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`.

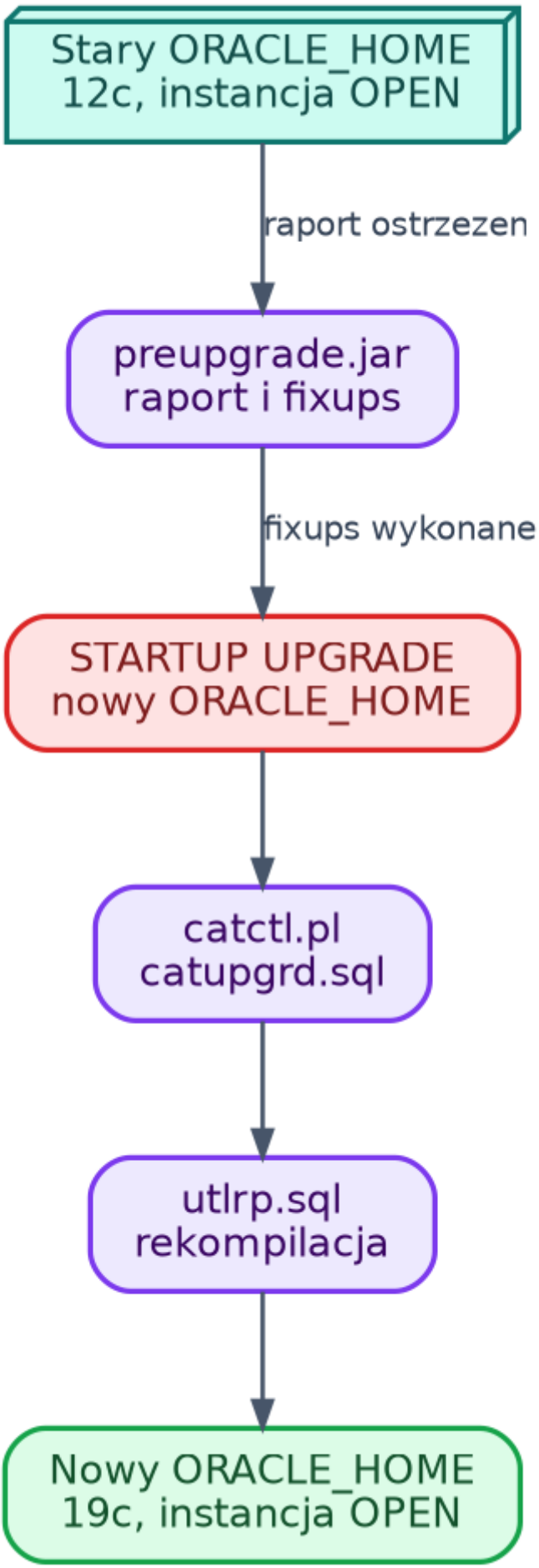
Skróty użyte na tej stronie (2)

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)

Revision #43
Created 2025-03-13 12:55:48 UTC by Tomasz Konieczny
Updated 2026-10-26 14:09:13 UTC by Tomasz Konieczny