Rozmiar bazy: pliki, tablespace'y, autoextend

> Tabela, z której skasowaliśmy dziewięćdziesiąt procent wierszy, w większości silników nie oddaje dyskowi ani jednego bajtu miejsca — ta strona tłumaczy dlaczego, i co faktycznie trzeba zrobić, żeby to miejsce odzyskać.

Skróty użyte na tej stronie (3)

Skrót	Co znaczy
DBCC	Database Console Commands: rodzina poleceń MSSQL do sprawdzania i naprawy spójności bazy, na przykład DBCC CHECKDB
GB	Gigabyte: miliard bajtów, jednostka pojemności pamięci i plików, tu najczęściej rozmiar modelu albo pamięci karty graficznej
SQL	Structured Query Language: język zapytań do relacyjnych baz danych

Fiszka

TYLKO ODCZYT

Rozmiar bazy: PostgreSQL

```
SELECT pg_size_pretty(pg_database_size('NazwaBazy'));
```

Przykład:

```
 pg_size_pretty
----------------
 42 GB
(1 row)
```

TYLKO ODCZYT

Rozmiar największych tabel i indeksów: PostgreSQL

```
SELECT relname, pg_size_pretty(pg_total_relation_size(relid)) AS rozmiar
FROM pg_catalog.pg_statio_user_tables
ORDER BY pg_total_relation_size(relid) DESC
LIMIT 5;
```

`pg_total_relation_size` liczy tabelę razem z jej indeksami i danymi TOAST — samo `pg_relation_size` pokazałoby tylko dane bez tego dodatku.

Przykład:

```
   relname   | rozmiar
-------------+---------
 zamowienia  | 8412 MB
 pozycje     | 6103 MB
(2 rows)
```

TYLKO ODCZYT

Wolne miejsce w tablespace: PostgreSQL

```
SELECT spcname, pg_size_pretty(pg_tablespace_size(spcname)) AS rozmiar
FROM pg_tablespace;
```

PostgreSQL nie prowadzi osobnej księgowości „wolnego miejsca w tablespace” w tym samym sensie co Oracle — wolne miejsce sprawdzamy na poziomie systemu plików, na którym leży katalog danego tablespace, poleceniem `df`.

Przykład:

```
  spcname   | rozmiar
------------+---------
 pg_default | 42 GB
(1 row)
```

TYLKO ODCZYT

Rozmiar bazy i tabel: Oracle

```
SELECT tablespace_name, round(sum(bytes)/1024/1024/1024, 2) AS rozmiar_gb
FROM dba_data_files
GROUP BY tablespace_name;
```

Przykład:

```
TABLESPACE_NAME    ROZMIAR_GB
------------------ ----------
USERS                   85.50
DANE                   210.75
```

TYLKO ODCZYT

Wolne miejsce i ustawienie autoextend: Oracle

```
SELECT file_name, autoextensible, maxbytes/1024/1024/1024 AS maxsize_gb
FROM dba_data_files
WHERE tablespace_name = 'NazwaTablespace';
```

Przykład:

```
FILE_NAME                          AUTOEXTENSIBLE   MAXSIZE_GB
---------------------------------- ---------------- ------------
+DATA/orcl/datafile/dane.dbf          YES                      500
```

TYLKO ODCZYT

Rozmiar bazy i plików: SQL Server

```
EXEC sp_spaceused;
```

Przykład:

```
database_name   database_size   unallocated space
--------------- --------------- -------------------
produkcja       8192.00 MB      612.25 MB
```

SQL = Structured Query Language: język zapytań do relacyjnych baz danych · **exec** = zastąp bieżący proces powłoki wskazanym programem, bez tworzenia nowego procesu

TYLKO ODCZYT

Ustawienie autogrowth: SQL Server

```
SELECT name, growth, is_percent_growth, max_size
FROM sys.database_files;
```

`growth` w jednostkach 8 KB stron przy `is_percent_growth = 0`, albo w procentach przy wartości `1`. `max_size` równe `-1` oznacza brak limitu — plik rośnie, dopóki starcza miejsca na dysku.

Przykład:

```
name           growth   is_percent_growth   max_size
-------------- -------- ------------------- ----------
produkcja      12800    0                   -1
produkcja_log  10       1                   268435456
```

SQL = Structured Query Language: język zapytań do relacyjnych baz danych

TYLKO ODCZYT

Rozmiar bazy i tabel: MySQL

```
SELECT table_name, round((data_length + index_length)/1024/1024, 2) AS rozmiar_mb
FROM information_schema.tables
WHERE table_schema = 'NazwaBazy'
ORDER BY (data_length + index_length) DESC
LIMIT 5;
```

Przykład:

```
+------------+------------+
| table_name | rozmiar_mb |
+------------+------------+
| zamowienia |     3841.5 |
| pozycje    |     2910.2 |
```

```
+------------+------------+
```

Po co to istnieje

> **Pojęcie: przestrzeń tabel (tablespace).** Logiczny kontener grupujący jeden albo więcej fizycznych plików na dysku, do którego silnik przypisuje tabele i indeksy w momencie ich tworzenia. Tablespace jest warstwą pośrednią między obiektem bazy danych (tabelą, indeksem) a konkretnym plikiem systemowym — pozwala rozłożyć dane na kilka dysków fizycznych, przenieść wybrane tabele na szybszy albo wolniejszy nośnik, i zarządzać przydziałem miejsca niezależnie od tego, ile tabel akurat w danym tablespace istnieje.

Rozmiar bazy widziany „z zewnątrz” — suma rozmiarów jej plików na dysku — nie jest tym samym, co rozmiar rzeczywiście przechowywanych danych. Plik danych, który raz urósł do określonego rozmiaru, w większości silników nie kurczy się sam z siebie, nawet jeśli większość zapisanych w nim wierszy zostanie później skasowana — miejsce zwolnione przez `DELETE` zostaje oznaczone jako wolne wewnątrz tego pliku, gotowe do ponownego użycia przez przyszłe zapisy tej samej tabeli, ale plik na poziomie systemu operacyjnego pozostaje tej samej wielkości. Baza, która przez rok rosła i malała wielokrotnie, może zajmować na dysku znacznie więcej, niż wynosi suma danych faktycznie w niej obecnych w danej chwili — różnicę między tymi dwiema liczbami trzeba świadomie mierzyć osobno, nie zakładać, że rozmiar pliku równa się rozmiarowi danych.

Jak to działa

Struktura plikowa czterech silników. Każdy z czterech silników organizuje swoje pliki inaczej, choć koncepcja przestrzeni tabel powtarza się pod różnymi nazwami niemal wszędzie.

Silnik	Jednostka organizacyjna	Pliki fizyczne	Automatyczny wzrost
PostgreSQL	katalog bazy wewnątrz klastra; tablespace jako dowiązanie symboliczne do innej lokalizacji	pliki segmentów po 1 GB na tabelę/indeks, w katalogu `base/`	rośnie automatycznie w miarę zapisu, bez osobnego ustawienia „autoextend”
Oracle	tablespace złożony z jednego lub wielu datafile	pliki `.dbf` (datafile)	`AUTOEXTEND ON` z opcjonalnym `MAXSIZE`, ustawiane per plik
SQL Server	filegroup grupujący pliki danych	plik główny `.mdf`, dodatkowe pliki danych `.ndf`, plik logu `.ldf`	`autogrowth` ustawiane per plik, w MB albo procentach, z opcjonalnym limitem maksymalnym

Silnik	Jednostka organizacyjna	Pliki fizyczne	Automatyczny wzrost
MySQL / MariaDB (InnoDB)	tablespace systemowy (`ibdata1`) albo osobny plik na tabelę	`ibdata1` (współdzielony) albo `NazwaTabeli.ibd` przy `innodb_file_per_table = ON` (domyślnie od 5.6)	pliki `.ibd` rosną automatycznie wraz z danymi tabeli

> **Pojęcie: autoextend / autogrowth.** Mechanizm automatycznego powiększania pliku danych przez silnik w momencie, gdy zapis nie mieści się już w jego bieżącym rozmiarze, bez potrzeby ręcznej interwencji administratora. Wygodny jako zabezpieczenie przed nagłym zapełnieniem pliku w środku dnia roboczego, ale nieskonfigurowany limit maksymalny (`MAXSIZE` w Oracle, limit w MB przy `autogrowth` w SQL Server) pozwala plikowi rosnąć aż do wyczerpania całego dostępnego miejsca na dysku, zabierając je również innym plikom i procesom systemowym współdzielącym ten sam wolumen.

Dlaczego DELETE nie zwalnia miejsca na dysku

`DELETE` w każdym z czterech silników usuwa wiersz logicznie — czyni go niewidocznym dla przyszłych zapytań — ale w żadnym z nich nie kurczy od razu pliku fizycznego, w którym ten wiersz leżał. Miejsce po skasowanym wierszu zostaje oznaczone jako wolne wewnątrz istniejącej struktury pliku i czeka na ponowne wykorzystanie przez kolejne zapisy tej samej tabeli — sam plik na dysku pozostaje tej samej wielkości, bo zmniejszenie pliku systemowego jest operacją kosztowną i blokującą, więc żaden silnik nie wykonuje jej automatycznie po każdym `DELETE`. Tabela, z której skasowano większość wierszy, ale do której nikt nic więcej nie wstawia, może latami zajmować tyle samo miejsca na dysku, ile zajmowała przed usunięciem danych.

Odzyskanie tego miejsca wymaga świadomej operacji reorganizacji, różnej w każdym silniku:

PostgreSQL: `VACUUM FULL`**.** Przepisuje całą tabelę do nowego, zwartego pliku bez martwych wersji wierszy, a stary plik zwalnia z powrotem systemowi plików. W odróżnieniu od zwykłego `VACUUM`, który tylko oznacza miejsce jako wolne do ponownego użycia bez zwracania go systemowi, `VACUUM FULL` blokuje tabelę wyłącznie na czas operacji (`ACCESS EXCLUSIVE`) — żadne zapytanie, ani odczyt, ani zapis, nie dostanie się do tabeli, dopóki operacja trwa.

SQL Server: `DBCC SHRINKFILE`**.** Zmniejsza fizyczny rozmiar wskazanego pliku danych albo logu, przesuwając zajęte strony w kierunku początku pliku i obcinając nieużywaną resztę. Operacja jest blokująca dla obiektów, których dotyczy przesunięcie stron, i notorycznie fragmentuje indeksy — przesuwane podczas shrink strony trafiają w przypadkowej kolejności, rozbijając wcześniejszy porządek fizyczny indeksu, temat rozwinięty na stronie Fragmentacja i przebudowa indeksów.

MySQL: `OPTIMIZE TABLE`**.** Dla tabel InnoDB odtwarza tabelę od nowa w nowym pliku `.ibd` (przy `innodb_file_per_table` włączonym), analogicznie do `VACUUM FULL` w PostgreSQL — blokuje tabelę na czas operacji i wymaga tymczasowo dodatkowego miejsca na dysku równego mniej więcej rozmiarowi przepisywanej tabeli.

Oracle: reorganizacja tablespace. Oracle nie ma jednego prostego odpowiednika — miejsce w obrębie tablespace odzyskuje się przez przeniesienie segmentu (`ALTER TABLE ... MOVE`) do nowego miejsca w tym samym albo innym tablespace, a następnie przebudowanie zależnych indeksów, które po przeniesieniu tabeli tracą ważność (`ALTER INDEX ... REBUILD`). Zmniejszenie samego pliku datafile (`ALTER DATABASE DATAFILE ... RESIZE`) jest możliwe tylko wtedy, gdy wolne miejsce znajduje się na samym końcu pliku — segment przeniesiony gdzie indziej w środku pliku wcześniej blokuje resize dopóki nie zwolni się przestrzeń na końcu.

Każda z tych operacji jest blokująca w takim czy innym stopniu i fragmentująca indeksy w trakcie samej reorganizacji — żadnej z nich nie uruchamiamy odruchowo, „bo baza urosła”, tylko po konkretnej decyzji, że odzyskane miejsce jest warte przestoju czy spowolnienia, jakie operacja spowoduje, najlepiej w oknie serwisowym poza godzinami szczytu.

Prognozowanie przyrostu

Regularne zapisywanie rozmiaru bazy i jej największych tabel w czasie — nie tylko jednorazowy odczyt w chwili problemu — pozwala policzyć tempo przyrostu i oszacować, za ile tygodni czy miesięcy bieżący wolumen się zapełni, zanim to się faktycznie stanie. Baza rosnąca liniowo o kilka gigabajtów tygodniowo pozwala zaplanować rozszerzenie dysku z wyprzedzeniem, zamiast reagować dopiero na alert o przekroczonym progu wolnego miejsca opisany na stronie Co monitorujemy w bazie danych — różnica między planowanym rozszerzeniem pojemności a awaryjnym dokładaniem dysku w środku nocy jest różnicą między rutynową czynnością a incydentem.

Częste błędy

Traktowanie DELETE jako sposobu na zwolnienie miejsca na dysku. Opisane wyżej — skasowanie milionów wierszy z przekonaniem, że baza natychmiast skurczy się na dysku, kończy się rozczarowaniem: plik pozostaje tej samej wielkości, dopóki ktoś nie wykona świadomej operacji reorganizacji.

Reorganizacja (shrink) wykonywana rutynowo, bez potrzeby. `DBCC SHRINKFILE` czy `VACUUM FULL` uruchamiane cyklicznie „dla porządku” zamiast w odpowiedzi na konkretną potrzebę odzyskania miejsca fragmentuje indeksy przy każdym takim przebiegu i blokuje tabelę bez realnej korzyści — miejsce, które i tak zostanie ponownie zajęte przez kolejne zapisy tej samej tabeli, nie musi być oddawane systemowi plików tylko po to, żeby chwilę później znów je zająć.

Brak limitu maksymalnego na autoextend/autogrowth. Plik z `AUTOEXTEND ON` bez ustawionego `MAXSIZE`, albo z `autogrowth` bez limitu, rośnie, dopóki starcza miejsca na całym wolumenie dyskowym — łącznie z miejscem potrzebnym innym bazom, plikom systemowym czy logom tej samej maszyny. Zapełnienie całego dysku w ten sposób bywa trudniejsze do opanowania niż zaplanowane, kontrolowane powiększenie tablespace o konkretną, przemyślaną wartość.

□□ Ćwiczenia

> Zadania sprawdzające.

Ćwiczenie 1 — diagnoza rozbieżności rozmiarów

Tabela `archiwum_zamowien` w PostgreSQL zajmuje na dysku 40 GB, ale zapytanie `SELECT count(*)` zwraca liczbę wierszy odpowiadającą, wedle szacunków, najwyżej 4 GB rzeczywistych danych. Jakie polecenie sprawdzimy, żeby potwierdzić, że różnica to bloat, i jaką operację rozważymy, żeby odzyskać miejsce?

Odpowiedź

Sprawdzamy `SELECT last_vacuum, last_autovacuum FROM pg_stat_user_tables WHERE relname = 'archiwum_zamowien';` — jeśli tabela od dawna nie była odkurzana mimo dużej liczby usunięć czy aktualizacji, potwierdza to narastający bloat. Odzyskanie miejsca wymaga `VACUUM FULL archiwum_zamowien;`, wykonanego świadomie w oknie serwisowym, bo operacja blokuje tabelę na czas przepisania jej do nowego, zwartego pliku.

Pytania kontrolne

1. Dlaczego `DBCC SHRINKFILE` w SQL Server jest operacją, którą wykonujemy świadomie i rzadko, a nie rutynowo?

Odpowiedź

Bo przesuwanie zajętych stron w kierunku początku pliku w celu obcięcia nieużywanej reszty blokuje dostęp do obiektów, których dotyczy, i przenosi strony indeksów w przypadkowej kolejności, rozbijając ich wcześniejszy porządek fizyczny — efektem rutynowego shrinku jest

silnie sfragmentowany indeks, który potem trzeba dodatkowo przebudować, więc operacja bywa kosztowniejsza niż korzyść z odzyskanego miejsca.

2. Czym różni się plik `.ibd` przy włączonym `innodb_file_per_table` od współdzielonego tablespace `ibdata1`?

Odpowiedź

Przy `innodb_file_per_table` każda tabela dostaje własny plik `.ibd`, więc operacje jak `OPTIMIZE TABLE` czy usunięcie tabeli (`DROP TABLE`) od razu zwalniają miejsce na poziomie systemu plików dla tej konkretnej tabeli. Współdzielony tablespace `ibdata1` trzyma dane wielu tabel w jednym pliku — usunięcie jednej tabeli zwalnia miejsce tylko wewnątrz tego wspólnego pliku, do ponownego użycia przez inne tabele InnoDB, ale sam plik `ibdata1` w praktyce nigdy się nie kurczy.

Powiązane: Fragmentacja i przebudowa indeksów · Co monitorujemy w bazie danych · MVCC: dlaczego czytający nie blokuje piszącego · Log transakcyjny: WAL, redo, transaction log

Źródła: dokumentacja PostgreSQL, rozdział „Database File Layout” i „Routine Vacuuming” · Oracle Database Administrator's Guide, rozdział „Managing Tablespaces” · Microsoft Docs, „Database Files and Filegroups” i „DBCC SHRINKFILE” · dokumentacja MySQL, rozdział „InnoDB File-Per-Table Tablespaces”

Revision #44
Created 2026-09-24 23:07:50 UTC by Tomasz Konieczny
Updated 2026-10-26 14:12:00 UTC by Tomasz Konieczny