Sesje: aktywne, nieaktywne, najdłuższe

Poziom: podstawy **Dotyczy: Oracle Database 19c** **Czas: ~20 min**

Zgłoszenie „baza się zawiesiła" prawie zawsze zaczyna się od tego samego widoku — `v$session` — i kończy tym samym poleceniem — `ALTER SYSTEM KILL SESSION`. Ta strona pokazuje całą drogę pomiędzy tymi dwoma punktami.

Skróty użyte na tej stronie (4)

Skrót	Co znaczy
PID	Process ID: numer procesu w systemie
RMAN	Recovery Manager: narzędzie Oracle do kopii zapasowych
SGA	System Global Area: wspólna pamięć instancji Oracle
WMS	Warehouse Management System: system zarządzania magazynem

Fiszka

TYLKO ODCZYT

Sprawdzenie aktywnych sesji

```
SELECT username FROM v$session WHERE username IS NOT NULL ORDER BY username ASC;
```

Lista wszystkich zalogowanych użytkowników — pierwszy, najprostszy rzut oka na to, kto w ogóle jest połączony do instancji, zanim zejdziemy do szczegółów pojedynczej sesji.

Przykład:

```
USERNAME
------------------------------
```

```
APP_USER
APP_USER
REPORTS_USER
SYSTEM
ZABBIX
```

ASC = Automatic Self-Calibration: automatyczna samokalibracja czujnika CO2, która zakłada, że co tydzień czujnik zobaczy czyste powietrze o stężeniu około 400 ppm

TYLKO ODCZYT

Sprawdzenie SID i SERIAL# wszystkich sesji

```
SELECT sid, serial#, username, command, program FROM v$session;
```

Przykład:

```
       SID    SERIAL# USERNAME             COMMAND PROGRAM
---------- ---------- -------------------- ------- ------------------------------
       142       5501 APP_USER                     3 JDBC Thin Client
       201       8832 REPORTS_USER                 3 sqlplus@app01
       305         12 ZABBIX                        3 zbx_monitor
```

SID = w Oracle: System Identifier, nazwa instancji; w Windows: Security Identifier, numer konta

TYLKO ODCZYT

Sprawdzenie sesji nieaktywnych

```
SELECT sid, serial#, status, program, username FROM v$session WHERE status = 'INACTIVE';
```

Sesja `INACTIVE` ma otwarte połączenie, ale w danym momencie nic nie wykonuje — z reguły to aplikacja trzymająca połączenie w puli, nie problem sam w sobie. Dużo takich sesji naraz bywa jednak sygnałem źle skonfigurowanego connection poolingu po stronie aplikacji.

Przykład:

```
       SID    SERIAL# STATUS   PROGRAM                          USERNAME
---------- ---------- -------- ------------------------------ --------------------
        88       2210 INACTIVE JDBC Thin Client                APP_USER
```

```
       112       4471 INACTIVE JDBC Thin Client                APP_USER
```

◫◫ TYLKO ODCZYT

Wyświetlenie informacji o procesie na podstawie SID i SERIAL

```
SELECT p.spid, s.sid, s.serial#, s.username, s.machine, s.program
FROM v$session s
JOIN v$process p ON (s.paddr = p.addr)
WHERE s.sid = NumerSID AND s.serial# = NumerSerial;
```

Łączymy `v$session` z `v$process` przez `paddr`/`addr`, żeby dojść od logicznej sesji Oracle do fizycznego procesu systemu operacyjnego, który ją obsługuje — `spid` w wyniku to właśnie systemowy PID, ten sam, który zobaczymy w wyniku `ps -ef`.

Przykład:

```
SPID          SID    SERIAL# USERNAME             MACHINE          PROGRAM
--------- ---------- ---------- -------------------- ----------------
------------------------------
28471            142       5501 APP_USER             app01        JDBC Thin Client
```

◫◫ **SID** = w Oracle: System Identifier, nazwa instancji; w Windows: Security Identifier, numer konta · **PID** = Process ID: numer procesu w systemie · **join** = połącz dwa pliki tekstowe po wspólnej kolumnie, jak złączenie tabel w bazie danych

◫◫ TYLKO ODCZYT

Sprawdzenie najdłużej wykonujących się kwerend

```
COLUMN sql_id FORMAT A15
COLUMN username FORMAT A15
COLUMN sql_text FORMAT A40
COLUMN event FORMAT A20
SELECT s.sid, s.serial#, s.username, s.sql_id, s.last_call_et AS elapsed_seconds,
       s.status, s.event, q.sql_text
FROM v$session s
JOIN v$sql q ON s.sql_id = q.sql_id
WHERE s.status = 'ACTIVE'
ORDER BY s.last_call_et DESC
```

```
FETCH FIRST 10 ROWS ONLY;
```

`last_call_et` to liczba sekund, jaka minęła od ostatniej interakcji sesji z instancją — dla sesji `ACTIVE` w praktyce oznacza to czas trwania bieżącego zapytania. Sortujemy malejąco, żeby najdłużej trwające zapytanie znalazło się na górze wyniku.

Przykład:

```
       SID    SERIAL# USERNAME        SQL_ID          ELAPSED_SECONDS STATUS
EVENT                SQL_TEXT
---------- ---------- -------------- -------------- -------------- --------
------------------- ----------------------------------------
       142       5501 APP_USER        7fzg1k3mn82qw        872 ACTIVE   db file
sequential read UPDATE zamowienia SET status = 'zamkniete'...
```

join = połącz dwa pliki tekstowe po wspólnej kolumnie, jak złączenie tabel w bazie danych

ZMIENIA SYSTEM

Sprawdzenie najdłużej działających programów: v$session_longops

```
SELECT s.sid, s.serial#, s.username, s.osuser, s.program, s.machine,
       l.elapsed_seconds, l.start_time, l.last_update_time, l.target_desc, l.sofar,
l.totalwork
FROM v$session s, v$session_longops l
WHERE s.sid = l.sid
  AND s.username IS NOT NULL
  AND l.elapsed_seconds > 60
ORDER BY l.elapsed_seconds DESC;
```

> **Pojęcie: v$session_longops.** Widok dedykowany operacjom, które z natury trwają długo i potrafią raportować postęp w trakcie wykonania — backup RMAN, przebudowa indeksu, `CREATE TABLE ... AS SELECT` na dużej tabeli. `sofar` i `totalwork` to jednostki postępu specyficzne dla danej operacji — ich iloraz razy 100 daje procent ukończenia.

Przykład:

```
       SID    SERIAL# USERNAME  OSUSER  PROGRAM         MACHINE     ELAPSED_SECONDS
TARGET_DESC          SOFAR TOTALWORK
```

```
---------- ---------- --------- ------- --------------- ----------- ---------------
------------------- ----- ---------
       305         12 SYSTEM    oracle  rman@ora01   ora01               4120 RMAN: full
datafile  71230    112000
```

RMAN = Recovery Manager: narzędzie Oracle do kopii zapasowych

TYLKO ODCZYT

Sprawdzenie historii aktywności konkretnych sesji: v$active_session_history

```
SELECT a.sid, a.sql_id, a.sql_text
FROM v$active_session_history a
WHERE a.sid IN (NumerSID)
ORDER BY a.sid, a.sample_time;
```

`v$active_session_history` próbkuje stan aktywnych sesji co sekundę i trzyma krótką historię w pamięci — użyteczne, gdy sesja już się zakończyła, ale chcemy odtworzyć, co dokładnie robiła tuż przed problemem, zanim zniknęła z `v$session`.

Przykład:

```
       SID SQL_ID          SQL_TEXT
---------- --------------- ----------------------------------------
       142 7fzg1k3mn82qw   UPDATE zamowienia SET status = 'zamkniete'...
       142 7fzg1k3mn82qw   UPDATE zamowienia SET status = 'zamkniete'...
```

ZMIENIA SYSTEM

Zakończenie sesji: ALTER SYSTEM KILL SESSION z IMMEDIATE

```
ALTER SYSTEM KILL SESSION 'NumerSID,NumerSerial' IMMEDIATE;
```

Przykład:

```
SQL> ALTER SYSTEM KILL SESSION '142,5501' IMMEDIATE;


System altered.
```

⚠ ZMIENIA SYSTEM

Zakończenie sesji bez IMMEDIATE

```
ALTER SYSTEM KILL SESSION 'NumerSID,NumerSerial';
```

Sesja zostaje oznaczona do zamknięcia, ale faktyczne zwolnienie zasobów następuje dopiero, gdy sesja sama dojdzie do bezpiecznego punktu — do tego czasu w `v$session.status` widzimy `KILLED`, a nie natychmiastowe zniknięcie wiersza.

Przykład:

```
SQL> ALTER SYSTEM KILL SESSION '142,5501';


System altered.
```

⚠ ZMIENIA SYSTEM

Rozłączenie sesji: ALTER SYSTEM DISCONNECT SESSION

```
ALTER SYSTEM DISCONNECT SESSION 'NumerSID,NumerSerial' IMMEDIATE;
```

> **Pojęcie: KILL SESSION kontra DISCONNECT SESSION.** `KILL SESSION` zamyka sesję logiczną, ale przy sesjach dedykowanych proces systemowy po stronie serwera potrafi jeszcze przez chwilę istnieć, zanim faktycznie zniknie. `DISCONNECT SESSION` idzie krok dalej i zamyka też samo połączenie sieciowe do klienta — używamy go, gdy `KILL SESSION` nie wystarcza, bo sesja jest zawieszona na poziomie samego połączenia (np. martwy socket po awarii sieci), a nie na poziomie wykonywanego zapytania.

Przykład:

```
SQL> ALTER SYSTEM DISCONNECT SESSION '142,5501' IMMEDIATE;


System altered.
```

⚠ ZMIENIA SYSTEM

Zakończenie wszystkich sesji danego użytkownika

```
BEGIN
  FOR r IN (SELECT sid, serial# FROM v$session WHERE username = 'NazwaUsera') LOOP
    EXECUTE IMMEDIATE 'ALTER SYSTEM KILL SESSION ''' || r.sid || ',' || r.serial# || '''
IMMEDIATE';
  END LOOP;
END;
/
```

Blok PL/SQL zamiast pojedynczej komendy, bo `ALTER SYSTEM KILL SESSION` przyjmuje dokładnie jedną parę `SID,SERIAL#` na wywołanie — pętla po `v$session` odfiltrowanym do jednego użytkownika zabija wszystkie jego sesje naraz, bez ręcznego przepisywania każdej pary z osobna.

Przykład:

```
SQL> BEGIN
  2    FOR r IN (SELECT sid, serial# FROM v$session WHERE username = 'APP_USER') LOOP
  3      EXECUTE IMMEDIATE 'ALTER SYSTEM KILL SESSION ''' || r.sid || ',' || r.serial#
|| ''' IMMEDIATE';
  4    END LOOP;
  5  END;
  6  /

PL/SQL procedure successfully completed.
```

□□ **SID** = w Oracle: System Identifier, nazwa instancji; w Windows: Security Identifier, numer konta · **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

□□ **TYLKO ODCZYT**

Sprawdzenie systemowego PID procesu bazy danych

```
SELECT p.spid
FROM v$session s JOIN v$process p ON s.paddr = p.addr
WHERE s.sid = NumerSID;
```

Przykład:

```
SPID
------------------------
28471
```

□□ **PID** = Process ID: numer procesu w systemie

□□ OSTROŻNIE · NIEODWRACALNE

Ubicie procesu systemowego: kill -9

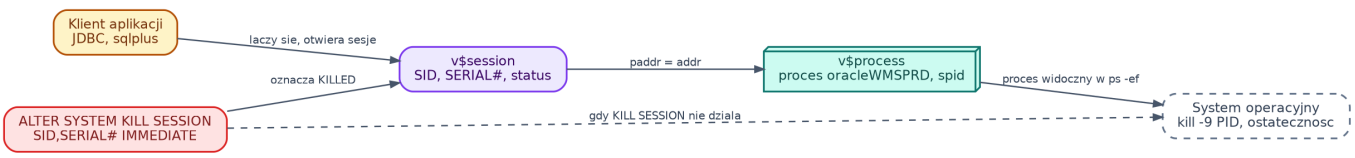
```
kill -9 NumerPID
```

Sięgamy po `kill -9` na procesie systemowym Oracle tylko wtedy, gdy `ALTER SYSTEM KILL SESSION IMMEDIATE` i `ALTER SYSTEM DISCONNECT SESSION` obie zawiodły, a proces nadal wisi — to znaczy, że sam mechanizm zamykania sesji wewnątrz instancji nie działa, nie że proces jest po prostu wolny. `kill -9` nie daje procesowi Oracle szansy na uporządkowane zwolnienie swoich struktur w SGA — proces PMON musi potem posprzątać po nim ręcznie, co dla pojedynczej sesji zwykle przebiega bez problemu, ale w skrajnym przypadku, przy uszkodzonych strukturach pamięci współdzielonej, potrafi doprowadzić do niestabilności całej instancji, a nie tylko jednej sesji. Traktujemy to jako ostateczność, nie pierwszy odruch.

□□ **SGA** = System Global Area: wspólna pamięć instancji Oracle · **kill** = wyślij sygnał do procesu, domyślnie SIGTERM proszący o zamknięcie

Widok v$session

Pojęcie: v$session. Widok dynamiczny z jednym wierszem na każde połączenie do instancji — identyfikator sesji (`SID`), numer seryjny (`SERIAL#`), użytkownik, program kliencki, status i wiele innych atrybutów. To jest odpowiednik `pg_stat_activity` z PostgreSQL, tylko z inną nazwą i inną kolumną identyfikującą sesję.



Sesja logiczna z v$session łączy się z fizycznym procesem systemowym przez v$process, a KILL SESSION działa najpierw na sesji, dopiero potem na procesie (kliknięcie otwiera pełny rozmiar)

> **Pojęcie: SID i SERIAL#.** `SID` to numer sesji, unikalny w danym momencie, ale wielokrotnego użytku — po zakończeniu sesji Oracle prędzej czy później przydzieli ten sam `SID` nowemu połączeniu. `SERIAL#` to numer seryjny, który razem z `SID` jednoznacznie identyfikuje konkretną sesję nawet po tym, jak stary `SID` zostanie ponownie użyty. Każda komenda administracyjna działająca na konkretnej sesji — zakończenie, sprawdzenie procesu — wymaga obu wartości naraz, właśnie po to, żeby nie trafić przypadkiem w zupełnie inną, nowszą sesję.

Kończenie sesji

> **Pojęcie: ALTER SYSTEM KILL SESSION.** Polecenie administracyjne, które oznacza sesję do zamknięcia. Bez `IMMEDIATE` Oracle czeka na naturalny punkt zakończenia bieżącej operacji sesji i oznacza ją jako `KILLED` do tego czasu — sesja może w tym stanie wisieć jeszcze przez chwilę. Z `IMMEDIATE` wycofuje bieżącą transakcję i zwalnia zasoby sesji od razu.

Częste błędy

KILL SESSION bez IMMEDIATE i oczekiwanie natychmiastowego efektu. Sesja przechodzi w stan `KILLED`, ale nie znika z `v$session` od razu — dopiero gdy sama dojdzie do punktu, w którym instancja może bezpiecznie zwolnić jej zasoby. Przy sesji zawieszonej w oczekiwaniu na blokadę to może nigdy nie nastąpić bez `IMMEDIATE`.

Pomylenie starego SID z nowym. `SID` jest wielokrotnego użytku — komenda wykonana na `SID` bez sprawdzenia aktualnego `SERIAL#` potrafi trafić w zupełnie inną, świeżo nawiązaną sesję, jeśli między odczytaniem `SID` a wykonaniem komendy minęło zbyt dużo czasu.

Odruchowe `kill -9` zamiast ALTER SYSTEM KILL SESSION. Zabicie procesu systemowego z pominięciem mechanizmów instancji jest zawsze rozwiązaniem awaryjnym, nie standardową procedurą kończenia sesji — najpierw próbujemy zamknąć sesję na poziomie samej bazy.

Z życia wzięte

□□ Z ŻYCIA WZIĘTE

O 6:10 Marek zgłasza, że skanery w Swarzędzu wiszą przy logowaniu do WMS. Tomek sprawdza `v$session` na WMSPRD i widzi kilkadziesiąt sesji APP_USER w stanie ACTIVE, z jedną trzymającą to samo zapytanie od dwudziestu minut. Po znalezieniu jej systemowego PID przez `v$process` i zabiciu jej poleceniem `ALTER SYSTEM KILL SESSION IMMEDIATE` kolejka skanerów rusza z miejsca.

Powiązane: Rozpoznanie instancji: wersja, parametry, komponenty · Blokady i obciążenie · Scheduler i zadania w bazie · Architektura Oracle: instancja, baza, SGA, PGA, procesy tła

Źródła: notatki własne, dokumentacja Oracle Database Reference 19c

---

Revision #43
Created 2026-09-24 23:34:17 UTC by Tomasz Konieczny
Updated 2026-10-26 14:08:47 UTC by Tomasz Konieczny