej lokalizacji razem z resztą. Poleceniem `ALTER DATABASE MOVE DATAFILE` przenosi go bez zatrzymywania instancji, żeby WMS w Swarzędzu nie zauważył przerwy.

Czym jest datafile i tempfile

> **Pojęcie: datafile.** Fizyczny plik na dysku, w którym leżą dane jednego tablespace — sam podział na tablespace i to, jak ma się on do datafile i segmentu, opisuje strona Tablespace: sprawdzenie, powiększenie, utworzenie, usunięcie. Każdy datafile ma swój numer (`FILE_ID`), ścieżkę, rozmiar i, opcjonalnie, ustawienia autoextend.

> **Pojęcie: tempfile.** Plik fizyczny należący do tablespace typu TEMP. Różni się od zwykłego datafile tym, że nie trzyma trwałych danych — leżą w nim tylko tymczasowe struktury sortowania i operacji, które się w nim nie mieszczą w pamięci. Dlatego tempfile nie jest częścią backupu bazy w takim samym sensie jak datafile — w razie utraty odtwarza się go samym poleceniem tworzącym nowy plik, bez przywracania z kopii zapasowej.

Zmiana nazwy albo lokalizacji datafile jest operacją, po którą sięgamy zwykle w dwóch sytuacjach: przenosimy dane na inny wolumen z powodu miejsca na dysku, albo porządkujemy nazewnictwo po migracji bazy na inny serwer, gdzie pliki wylądowały pod inną ścieżką niż docelowa.

Częste błędy

Zapomnienie o zdjęciu tablespace offline przy metodzie klasycznej. `ALTER DATABASE RENAME FILE` bez wcześniejszego `ALTER TABLESPACE ... OFFLINE` kończy się błędem — baza nie pozwoli podmienić ścieżki pliku, który jest aktualnie otwarty i używany.

Usunięcie starego pliku z dysku przed weryfikacją. Dopóki nie potwierdzimy zapytaniem po `dba_data_files`, że baza faktycznie otworzyła plik pod nową ścieżką i tablespace wrócił online bez błędu, stara kopia zostaje na dysku — to jedyna droga odwrotu, gdyby coś poszło nie tak.

Próba `MOVE DATAFILE` na wersji starszej niż 12c. Polecenie po prostu nie istnieje w starszych wersjach — jedyną drogą jest tam procedura klasyczna z offline tablespace.

Powiązane: Tablespace: sprawdzenie, powiększenie, utworzenie, usunięcie · UNDO i TEMP · Migracja bazy danych na inny serwer

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

UNDO i TEMP

Poziom: operacyjny **Dotyczy: Oracle Database 12c–19c** **Czas: ~20 min**

Dwa specjalne tablespace, które nie trzymają danych aplikacji, tylko obsługują mechanikę silnika — UNDO pod wycofanie transakcji i wielowersyjność, TEMP pod sortowania, które nie zmieściły się w pamięci. Oba potrafią się zapchać w zupełnie inny sposób niż zwykły tablespace.

Skróty użyte na tej stronie (1)

Skrót	Co znaczy
PGA	Program Global Area: prywatna pamięć procesu serwera Oracle

Fiszka

TYLKO ODCZYT

Sprawdzenie zajętości UNDO wg statusu extentów

```
SELECT tablespace_name, status, SUM(blocks)*8192/1024/1024/1024 AS gb
FROM dba_undo_extents
GROUP BY tablespace_name, status;
```

Sprawdzamy, ile miejsca w tablespace UNDO zajmują extenty w poszczególnych stanach — `ACTIVE` (aktualnie potrzebne żywym transakcjom), `UNEXPIRED` (jeszcze w oknie `UNDO_RETENTION`, przydatne do zapytań flashback) i `EXPIRED` (już niepotrzebne, gotowe do ponownego użycia).

Przykład:

```
TABLESPACE_NAME  STATUS          GB
---------------- --------- ---------
UNDOTBS1         ACTIVE         .42
UNDOTBS1         UNEXPIRED     1.15
UNDOTBS1         EXPIRED       3.68
```

⚠ ZMIENIA SYSTEM

Regeneracja tablespace UNDO: utworzenie nowego i przełączenie

Zanim zaczniemy: regenerację robimy, kiedy stary tablespace UNDO urósł ponad rozsądny rozmiar (zwykle po jednorazowym szczycie transakcyjnym) i chcemy odzyskać miejsce na dysku, zamiast czekać, aż same extenty się zwolnią. Baza cały czas pozostaje online — operacja jest w pełni bezobsługowa dla aplikacji, poza chwilowym większym obciążeniem podczas przełączania.

Tworzymy nowy tablespace UNDO obok istniejącego:

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

Przykład:

```
CREATE UNDO TABLESPACE UNDOTBS2 DATAFILE '/oracle/db/wmstest/oradata/undotbs02.dbf' SIZE
1024M;
```

```
Tablespace created.
```

Przełączamy instancję na nowy tablespace parametrem `UNDO_TABLESPACE`:

```
ALTER SYSTEM SET undo_tablespace='NazwaTablespace' SCOPE=BOTH;
```

Przykład:

```
ALTER SYSTEM SET undo_tablespace='UNDOTBS2' SCOPE=BOTH;
```

```
System altered.
```

Od tego momentu wszystkie nowe transakcje piszą undo do nowego tablespace, a stary trzyma jeszcze dane starszych, wciąż aktywnych transakcji — dopiero kiedy się od nich uwolni, można go bezpiecznie usunąć:

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

Przykład:

```
DROP TABLESPACE UNDOTBS1 INCLUDING CONTENTS AND DATAFILES;
```

```
Tablespace dropped.
```

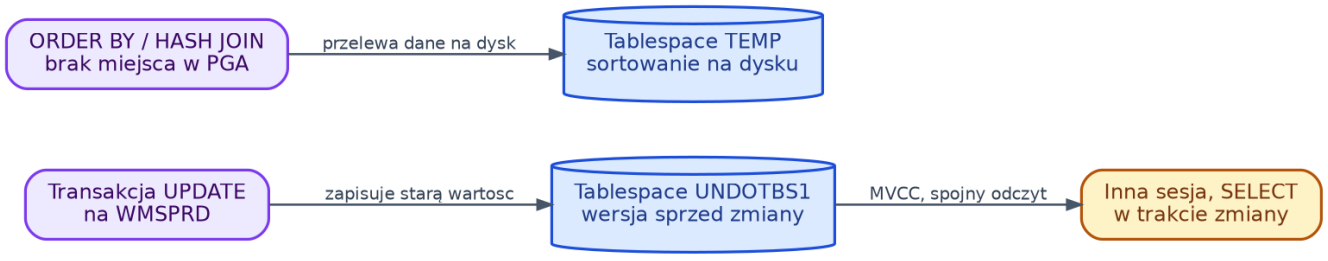
Jeżeli `DROP TABLESPACE` odmówi z powodu wciąż aktywnych extentów w starym tablespace, odczekujemy chwilę i sprawdzamy ponownie fiszką zajętości UNDO wyżej — extenty muszą przejść w stan `EXPIRED`, zanim miejsce faktycznie się zwolni.

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

Czym jest segment wycofania i po co istnieje

Pojęcie: segment wycofania (undo). Miejsce, w którym Oracle zapisuje poprzednią wersję zmienianych danych, zanim nadpisze je nową wartością. Kiedy transakcja robi `ROLLBACK`, silnik odtwarza stan sprzed zmiany właśnie z segmentu wycofania. To samo undo napędza mechanizm wielowersyjności — czytający proces widzi dane w takim stanie, w jakim były w chwili rozpoczęcia jego zapytania, nawet jeśli w międzyczasie ktoś inny je zmienia, ponieważ Oracle w razie potrzeby sięga po starszą wersję wiersza właśnie z UNDO.

Mechanizm ten szerzej opisuje strona MVCC: dlaczego czytający nie blokuje piszącego.



Transakcja zapisuje starą wartość wiersza do UNDO, dzięki czemu inna sesja czyta spójny stan sprzed zmiany, a osobno sortowanie bez miejsca w PGA przelewa dane do TEMP (kliknięcie otwiera pełny rozmiar)

UNDO leży w specjalnym tablespace, zwykle nazwanym `UNDOTBS1`, wskazanym parametrem `UNDO_TABLESPACE`. W przeciwieństwie do zwykłego tablespace danych, jego zajętość jest z natury zmienna w czasie — rośnie wraz z długimi, aktywnymi transakcjami i kurczy się, kiedy dane

wycofania przestają być komukolwiek potrzebne.

UNDO_RETENTION i ORA-01555

> **Pojęcie: UNDO_RETENTION.** Parametr określający, przez ile sekund Oracle stara się zachować dane wycofania po zakończeniu transakcji, zamiast od razu nadpisywać je nowszymi danymi — im dłuższe okno, tym dłużej działają zapytania flashback i tym mniejsze ryzyko błędu przy długo trwających zapytaniach na spójnych danych.

Błąd ORA-01555 (snapshot too old) pojawia się, kiedy długo działające zapytanie próbuje odczytać starszą wersję wiersza, a odpowiadające jej dane wycofania zostały już nadpisane przez inne transakcje, zanim zapytanie zdążyło po nie sięgnąć. Najczęstsza przyczyna to zbyt mały tablespace UNDO albo zbyt krótki `UNDO_RETENTION` w stosunku do czasu trwania najdłuższych zapytań w bazie — rozwiązaniem jest zwykle powiększenie UNDO albo wydłużenie retencji, rzadziej skrócenie samego zapytania.

Czym jest TEMP

> **Pojęcie: tablespace TEMP.** Przestrzeń na dane tymczasowe — sortowania (`ORDER BY`, `GROUP BY`), operacje `HASH JOIN` i inne struktury pośrednie, które nie zmieściły się w przydzielonej pamięci PGA. W przeciwieństwie do zwykłego tablespace, miejsce zwolnione przez zakończone sortowanie od razu wraca do puli wolnego miejsca w tym samym pliku, więc TEMP praktycznie nigdy nie rośnie trwale — chyba że jedna operacja jest naprawdę ogromna albo coś trzyma sesję otwartą z niedokończonym sortowaniem.

Sprawdzenie sesji zajmujących TEMP

```
SELECT s.sid, s.serial#, s.username, u.tablespace, ROUND(u.blocks*8/1024,2) AS mb
FROM v$session s
JOIN v$tempseg_usage u ON u.session_addr = s.saddr
ORDER BY u.blocks DESC;
```

Sprawdzamy, która konkretnie sesja aktualnie zajmuje TEMP i ile — przydaje się, kiedy zapytanie z fiszki o zajętości TEMP na stronie Tablespace: sprawdzenie, powiększenie, utworzenie, usunięcie pokazuje wysokie zużycie, a szukamy winnego zapytania.

Przykład:

```
   SID  SERIAL# USERNAME   TABLESPACE           MB
------- -------- ---------- ------------ -----------
    412     8821 WMS_APP     TEMP               1240
    118     2054 WMS_ADMIN   TEMP                310
```

Odtworzenie tablespace TEMP

Zanim zaczniemy: odtwarzamy TEMP od zera, kiedy plik tymczasowy uległ uszkodzeniu albo kiedy chcemy zmienić jego rozmiar bazowy bez zostawiania rozdętego pliku. Tempfile nie jest częścią backupu bazy w klasycznym sensie — operacja jest więc czysto logiczna, bez odtwarzania z kopii zapasowej.

Tworzymy nowy tablespace tymczasowy obok istniejącego:

```
CREATE TEMPORARY TABLESPACE NazwaTablespace TEMPFILE 'SciezkaDoPliku' SIZE RozmiarMB
AUTOEXTEND ON NEXT 500M MAXSIZE 8192M;
```

Przełączamy go na domyślny tablespace tymczasowy bazy:

```
ALTER DATABASE DEFAULT TEMPORARY TABLESPACE NazwaTablespace;
```

Usuwamy stary tablespace TEMP razem z plikiem:

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

Przykład:

```
Tablespace created.


Database altered.


Tablespace dropped.
```

Z życia wzięte

Z ŻYCIA WZIĘTE

W środku miesiąca Marek zgłasza, że raport rozliczenia magazynu urywa się błędem po czterdziestu minutach działania. Tomek sprawdza `DBA_UNDO_EXTENTS` na WMSPRD i widzi, że tablespace UNDOTBS1 jest zbyt mały wobec długości tego zapytania, więc starsze dane wycofania zostały nadpisane, zanim raport zdążył po nie sięgnąć. Po podniesieniu `UNDO_RETENTION` i dołożeniu miejsca do UNDO ten sam raport kończy się bez błędu ORA-01555.

Częste błędy

Próba dropnięcia UNDO, który wciąż jest wskazany parametrem UNDO_TABLESPACE. Baza odmówi — najpierw trzeba przełączyć instancję na inny tablespace UNDO, dopiero potem można usunąć poprzedni.

Traktowanie wysokiej zajętości TEMP jak wycieku. Zanim zaczniemy powiększać TEMP, warto sprawdzić fiszką sesji, czy to nie jest jedno bardzo duże, ale krótkotrwałe zapytanie — miejsce samo wróci do puli po jego zakończeniu.

Za krótki UNDO_RETENTION przy długich raportach. Baza domyślnie próbuje utrzymać retencję automatycznie na podstawie historii obciążenia, ale przy nagłym skoku długości zapytań i tak można trafić na ORA-01555 — w takiej sytuacji retencję warto podnieść ręcznie, a nie tylko powiększać UNDO.

Powiązane: Tablespace: sprawdzenie, powiększenie, utworzenie, usunięcie · MVCC: dlaczego czytający nie blokuje piszącego · Pliki danych: konfiguracja, przenoszenie, zmiana nazwy

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

Redo logi: grupy, przełączanie, rozmiar

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

Redo logi to dziennik zapisu, bez którego Oracle nie potrafiłby odtworzyć niezacommitowanych zmian po awarii ani zbudować backupu gorącego — te fiszki pokazują zarządzanie grupami, dobór rozmiaru i to, jak często baza je przełącza.

Skróty użyte na tej stronie (2)

Skrót	Co znaczy
GB	Gigabyte: miliard bajtów, jednostka pojemności pamięci i plików, tu najczęściej rozmiar modelu albo pamięci karty graficznej
WMS	Warehouse Management System: system zarządzania magazynem

Fiszka

TYLKO ODCZYT

Sprawdzenie wszystkich grup redo logów: v$log

```
SELECT group#, bytes/1024/1024 mb, status FROM v$log ORDER BY group#;
```

Sprawdzamy, ile grup istnieje, jaki mają rozmiar i w jakim są stanie — `CURRENT` (grupa, do której baza pisze teraz), `ACTIVE` (potrzebna jeszcze do odzyskiwania), `INACTIVE` (wolna, gotowa do ponownego użycia).

Przykład:

```
    GROUP#         MB STATUS
--------- ---------- ----------------
        1        512 INACTIVE
```

```
         2        512 CURRENT
         3        512 INACTIVE
         4        512 INACTIVE
         5        512 INACTIVE
```

Pełny widok ze wszystkimi kolumnami, kiedy potrzebujemy więcej szczegółów niż tylko rozmiar i status:

```
SELECT * FROM v$log;
```

👁️ TYLKO ODCZYT

Sprawdzenie ścieżek plików każdej grupy: v$logfile

```
SELECT group#, member, status FROM v$logfile;
```

Sprawdzamy, gdzie na dysku leżą fizyczne pliki każdej grupy i każdego jej członka — potrzebne przed dodaniem albo usunięciem grupy, żeby wiedzieć, jakich ścieżek użyć albo które pliki skasować ręcznie z dysku.

Przykład:

```
   GROUP# MEMBER                                          STATUS
--------- ---------------------------------------------- --------
        1 /oracle/db/wmstest/redoA/redo_g01m01.log
        1 /oracle/db/wmstest/redoB/redo_g01m02.log
        2 /oracle/db/wmstest/redoA/redo_g02m01.log
        2 /oracle/db/wmstest/redoB/redo_g02m02.log
```

Sama ścieżka do konkretnej grupy, bez reszty:

```
SELECT member FROM v$logfile WHERE group# = NumerGrupy;
```

⚠️ ZMIENIA SYSTEM

Dodanie nowej grupy redo logów: ALTER DATABASE ADD LOGFILE GROUP

```
ALTER DATABASE ADD LOGFILE GROUP NumerGrupy ('SciezkaDoPliku', 'SciezkaDoPliku') SIZE
RozmiarMB;
```

Dodajemy nową grupę z dwoma członkami — pierwszy na jednym dysku, drugi na drugim, zgodnie z zasadą, że każdy członek danej grupy powinien leżeć na osobnym fizycznym dysku.

Przykład:

```
ALTER DATABASE ADD LOGFILE GROUP 5
  ('/oracle/db/wmsprd/redoA/redo_g05m01.log',
   '/oracle/db/wmsprd/redoB/redo_g05m02.log')
SIZE 2048M;
```

```
Database altered.
```

Przy zakładaniu wielu grup naraz w nowej instancji, konsekwentnie trzymamy nieparzyste numery grup na dysku redoA, a parzyste na redoB:

```
ALTER DATABASE ADD LOGFILE GROUP 7 ('/oracle/db/wmsprd/redoA/redo_g07m1.log',
'/oracle/db/wmsprd/redoB/redo_g07m2.log') SIZE 512M;
ALTER DATABASE ADD LOGFILE GROUP 8 ('/oracle/db/wmsprd/redoB/redo_g08m1.log',
'/oracle/db/wmsprd/redoB/redo_g08m2.log') SIZE 512M;
```

◫◫ OSTROŻNIE · NIEODWRACALNE

Usunięcie wybranej grupy redo logów: ALTER DATABASE DROP LOGFILE GROUP

```
ALTER DATABASE DROP LOGFILE GROUP NumerGrupy;
```

Usuwamy wpis grupy z bazy — grupa musi być w stanie `INACTIVE`, nie da się usunąć grupy aktualnie używanej (`CURRENT`) ani jeszcze potrzebnej do odzyskiwania (`ACTIVE`). To polecenie nie usuwa plików grupy z dysku — po jego wykonaniu kasujemy je ręcznie systemowym poleceniem `rm`.

Przykład:

```
ALTER DATABASE DROP LOGFILE GROUP 5;
```

```
Database altered.
```

⚠ OSTROŻNIE · NIEODWRACALNE

Usunięcie kilku grup w pętli PL/SQL

```
BEGIN
  FOR i IN NumerGrupyOd..NumerGrupyDo LOOP
    EXECUTE IMMEDIATE 'ALTER DATABASE DROP LOGFILE GROUP ' || i;
  END LOOP;
END;
/
```

Kiedy porządkujemy po sobie kilka grup naraz, na przykład po zmniejszeniu ich liczby albo po migracji, wygodniej przejechać zakres numerów jedną pętlą niż wołać `DROP LOGFILE GROUP` osobno dla każdej.

Przykład:

```
BEGIN
  FOR i IN 6..9 LOOP
    EXECUTE IMMEDIATE 'ALTER DATABASE DROP LOGFILE GROUP ' || i;
  END LOOP;
END;
/
```

```
PL/SQL procedure successfully completed.
```

execute = FortiOS: polecenia, które coś wykonują na urządzeniu, na przykład execute backup config wysyła kopię konfiguracji, a execute ha manage otwiera sesję na drugiej zaporze w parze

⚠ ZMIENIA SYSTEM

Wymuszenie przełączenia na kolejną grupę: ALTER SYSTEM SWITCH LOGFILE

```
ALTER SYSTEM SWITCH LOGFILE;
```

Wymuszamy natychmiastowe przejście na kolejną grupę, zanim baza sama by to zrobiła po zapełnieniu bieżącej — przydaje się przed operacjami, które chcemy mieć na pewno zarchiwizowane od razu, na przykład przed backupem albo przed planowanym wyłączeniem instancji.

Przykład:

```
System altered.
```

◫◫ ZMIENIA SYSTEM

Wymuszenie checkpointu: ALTER SYSTEM CHECKPOINT

```
ALTER SYSTEM CHECKPOINT;
```

> **Pojęcie: checkpoint.** Moment, w którym Oracle zrzuca na dysk wszystkie zmodyfikowane bloki z pamięci (bufory), synchronizując pliki danych ze stanem zapisanym w redo logach — po checkpoincie redo log nie musi już pamiętać zmian sprzed niego, bo są one już trwale zapisane w plikach danych. Wymuszamy go ręcznie rzadko, głównie przy diagnozowaniu problemów z odzyskiwaniem albo tuż przed operacjami na plikach fizycznych bazy.

Przykład:

```
System altered.
```

◫◫ ZMIENIA SYSTEM

Sprawdzenie odstępów między przełączeniami grup: v$log_history

```
WITH hist AS (
  SELECT first_time,
         LAG(first_time) OVER (ORDER BY first_time) prev_time
  FROM v$log_history
  WHERE first_time >= SYSDATE - 1
)
SELECT ROUND(AVG((first_time - prev_time)*24*60), 2) AS avg_mins_between_switch
FROM hist
```

```
WHERE prev_time IS NOT NULL;
```

Liczymy, co ile minut w ciągu ostatniej doby baza przełączała redo logi — jeżeli wynik jest bardzo niski, przełączenia są zbyt częste, a jeśli bardzo wysoki, redo logi mogą być za duże w stosunku do ruchu w bazie, co wydłuża czas odzyskiwania po awarii.

Przykład:

```
AVG_MINS_BETWEEN_SWITCH
-----------------------
                   18.4
```

Trzymamy się reguły, że redo logi powinny się rotować nie częściej niż cztery razy na godzinę (czyli nie częściej niż co 15 minut) i nie rzadziej niż raz na 1–2 godziny. Wartość tę warto dopasować do częstotliwości backupów archive logów — najczęściej przyjmuje się 30 minut jako wartość wyjściową.

Liczba przełączeń na godzinę w ostatniej dobie, tym samym zapytaniem rozbitym na godziny:

```
SELECT TO_CHAR(first_time,'YYYY-MM-DD HH24') AS hour, COUNT(*) AS switches
FROM v$log_history
WHERE first_time >= SYSDATE - 1
GROUP BY TO_CHAR(first_time,'YYYY-MM-DD HH24')
ORDER BY 1;
```

Przykład:

```
HOUR             SWITCHES
---------------- ----------
2026-09-24 08             2
2026-09-24 09             3
2026-09-24 10             5
2026-09-24 11             2
```

Jeżeli wygenerowana liczba przełączeń wykracza poza wspomnianą regułę, zmieniamy rozmiar redo logów albo wymuszamy regularny switch parametrem `ARCHIVE_LAG_TARGET` — na przykład ustawiając wymuszenie przełączenia co najwyżej co 30 minut:

```
ALTER SYSTEM SET archive_lag_target = 1800 SCOPE = BOTH;
```

LAG = Link Aggregation Group: grupa łączy spięta w jedno logiczne; wariant MC-LAG rozciąga ją na dwa przełączniki · **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

ZMIENIA SYSTEM

Wyliczenie ilości generowanego redo w MB na godzinę

```
SELECT TRUNC(completion_time,'HH24') AS hour,
       SUM(blocks) * MAX(block_size) / 1024 / 1024 AS redo_mb_per_hour
FROM v$archived_log
WHERE completion_time >= SYSDATE - 7
  AND standby_dest = 'NO'
  AND resetlogs_id = (SELECT resetlogs_id FROM v$database)
GROUP BY TRUNC(completion_time,'HH24')
ORDER BY redo_mb_per_hour DESC
FETCH FIRST 10 ROWS ONLY;
```

Liczymy godzinowy szczyt generowania redo z ostatniego tygodnia na podstawie już zarchiwizowanych logów — to jest dane wejściowe do wyliczenia sensownego rozmiaru redo logów, w kolejnej fiszce.

Przykład:

```
HOUR             REDO_MB_PER_HOUR
---------------- ------------------
2026-09-22 10                  980
2026-09-23 09                  845
2026-09-24 14                  790
```

Zużycie na poziomie metryk systemowych, gdy interesuje nas bieżąca średnia zamiast historii archiwum:

```
SELECT ROUND(MAX(value)/1024/1024*60, 2) AS redo_mb_per_min
FROM v$sysmetric_history
WHERE metric_name = 'Redo Generated Per Sec';
```

TYLKO ODCZYT

Dobór rozmiaru redo logów na podstawie szczytowego generowania

Nie jest to gotowa komenda, tylko formuła, którą stosujemy po wyliczeniu szczytowego tempa generowania redo poprzednią fiszką:

```
target_redo_log_size_MB = peak_redo_generation_rate_MB_per_min ×
desired_minutes_between_log_switches
```

Za `desired_minutes_between_log_switches` domyślnie przyjmujemy 30 minut, zgodnie z regułą częstotliwości przełączeń podaną wyżej. Wynik warto zaokrąglić w górę do najbliższej potęgi dwójki — na przykład wynik 126 MB zaokrąglamy do 256 MB.

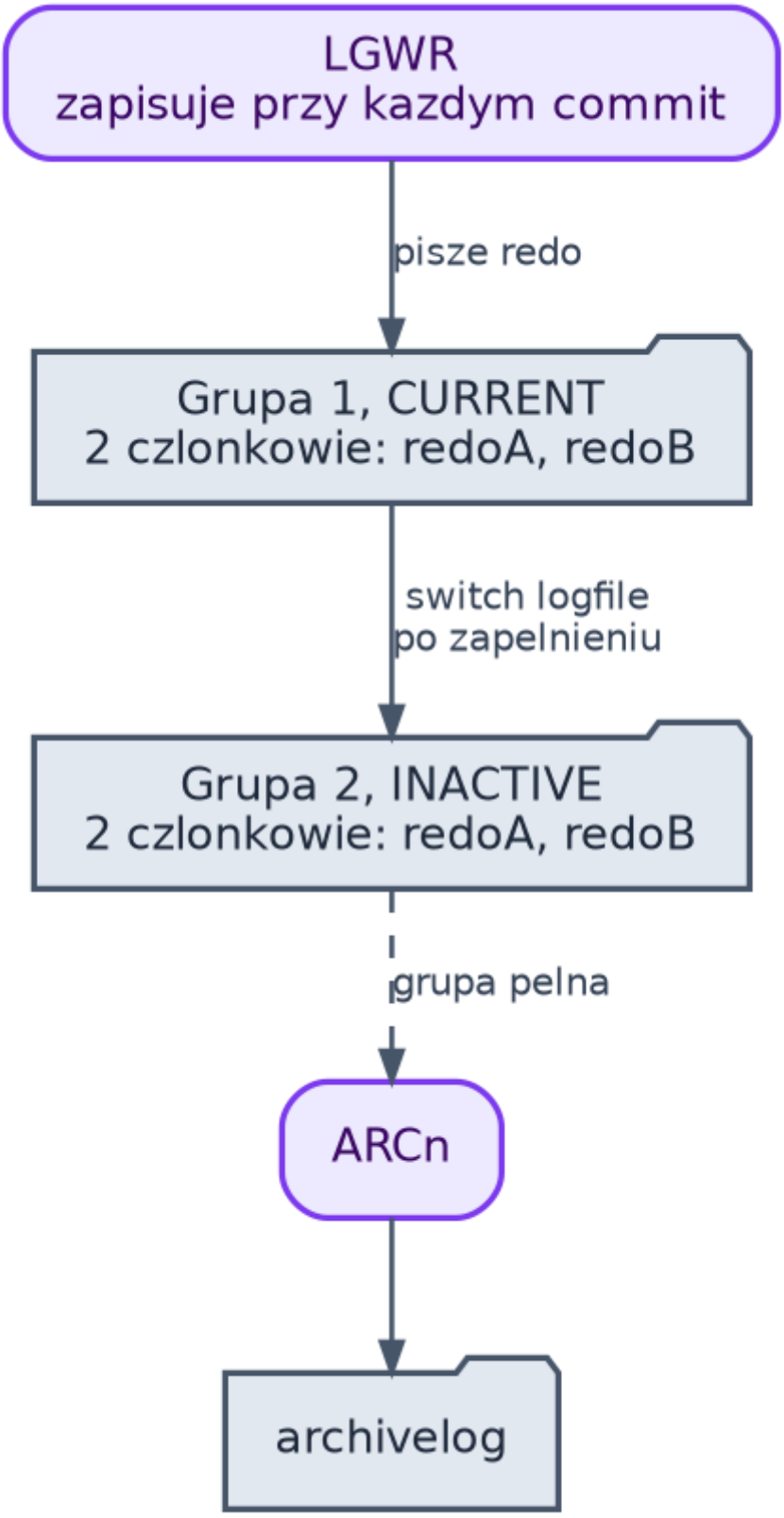
Z życia wzięte

Z ŻYCIA WZIĘTE

Nocą, przy zamknięciu miesiąca, Tomek zauważa w alert logu WMSPRD przełączenia redo logów co dwie, trzy minuty. Rano Marek dopytuje, dlaczego WMS działał wolniej niż zwykle akurat wtedy, kiedy skanery logowały największy ruch zmiany. Po sprawdzeniu `v$log_history` zespół dokłada większe grupy logów, żeby przełączenia wróciły do rozsądnych kilkunastu minut.

Czym jest grupa redo logów i jej członek

Pojęcie: grupa redo logów. Zbiór jednego lub więcej identycznych plików, do których Oracle zapisuje w kółko strumień zmian zachodzących w bazie — każda transakcja, zanim zostanie potwierdzona, musi mieć swój zapis w aktualnie używanej grupie redo logów. Baza wymaga minimum dwóch grup, żeby móc przełączać się między nimi w kółko, w praktyce najczęściej zakłada się pięć grup jako wartość wyjściową.



LGWR pisze do bieżącej, aktywnej grupy redo logów, po jej zapełnieniu baza przełącza się na kolejną, a ARCn kopiuje pełną grupę do archivelogu (kliknięcie otwiera pełny rozmiar)

> **Pojęcie: członek grupy.** Pojedynczy plik fizyczny wchodzący w skład grupy — jeśli grupa ma więcej niż jednego członka, Oracle pisze do wszystkich naraz (multipleksacja), więc utrata jednego pliku dzięki drugiej kopii nie oznacza utraty danych. Zalecana praktyka to trzymanie każdego członka danej grupy na osobnym dysku fizycznym, żeby awaria jednego dysku nie zabrała całej grupy naraz.

Zalecany rozmiar pojedynczego redo loga to przedział 512 MB – 2 GB — górna granica nie jest sztywna i bywa przekraczana w bazach o dużym ruchu transakcyjnym, ale poniżej 512 MB rzadko ma to sens, bo prowadzi do zbyt częstych przełączeń.

Częste błędy

Włączenie NOLOGGING bez świadomości konsekwencji. Tryb `NOLOGGING` na operacji (na przykład `CREATE TABLE ... NOLOGGING` albo `INSERT /*+ APPEND */`) pomija zapis pełnego redo dla tej operacji, co ją przyspiesza, ale oznacza, że w razie odtwarzania bazy z backupu sprzed tej operacji dane wstawione w trybie NOLOGGING nie da się odzyskać z redo logów — trzeba je wstawić ponownie albo mieć backup wykonany już po tej operacji. Wymuszenie pełnego logowania mimo `NOLOGGING` na poziomie obiektów opisuje strona Tryb ARCHIVELOG i logowanie bazy, sekcja o `FORCE LOGGING`.

Zbyt małe redo logi przy dużym ruchu transakcyjnym. Prowadzi do przełączeń częstszych niż raz na kilkanaście minut, co obciąża bazę i, przy włączonym archivelog, generuje nadmiar małych plików archiwalnych.

Usunięcie grupy bez wcześniejszego usunięcia jej plików z dysku. `DROP LOGFILE GROUP` czyści tylko wpis w bazie — pliki `.log` zostają fizycznie na dysku i trzeba je skasować ręcznie, inaczej z czasem zaśmiecają katalog.

Powiązane: Tryb ARCHIVELOG i logowanie bazy · Pliki kontrolne · Backup pełny, różnicowy, przyrostowy, logi

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

Pliki kontrolne

Poziom: operacyjny **Dotyczy: Oracle Database 12c–19c** **Czas: ~15 min**

Bez pliku kontrolnego baza nie wie, z jakich plików się składa ani w jakim jest stanie — jeden zniszczony control file bez multipleksacji potrafi zablokować start całej instancji.

Skróty użyte na tej stronie (3)

Skrót	Co znaczy
RMAN	Recovery Manager: narzędzie Oracle do kopii zapasowych
SQL	Structured Query Language: język zapytań do relacyjnych baz danych
WMS	Warehouse Management System: system zarządzania magazynem

Fiszka

TYLKO ODCZYT

Sprawdzenie lokalizacji plików kontrolnych: v$controlfile

```
SELECT * FROM v$controlfile;
```

Sprawdzamy, ile kopii pliku kontrolnego baza faktycznie widzi i gdzie leżą — pierwsze zapytanie przy diagnozowaniu problemów ze startem instancji albo przy planowaniu przeniesienia plików kontrolnych na inny dysk.

Przykład:

```
STATUS  NAME
------- --------------------------------------------------
        /oracle/db/wmstest/oradata/control/control01.ctl
        /oracle/db/wmstest/redoA/control/control02.ctl
        /oracle/db/wmstest/redoB/control/control03.ctl
```

Ten sam zestaw ścieżek widać też w parametrze instancji:

```
SHOW PARAMETER control_files
```

Przykład:

```
NAME           TYPE        VALUE
-------------- ----------- ------------------------------------------------------
control_files  string      /oracle/db/wmstest/oradata/control/control01.ctl,
                            /oracle/db/wmstest/redoA/control/control02.ctl,
                            /oracle/db/wmstest/redoB/control/control03.ctl
```

show = FortiOS: wypisuje konfigurację, w podstawowej postaci tylko ustawienia różniące się od domyślnych, a z wariantem full-configuration także wartości domyślne

ZMIENIA SYSTEM

Zmiana lokalizacji plików kontrolnych: parametr CONTROL_FILES

```
ALTER SYSTEM SET control_files='SciezkaDoPliku','SciezkaDoPliku','SciezkaDoPliku'
SCOPE=SPFILE;
```

Ustawiamy nowe ścieżki plików kontrolnych w SPFILE — konieczne przy zmianie ich lokalizacji na inny dysk albo po uszkodzeniu jednej z kopii. Zmiana zapisuje się tylko do parametrów startowych, a baza wciąż działa na starych ścieżkach aż do restartu, więc kolejnym krokiem jest zamknięcie instancji, fizyczne skopiowanie plików na nowe ścieżki systemowym poleceniem, i dopiero wtedy ponowne uruchomienie.

Przykład:

```
ALTER SYSTEM SET control_files=
  '/oracle/db/wmstest/oradata/control/control01.ctl',
  '/oracle/db/wmstest/redoA/control/control02.ctl',
  '/oracle/db/wmstest/redoB/control/control03.ctl'
SCOPE=SPFILE;
```

```
System altered.
```

Zamykamy instancję, kopiujemy plik(i) kontrolne systemowo pod nowe ścieżki z parametru, po czym uruchamiamy bazę ponownie:

```
STARTUP;
```

□□ **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

□□ ZMIENIA SYSTEM

Backup pliku kontrolnego do trace: ALTER DATABASE BACKUP CONTROLFILE TO TRACE

```
ALTER DATABASE BACKUP CONTROLFILE TO TRACE;
```

Zrzucamy zawartość bieżącego pliku kontrolnego do czytelnego skryptu SQL w katalogu trace instancji — plik ten zawiera gotowe polecenie `CREATE CONTROLFILE`, które w razie utraty wszystkich kopii pozwala odtworzyć strukturę bazy ręcznie, bez sięgania po backup RMAN. Wykonujemy to okresowo albo tuż przed planowaną zmianą struktury plików bazy.

Przykład:

```
Database altered.
```

Wygenerowany plik trafia do katalogu wskazanego parametrem `diagnostic_dest`, zwykle pod ścieżką podobną do `.../diag/rdbms/NazwaBazy/NazwaInstancji/trace/`.

□□ **SQL** = Structured Query Language: język zapytań do relacyjnych baz danych · **RMAN** = Recovery Manager: narzędzie Oracle do kopii zapasowych

□□ ZMIENIA SYSTEM

Odtworzenie pliku kontrolnego z backupu RMAN

```
RESTORE CONTROLFILE FROM AUTOBACKUP;
```

Kiedy tracimy wszystkie kopie pliku kontrolnego naraz, odtwarzamy go z automatycznego backupu RMAN — instancja musi być w stanie `NOMOUNT`, bo bez pliku kontrolnego nie da się jej zamontować. Pełny przebieg odtworzenia bazy razem z plikiem kontrolnym z backupu na innym hoście pokazuje strona Odtworzenie z backupu Networkera na innym hoście.

Przykład:

```
Starting restore at 25-SEP-26
using target database control file instead of recovery catalog
```

```
allocated channel: ORA_DISK_1
channel ORA_DISK_1: restoring control file
channel ORA_DISK_1: restore complete, elapsed time: 00:00:04
output file name=/oracle/db/wmsprd/oradata/control01.ctl
output file name=/oracle/db/wmsprd/redoA/control02.ctl
Finished restore at 25-SEP-26
```

Po odtworzeniu montujemy bazę i przechodzimy do odtworzenia właściwych danych:

```
ALTER DATABASE MOUNT;
```

RMAN = Recovery Manager: narzędzie Oracle do kopii zapasowych

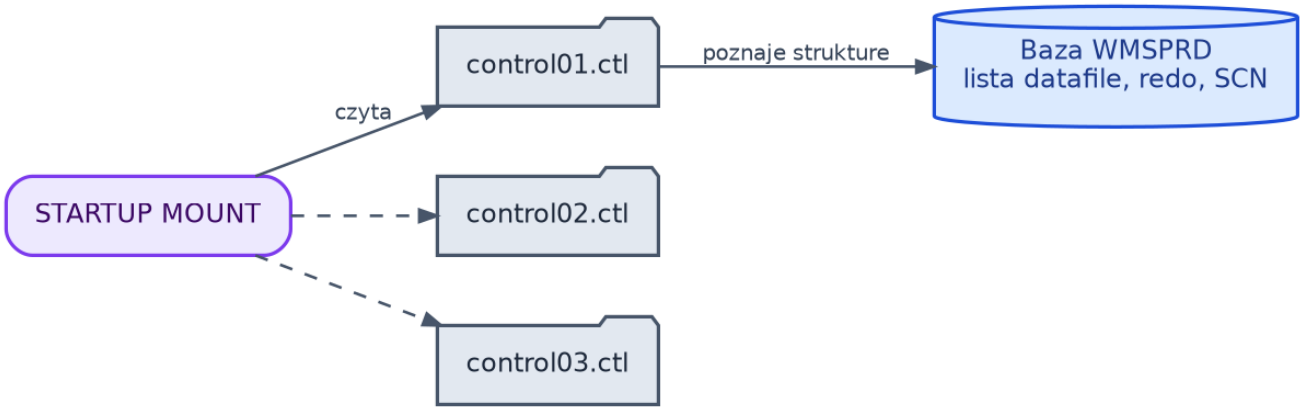
Z życia wzięte

Z ŻYCIA WZIĘTE

Po awarii jednego z dysków na ora01 Tomek sprawdza `v$controlfile` i odkrywa, że wszystkie trzy kopie pliku kontrolnego WMSPRD leżały na tym samym woluminie. Instancja i tak wstaje, bo dysk akurat przeżył, ale Marek słyszy jasno, że przy innym scenariuszu WMS stałby do czasu pełnego odtworzenia z backupu. Zespół rozkłada kopie plików kontrolnych na trzy osobne dyski fizyczne, zanim coś podobnego zdarzy się po raz drugi.

Czym jest plik kontrolny

Pojęcie: plik kontrolny (control file). Mały plik binarny, w którym Oracle trzyma metadane o fizycznej strukturze bazy — nazwę bazy, listę wszystkich datafile i redo logów wraz ze ścieżkami, aktualny numer SCN, informacje o backupach zarejestrowanych przez RMAN oraz historię przełączeń redo logów. Instancja czyta plik kontrolny już na etapie `STARTUP MOUNT`, zanim jeszcze otworzy same pliki danych — bez niego baza nie wie, czego w ogóle szukać na dysku.



Startup mount czyta pierwszą dostępną kopię pliku kontrolnego z trzech multipleksowanych, żeby poznać strukturę bazy WMSPRD (kliknięcie otwiera pełny rozmiar)

> **Pojęcie: multipleksacja plików kontrolnych.** Trzymanie kilku identycznych kopii pliku kontrolnego na różnych dyskach fizycznych, uaktualnianych przez Oracle jednocześnie przy każdej zmianie. Podobnie jak przy członkach grupy redo logów, chodzi o to, żeby awaria jednego dysku nie zabrała jedynej kopii tego, bez czego baza nie wystartuje. Ścieżki wszystkich kopii wskazuje parametr `CONTROL_FILES`.

Częste błędy

Zmiana parametru CONTROL_FILES bez wcześniejszego skopiowania plików na nowe ścieżki. Baza po restarcie szuka plików pod nowymi ścieżkami z SPFILE — jeśli fizycznie ich tam nie ma, instancja nie wystartuje nawet do `NOMOUNT`.

Brak multipleksacji na produkcji. Jedna kopia pliku kontrolnego na jednym dysku oznacza, że awaria tego jednego dysku wymaga pełnego odtworzenia z backupu zamiast prostego przełączenia na drugą kopię — koszt utrzymania kilku kopii jest znikomy w porównaniu z ryzykiem.

Poleganie wyłącznie na backupie do trace jako jedynym zabezpieczeniu. Skrypt z `BACKUP CONTROLFILE TO TRACE` odtwarza strukturę bazy, ale nie zawiera samych danych — jest uzupełnieniem backupu RMAN, a nie jego zamiennikiem.

Powiązane: Redo logi: grupy, przełączanie, rozmiar · RMAN: backup, walidacja, czyszczenie archiwów · Odtworzenie z backupu Networkera na innym hoście

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