Sesje i aktywność

Poziom: podstawy **Dotyczy: Sybase ASE 15.7 / 16.0 (SAP ASE)** **Czas: ~15 min**

Kto jest podłączony do serwera, co w danej chwili robi i co zrobić, gdy trzeba tę sesję przerwać.

Skróty użyte na tej stronie (3)

Skrót	Co znaczy
ASE	Adaptive Server Enterprise: nazwa silnika Sybase od 1996 roku, dziś rozwijanego przez SAP jako SAP ASE
SAP	Systeme, Anwendungen und Produkte in der Datenverarbeitung: niemiecki producent oprogramowania klasy ERP, właściciel silnika HANA i przejętego wcześniej Sybase ASE
SQL	Structured Query Language: język zapytań do relacyjnych baz danych

Fiszka

Komendy wykonujemy zalogowani do `isql`, opisanego na stronie isql: klient wiersza poleceń.

TYLKO ODCZYT

Wyświetlenie ilości aktywnych sesji per baza danych

```
SELECT COUNT(DISTINCT spid) AS Liczba_Sesji
FROM master..sysprocesses
WHERE dbid = DB_ID('NazwaBazy')
go
```

Liczymy, ile odrębnych procesów (`spid`) ma aktualnie otwarte połączenie do wskazanej bazy — pierwsze pytanie przed jakąkolwiek operacją, która wymaga wyłączności na bazę, na przykład przed odtwarzaniem opisanym na stronie Backup i odtwarzanie.

> **Pojęcie: spid.** Server Process ID — numer procesu nadawany przez silnik Sybase każdemu połączeniu w momencie jego nawiązania. Nie ma nic wspólnego z PID-em procesu systemu operacyjnego; to wewnętrzny numer samego dataservera, unikalny w obrębie jednej instancji, i to właśnie ten numer podajemy w poleceniu `kill`.

Przykład:

```
 Liczba_Sesji
 -------------
             7


(1 row affected)
```

TYLKO ODCZYT

Wyświetlenie wszystkich aktywnych sesji: sp_who

```
sp_who
go
```

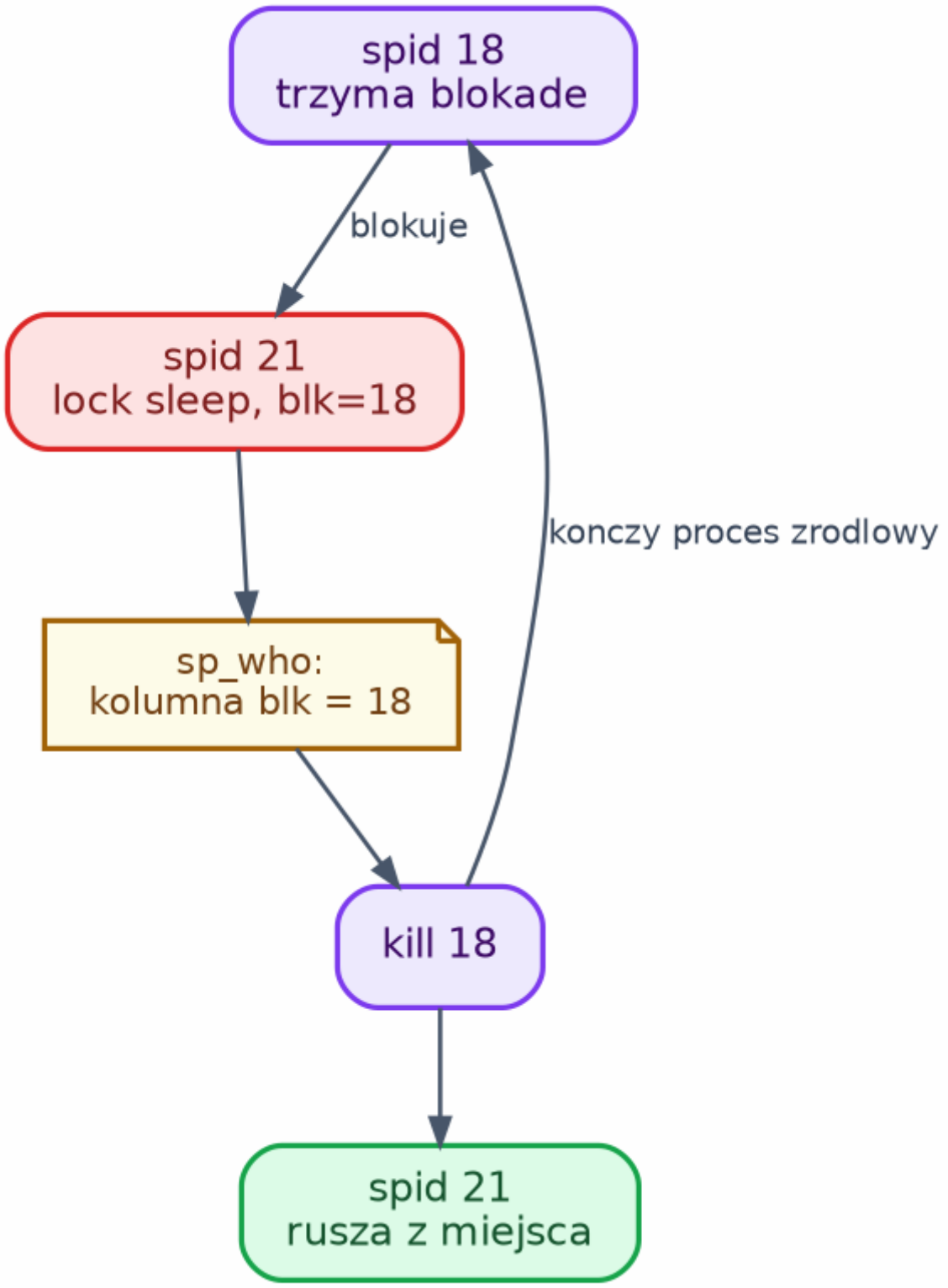
Zwracamy listę wszystkich procesów podłączonych do serwera, niezależnie od bazy — login, bazę, na której aktualnie pracują, oraz status procesu.

Przykład:

```
 spid  status      loginame    hostname    blk  dbname     cmd
 ----- ----------- ----------- ----------- ---- ---------- --------------
 12    running     sa          app01   0    master     SELECT
 18    sleeping    adm_user    app01   0    produkcja  AWAITING COMMAND
 21    lock sleep  adm_user    app01   18   produkcja  UPDATE


(3 rows affected)
```

Kolumna `blk` różna od zera pokazuje `spid` procesu blokującego — w powyższym przykładzie sesja `21` czeka na zwolnienie blokady trzymanej przez sesję `18`.



Odnalezienie procesu źródłowego blokady przez kolumnę blk i zabicie go, zamiast zabijania samej ofiary (kliknięcie otwiera pełny rozmiar)

🔒 TYLKO ODCZYT

Zawężenie listy sesji do jednej bazy danych: sp_who z parametrem

```
sp_who 'NazwaLoginu'
go
```

Podanie nazwy loginu jako argumentu zawęża wynik `sp_who` do sesji otwartych przez tego konkretnego użytkownika, zamiast wypisywać wszystkie połączenia na serwerze.

🔒 TYLKO ODCZYT

Wyświetlenie blokad na serwerze: sp_lock

```
sp_lock
go
```

Zwracamy listę wszystkich blokad aktualnie trzymanych na serwerze — który `spid` trzyma jaką blokadę, na jakim obiekcie i w jakim trybie.

Przykład:

```
 spid   locktype        table_id    page   dbname     class
 ------ --------------- ----------- ------ ---------- -----------
 18     Ex_row-blk       1234567     0      produkcja  Non Cursor Lock
 21     Sh_page-blk      1234567     0      produkcja  Non Cursor Lock

(2 rows affected)
```

🔒 TYLKO ODCZYT

Sprawdzenie blokad filtrem po nazwie bazy: monLocks

```
SELECT * FROM master..monLocks
WHERE DBName LIKE 'NazwaBazy%'
go
```

Odpytujemy tabelę monitorującą `monLocks` bezpośrednio, zawężając wynik do konkretnej bazy filtrem `LIKE` — wygodniejsze niż przeglądanie całego wyniku `sp_lock` na serwerze, na którym równolegle pracuje wiele baz naraz. Wymaga włączonego mechanizmu Monitoring

Tables (`sp_configure 'enable monitoring'`).

◫◫ ZMIENIA SYSTEM

Zakończenie procesu: kill

```
kill NumerSPID
go
```

Przerywamy wskazany proces po numerze `spid` odczytanym wcześniej z `sp_who` albo z `sysprocesses` — używamy tego, gdy sesja wisi, trzyma blokadę blokującą inne procesy, albo gdy musimy zwolnić bazę przed jej odtworzeniem, jak w procedurze opisanej na stronie Backup i odtwarzanie.

Uważaj: `kill` nie działa natychmiast, jeśli proces akurat wykonuje długą operację zapisu — silnik oznacza go do zabicia i kończy transakcję (rollback), co przy dużej transakcji potrafi potrwać dłużej niż samo jej wykonanie. Ponowne `kill NumerSPID with statusonly` (bez `go` osobno, jako druga forma tej samej komendy) pokazuje postęp wycofywania.

◫◫ **kill** = wyślij sygnał do procesu, domyślnie SIGTERM proszący o zamknięcie

◫◫ TYLKO ODCZYT

Sprawdzenie zajętych zasobów silnika: sp_monitor

```
sp_monitor
go
```

Zwracamy migawkę bieżącego wykorzystania zasobów serwera od jego uruchomienia — liczbę operacji odczytu i zapisu, wykorzystanie procesora i pamięci podręcznej.

Przykład:

```
last_run                    current_run                 seconds
--------------------------- --------------------------- --------
Sep 25 2026  8:00:00:000AM   Sep 25 2026 10:15:00:000AM    8100

cpu_busy         io_busy          idle             packets_received
---------------- ---------------- ---------------- ------------------
412(5%)-412(5%)  88(1%)-88(1%)    7600(94%)-7600(94%)  184291
```

```
(1 row affected)
```

🔒 TYLKO ODCZYT

Monitorowanie wydajności silnika przez określony czas: sp_sysmon

```
sp_sysmon "00:01:00"
go
```

Uruchamiamy zbieranie szczegółowych statystyk wydajnościowych silnika przez podany czas (tu jedną minutę) i po jego upływie otrzymujemy wielostronicowy raport — wykorzystanie wątków silnika, cache'y danych, blokad i I/O. Sięgamy po to przy podejrzeniu realnego problemu wydajnościowego, nie jako rutynowe sprawdzenie, bo sam pomiar obciąża serwer.

Z życia wzięte

🔒 Z ŻYCIA WZIĘTE

W środę rano skrypt eksportujący dane do systemu kadrowego wisi od godziny, blokując resztę zapytań do PLACE_ARCH na syb01. sp_who pokazuje sesję w stanie lock sleep, czekającą na blokadę trzymaną przez inny proces, który sam nic już nie robi od dłuższego czasu. Zamiast zabijać pierwszą lepszą sesję, sprawdzamy najpierw kolumnę blk, żeby trafić w proces źródłowy, nie w ofiarę blokady. Po kill numeru spid procesu źródłowego cała kolejka zapytań rusza z miejsca.

Częste błędy

Zabicie sesji bez sprawdzenia `blk`. Zabicie procesu, który sam jest zablokowany przez inny (kolumna `blk` w `sp_who` różna od zera), nic nie rozwiąże — trzeba dotrzeć do procesu źródłowego, blokującego resztę.

Mylenie `spid` z PID-em systemowym. `kill NumerSPID` w `isql` i `kill` systemowe z powłoki bash to dwie zupełnie różne komendy działające na dwóch różnych przestrzeniach numerów — próba zabicia procesu Sybase komendą systemową `kill` nie przerywa sesji SQL, tylko w skrajnym przypadku ubija cały dataserver.

Powiązane: isql: klient wiersza poleceń · Rozpoznanie serwera i baz · Backup i odtwarzanie

Źródła: notatki własne administratora Sybase ASE · dokumentacja SAP ASE, rozdział „System Administration Guide" i „Monitoring Tables Reference"

Revision #42
Created 2025-03-09 16:23:37 UTC by Tomasz Konieczny
Updated 2026-10-26 14:08:05 UTC by Tomasz Konieczny