Architektura Sybase ASE: serwer, device, baza, segment

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

Cztery pojęcia, bez których żadna komenda w tej książce nie ma sensu: czym jest serwer Sybase, czym jest device, jak device ma się do bazy danych i czym w tym wszystkim jest segment.

Skróty użyte na tej stronie (7)

Skrót	Co znaczy
ASE	Adaptive Server Enterprise: nazwa silnika Sybase od 1996 roku, dziś rozwijanego przez SAP jako SAP ASE
LVM	Logical Volume Manager: warstwa w Linuksie składająca dyski w elastyczne wolumeny
MSSQL	Microsoft SQL Server: silnik bazy danych Microsoftu, na nim w Zielonej stoi księgowość Optima
RAID	Redundant Array of Independent Disks: sposób łączenia kilku fizycznych dysków w jeden, widziany przez system jako pojedynczy dysk, zwykle z dodatkową odpornością na awarię jednego z nich
SAN	w certyfikatach: Subject Alternative Name, dodatkowe nazwy hosta w certyfikacie; w pamięciach: Storage Area Network, sieć dyskowa
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

Po co to istnieje

> **Pojęcie: Adaptive Server Enterprise.** Sybase ASE (Adaptive Server Enterprise), dziś sprzedawany pod marką SAP ASE po przejęciu Sybase przez SAP w 2010 roku, to relacyjny silnik bazodanowy zaprojektowany w latach osiemdziesiątych z myślą o dużej liczbie

> równoległych połączeń i pracy w architekturze klient-serwer. Kod, na którym oparty był kiedyś wczesny Microsoft SQL Server, wywodzi się właśnie z Sybase ASE — obie rodziny rozeszły się na początku lat dziewięćdziesiątych, ale część słownictwa (`sysprocesses`, `sp_who`, `sp_helpdb`) do dziś wygląda znajomo dla administratora MSSQL.

Zanim zaczniemy uruchamiać komendy, warto rozumieć, z ilu osobnych procesów faktycznie składa się instalacja Sybase, i czym różni się device od bazy danych — to jest najczęstsze źródło pomyłek u kogoś, kto przychodzi z PostgreSQL albo z Oracle.

Procesy silnika

> **Pojęcie: dataserver.** Główny proces silnika, ten, który faktycznie przyjmuje połączenia klientów, wykonuje zapytania SQL i zarządza pamięcią oraz danymi na dysku. To jest to, co potocznie nazywamy „Sybase" albo „ASE" — jeden proces systemowy, uruchamiany zwykle jako usługa `sybase` albo bezpośrednio binarką `dataserver`.

> **Pojęcie: backupserver.** Osobny, niezależny proces, który wykonuje faktyczne operacje `dump` i `load` — backup i odtwarzanie baz danych. Dataserver nigdy sam nie zapisuje kopii zapasowej na dysk czy taśmę: zleca to backupserverowi przez wewnętrzny protokół. Rozdzielenie tych dwóch ról ma sens historyczny i praktyczny naraz — backup dużej bazy potrafi trwać godzinami, a wydzielenie go do osobnego procesu pozwala silnikowi zarządzać tym obciążeniem niezależnie od obsługi zapytań. Bez uruchomionego backupservera komendy `dump database` i `load database`, opisane na stronie Backup i odtwarzanie, kończą się błędem połączenia.

Oba procesy nasłuchują na osobnych portach i muszą umieć się nawzajem znaleźć — do tego służy plik `interfaces` (w nowszych wersjach `sql.ini` na Windows), trzymany zwykle w katalogu `$SYBASE/interfaces`. Zawiera on wpis dla każdego serwera logicznego z jego nazwą, adresem i portem:

```
SRV_DB01
        master tcp ether syb01.zielona.example 5000
        query tcp ether syb01.zielona.example 5000
SRV_DB01_BS
        master tcp ether syb01.zielona.example 5001
        query tcp ether syb01.zielona.example 5001
```

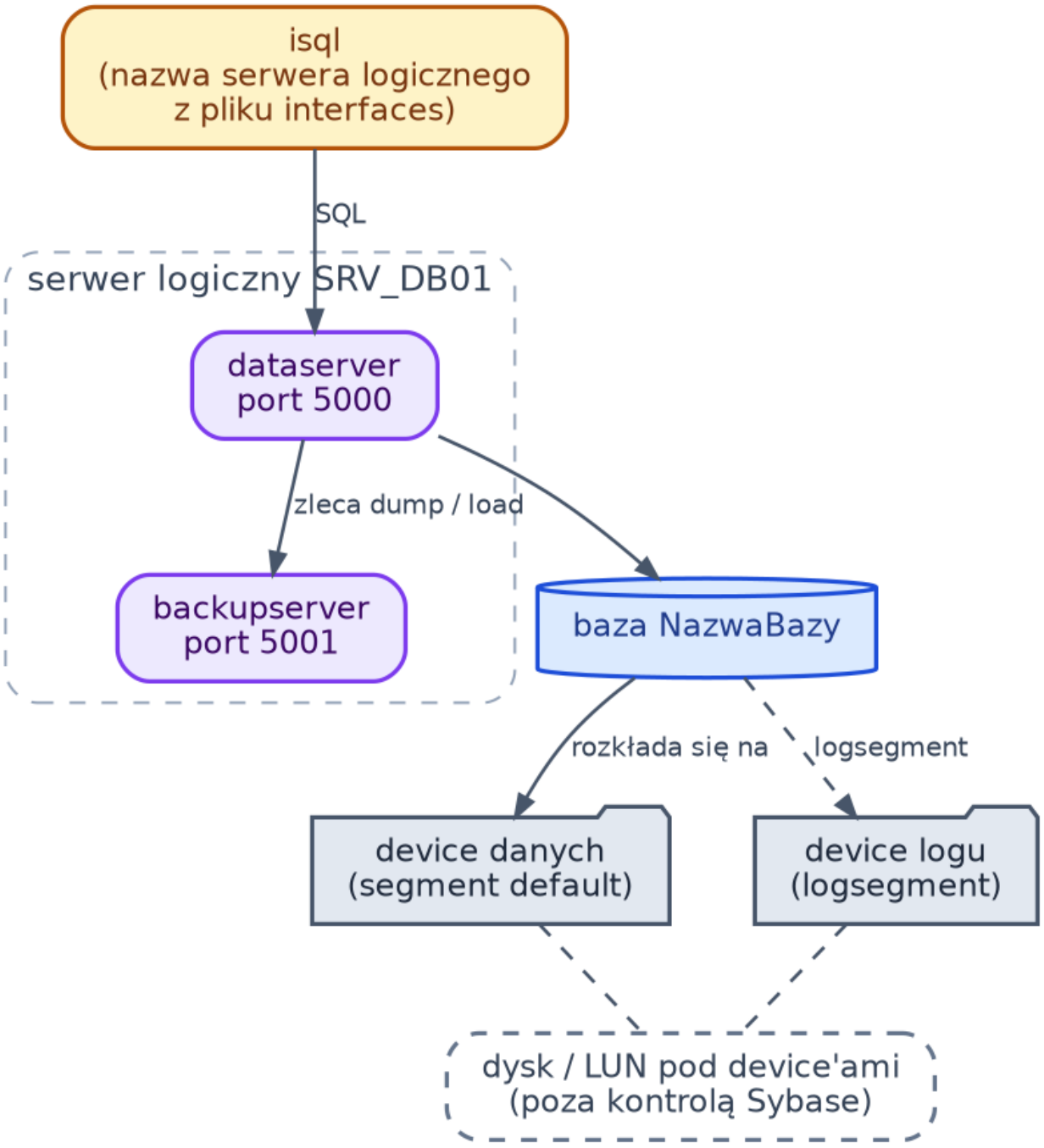
Nazwa serwera logicznego z tego pliku, nie nazwa hosta systemu operacyjnego, jest tym, co podajemy potem jako `NazwaSerwera` w kliencie `isql`, opisanym na stronie isql: klient wiersza

poleceń.

> **Pojęcie: SYBASE i SYBASE_ASE.** Dwie zmienne środowiskowe, od których zależy, czy narzędzia w ogóle znajdą swoją instalację. `SYBASE` wskazuje katalog główny instalacji, na przykład `/opt/sybase`, a `SYBASE_ASE` wskazuje podkatalog konkretnej wersji silnika wewnątrz niego, na przykład `ASE-16_0`. Skrypt `SYBASE.sh` (albo `SYBASE.csh` dla powłok z rodziny csh), leżący w katalogu głównym instalacji, ustawia obie te zmienne razem z `PATH` i `LD_LIBRARY_PATH` — bez jego wczytania polecenia `isql` czy `dataserver` z linii komend zwykle nie są nawet widoczne.

Device, baza i segment

To jest rozróżnienie, które warto mieć jasne, zanim ruszymy dalej, bo cała rodzina komend do powiększania miejsca, opisana na stronie Powiększanie baz i urządzeń, operuje właśnie na tych trzech pojęciach naraz, i myli się je najczęściej.



Droga od isql przez dataserver do bazy, ktora rozklada sie na device'y z osobnym segmentem logu (kliknięcie otwiera pełny rozmiar)

> **Pojęcie: device.** Urządzenie w rozumieniu Sybase (`disk init`, `sp_helpdevice`) to nie dysk fizyczny ani partycja systemu operacyjnego, tylko plik albo surowa partycja, którą Sybase sam sformatował na swój wewnętrzny użytek i którą zarządza porcjami zwanymi stronami. Jeden device to kawałek przestrzeni zarezerwowany dla silnika — silnik nie wie ani nie chce wiedzieć, na jakim fizycznie dysku ten plik leży, to jest już sprawa administratora systemu i warstwy LVM czy SAN pod spodem.

> **Pojęcie: baza danych.** Baza danych w Sybase to obiekt logiczny, który dopiero rozkłada się na jeden lub więcej device'ów. Jedna baza może zajmować fragmenty kilku różnych device'ów naraz, a jeden device może być współdzielony przez kilka baz — to jest coś, czego administrator przyzwyczajony do plików danych „jeden do jednego" z innych silników nie spodziewa się od razu. Mapowanie, który fragment którego device'a należy do której bazy, trzyma systemowa tabela `sysusages` w bazie `master`, a jej czytelne podsumowanie zwraca `sp_helpdb`, opisane na stronie Rozpoznanie serwera i baz.

> **Pojęcie: segment.** Segment to nazwana pula miejsca wewnątrz bazy danych, wskazująca, na których device'ach wolno zapisywać dany rodzaj obiektu. Domyślnie istnieją trzy segmenty: `system` (katalog systemowy bazy), `default` (zwykłe tabele i indeksy) i `logsegment` (log transakcyjny). Rozdzielenie logu transakcyjnego na osobny device przypięty wyłącznie do `logsegment` jest jedną z pierwszych rzeczy, które sprawdzamy na nieznanym serwerze — log rosnący na tym samym device'ie co dane potrafi w chwili awarii zapełnić się razem z nimi i zablokować bazę w najgorszym możliwym momencie.

Zależność w jedną stronę wygląda więc tak: dysk fizyczny (poza kontrolą silnika) trzyma plik, z którego `disk init` robi device; device zostaje przypisany do jednej lub więcej baz przez `create database` albo `alter database ... on`; wewnątrz bazy segment decyduje, który rodzaj danych trafia na który z przypisanych device'ów.

Analogia

Device, baza i segment układają się w tę samą zależność co magazyn, hala i regał.

Device to fizyczny magazyn — pomieszczenie, które administrator budynku (system operacyjny i warstwa dysków pod spodem) oddaje do dyspozycji jednej firmie transportowej (Sybase) i o którego fizyczną konstrukcję ta firma już się nie martwi. Baza danych to hala robocza wydzielona wewnątrz jednego magazynu albo rozciągnięta na kilka magazynów naraz, jeśli jeden nie starczał — dokładnie tak jak jedna baza potrafi zajmować kilka device'ów. Segment to z kolei oznaczenie na regałach wewnątrz hali: „tu tylko dokumentacja" (system), „tu towar bieżący" (default), „tu księga przyjęć i wydań" (logsegment) — rozdzielenie, które nie zmienia tego, w którym magazynie coś fizycznie leży, tylko pilnuje, żeby różne rodzaje rzeczy nie trafiały na wspólną półkę.

Ograniczenie tej analogii jest takie samo jak ograniczenie samego mechanizmu: firma transportowa (silnik) nie ma wglądu w to, czy magazyn stoi na bagnistym gruncie czy na betonowej płycie — awaria dysku pod device'em jest dla Sybase zupełnie niewidoczna, dopóki nie zacznie zwracać błędów odczytu. Odporność na taką awarię załatwia się o piętro niżej, w warstwie RAID czy replikacji dyskowej, nie w samym silniku.

Engines — silnik jako zestaw wątków roboczych

> **Pojęcie: engine.** Wewnątrz jednego procesu `dataserver` zapytania obsługuje nie jeden wątek, tylko pula wątków roboczych zwanych w dokumentacji Sybase właśnie „engines" — każdy z nich potrafi przejąć obsługę dowolnego połączenia klienckiego czekającego w kolejce. To osobne znaczenie słowa „silnik" od tego, którego używamy w tytule tej strony na określenie całego Adaptive Server Enterprise — tu chodzi o wewnętrzny wątek roboczy, nie o cały produkt.

Liczbę takich wątków konfiguruje parametr `sp_configure 'max online engines'`, a zmiana wymaga restartu serwera, żeby zaczęła obowiązywać:

```
sp_configure 'max online engines'
go
```

```
 Parameter Name        Default    Memory Used  Config Value  Run Value
 --------------------- ---------- ------------ ------------- ----------
 max online engines     1          0            4             4

(1 row affected)
```

Zbyt mała liczba wątków roboczych względem liczby rdzeni procesora zostawia moc obliczeniową niewykorzystaną; zbyt duża, przekraczająca sensownie liczbę fizycznie dostępnych rdzeni, prowadzi do rywalizacji wątków o procesor i w praktyce potrafi zapytania spowolnić zamiast przyspieszyć.

Bazy systemowe

Każda instalacja Sybase ASE, zaraz po `dataserver -d`, ma już cztery bazy założone automatycznie — zanim jeszcze administrator utworzy pierwszą bazę produkcyjną.

Baza	Rola
`master`	katalog całego serwera — loginy, konfiguracja, lista wszystkich pozostałych baz i ich device'ów. Bez `master` serwer w ogóle nie wstaje

Baza	Rola
`model`	szablon, z którego kopiowana jest struktura każdej nowej bazy tworzonej przez `create database`
`tempdb`	przestrzeń robocza na tabele tymczasowe, sortowania i wyniki pośrednie zapytań — czyszczona przy każdym restarcie serwera
`sybsystemprocs`	przechowuje definicje wszystkich procedur składowanych systemowych zaczynających się od `sp_` — `sp_helpdb`, `sp_who`, `sp_addlogin` i reszta fizycznie mieszkają tutaj, nie w `master`

Uważaj: uszkodzenie bazy `master`, w odróżnieniu od uszkodzenia bazy produkcyjnej, zwykle oznacza, że serwer nie jest w stanie się nawet uruchomić — dlatego backup `master` po każdej zmianie konfiguracji (nowy login, nowy device, zmiana `sp_configure`) jest regułą, nie luksusem.

Strona logiczna

> **Pojęcie: logical page size.** Rozmiar strony to jednostka, którą Sybase przydziela i rozlicza przestrzeń — zarówno na device'ach, jak i w pamięci podręcznej danych. Starsze instalacje ASE pracowały domyślnie na stronach 2 KB; nowsze instalacje 15.x/16.0 zakładane są zwykle ze stronami 4 KB, 8 KB albo 16 KB, bo większa strona ogranicza liczbę operacji I/O potrzebnych do przeczytania tej samej ilości danych, kosztem trochę większego marnotrawstwa miejsca na małych wierszach. Rozmiar strony ustala się raz, w momencie inicjalizacji serwera (`dataserver -b`), i nie da się go zmienić bez odtworzenia serwera od zera z nowym `master` device — to jedna z niewielu decyzji w Sybase, której nie da się cofnąć poleceniem administracyjnym. Zmienna systemowa `@@maxpagesize`, widoczna w zapytaniach na stronie Rozpoznanie serwera i baz, zwraca właśnie tę wartość w bajtach.

Jak to działa: przydział strony przy zapisie

Kiedy transakcja wstawia nowy wiersz, silnik nie szuka wolnego miejsca na całym serwerze — szuka go wyłącznie w granicach segmentu właściwego dla danego obiektu, wewnątrz device'ów przypisanych do bazy. Dla zwykłej tabeli oznacza to segment `default`; dla zapisu do logu transakcyjnego — `logsegment`, fizycznie odizolowany na innym device'ie, jeśli administrator tak to skonfigurował. Strona (jednostka o rozmiarze `logical page size`, opisanym niżej) jest najmniejszą porcją, jaką silnik potrafi przydzielić i odczytać — nawet wiersz mniejszy niż jedna strona rezerwuje

ją w całości, co przy bardzo wąskich tabelach z bardzo dużą liczbą wierszy potrafi zostawiać zauważalny procent zarezerwowanego, ale niewykorzystanego miejsca.

Kiedy żaden z device'ów przypisanych do danego segmentu nie ma już wolnych stron, kolejny `INSERT` kończy się błędem „Sybase database's ... segment is full" — nawet jeśli inne segmenty tej samej bazy albo inne bazy na tym samym serwerze mają mnóstwo wolnego miejsca. To jest bezpośrednia konsekwencja modelu segmentowego: wolne miejsce nie przepływa samo między segmentami, trzeba je świadomie przydzielić poleceniem `alter database ... on NazwaDevice`, opisanym na stronie Powiększanie baz i urządzeń.

Z życia wzięte

> **Z ŻYCIA WZIĘTE**
>
> Tomek dostaje zadanie sprawdzenia archiwum płac na syb01 i próbuje połączyć się isql-em przez nazwę hosta systemu operacyjnego zamiast nazwy serwera logicznego z pliku interfaces. Połączenie kończy się komunikatem o nierozpoznanym serwerze, mimo że syb01 odpowiada na ping bez problemu. Dopiero podanie nazwy SRV_DB01 z pliku interfaces pozwala się połączyć i zobaczyć bazę PLACE_ARCH. Tomek zapisuje sobie różnicę między nazwą hosta a nazwą serwera logicznego, żeby drugi raz nie szukać tego przez pół godziny.

Częste błędy

Mylenie device'a z bazą danych. Powiększenie device'a poleceniem `disk resize`, opisanym na stronie Powiększanie baz i urządzeń, nie powiększa automatycznie żadnej bazy — dokłada tylko wolne miejsce na device'ie. Dopiero `alter database ... on NazwaDevice` przydziela to miejsce konkretnej bazie.

Brak oddzielnego device'a pod log transakcyjny. Bazy zakładane „na szybko" jednym device'em pod dane i log naraz tracą jedną z podstawowych linii obrony przy pełnym dysku — log i dane rosną razem i razem się kończą.

Nieznajomość nazwy serwera logicznego. Próba połączenia się `isql`-em przez nazwę hosta systemu operacyjnego zamiast nazwy z pliku `interfaces` kończy się komunikatem o nierozpoznanym serwerze, mimo że host odpowiada na ping.

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

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

Revision #42
Created 2026-09-25 00:00:42 UTC by Tomasz Konieczny
Updated 2026-10-26 14:08:01 UTC by Tomasz Konieczny