Procedury

Start i stop instancji, utworzenie nowej bazy, upgrade silnika i migracja na inny serwer.

- Uruchamianie i zatrzymywanie instancji
- Utworzenie nowej instancji bazy danych w wersji 19c i wyższej
- Upgrade silnika bazy danych oraz klienta
- Migracja bazy danych na inny serwer

Uruchamianie i zatrzymywanie instancji

Typ: procedura **Dotyczy: Oracle Database 12c–19c** **Czas: ~10 min**

Efekt: instancja zatrzymana lub uruchomiona w kontrolowany sposób, z listenerem sprawdzonym po drodze, tak żeby aplikacje klienckie nie łapały błędów połączenia zaraz po starcie.

Skróty użyte na tej stronie (4)

Skrót	Co znaczy
RAM	Random Access Memory: pamięć operacyjna
SGA	System Global Area: wspólna pamięć instancji Oracle
SQL	Structured Query Language: język zapytań do relacyjnych baz danych
TNS	Transparent Network Substrate: warstwa sieciowa Oracle, przez którą klient znajduje i łączy się z listenerem bazy

Zanim zaczniemy

> **Z ŻYCIA WZIĘTE**
>
> Przed planowaną aktualizacją systemu operacyjnego na ora01 Tomek chce zamknąć instancję WMSPRD w kontrolowany sposób, zamiast czekać na restart serwera. Robi `SHUTDOWN IMMEDIATE`, ale po dziesięciu minutach polecenie wciąż wisi, bo jedna z sesji wsadowych ma otwartą długą transakcję. Marek potwierdza, że to zadanie nocne, więc zespół czeka jeszcze chwilę zamiast sięgać od razu po `SHUTDOWN ABORT`.

Logujemy się na serwer bazy na konto systemowe `oracle` i łączymy się z instancją lokalnie, z uprawnieniami `sysdba`:

```
sudo su - oracle
sqlplus / as sysdba
```

Dla świętego spokoju, przed restartem serwera, na którym stoi instancja Oracle, zawsze wyłączamy instancję jawnie przed restartem systemu operacyjnego — zamiast liczyć na to, że system sam grzecznie zamknie bazę w swojej sekwencji wyłączania.

Zamknięcie instancji: tryby SHUTDOWN

> **Pojęcie: zamknięcie instancji.** Zamknięcie bazy przechodzi przez kilka etapów w drugą stronę niż uruchomienie — najpierw zamyka bazę (odłącza pliki danych), potem odmontowuje (`dismount`, traci powiązanie z plikem kontrolnym), na końcu zatrzymuje samą instancję (usuwa struktury pamięci SGA/PGA z RAM). To, jak dokładnie Oracle traktuje sesje i transakcje otwarte w tym momencie, zależy od wybranego trybu.

Tryb	Co robi	Kiedy sięgamy
`SHUTDOWN NORMAL`	czeka, aż wszyscy użytkownicy sami się rozłączą, nowych połączeń nie przyjmuje	prawie nigdy w praktyce — może czekać w nieskończoność
`SHUTDOWN TRANSACTIONAL`	czeka na zakończenie trwających transakcji, ale nie pozwala zaczynać nowych, i od razu rozłącza sesje bezczynne	planowe wyłączenie z minimalnym ryzykiem przerwania czyjejś transakcji
`SHUTDOWN IMMEDIATE`	wycofuje (`rollback`) wszystkie niezacommitowane transakcje i rozłącza wszystkich natychmiast	standardowy wybór do rutynowego, kontrolowanego wyłączenia — najczęściej używany tryb
`SHUTDOWN ABORT`	zatrzymuje instancję natychmiast, bez porządków — jak nagłe wyłączenie zasilania	tylko gdy `IMMEDIATE` wisi i nie kończy się w rozsądnym czasie, albo instancja jest w stanie, z którego nie da się jej normalnie zamknąć

Po `SHUTDOWN ABORT` kolejny `STARTUP` wymusza pełne odzyskiwanie z redo logów (instance recovery) — baza wciąż wstaje poprawnie, ale trwa to dłużej niż po zamknięciu w trybie `IMMEDIATE`, gdzie dane są już spójnie zrzucone na dysk.

Zamknięcie instancji w trybie IMMEDIATE

```
SHUTDOWN IMMEDIATE;
```

Wyłączamy bazę i instancję w jednym poleceniu — proces obejmuje zamknięcie bazy, odmontowanie, a potem zatrzymanie instancji i może zająć nawet do kilku minut przy dużej liczbie

aktywnych transakcji do wycofania.

Przykład:

```
Database closed.
Database dismounted.
ORACLE instance shut down.
```

Awaryjne zamknięcie instancji: ABORT

```
SHUTDOWN ABORT;
```

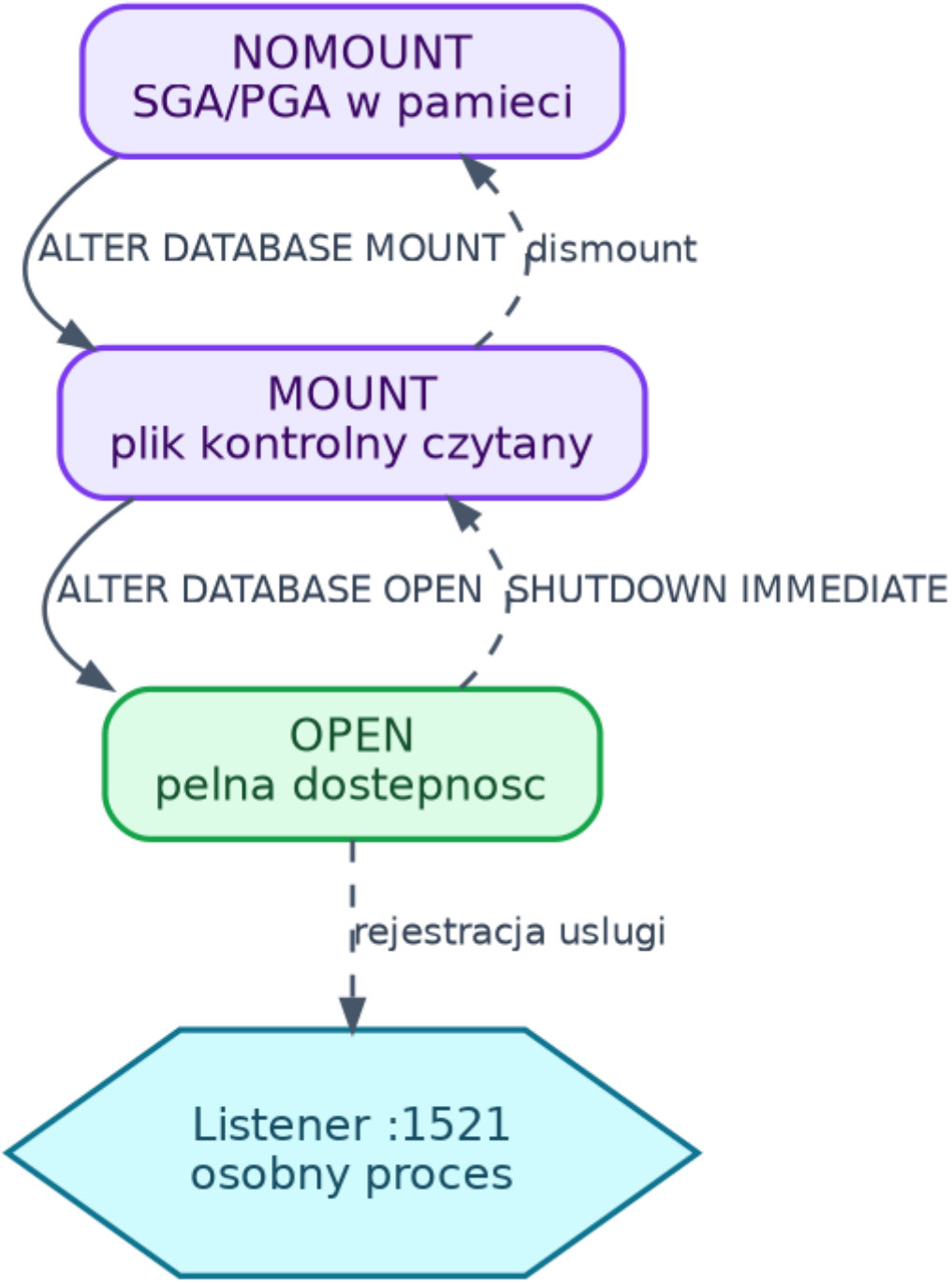
Sięgamy po ten tryb dopiero, kiedy `SHUTDOWN IMMEDIATE` wisi bez końca — na przykład przy bardzo długiej transakcji do wycofania albo zawieszonym procesie w tle.

Przykład:

```
ORACLE instance shut down.
```

Uruchomienie instancji: etapy NOMOUNT / MOUNT / OPEN

> **Pojęcie: uruchomienie instancji.** Uruchomienie przechodzi przez trzy etapy w kolejności rosnącej dostępności — każdy kolejny etap wymaga poprzedniego i odsłania coraz więcej funkcji bazy.



Instancja przechodzi kolejno przez NOMOUNT, MOUNT i OPEN, a zamknięcie w trybie IMMEDIATE cofa ją tymi samymi etapami w drugą stronę (kliknięcie otwiera pełny rozmiar)

Etap	Co jest dostępne	Do czego służy
`NOMOUNT`	tylko struktury pamięci instancji (SGA/PGA), żadnych plików bazy jeszcze nieotwartych	odtwarzanie pliku kontrolnego z backupu, tworzenie nowej bazy

Etap	Co jest dostępne	Do czego służy
`MOUNT`	plik kontrolny odczytany, instancja zna strukturę bazy (listę plików), ale same pliki danych zamknięte	zmiana trybu ARCHIVELOG, odtwarzanie plików danych, operacje na plikach fizycznych
`OPEN`	pełna dostępność, wszystkie pliki danych otwarte, aplikacje mogą się łączyć i pracować	normalna praca bazy

Uruchomienie instancji do pełnej dostępności: STARTUP

```
STARTUP;
```

Włączenie, zamontowanie oraz otwarcie bazy danych w ramach jednego polecenia — domyślne zachowanie `STARTUP` bez dodatkowych parametrów, kiedy chcemy od razu przejść do pełnej dostępności.

Przykład:

```
ORACLE instance started.

Total System Global Area 7180646608 bytes
Fixed Size                  8956112 bytes
Variable Size            1191182336 bytes
Database Buffers         5972688896 bytes
Redo Buffers                7819264 bytes
Database mounted.
Database opened.
```

Zatrzymanie na etapie MOUNT: STARTUP MOUNT

```
STARTUP MOUNT;
```

Uruchamiamy instancję i montujemy bazę, celowo nie otwierając jej do pracy — potrzebne przed operacjami wymagającymi stanu `MOUNT`, na przykład zmianą trybu ARCHIVELOG opisaną na stronie Tryb ARCHIVELOG i logowanie bazy.

Przykład:

```
ORACLE instance started.
Database mounted.
```

Otwarcie zamontowanej bazy: ALTER DATABASE OPEN

```
ALTER DATABASE OPEN;
```

Kiedy instancja jest już zamontowana (`MOUNT`), a chcemy dopiero teraz otworzyć ją do pracy — na przykład po zakończonej operacji, która wymagała stanu `MOUNT`.

Przykład:

```
Database altered.
```

Kolejność przy listenerze

Sam start instancji nie uruchamia listenera TNS — to osobny proces, o który trzeba zadbać po kolei. Jeśli po starcie bazy aplikacje klienckie (SQL Developer, aplikacja produkcyjna) nadal nie mogą się połączyć „z zewnątrz", sprawdzamy stan listenera:

```
lsnrctl status
```

Przykład, kiedy listener działa poprawnie i widzi usługę bazy:

```
LSNRCTL for Linux: Version 19.0.0.0.0 - Production
Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=ora01)(PORT=1521)))
Services Summary...
Service "wmsprd" has 1 instance(s).
  Instance "wmsprd", status READY, has 1 handler(s) for this service.
The command completed successfully
```

Jeżeli zwrotka jest inna — usługa bazy nie figuruje na liście — uruchamiamy listener jawnie:

```
lsnrctl start
```

Weryfikacja stanu

Sprawdzamy, czy instancja jest otwarta i w jakim jest stanie:

```
SELECT instance_name, status, database_status FROM v$instance;
```

Przykład:

```
INSTANCE_NAME    STATUS       DATABASE_STATUS
---------------- ------------ -----------------
wmsprd           OPEN         ACTIVE
```

Alert log jako pierwsze miejsce do zajrzenia

> **Pojęcie: alert log.** Główny plik dziennika instancji, w którym Oracle zapisuje każdy start, zamknięcie, zmianę parametrów, przełączenie redo loga i każdy poważniejszy błąd (`ORA-`). Jeśli `STARTUP` albo `SHUTDOWN` kończy się czymś niespodziewanym, albo instancja po prostu „nie chce wstać", alert log jest pierwszym miejscem do sprawdzenia — zanim zaczniemy szukać dalej, zwykle jest tam już jasno napisane, na czym stanęło uruchamianie. Plik leży pod ścieżką wskazaną parametrem `diagnostic_dest`, zwykle w katalogu `.../diag/rdbms/NazwaBazy/NazwaInstancji/trace/alert_NazwaInstancji.log`.

Częste błędy

Rutynowe używanie `SHUTDOWN ABORT` zamiast `IMMEDIATE`. `ABORT` jest szybszy, ale zostawia bazę w stanie wymagającym pełnego odzyskiwania przy następnym starcie — dla planowych wyłączeń zawsze zaczynamy od `IMMEDIATE` i sięgamy po `ABORT` dopiero, gdy `IMMEDIATE` faktycznie się zawiesza.

Zapomnienie o listenerze po starcie bazy. Baza działa i jest otwarta, a mimo to aplikacje nie mogą się połączyć — sam listener czasem trzeba uruchomić osobno, restart instancji go nie dotyka.

Restart serwera bez wcześniejszego zamknięcia instancji. System operacyjny wymusza wtedy na Oracle odpowiednik `SHUTDOWN ABORT` w trakcie swojej sekwencji zamykania — bezpieczniej jest samemu zamknąć bazę w trybie `IMMEDIATE`, zanim wydamy polecenie restartu systemu.

Powiązane: Tryb ARCHIVELOG i logowanie bazy · Pliki kontrolne · Utworzenie nowej instancji bazy danych w wersji 19c i wyższej

Źródła: notatki własne, dokumentacja Oracle Database (Administrator's Guide — Starting Up and Shutting Down)

Utworzenie nowej instancji bazy danych w wersji 19c i wyższej

Typ: procedura **Dotyczy: Oracle Database 19c+ , DBCA w trybie silent**

Czas: ~1–2 godz.

Efekt: nowa instancja bazy danych utworzona, zarejestrowana w `oratab` i w listenerze, gotowa do przyjęcia pierwszego schematu.

Skróty użyte na tej stronie (3)

Skrót	Co znaczy
DBCA	Database Configuration Assistant: graficzny albo tekstowy kreator tworzący nową instancję i bazę Oracle
PGA	Program Global Area: prywatna pamięć procesu serwera Oracle
SGA	System Global Area: wspólna pamięć instancji Oracle

Zanim zaczniemy

> **Z ŻYCIA WZIĘTE**
>
> Ola zgłasza, że potrzebuje osobnej instancji testowej WMSTEST, żeby nie testować nowej wersji integracji na produkcyjnym WMSPRD. Tomek uruchamia DBCA w trybie silent na ora01, podając rozmiary SGA i PGA policzone wcześniej z RAM-u serwera. Po zakończeniu tworzenia zapomina dopisać wpis do `/etc/oratab`, więc po najbliższym restarcie serwera baza WMSTEST się nie podnosi, dopóki ktoś nie doda brakującej linii ręcznie.

Software Oracle Database (`ORACLE_HOME`) musi być już zainstalowany na serwerze, zanim przystąpimy do tworzenia samej instancji — instalacja binariów i utworzenie bazy to dwa osobne kroki. Przygotowujemy sobie z wyprzedzeniem:

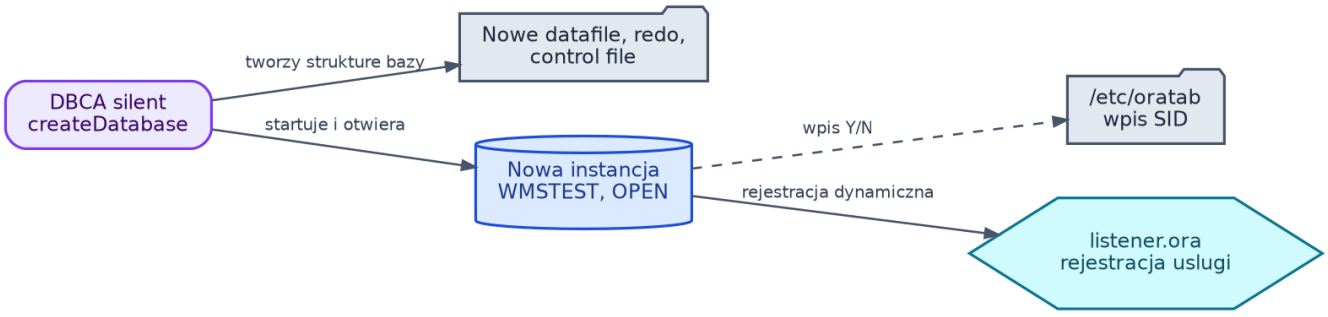
Wymaganie	Uwaga
Zainstalowany `ORACLE_HOME` w wersji 19c lub wyższej	ta procedura zakłada gotowy software, nie instalację silnika
Nazwa SID i nazwa globalna bazy	zwykle takie same dla bazy niekonteneryzowanej
Wolumen pod dane, zgodny z konwencją `vgDB` / `lvOracle`	patrz strona Instalacja Rocky Linux z własnym układem VG/LV
Zaplanowane rozmiary SGA/PGA	patrz notatki o `sga_target` / `pga_aggregate_target` na stronie architektury pamięci instancji
Wpis w `/etc/oratab` gotowy do dodania	rejestrujemy nową instancję, żeby skrypty startowe ją widziały

Uważaj: DBCA w trybie silent domyślnie tworzy bazę w trybie `NOARCHIVELOG` — jeśli baza ma iść na produkcję, włączenie trybu ARCHIVELOG (strona Tryb ARCHIVELOG i logowanie bazy) robimy zaraz po utworzeniu, zanim ktokolwiek zacznie w niej pracować.

Przebieg

1. Uruchomienie DBCA w trybie silent

Logujemy się na konto `oracle`, ustawiamy środowisko na docelowy `ORACLE_HOME` i uruchamiamy DBCA bez interfejsu graficznego, podając parametry bazy wprost w linii poleceń:



DBCA tworzy strukturę nowej bazy i od razu startuje oraz otwiera instancję, która dopiero potem trafia do oratab i rejestruje się w listenerze (kliknięcie otwiera pełny rozmiar)

```
dbca -silent -createDatabase \
  -templateName General_Purpose.dbc \
  -gdbname NazwaBazy \
```

```
  -sid NazwaSID \
  -sysPassword HasloSys \
  -systemPassword HasloSystem \
  -createAsContainerDatabase false \
  -characterSet AL32UTF8 \
  -memoryPercentage 60 \
  -emConfiguration NONE \
  -datafileDestination SciezkaDoKatalogu \
  -recoveryAreaDestination SciezkaDoKatalogu \
  -storageType FS
```

Przykład wywołania:

```
dbca -silent -createDatabase \
  -templateName General_Purpose.dbc \
  -gdbname wmstest \
  -sid wmstest \
  -sysPassword ******** \
  -systemPassword ******** \
  -createAsContainerDatabase false \
  -characterSet AL32UTF8 \
  -memoryPercentage 60 \
  -emConfiguration NONE \
  -datafileDestination /oracle/db/wmstest/oradata \
  -recoveryAreaDestination /oracle/db/wmstest/fra \
  -storageType FS
```

Przykład:

```
Prepare for db operation
7% complete
Copying database files
29% complete
Creating and starting Oracle instance
42% complete
Completing Database Creation
78% complete
Executing Post Configuration Actions
100% complete
Database creation complete. For details check the logfiles at:
```

```
 /oracle/db/wmstest/cfgtoollogs/dbca/wmstest.
Database Information:
Global Database Name:wmstest
System Identifier(SID):wmstest
Look at the log file "/oracle/db/wmstest/cfgtoollogs/dbca/wmstest/wmstest.log" for further
details.
```

Cały proces trwa zwykle 30–60 minut, zależnie od szablonu i rozmiaru bazowych plików danych.

2. Alternatywa: utworzenie bazy poleceniem CREATE DATABASE

Kiedy potrzebujemy pełnej kontroli nad każdym plikiem i parametrem (albo automatyzujemy tworzenie skryptem), zamiast DBCA używamy klasycznego `CREATE DATABASE` na już uruchomionej w trybie `NOMOUNT` instancji z minimalnym plikiem `init.ora`:

```
sqlplus / as sysdba
```

```
STARTUP NOMOUNT PFILE='SciezkaDoPliku';
```

```
CREATE DATABASE NazwaBazy
  USER SYS IDENTIFIED BY HasloSys
  USER SYSTEM IDENTIFIED BY HasloSystem
  LOGFILE GROUP 1 ('SciezkaDoPliku') SIZE 512M,
          GROUP 2 ('SciezkaDoPliku') SIZE 512M
  MAXLOGFILES 16
  DATAFILE 'SciezkaDoPliku' SIZE 800M AUTOEXTEND ON NEXT 100M MAXSIZE UNLIMITED
  SYSAUX DATAFILE 'SciezkaDoPliku' SIZE 600M
  DEFAULT TABLESPACE users DATAFILE 'SciezkaDoPliku' SIZE 200M
  DEFAULT TEMPORARY TABLESPACE temp TEMPFILE 'SciezkaDoPliku' SIZE 200M
  UNDO TABLESPACE undotbs1 DATAFILE 'SciezkaDoPliku' SIZE 500M
  CHARACTER SET AL32UTF8;
```

Po utworzeniu bazy tą metodą uruchamiamy jeszcze skrypty budujące słownik danych i pakiety standardowe — dopiero po nich baza jest kompletna:

```
@?/rdbms/admin/catalog.sql
@?/rdbms/admin/catproc.sql
```

Metoda przez `CREATE DATABASE` jest pracochłonniejsza i rzadziej używana w praktyce niż DBCA — sięgamy po nią głównie przy tworzeniu baz z automatyzacji (Ansible, skrypty) albo kiedy potrzebujemy nietypowego układu plików, którego szablon DBCA nie przewiduje.

3. Rejestracja w /etc/oratab

```
echo "NazwaSID:SciezkaDoORACLE_HOME:N" | sudo tee -a /etc/oratab
```

> **Pojęcie: oratab.** Plik rejestrujący wszystkie instancje Oracle skonfigurowane na danym serwerze — trzecia kolumna (`Y` / `N`) mówi skryptom startowym (`dbstart` / `dbshut`), czy dana instancja ma wystartować automatycznie razem z systemem.

Przykład:

```
wmstest:/oracle/db/19000:N
```

DBCA zwykle dopisuje wpis do `oratab` sam — warto to jednak zawsze zweryfikować ręcznie, zwłaszcza po tworzeniu bazy metodą `CREATE DATABASE`.

4. Rejestracja w listenerze

Sprawdzamy, czy listener widzi nową instancję (rejestracja dynamiczna działa automatycznie po starcie instancji, o ile `local_listener` jest poprawnie ustawiony):

```
SHOW PARAMETER local_listener
```

Jeśli listener nasłuchuje na niestandardowym porcie albo adresie, dopisujemy wpis statyczny do `listener.ora`:

```
SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (GLOBAL_DBNAME = NazwaBazy)
      (ORACLE_HOME = SciezkaDoORACLE_HOME)
      (SID_NAME = NazwaSID)
    )
  )
```

Przeładowujemy listener, żeby podjął zmianę:

```
lsnrctl reload
```

Weryfikacja

Sprawdzamy stan instancji:

```
SELECT instance_name, status, database_status FROM v$instance;
```

Przykład:

```
INSTANCE_NAME    STATUS       DATABASE_STATUS
---------------- ------------ -----------------
wmstest           OPEN         ACTIVE
```

Sprawdzamy, że listener widzi nową usługę:

```
lsnrctl status
```

Przykład:

```
Services Summary...
Service "wmstest" has 1 instance(s).
  Instance "wmstest", status READY, has 1 handler(s) for this service.
```

Sprawdzamy, że wszystkie komponenty słownika danych są w stanie `VALID` — patrz fiszki walidacji na stronie Upgrade silnika bazy danych oraz klienta.

Wycofanie zmian

Kiedy baza została utworzona błędnie (zła nazwa, zły character set, zły układ plików) i jeszcze nikt jej nie używa, najprościej jest usunąć ją całą i zacząć od nowa, zamiast poprawiać ustawienia po fakcie:

```
dbca -silent -deleteDatabase -sourceDB NazwaSID
```

Usuwamy też wpis z `oratab` i ewentualny statyczny wpis z `listener.ora`, jeśli zdążyliśmy go dodać.

Gdy coś nie zadziała

Objaw	Przyczyna	Naprawa
DBCA kończy się błędem o braku miejsca	wolumen pod `datafileDestination` za mały na wybrany szablon	sprawdzamy `df -h` przed startem, poprawiamy rozmiar wolumenu albo wybieramy mniejszy szablon
`ORA-01034` po starcie instancji	zmienna `ORACLE_SID` w sesji nie zgadza się z nazwą utworzonej instancji	ustawiamy poprawnie `. oraenv` przed połączeniem `sqlplus`
Listener nie widzi nowej usługi	rejestracja dynamiczna nie zdążyła się jeszcze wykonać albo `local_listener` wskazuje zły adres	czekamy do minuty na rejestrację dynamiczną albo dodajemy wpis statyczny do `listener.ora` i `lsnrctl reload`
`CREATE DATABASE` kończy się błędem o brakujących uprawnieniach do katalogów	katalogi docelowe nie istnieją albo nie należą do użytkownika `oracle`	tworzymy katalogi i nadajemy właściciela `oracle:oinstall` przed ponowną próbą

Powiązane: Uruchamianie i zatrzymywanie instancji · Tryb ARCHIVELOG i logowanie bazy · Upgrade silnika bazy danych oraz klienta · Instalacja Rocky Linux z własnym układem VG/LV

Źródła: notatki własne, dokumentacja Oracle Database (Database Installation Guide — DBCA Silent Mode)

Upgrade silnika bazy danych oraz klienta

Typ: procedura **Dotyczy: Oracle Database, upgrade do 19c** **Czas: ~3–6 godz.**

Efekt: silnik bazy danych i klient Oracle podniesione do nowej wersji, komponenty słownika danych zwalidowane, `oratab` i zmienne środowiskowe przełączone na nowy `ORACLE_HOME`.

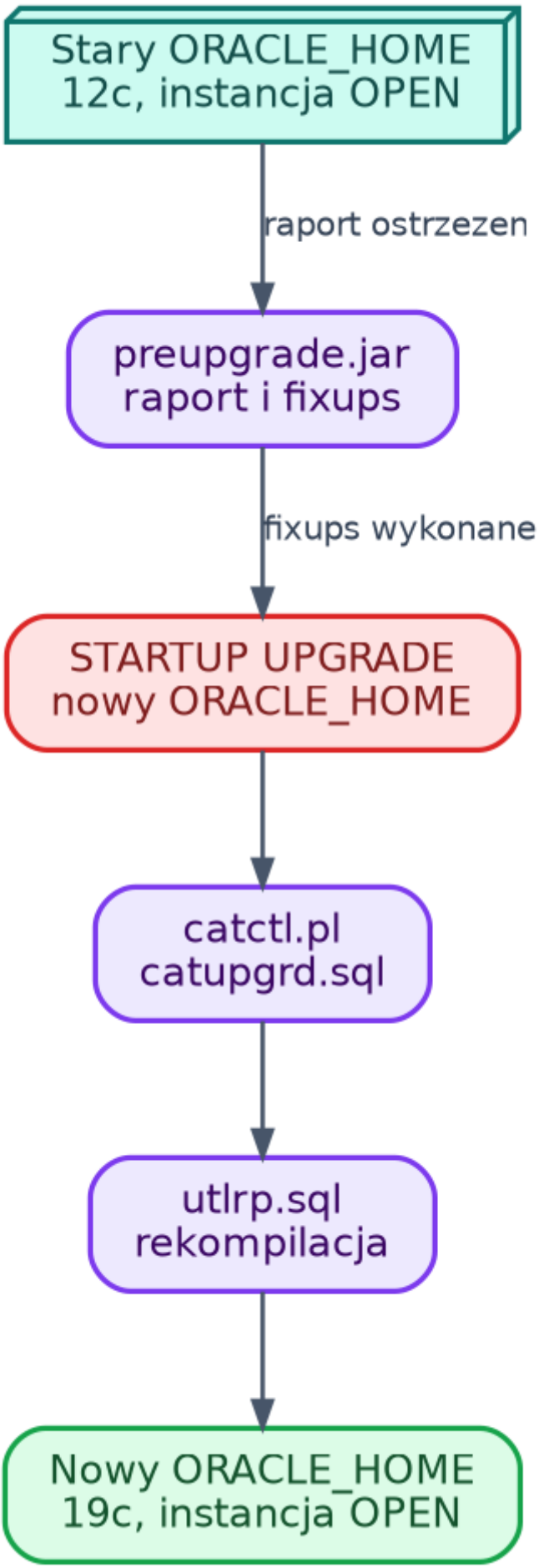
Skróty użyte na tej stronie (2)

Skrót	Co znaczy
GUI	Graphical User Interface: okienka i myszka zamiast komend
WMS	Warehouse Management System: system zarządzania magazynem

Zanim zaczniemy

> **Z ŻYCIA WZIĘTE**
>
> Przed planowanym upgrade'em WMSPRD z wersji 12c na 19c Tomek uruchamia narzędzie preupgrade i dostaje listę ostrzeżeń, których nikt wcześniej nie widział. Marek pyta tylko, ile potrwa okno serwisowe, bo zmiana w Swarzędzu zaczyna się o 6:00 i nie może zastać WMS offline. Zespół planuje upgrade na sobotnią noc, sprawdza `DBA_REGISTRY` przed startem i po `catctl.pl` uruchamia `utlrp.sql`, zanim ktokolwiek zdąży się zalogować w poniedziałek.

Upgrade silnika to operacja z realnym oknem niedostępności — instancja jest zamknięta na czas kompilacji nowego słownika danych. Przed przystąpieniem:


Stary ORACLE_HOME
12c, instancja OPEN
raport ostrzezen
preupgrade.jar
raport i fixups
fixups wykonane
STARTUP UPGRADE
nowy ORACLE_HOME
catctl.pl
catupgrd.sql
utlrp.sql
rekompilacja
Nowy ORACLE_HOME
19c, instancja OPEN

Upgrade przechodzi od starego ORACLE_HOME przez preupgrade i tryb STARTUP UPGRADE do catctl.pl i utlrp.sql, kończąc na nowym ORACLE_HOME z otwartą instancją (kliknięcie otwiera pełny rozmiar)

Wymaganie	Uwaga
Pełny backup bazy przed startem	patrz strona RMAN: backup, walidacja, czyszczenie archiwów — bez świeżego backupu nie zaczynamy
Nowy `ORACLE_HOME` zainstalowany obok starego	upgrade nie nadpisuje starej wersji, tylko instaluje nową równolegle
Sprawdzone `preupgrade` na starej wersji	narzędzie ostrzega o niekompatybilnościach, zanim jeszcze cokolwiek ruszymy
Wszystkie komponenty w `dba_registry` w stanie `VALID` przed startem	upgrade nieważnego komponentu zwykle kończy się błędem w połowie procesu
Okno serwisowe zaakceptowane z właścicielami aplikacji	baza jest offline od zamknięcia do zakończenia `utlrp.sql`

Uważaj: po `STARTUP UPGRADE` baza jest w specjalnym trybie ograniczonym — dostępna tylko dla operacji upgrade'u, nie dla normalnego ruchu aplikacji. Nie przerywamy procesu w połowie bez planu awaryjnego — połowicznie zaktualizowany słownik danych bywa trudniejszy do naprawienia niż dokończenie upgrade'u.

Przebieg

1. Sprawdzenie stanu komponentów przed upgrade'em

Zanim zaczniemy, sprawdzamy, że słownik danych jest w porządku na starej wersji:

```
SELECT comp_name, version, status FROM dba_registry WHERE status <> 'VALID';
```

Zapytanie nie powinno zwrócić żadnego wiersza — jeśli coś jest niepoprawne (`INVALID`), naprawiamy to przed upgrade'em, a nie w jego trakcie.

Przykład:

```
no rows selected
```

Sprawdzamy też wersję i status samej instancji:

```
SELECT instance_name, status, version FROM v$instance;
```

Przykład:

```
INSTANCE_NAME    STATUS       VERSION
---------------- ------------ -----------------
wmsprd           OPEN         12.2.0.1.0
```

2. Uruchomienie preupgrade tool na starej wersji

Uruchamiamy narzędzie preupgrade z katalogu nowego `ORACLE_HOME`, wskazując mu jeszcze działającą starą instancję:

```
$NEW_ORACLE_HOME/jdk/bin/java -jar $NEW_ORACLE_HOME/rdbms/admin/preupgrade.jar TERMINAL TEXT
```

Narzędzie generuje dwa skrypty — `preupgrade_fixups.sql` (uruchamiany na starej wersji przed upgrade'em) i `postupgrade_fixups.sql` (uruchamiany po) — oraz listę ostrzeżeń, które trzeba przejrzeć ręcznie przed dalszymi krokami.

Przykład:

```
Preupgrade Actions
    <Actions Required>
Preupgrade tool was able to fix or update the settings automatically.
No SEVERE errors found.
Report generated at:
/oracle/db/wmsprd/cfgtoollogs/wmsprd/preupgrade/preupgrade.log
```

Uruchamiamy wygenerowany skrypt naprawczy na starej wersji, zanim zamkniemy instancję:

```
@preupgrade_fixups.sql
```

3. Zamknięcie instancji i uruchomienie w trybie UPGRADE na nowym ORACLE_HOME

Zamykamy bazę na starym `ORACLE_HOME`:

```
SHUTDOWN IMMEDIATE;
```

Przełączamy środowisko na nowy `ORACLE_HOME`, zachowując ten sam `ORACLE_SID`, i uruchamiamy instancję w trybie upgrade:

```
export ORACLE_HOME=SciezkaDoNowegoORACLE_HOME
export PATH=$ORACLE_HOME/bin:$PATH
sqlplus / as sysdba
```

```
STARTUP UPGRADE;
```

Przykład:

```
ORACLE instance started.

Total System Global Area 7180646608 bytes
Database mounted.
Database opened.
```

4. Wariant ręczny: dbupgrade (catctl.pl)

Zamiast graficznego DBUA, na serwerach bez GUI wykonujemy upgrade skryptem `dbupgrade`, który wewnętrznie woła `catctl.pl` — narzędzie równoległe, znacznie szybsze niż uruchamianie starych skryptów `catupgrd.sql` wprost:

```
cd $ORACLE_HOME/rdbms/admin
$ORACLE_HOME/perl/bin/perl catctl.pl catupgrd.sql
```

Przykład:

```
Argument list for [catctl.pl]
For Oracle internal use only A = 0
Catcon: ALL catcon-related output will be written to catupgrd_catcon_*.log
Number of Cpus = 8
Serial Phase #: 0  Files: 1     Time: 173s
Serial Phase #: 1  Files: 5     Time: 620s
...
Grand Total Time: 3814s
```

```
--------------------------------------------------
Phases [0-107]         End Time:[2026_09_25 12:47:31]
--------------------------------------------------
Component: 'CATALOG'   [   1657s]
Component: 'CATPROC'   [   980s]
Component: 'JAVAVM'    [   612s]
--------------------------------------------------
Total Upgrade Time: [63.6] mins
--------------------------------------------------
```

Jeśli baza korzysta z modułu OLAP, przed samym upgrade'em, po wybraniu środowiska, włączamy go jawnie:

```
chopt enable olap
```

5. Wariant graficzny: DBUA

Tam, gdzie mamy dostęp do GUI (albo X11 forwarding), zamiast ręcznej sekwencji uruchamiamy asystenta:

```
dbua
```

DBUA przechodzi przez te same kroki co wariant ręczny (backup, preupgrade, `catctl.pl`, `utlrp.sql`) w formie kreatora — wybieramy instancję do upgrade'u, potwierdzamy backup i zatwierdzamy plan, reszta dzieje się automatycznie z paskiem postępu.

6. Rekompilacja obiektów nieprawidłowych: utlrp.sql

Po zakończeniu głównego skryptu upgrade'u rekompilujemy wszystkie obiekty PL/SQL, które w trakcie procesu zostały oznaczone jako nieprawidłowe:

```
@$ORACLE_HOME/rdbms/admin/utlrp.sql
```

Przykład:

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

```
Comp Timestamp Utlrp_Bgn  2026-09-25 13:02:14

DOC> The following utlrp_bgn.sql script will compile invalid
DOC> objects in the database. ...

PL/SQL procedure successfully completed.
```

7. Otwarcie bazy do normalnej pracy

```
STARTUP;
```

Po zamknięciu instancji z trybu upgrade uruchamiamy ją zwyczajnie — od tego momentu baza działa już jako nowa wersja.

8. Aktualizacja klienta Oracle

Na maszynach klienckich (aplikacje, serwery pośredniczące) instalujemy nową wersję Oracle Instant Client albo pełnego klienta, obok starej — analogicznie jak przy silniku, równolegle, nie nadpisując poprzedniej wersji, dopóki nie potwierdzimy, że aplikacje działają poprawnie na nowym kliencie.

9. Przełączenie oratab i zmiennych środowiskowych

Aktualizujemy wpis w `/etc/oratab`, żeby wskazywał nowy `ORACLE_HOME`:

```
sudo sed -i 's|SciezkaDoStaregoORACLE_HOME|SciezkaDoNowegoORACLE_HOME|' /etc/oratab
```

Poprawiamy też profile powłoki (`.bash_profile` konta `oracle`, aliasy `oraenv`), żeby domyślnie ładowały nowy `ORACLE_HOME` przy kolejnych logowaniach.

Weryfikacja

Sprawdzamy wersję instancji po upgrade:

```
SELECT instance_name, status, version FROM v$instance;
```

Przykład:

```
INSTANCE_NAME    STATUS       VERSION
---------------- ------------ -----------------
wmsprd           OPEN         19.22.0.0.0
```

Sprawdzamy, że wszystkie komponenty słownika danych wróciły do stanu `VALID`:

```
SELECT comp_name, version_full, status FROM dba_registry;
```

Przykład:

```
COMP_NAME                     VERSION_FULL     STATUS
----------------------------- ---------------- ---------
Oracle Database Catalog Views  19.22.0.0.0      VALID
Oracle Database Packages       19.22.0.0.0      VALID
JServer JAVA Virtual Machine   19.22.0.0.0      VALID
Oracle XML Database            19.22.0.0.0      VALID
```

Sprawdzamy, że nie zostało nic w stanie innym niż `VALID`:

```
SELECT comp_name, version, status FROM dba_registry WHERE status <> 'VALID';
```

Przykład:

```
no rows selected
```

Wycofanie zmian

Powrót do starej wersji po upgrade'u wymaga odtworzenia bazy z backupu wykonanego przed rozpoczęciem procedury — nie ma prostej ścieżki „downgrade" analogicznej do samego upgrade'u, poza nielicznymi przypadkami przewidzianymi wprost przez Oracle dla konkretnych par wersji. Dlatego backup z kroku pierwszego jest warunkiem koniecznym całej procedury, nie tylko dobrą praktyką — w razie problemu wracamy do niego procedurą opisaną na stronie RMAN: backup, walidacja, czyszczenie archiwów, a stary `ORACLE_HOME` zostawiamy nietknięty właśnie na taką okoliczność.

Gdy coś nie zadziała

Objaw	Przyczyna	Naprawa
`catctl.pl` kończy się błędem w jednej z faz	komponent był już `INVALID` przed upgrade'em albo preupgrade fixups nie zostały uruchomione	sprawdzamy logi `catupgrd*.log` w katalogu uruchomienia, poprawiamy zgłoszony komponent i wznawiamy `catctl.pl` — potrafi kontynuować od miejsca przerwania
Komponenty zostają `INVALID` po `utlrp.sql`	`utlrp.sql` nie zawsze rekompiluje wszystko za jednym razem przy dużej liczbie obiektów	uruchamiamy `utlrp.sql` ponownie — druga runda zwykle domyka resztę
Aplikacja nie łączy się po zmianie klienta	stary klient wciąż wskazywany w `PATH` / `LD_LIBRARY_PATH` aplikacji	poprawiamy zmienne środowiskowe procesu aplikacji, nie tylko konta systemowego DBA
`STARTUP UPGRADE` kończy się błędem `ORA-` przy starcie	preupgrade fixups nie zostały uruchomione na starej wersji przed zamknięciem instancji	wracamy do starego `ORACLE_HOME`, uruchamiamy `preupgrade_fixups.sql`, dopiero potem ponawiamy zamknięcie i upgrade

Powiązane: RMAN: backup, walidacja, czyszczenie archiwów · Uruchamianie i zatrzymywanie instancji · Utworzenie nowej instancji bazy danych w wersji 19c i wyższej

Źródła: notatki własne, dokumentacja Oracle Database (Upgrade Guide)

Migracja bazy danych na inny serwer

Typ: procedura **Dotyczy: Oracle Database 12c–19c, Data Pump, RMAN**

Czas: ~2–6 godz., zależnie od rozmiaru bazy i wariantu

Efekt: baza `WMSPRD` działająca na nowym serwerze (w przykładach `NazwaNowegoSerwera`), z aplikacją przełączoną na nowy adres, stary serwer `ora01` gotowy do wygaszenia po okresie karencji.

Skróty użyte na tej stronie (7)

Skrót	Co znaczy
DBID	Database Identifier: unikalny numer bazy Oracle, przydatny przy rozróżnianiu dwóch baz o tej samej nazwie
DNS	Domain Name System: książka telefoniczna internetu, zamienia nazwy na adresy IP
IP	Internet Protocol: adres komputera w sieci; w elektryce: stopień ochrony obudowy (Ingress Protection)
RMAN	Recovery Manager: narzędzie Oracle do kopii zapasowych
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
SID	w Oracle: System Identifier, nazwa instancji; w Windows: Security Identifier, numer konta
WMS	Warehouse Management System: system zarządzania magazynem

Zanim zaczniemy

Z ŻYCIA WZIĘTE

Stary serwer ora01 zaczyna brakować mocy pod rosnący WMS, więc zespół planuje migrację bazy WMSPRD na nowy serwer. Tomek wybiera wariant RMAN zamiast Data Pump, bo baza zostaje na tej samej wersji silnika, a przestój ma być jak najkrótszy przed startem zmiany magazynowej. Po `RESTORE DATABASE`, otwarciu z `RESETLOGS` i przełączeniu `tnsnames.ora` na aplikacjach Marek o 6:00 nie zauważa żadnej różnicy poza nowym adresem w logach

> skanerów.

Migracja bazy na inny serwer da się zrobić dwiema drogami, zależnie od tego, ile czasu przestoju możemy sobie pozwolić i czy zmienia się wersja silnika po drodze:



Baza źródłowa trafia przez backup RMAN albo zrzut Data Pump na nowy serwer, gdzie po odtworzeniu aplikacja przełącza się na nowy adres (kliknięcie otwiera pełny rozmiar)

Wariant	Kiedy wybieramy	Przestój
Data Pump (export/import logiczny)	migracja między różnymi wersjami silnika, zmiana struktury tablespace, potrzeba wybiórczej migracji schematów	dłuższy — export, transfer pliku, import to trzy sekwencyjne, czasochłonne kroki
RMAN (restore z backupu / duplicate)	ta sama wersja silnika, chcemy zachować identyczną fizyczną strukturę bazy, minimalizujemy przestój	krótszy — odtwarzamy gotowy backup zamiast eksportować dane na nowo

Wymaganie	Uwaga
Nowy serwer `NazwaNowegoSerwera` z zainstalowanym `ORACLE_HOME` w tej samej (lub kompatybilnej) wersji	patrz strona Utworzenie nowej instancji bazy danych w wersji 19c i wyższej
Świeży, pełny backup bazy źródłowej	przy wariancie RMAN warunek konieczny, przy Data Pump zalecany na wszelki wypadek
Ustalone okno przestoju z właścicielami aplikacji	migracja zawsze kończy się przełączeniem connection stringa aplikacji na nowy adres
Sprawdzone różnice w układzie dysków między serwerami	ścieżki plików danych mogą się różnić — trzeba to zmapować w obu wariantach

> (gdzie `NazwaNowegoSerwera` to nowy serwer Oracle, na który przenosimy bazę; Zielona ma dziś jeden serwer Oracle, `ora01`, więc nazwę nowego nadajemy przy zakładaniu maszyny)

Uważaj: dopóki aplikacja nie zostanie przełączona na nowy serwer, stary serwer wciąż przyjmuje ruch — jeśli w międzyczasie ktoś napisze coś do bazy źródłowej po zakończeniu eksportu czy backupu, te dane nie trafią do kopii na nowym serwerze. Okno przestoju musi obejmować też sam moment przełączenia aplikacji, nie tylko transfer danych.

Przebieg — wariant przez Data Pump

1. Eksport bazy źródłowej

Na `ora01` eksportujemy dane pełną bazą albo schemat po schemacie — pełny opis parametrów `expdp` pokazuje strona Data Pump: export i import:

```
expdp directory=EXPDIR dumpfile=wmsprd_export_%U.dmp logfile=wmsprd_export.log full=Y
parallel=4 compression=all
```

Przykład:

```
Starting "SYS"."SYS_EXPORT_FULL_01" with 4 workers
Estimate in progress using BLOCKS method...
Total estimation using BLOCKS method: 112.4 GB
Master table "SYS"."SYS_EXPORT_FULL_01" successfully loaded/unloaded
Dump file set for SYS.SYS_EXPORT_FULL_01 is:
  /oracle/db/wmsprd/datapump/wmsprd_export_01.dmp
Job "SYS"."SYS_EXPORT_FULL_01" successfully completed
```

2. Transfer plików zrzutu na serwer docelowy

Przenosimy wygenerowane pliki `.dmp` na `NazwaNowegoSerwera`, na przykład przez `scp` między kontami systemowymi `oracle`:

```
scp /oracle/db/wmsprd/datapump/wmsprd_export_*.dmp
oracle@NazwaNowegoSerwera:/oracle/db/wmsprd/datapump/
```

3. Utworzenie pustej bazy docelowej i katalogu Data Pump

Na `NazwaNowegoSerwera` zakładamy nową, pustą instancję o docelowej nazwie (procedura na stronie Utworzenie nowej instancji bazy danych w wersji 19c i wyższej) i rejestrujemy katalog Data Pump wskazujący na przeniesione pliki:

```
CREATE DIRECTORY EXPDIR AS '/oracle/db/wmsprd/datapump';
GRANT READ, WRITE ON DIRECTORY EXPDIR TO NazwaUsera;
```

4. Import na serwerze docelowym

```
impdp directory=EXPDIR dumpfile=wmsprd_export_%U.dmp logfile=wmsprd_import.log full=Y
parallel=4
```

Przykład:

```
Starting "SYS"."SYS_IMPORT_FULL_01" with 4 workers
Master table "SYS"."SYS_IMPORT_FULL_01" successfully loaded/unloaded
Processing object type DATABASE_EXPORT/SCHEMA/TABLE/TABLE_DATA
. . imported "WMS_APP"."ZAMOWIENIA"                      41.2 GB 18340210 rows
Job "SYS"."SYS_IMPORT_FULL_01" successfully completed
```

Jeżeli układ tablespace na serwerze docelowym różni się od źródłowego, dodajemy `REMAP_TABLESPACE`, opisane szerzej na stronie Data Pump: export i import.

Przebieg — wariant przez RMAN (restore i przełączenie ID bazy)

Zanim zaczniemy: ten wariant zakłada, że na `NazwaNowegoSerwera` istnieje już pusta instancja o docelowej nazwie SID, i że dysponujemy backupem bazy źródłowej dostępnym z poziomu `NazwaNowegoSerwera` (na dysku współdzielonym albo przez bibliotekę backupu, tak jak w procedurze Odtworzenie z backupu Networkera na innym hoście, jeśli backup leży w systemie backupowym typu NetWorker). Ten wariant jest szybszy niż Data Pump przy dużych bazach, bo RMAN kopiuje bloki fizyczne zamiast przetwarzać dane wiersz po wierszu.

1. Świeży backup bazy źródłowej

```
RMAN> BACKUP DATABASE PLUS ARCHIVELOG;
```

Pełny opis backupu i jego weryfikacji pokazuje strona RMAN: backup, walidacja, czyszczenie archiwów.

2. Uruchomienie instancji docelowej i ustawienie DBID bazy źródłowej

Na `NazwaNowegoSerwera` łączymy się do RMAN i ustawiamy DBID bazy, którą odtwarzamy — bez tego RMAN nie znajdzie właściwego backupu:

```
RMAN> STARTUP NOMOUNT;
RMAN> SET DBID NumerDBID;
```

3. Odtworzenie pliku kontrolnego i montowanie bazy

```
RMAN> RESTORE CONTROLFILE FROM AUTOBACKUP;
RMAN> STARTUP MOUNT;
```

4. Mapowanie ścieżek plików danych, jeśli różnią się między serwerami

Sprawdzamy ścieżki na serwerze źródłowym (fiszka pokazana na stronie Odtworzenie z backupu Networkera na innym hoście) i, jeśli różnią się od docelowego układu dysków, przygotowujemy `SET NEWNAME` dla każdego pliku:

```
RMAN> RUN {
2>   SET NEWNAME FOR DATAFILE 1 TO 'SciezkaDoPliku';
3>   SET NEWNAME FOR DATAFILE 2 TO 'SciezkaDoPliku';
4>   -- analogicznie dla pozostałych datafile bazy WMSPRD
5>   RESTORE DATABASE;
6>   SWITCH DATAFILE ALL;
7>   RECOVER DATABASE;
8> }
```

Przykład:

```
Starting restore at 25-SEP-26
using channel ORA_DISK_1
channel ORA_DISK_1: restoring datafile 00001 to /oracle/db/wmsprd/oradata/WMSPRD/system01.dbf
channel ORA_DISK_1: restoring datafile 00002 to /oracle/db/wmsprd/oradata/WMSPRD/sysaux01.dbf
Finished restore at 25-SEP-26

datafile 1 switched to datafile copy
datafile 2 switched to datafile copy

Starting recover at 25-SEP-26
media recovery complete, elapsed time: 00:04:02
Finished recover at 25-SEP-26
```

5. Otwarcie bazy z RESETLOGS

```
RMAN> ALTER DATABASE OPEN RESETLOGS;
```

Pełny, szczegółowy przebieg tego kroku razem z mapowaniem tempfile pokazuje strona Odtworzenie z backupu Networkera na innym hoście — techniczne kroki (SET NEWNAME, SWITCH, RESETLOGS) są identyczne niezależnie od tego, czy backup leży na dysku lokalnym, czy w systemie backupowym.

Przełączenie aplikacji

Niezależnie od wybranego wariantu, ostatni krok migracji jest ten sam — przełączenie aplikacji na nowy adres bazy:

Aktualizujemy `tnsnames.ora` po stronie serwerów aplikacyjnych, wskazując `HOST=NazwaNowegoSerwera` zamiast `ora01`, albo, jeśli aplikacja łączy się przez DNS/VIP, przełączamy sam wpis DNS na nowy adres IP.

Sprawdzamy, że aplikacja faktycznie nawiązuje połączenie z nowym serwerem:

```
SELECT sys_context('USERENV','SERVER_HOST') FROM dual;
```

Przykład:

```
SYS_CONTEXT('USERENV','SERVER_HOST')
-----------------------------------
NazwaNowegoSerwera
```

Weryfikacja

Sprawdzamy nazwę, DBID i tryb bazy na serwerze docelowym:

```
SELECT name, dbid, open_mode, log_mode FROM v$database;
```

Przykład:

```
NAME          DBID       OPEN_MODE            LOG_MODE
--------- ---------- -------------------- ------------
WMSPRD    3032243418 READ WRITE           ARCHIVELOG
```

Sprawdzamy rozmiar bazy po migracji i porównujemy go z rozmiarem źródłowym (fiszki na stronie Rozmiary tabel, segmentów i indeksów), żeby upewnić się, że nic nie zgubiło się po drodze:

```
SELECT ROUND(SUM(bytes)/1024/1024/1024, 2) AS database_size_gb FROM dba_data_files;
```

Sprawdzamy w aplikacji, że kluczowe operacje działają poprawnie na nowym serwerze, zanim ogłosimy migrację zakończoną.

Wycofanie zmian

Dopóki stary serwer `ora01` nie został wyłączony, wycofanie jest proste — przełączamy `tnsnames.ora` (albo wpis DNS) z powrotem na `ora01` i wracamy do pracy na starej bazie, tak jakby migracji nie było. Z tego powodu nie wygaszamy serwera źródłowego od razu po migracji — trzymamy go w gotowości przez ustalony okres karencji (typowo kilka dni do dwóch tygodni), na wypadek gdyby na nowym serwerze ujawnił się problem niewidoczny w pierwszych godzinach po przełączeniu.

Jeśli migracja szła wariantem RMAN i baza na `NazwaNowegoSerwera` została otwarta poleceniem `RESETLOGS`, cofnięcie samej bazy do stanu sprzed migracji nie jest możliwe bez ponownego odtworzenia z backupu — wycofanie w praktyce oznacza więc tylko przełączenie aplikacji z powrotem na `ora01`, nie unieważnienie tego, co stało się na `NazwaNowegoSerwera`.

Gdy coś nie zadziała

Objaw	Przyczyna	Naprawa
Import Data Pump kończy się błędem o brakującym tablespace	docelowa baza ma inny układ tablespace niż źródłowa	dodajemy `REMAP_TABLESPACE` do polecenia `impdp`, patrz strona Data Pump: export i import
RMAN nie znajduje backupu po `SET DBID`	zły DBID albo backup niedostępny z poziomu serwera docelowego	sprawdzamy DBID zapytaniem `SELECT dbid FROM v$database` na serwerze źródłowym i dostępność ścieżki/kanału backupu z `NazwaNowegoSerwera`
Aplikacja po przełączeniu zgłasza błędy uprawnień	migracja Data Pump nie przeniosła wszystkich grantów systemowych (np. nadanych bezpośrednio, poza rolami)	sprawdzamy uprawnienia zapytaniami z sekcji o uprawnieniach userów i uzupełniamy brakujące granty ręcznie
Wydajność zapytań gorsza na nowym serwerze zaraz po migracji	statystyki optymalizatora nie zostały przeniesione albo są nieaktualne po dużym imporcie	zbieramy statystyki na nowo (`DBMS_STATS.GATHER_DATABASE_STATS`) zanim uznamy różnicę wydajności za problem infrastrukturalny

Powiązane: Data Pump: export i import · RMAN: backup, walidacja, czyszczenie archiwów · Odtworzenie z backupu Networkera na innym hoście · Utworzenie nowej instancji bazy danych w wersji 19c i wyższej

Źródła: notatki własne, dokumentacja Oracle Database (Data Pump, Backup and Recovery User's Guide)