Rozpoznanie i diagnostyka

Wersja i edycja, rozmiary plików i tabel, sesje, blokady oraz indeksy.

- Rozpoznanie instancji i baz
- Rozmiary tabel i indeksów
- Konta serwisowe i uprawnienia do monitoringu
- Rozpoznanie procedur składowanych
- Sesje, zapytania i blokady
- Indeksy: status, fragmentacja, przebudowa

Rozpoznanie instancji i baz

Poziom: podstawy **Dotyczy: SQL Server 2016–2022** **Czas: ~15 min**

Zestaw zapytań, od których zaczynamy pracę na nieznanej wcześniej instancji — jaka to wersja i edycja, jakie bazy w niej siedzą, w jakim są trybie recovery i ile zajmują na dysku.

Skróty użyte na tej stronie (3)

Skrót	Co znaczy
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
SSMS	SQL Server Management Studio: graficzna konsola MSSQL

Fiszka

TYLKO ODCZYT

Sprawdzenie pełnej wersji silnika: @@VERSION

```
SELECT @@VERSION;
```

Sprawdzamy pełny opis wersji silnika — numer buildu, poprawkę CU, edycję i system operacyjny, na którym stoi instancja. Wynik jest jednym długim tekstem, wygodnym do wklejenia w zgłoszenie do supportu, ale niewygodnym do dalszego przetwarzania — do tego celu lepiej pasuje `SERVERPROPERTY`.

Przykład:

```
--------------------------------------------------------------------------------
Microsoft SQL Server 2019 (RTM-CU18) (KB5017593) - 15.0.4223.1 (X64)
        Sep  8 2022 12:34:56
        Copyright (C) 2019 Microsoft Corporation
        Standard Edition (64-bit) on Windows Server 2019 Standard 10.0 <X64> (Build
```

```
17763: )


(1 rows affected)
```

TYLKO ODCZYT

Sprawdzenie wersji edycji silnika MSSQL: SERVERPROPERTY

```
SELECT
    SERVERPROPERTY('Edition') AS Edycja,
    SERVERPROPERTY('ProductVersion') AS WersjaProduktu,
    SERVERPROPERTY('ProductLevel') AS PoziomProduktu,
    SERVERPROPERTY('EngineEdition') AS EngineEdition;
```

Sprawdzamy edycję silnika w formie gotowej do porównania w skrypcie — `EngineEdition` zwraca liczbę (1 = Personal/Desktop, 2 = Standard, 3 = Enterprise, 4 = Express, 5 = SQL Database Azure, 8 = Managed Instance Azure), a `ProductLevel` mówi, czy stoimy na RTM czy na konkretnym Service Packu. Limity poszczególnych edycji opisujemy na stronie Architektura MSSQL: instancja, baza, pliki MDF/NDF/LDF.

Przykład:

```
Edycja                                  WersjaProduktu  PoziomProduktu  EngineEdition
--------------------------------------- --------------- --------------- -------------
Standard Edition (64-bit)                15.0.4223.1     RTM             2


(1 rows affected)
```

MSSQL = Microsoft SQL Server: silnik bazy danych Microsoftu, na nim w Zielonej stoi księgowość Optima · **SQL** = Structured Query Language: język zapytań do relacyjnych baz danych · **MDF** = Master Data File: główny plik danych bazy MSSQL

TYLKO ODCZYT

Wylistowanie wszystkich baz danych na instancji: sys.databases

```
SELECT name AS NazwaBazy, state_desc AS Stan, recovery_model_desc AS TrybRecovery
FROM sys.databases
ORDER BY name;
```

Sprawdzamy, jakie bazy w ogóle istnieją na instancji, w jakim są stanie (`ONLINE`, `RESTORING`, `OFFLINE`) i w jakim trybie recovery każda z nich pracuje — pierwsze zapytanie na nieznanej instancji, zanim zaczniemy szukać konkretnego problemu.

Przykład:

```
NazwaBazy    Stan     TrybRecovery
------------ -------- ------------
master       ONLINE   SIMPLE
model        ONLINE   FULL
msdb         ONLINE   SIMPLE
tempdb       ONLINE   SIMPLE
CDN_Zielona       ONLINE   FULL
CDN_Zielona_TEST       ONLINE   SIMPLE

(6 rows affected)
```

□□ TYLKO ODCZYT

Sprawdzenie wszystkich plików, rozmiarów oraz trybu recovery: sys.master_files

```
SELECT
    d.name AS NazwaBazy,
    d.recovery_model_desc AS TrybRecovery,
    mf.type_desc AS TypPliku,
    mf.name AS NazwaLogiczna,
    CONVERT(decimal(12,2), mf.size * 8 / 1024.0) AS RozmiarMB
FROM sys.databases d
JOIN sys.master_files mf ON d.database_id = mf.database_id
ORDER BY d.name, mf.type_desc;
```

Sprawdzamy, z jakich fizycznych plików (danych i logu) składa się każda baza na instancji i ile miejsca każdy z nich zajmuje — jedno zapytanie zamiast klikania po właściwościach każdej bazy z osobna w SSMS. `sys.master_files` jest widokiem katalogowym na poziomie instancji, więc widzimy tu wszystkie bazy naraz, nie tylko tę, na której akurat siedzimy.

Przykład:

```
NazwaBazy   TrybRecovery   TypPliku   NazwaLogiczna     RozmiarMB
----------- ------------- ---------- ----------------- ----------
CDN_Zielona      FULL           ROWS       CDN_Zielona            8192.00
CDN_Zielona      FULL           LOG        CDN_Zielona_log        2048.00
CDN_Zielona_TEST      SIMPLE         ROWS       CDN_Zielona_TEST            512.00
CDN_Zielona_TEST      SIMPLE         LOG        CDN_Zielona_TEST_log         64.00

(4 rows affected)
```

□□ **SSMS** = SQL Server Management Studio: graficzna konsola MSSQL · **join** = połącz dwa pliki tekstowe po wspólnej kolumnie, jak złączenie tabel w bazie danych

□□ **TYLKO ODCZYT**

Sprawdzenie zajętości plików bazy danych: FILEPROPERTY

```
DECLARE @NazwaBazy NVARCHAR(128) = N'NazwaBazy';
DECLARE @sql NVARCHAR(MAX);

SET @sql = N'
USE ' + QUOTENAME(@NazwaBazy) + N';
SELECT
    ''' + @NazwaBazy + N''' AS NazwaBazy,
    mf.name AS NazwaPlikuLogiczna,
    mf.physical_name AS SciezkaFizyczna,
    mf.type_desc AS TypPliku,
    mf.size / 128.0 AS RozmiarMB,
    FILEPROPERTY(mf.name, ''SpaceUsed'') / 128.0 AS ZajeteMB,
    CAST((FILEPROPERTY(mf.name, ''SpaceUsed'') / 128.0) / (mf.size / 128.0) * 100 AS
decimal(5,2)) AS ZajeteProcent
FROM sys.master_files mf
WHERE mf.database_id = DB_ID(''' + @NazwaBazy + N''');';

EXEC sp_executesql @sql;
```

Sprawdzamy nie tylko rozmiar pliku na dysku, ale realną zajętość wewnątrz niego — różnica między `RozmiarMB` a `ZajeteMB` to zapas, o który plik urósł w wyniku autogrowth, ale którego dane jeszcze nie wypełniły. `FILEPROPERTY` działa wyłącznie w kontekście bazy, do której się odwołuje, dlatego zapytanie buduje dynamiczne `USE` przez `sp_executesql` — dzięki temu można je wywołać z poziomu `master`, podając nazwę docelowej bazy jako parametr, zamiast

ręcznie przełączać kontekst.

Przykład:

```
NazwaBazy   NazwaPlikuLogiczna   SciezkaFizyczna                          TypPliku
RozmiarMB   ZajeteMB   ZajeteProcent
----------- -------------------- ---------------------------------------- ----------
----------- ---------- -------------
CDN_Zielona      CDN_Zielona
D:\MSSQL\DATA\CDN_Zielona.mdf                 ROWS       8192.00     6534.25    79.76
CDN_Zielona      CDN_Zielona_log
E:\MSSQL\LOG\CDN_Zielona_log.ldf              LOG        2048.00     412.00     20.12

(2 rows affected)
```

Ten sam pomiar dla wszystkich baz online naraz, bez podawania nazwy z osobna, budujemy kursorem po `sys.databases`, sklejając wynik z każdej bazy przez `UNION ALL` — przydatne przy comiesięcznym przeglądzie zajętości całej instancji naraz, zamiast bazy po bazie.

declare = zadeklaruj zmienną powłoki z określonym typem, na przykład jako liczbę całkowitą albo tablicę · **SET** = Switch Embedded Teaming: zespół kart sieciowych zbudowany w przełączniku wirtualnym Hyper-V; łączy dwie karty w jedno łącze bez ustawień po stronie przełącznika fizycznego, więc bez LACP · **exec** = zastąp bieżący proces powłoki wskazanym programem, bez tworzenia nowego procesu

TYLKO ODCZYT

Sprawdzenie wielkości danych w msdb: sys.objects i sys.allocation_units

```
USE msdb;
GO

SELECT TOP (10)
    o.object_id,
    SCHEMA_NAME(o.schema_id) + '.' + o.name AS Obiekt,
    o.type,
    i.total_rows AS Wiersze,
    i.total_size AS RozmiarMB
FROM sys.objects o
```

```
JOIN (
    SELECT
        i.object_id,
        total_size = CAST(SUM(a.total_pages) * 8. / 1024 AS DECIMAL(18,2)),
        total_rows = SUM(CASE WHEN i.index_id IN (0, 1) AND a.type = 1 THEN p.rows END)
    FROM sys.indexes i
    JOIN sys.partitions p ON i.object_id = p.object_id AND i.index_id = p.index_id
    JOIN sys.allocation_units a ON p.partition_id = a.container_id
    WHERE i.is_disabled = 0 AND i.is_hypothetical = 0
    GROUP BY i.object_id
) i ON o.object_id = i.object_id
WHERE o.type IN ('V', 'U', 'S')
ORDER BY i.total_size DESC;
```

Sprawdzamy, które tabele wewnątrz `msdb` faktycznie zajmują miejsce — najczęściej okazuje się, że winowajcą jest historia backupów (`backupset`, `backupfile`) albo historia zadań SQL Agent (`sysjobhistory`), która latami nie była czyszczona. Purge tej historii opisujemy na stronie Backup baz danych.

Przykład:

```
object_id    Obiekt                    type   Wiersze     RozmiarMB
------------ ------------------------- ------ ----------- ----------
1361911459   dbo.backupfile            U      184320      612.50
1345911402   dbo.sysjobhistory         U      98304       284.75
1297911288   dbo.backupset             U      52480       118.25

(3 rows affected)
```

SQL = Structured Query Language: język zapytań do relacyjnych baz danych · **join** = połącz dwa pliki tekstowe po wspólnej kolumnie, jak złączenie tabel w bazie danych

Z życia wzięte

Z ŻYCIA WZIĘTE

W środę rano firma przejmuje od zewnętrznej firmy stary serwer sql01 po zwolnionym administratorze, bez żadnej dokumentacji poza hasłem do konta sa. Zanim ktokolwiek

dotknie czegokolwiek, sprawdzamy wersję silnika i listę baz zapytaniem do sys.databases. Okazuje się, że oprócz CDN_Zielona stoi tam jeszcze zapomniana baza testowa w trybie SIMPLE, o której nikt w zespole nie wiedział. Dopiero ten przegląd pozwala bezpiecznie zaplanować dalsze porządki.

Po co to istnieje

> **Pojęcie: sp_helpdb.** Wbudowana procedura systemowa zwracająca skróconą wersję informacji o bazie w jednym wywołaniu — rozmiar, właściciela, tryb recovery i listę plików. Wygodna do szybkiego zerknięcia interaktywnie, ale mniej elastyczna niż zapytania bezpośrednio do `sys.databases` i `sys.master_files`, bo nie da się jej łatwo przefiltrować ani połączyć z innymi danymi.

```
EXEC sp_helpdb N'NazwaBazy';
```

Częste błędy

Wnioskowanie o rozmiarze bazy z samego `sys.databases`. Ten widok katalogowy nie trzyma kolumny z rozmiarem — rozmiar trzeba policzyć z `sys.master_files`, sumując wszystkie pliki danej bazy, albo odczytać z `FILEPROPERTY`, jak w zapytaniu wyżej.

Ufanie samemu rozmiarowi pliku bez sprawdzenia zajętości. Plik `.mdf` mający 100 GB na dysku może mieć w środku 20 GB realnych danych, jeśli baza raz urosła pod duże obciążenie, a potem dane usunięto — miejsce zostaje zarezerwowane przez silnik i nie zwalnia się samo, dopóki nie wykonamy `DBCC SHRINKFILE`, co jednak ma swoje poważne wady opisane na stronie Log transakcyjny: rozmiar, shrink, zapchanie.

Powiązane: Architektura MSSQL: instancja, baza, pliki MDF/NDF/LDF · sqlcmd i narzędzia klienckie · Rozmiary tabel i indeksów · Backup baz danych

Źródła: dokumentacja Microsoft Learn, sekcja „sys.databases", „sys.master_files" i „SERVERPROPERTY"

Rozmiary tabel i indeksów

Poziom: podstawy **Dotyczy: SQL Server 2016–2022** **Czas: ~15 min**

Sprawdzenie, które tabele i indeksy faktycznie zajmują miejsce w bazie, zanim zaczniemy planować reindeksację albo szukać kandydata do archiwizacji.

Skróty użyte na tej stronie (1)

Skrót	Co znaczy
SQL	Structured Query Language: język zapytań do relacyjnych baz danych

Fiszka

TYLKO ODCZYT

Sprawdzenie rozmiaru pojedynczej tabeli: sp_spaceused

```
EXEC sp_spaceused N'NazwaTabeli';
```

Sprawdzamy rozmiar konkretnej tabeli wraz z jej indeksami — wbudowana procedura systemowa, najszybszy sposób na zerknięcie w jeden obiekt bez pisania własnego zapytania.

Przykład:

```
name          rows      reserved    data      index_size   unused
------------- --------- ----------- --------- ------------ --------
Zamowienia    1048576   2145280 KB  1802112 KB 340992 KB   2176 KB


(1 rows affected)
```

exec = zastąp bieżący proces powłoki wskazanym programem, bez tworzenia nowego procesu

TYLKO ODCZYT

Sortowanie tabel po rozmiarze: sys.dm_db_partition_stats

```
SELECT
    t.name AS NazwaTabeli,
    SUM(ps.reserved_page_count) * 8 AS ZarezerwowaneKB,
    SUM(ps.used_page_count) * 8 AS ZajeteKB
FROM sys.tables t
JOIN sys.dm_db_partition_stats ps ON t.object_id = ps.object_id
GROUP BY t.name
ORDER BY SUM(ps.used_page_count) DESC;
```

Sprawdzamy, które tabele w bieżącej bazie zajmują najwięcej miejsca — zapytanie działa w kontekście bazy, na której jest wykonane, więc wymaga wcześniejszego `USE NazwaBazy`. `sys.dm_db_partition_stats` liczy strony osobno dla każdej partycji każdego indeksu, dlatego sumujemy je po `object_id` tabeli, żeby dostać jeden wiersz na tabelę zamiast jednego na indeks.

Przykład:

```
NazwaTabeli        ZarezerwowaneKB   ZajeteKB
------------------ ----------------- ----------
Zamowienia         2145280           1802112
PozycjeZamowien    987136            845056
Klienci            65536             52224

(3 rows affected)
```

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

□□ **TYLKO ODCZYT**

Zajętość poszczególnych tabel wraz z indeksami: sys.allocation_units

```
SELECT TOP (30)
    SCHEMA_NAME(tab.schema_id) + '.' + tab.name AS Tabela,
    CAST(SUM(spc.used_pages * 8) / 1024.00 AS numeric(36, 2)) AS ZajeteMB,
    CAST(SUM(spc.total_pages * 8) / 1024.00 AS numeric(36, 2)) AS AlokowaneMB
FROM sys.tables tab
JOIN sys.indexes ind ON tab.object_id = ind.object_id
JOIN sys.partitions part ON ind.object_id = part.object_id AND ind.index_id =
```

```
part.index_id
JOIN sys.allocation_units spc ON part.partition_id = spc.container_id
GROUP BY SCHEMA_NAME(tab.schema_id) + '.' + tab.name
ORDER BY SUM(spc.used_pages) DESC;
```

Sprawdzamy trzydzieści największych tabel w bieżącej bazie, tym razem licząc bezpośrednio na poziomie jednostek alokacji, z rozróżnieniem miejsca faktycznie zajętego (`used_pages`) od zarezerwowanego dla obiektu (`total_pages`) — różnica między tymi dwiema kolumnami to zapas, który zwolniłby `DBCC SHRINKFILE`, gdybyśmy go wykonali, ale tego typowo nie robimy z powodów opisanych na stronie Log transakcyjny: rozmiar, shrink, zapchanie.

Przykład:

```
Tabela               ZajeteMB    AlokowaneMB
-------------------- ----------- ------------
dbo.Zamowienia        1760.25     1784.50
dbo.PozycjeZamowien    825.50     841.75
dbo.Klienci             51.00      52.25

(3 rows affected)
```

DBCC = Database Console Commands: rodzina poleceń MSSQL do sprawdzania i naprawy spójności bazy, na przykład DBCC CHECKDB · **join** = połącz dwa pliki tekstowe po wspólnej kolumnie, jak złączenie tabel w bazie danych

ZMIENIA SYSTEM

Weryfikacja TotalIndexSize przed reindeksacją

```
SELECT
    SUM(p.used_page_count) * 8 / 1024.0 / 1024.0 AS TotalIndexSizeGB
FROM sys.dm_db_partition_stats p
JOIN sys.indexes i ON p.object_id = i.object_id AND p.index_id = i.index_id
WHERE i.type > 0;
```

Sprawdzamy łączny rozmiar wszystkich indeksów w bieżącej bazie, z pominięciem samych stert (`index_id = 0`, `type = 0`) — ta liczba jest orientacyjnym punktem odniesienia przy planowaniu reindeksacji, bo `ALTER INDEX ... REBUILD` w trakcie działania potrzebuje dodatkowego miejsca na dysku, w skrajnym przypadku zbliżonego do rozmiaru przebudowywanego indeksu, dopóki stara wersja nie zostanie zwolniona po zakończeniu

operacji.

Przykład:

```
TotalIndexSizeGB
------------------
42.780000


(1 rows affected)
```

Uważaj: przy rebuild bez `ONLINE` silnik trzyma jednocześnie starą i nową wersję przebudowywanego indeksu do samego końca operacji — na dysku, na którym zostało niewiele wolnego miejsca, duży rebuild może się nie zmieścić i przerwać z błędem `Msg 1101, Level 17, State 12, Could not allocate a new page for database`.

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

Z życia wzięte

Z ŻYCIA WZIĘTE

W ostatni dzień miesiąca, kiedy księgowość pracuje do nocy, monitoring zgłasza, że dysk z danymi CDN_Zielona ma zaledwie kilkanaście procent wolnego miejsca. Zamiast zgadywać, sprawdzamy zapytaniem do sys.dm_db_partition_stats, które tabele zajmują najwięcej miejsca. Okazuje się, że tabela Zamowienia urosła w tym miesiącu wyjątkowo mocno, bo dział sprzedaży importował archiwalne zamówienia z poprzednich lat. Ten wynik pozwala nam dokupić miejsce na dysku, zamiast w nocy walczyć z zapchaną bazą.

Po co to istnieje

Rozmiar tabeli i jej indeksów to jedna z pierwszych rzeczy, o które pytamy, zanim podejmiemy dowolną decyzję operacyjną — czy starczy miejsca na reindeksację, czy backup zmieści się w oknie serwisowym, czy tabela jest kandydatem do partycjonowania albo archiwizacji starych wierszy. Bez tych liczb każda taka decyzja jest zgadywaniem.

> **Pojęcie: strona (page) i extent.** Silnik SQL Server przechowuje dane w stronach po 8 KB — to najmniejsza jednostka odczytu i zapisu z dysku. Osiem kolejnych stron tworzy extent (64 KB); większość zapytań o rozmiar liczy właśnie strony i mnoży je przez 8 KB, żeby dostać wynik w kilobajtach, tak jak w zapytaniach powyżej.

Częste błędy

Mylenie rozmiaru tabeli z rozmiarem samych danych bez indeksów. `sp_spaceused` i zapytania powyżej liczą tabelę razem z jej indeksami — tabela z dużą liczbą indeksów wtórnych bywa mniejsza od nich łącznie niż same dane, co jest częstym zaskoczeniem przy pierwszym spojrzeniu na wynik.

Branie `AlokowaneMB`/`ZarezerwowaneKB` za rozmiar realnie potrzebny po reindeksacji. Ta kolumna zawiera zapas i fragmentację nagromadzoną przez lata — po świeżym `REBUILD` rozmiar zwykle spada, czasem znacząco, co jest normalne i nie oznacza utraty danych.

Powiązane: Rozpoznanie instancji i baz · Indeksy: status, fragmentacja, przebudowa · Fragmentacja i przebudowa indeksów

Źródła: dokumentacja Microsoft Learn, sekcja „sp_spaceused" i „sys.dm_db_partition_stats"

Konta serwisowe i uprawnienia do monitoringu

Poziom: podstawy **Dotyczy: SQL Server 2016–2022** **Czas: ~15 min**

Nadawanie loginowi tylko tych uprawnień, których faktycznie potrzebuje — na przykładzie konta systemu monitorującego, które ma czytać stan instancji, a nie mieć dostęp do danych.

Skróty użyte na tej stronie (2)

Skrót	Co znaczy
SQL	Structured Query Language: język zapytań do relacyjnych baz danych
SSMS	SQL Server Management Studio: graficzna konsola MSSQL

Fiszka

ZMIENIA SYSTEM

Nadanie uprawnienia do stanu serwera: VIEW SERVER STATE

```
GRANT VIEW SERVER STATE TO NazwaLoginu;
```

Nadajemy loginowi prawo odpytywania większości widoków dynamicznego zarządzania (`sys.dm_exec_*`, `sys.dm_os_*`) na poziomie całej instancji — bez tego uprawnienia zapytania do tych widoków zwracają puste wyniki albo błąd odmowy dostępu, mimo że sama składnia jest poprawna.

ZMIENIA SYSTEM

Nadanie uprawnienia do definicji obiektów: VIEW ANY DEFINITION

```
GRANT VIEW ANY DEFINITION TO NazwaLoginu;
```

Nadajemy prawo odczytu definicji obiektów (treści procedur, widoków, funkcji) na całej instancji, bez nadawania prawa do samych danych — przydatne dla narzędzi dokumentujących strukturę bazy albo systemów monitorujących, które muszą znać nazwy i definicje obiektów, ale nie mają czytać wierszy tabel.

□□ ZMIENIA SYSTEM

Nadanie odczytu do konkretnej tabeli systemowej: SELECT na obiekcie

```
GRANT SELECT ON OBJECT::msdb.dbo.sysjobs TO NazwaLoginu;
```

Nadajemy odczyt pojedynczej tabeli systemowej wewnątrz `msdb` — typowy krok przy podłączaniu zewnętrznego systemu monitoringu, który sam odpytuje stan zadań SQL Agent zamiast czekać na alert wysłany przez sam silnik.

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

□□ ZMIENIA SYSTEM

Nadanie prawa do wykonania procedury systemowej: EXECUTE na obiekcie

```
GRANT EXECUTE ON OBJECT::msdb.dbo.agent_datetime TO NazwaLoginu;
```

Nadajemy prawo wywołania konkretnej procedury systemowej bez nadawania szerszych uprawnień do całej bazy `msdb` — zasada najmniejszych uprawnień w praktyce: konto monitorujące dostaje dokładnie te trzy czy cztery uprawnienia, które są mu rzeczywiście potrzebne, zamiast roli `db_owner` na wyrost.

□□ ZMIENIA SYSTEM

Zestaw uprawnień dla konta monitorującego SQL Agent

```
USE msdb;
GO

GRANT SELECT ON OBJECT::msdb.dbo.sysjobs TO NazwaLoginu;
GRANT SELECT ON OBJECT::msdb.dbo.sysjobactivity TO NazwaLoginu;
```

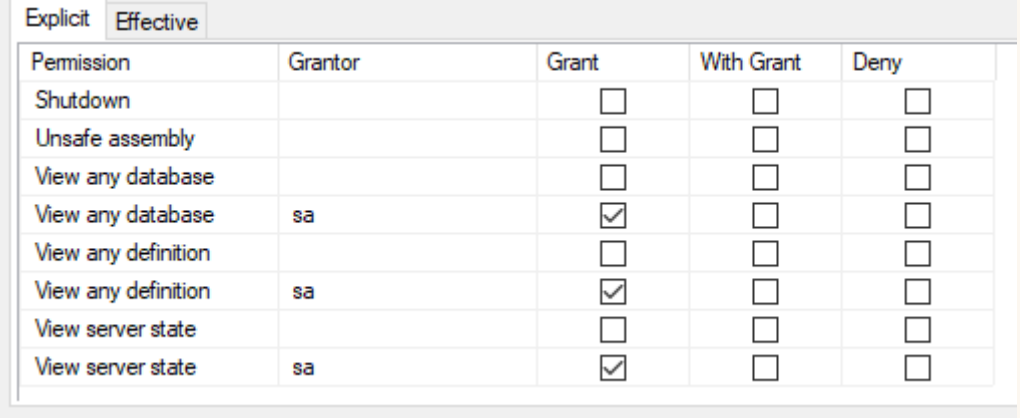
```
GRANT SELECT ON OBJECT::msdb.dbo.sysjobservers TO NazwaLoginu;
GRANT EXECUTE ON OBJECT::msdb.dbo.agent_datetime TO NazwaLoginu;
GO

USE master;
GO

GRANT VIEW SERVER STATE TO NazwaLoginu;
GRANT VIEW ANY DEFINITION TO NazwaLoginu;
GO
```

Komendy wykonujemy jako blok, w tej kolejności, ponieważ pierwsza część działa w kontekście `msdb`, a druga w kontekście `master` — to jedyna sytuacja, w której łączymy kilka poleceń `GRANT` w jeden blok, bo razem tworzą kompletny, gotowy do wklejenia zestaw uprawnień pod jedno konto serwisowe, na przykład konto systemu Zabbix albo innego narzędzia monitorującego instancję z zewnątrz.

Ten sam komplet uprawnień widać w oknie właściwości loginu w SSMS, na zakładce Securables — kolumna Grant pokazuje, co konto faktycznie dostało.



Zakładka Securables we właściwościach loginu monitorującego: nadane View any database, View any definition i View server state. (kliknięcie otwiera pełny rozmiar)

SQL = Structured Query Language: język zapytań do relacyjnych baz danych · **SSMS** = SQL Server Management Studio: graficzna konsola MSSQL

Z życia wzięte

Z ŻYCIA WZIĘTE

We wtorek Ola prosi o konto dla nowego eksportera Zabbixa, żeby monitorować kondycję sql01 razem z resztą infrastruktury. Zamiast dodać je od razu do roli sysadmin, bo tak najszybciej, nadajemy dokładnie te uprawnienia, których potrzebuje: VIEW SERVER STATE i odczyt kilku tabel w msdb. Miesiąc później hasło tego konta wycieka razem z konfiguracją Zabbixa skopiowaną na złe repozytorium, ale ryzyko ogranicza się do samego monitoringu, nie do całej instancji.

Po co to istnieje

Pojęcie: login i użytkownik bazy danych. W SQL Server to dwa osobne poziomy uprawnień. Login (`CREATE LOGIN`) istnieje na poziomie całej instancji i decyduje, czy dane konto w ogóle może się połączyć. Użytkownik bazy danych (`CREATE USER`) istnieje wewnątrz konkretnej bazy i mapuje się na login — dopiero on decyduje, do czego dane konto ma dostęp wewnątrz tej konkretnej bazy. Ten sam login może mieć różnych użytkowników — a więc różne uprawnienia — w różnych bazach na tej samej instancji. Powiązanie między loginem a użytkownikiem, oraz sytuacje, w których się ono rozjeżdża po odtworzeniu bazy z innego serwera, opisujemy na stronie Naprawa uprawnień użytkownika SQL po odtworzeniu baz.

Konta systemów monitorujących (Zabbix, Nagios, Prometheus z odpowiednim exporterem) potrzebują dostępu tylko do odczytu stanu instancji — nigdy do danych biznesowych. Nadanie takiemu kontu roli `sysadmin` na wyrost, bo „tak jest szybciej", jest najczęstszym błędem przy podłączaniu monitoringu i tworzy niepotrzebne ryzyko: kompromitacja hasła narzędzia monitorującego oznacza wtedy pełny dostęp administracyjny do wszystkich baz na instancji.

Jak to działa

Uprawnienia `GRANT ... ON OBJECT::` są uprawnieniami na poziomie pojedynczego obiektu, nie roli. Zamiast dodawać login do wbudowanej roli bazodanowej `db_datareader`, która dałaby odczyt do wszystkich tabel bazy `msdb` włącznie z tymi, których monitoring nigdy nie odpyta, nadajemy `SELECT` na dokładnie tych trzech, czterech tabelach, których faktycznie potrzebuje zapytanie widoczne po stronie narzędzia monitorującego.

> **Pojęcie: rola serwerowa i rola bazodanowa.** Rola serwerowa (`sysadmin`, `serveradmin`, `dbcreator`) działa na poziomie całej instancji. Rola bazodanowa (`db_owner`, `db_datareader`, `db_datawriter`, `db_backupoperator`) działa wewnątrz jednej bazy. Nadanie loginowi członkostwa w roli jest szybsze niż wypisywanie uprawnień obiekt po obiekcie, ale nadaje od razu cały zestaw uprawnień tej roli — dla kont serwisowych o wąskim zakresie odpowiedzialności wypisanie pojedynczych `GRANT` jest bardziej precyzyjne, choć bardziej pracochłonne przy pierwszym ustawieniu.

Częste błędy

Nadanie roli `sysadmin` kontu monitorującemu „na wszelki wypadek". To najszybsza droga do sytuacji, w której hasło jednego konta serwisowego, wycieknięte z konfiguracji narzędzia trzeciego, daje pełną kontrolę nad wszystkimi bazami na instancji.

Zapomnienie o `VIEW SERVER STATE` przy podłączaniu monitoringu. Zapytania do `sys.dm_exec_sessions` czy `sys.dm_os_performance_counters` bez tego uprawnienia zwracają pusty wynik zamiast błędu — co bywa mylące, bo wygląda jak brak aktywności na serwerze, a nie jak brak uprawnień.

Powiązane: Naprawa uprawnień użytkownika SQL po odtworzeniu baz · Konta, role i uprawnienia: GRANT, REVOKE, zasada najmniejszych uprawnień · SQL Agent: zadania i harmonogramy

Źródła: dokumentacja Microsoft Learn, sekcja „GRANT (Transact-SQL)" i „Server-Level Roles"

Rozpoznanie procedur składowanych

Poziom: podstawy **Dotyczy: SQL Server 2016–2022** **Czas: ~10 min**

Sprawdzenie, jakie procedury składowane istnieją w bazie i jak wygląda ich definicja, kiedy trafiamy na nieznaną wcześniej bazę bez dokumentacji.

Skróty użyte na tej stronie (1)

Skrót	Co znaczy
SQL	Structured Query Language: język zapytań do relacyjnych baz danych

Fiszka

TYLKO ODCZYT

Wylistowanie procedur składowanych w bazie: INFORMATION_SCHEMA.ROUTINES

```
SELECT ROUTINE_SCHEMA, ROUTINE_NAME, ROUTINE_TYPE, CREATED, LAST_ALTERED
FROM NazwaBazy.INFORMATION_SCHEMA.ROUTINES
WHERE ROUTINE_TYPE = 'PROCEDURE'
ORDER BY ROUTINE_NAME;
```

Sprawdzamy, jakie procedury składowane istnieją we wskazanej bazie — `INFORMATION_SCHEMA.ROUTINES` obejmuje zarówno procedury (`PROCEDURE`), jak i funkcje (`FUNCTION`), stąd filtr na `ROUTINE_TYPE`. To pierwsze zapytanie na bazie, o której wiemy tylko, że jest zasilana logiką z poziomu procedur, a nie mamy do niej dokumentacji.

Przykład:

```
ROUTINE_SCHEMA   ROUTINE_NAME            ROUTINE_TYPE   CREATED
LAST_ALTERED
---------------- ----------------------- -------------- ----------------------
----------------------
dbo              IndexOptimize           PROCEDURE      2022-03-14 09:12:33.000 2023-
11-02 15:40:11.000
dbo              usp_ArchiwizujZamowienia PROCEDURE      2021-06-01 10:00:00.000 2021-
06-01 10:00:00.000

(2 rows affected)
```

TYLKO ODCZYT

Wyświetlenie definicji procedury: sp_helptext

```
EXEC sp_helptext N'NazwaProcedury';
```

Sprawdzamy pełną treść procedury — jej definicję `CREATE PROCEDURE` w oryginalnym sformatowaniu, tak jak została zapisana przy tworzeniu. Wykonujemy to zapytanie na bazie, która zawiera daną procedurę, więc wcześniej przełączamy kontekst poleceniem `USE NazwaBazy`.

Przykład:

```
Text
------------------------------------------------------------
CREATE PROCEDURE dbo.IndexOptimize
    @Databases NVARCHAR(MAX) = NULL,
    @FragmentationLow NVARCHAR(MAX) = NULL
AS
BEGIN
    SET NOCOUNT ON;
    -- ...

(18 rows affected)
```

Ten sam efekt daje odpytanie `sys.sql_modules` bezpośrednio, co bywa wygodniejsze w skryptach, bo zwraca definicję jako pojedynczy tekst w kolumnie `definition`, zamiast wielu wierszy jak `sp_helptext`:

```
SELECT definition FROM sys.sql_modules WHERE object_id = OBJECT_ID(N'NazwaProcedury');
```

exec = zastąp bieżący proces powłoki wskazanym programem, bez tworzenia nowego procesu

Z życia wzięte

Z ŻYCIA WZIĘTE

Tomek dostaje zgłoszenie, że raport miesięczny w Optimie czasem się zawiesza, ale w aplikacji nie widać żadnego kodu odpowiedzialnego za jego generowanie. Sprawdza listę procedur składowanych w CDN_Zielona i znajduje usp_ArchiwizujZamowienia, o której nikt w zespole wcześniej nie słyszał. Próba odczytania jej treści przez sp_helptext kończy się pustym wynikiem, bo procedura została utworzona z opcją WITH ENCRYPTION. Dopiero rozmowa z Olą ujawnia, że to stary skrypt jednego z poprzednich programistów, zaszyfrowany na wszelki wypadek.

Po co to istnieje

Procedury składowane bywają jedynym miejscem, w którym opisana jest logika biznesowa działająca na bazie — harmonogramy konserwacyjne (jak popularna procedura `IndexOptimize` ze zbioru narzędzi Ola Hallengren, opisana szerzej na stronie Indeksy: status, fragmentacja, przebudowa), procedury archiwizujące dane albo integracje z zewnętrznymi systemami. Zanim zmienimy cokolwiek w bazie, której nie znamy, sprawdzamy, jakie procedury na niej działają i co dokładnie robią — `sp_helptext` jest do tego szybsze niż szukanie kodu źródłowego w repozytorium, którego może w ogóle nie być.

Częste błędy

Brak wyniku z `sp_helptext` dla procedury zaszyfrowanej. Procedura utworzona z opcją `WITH ENCRYPTION` nie zwraca definicji przez `sp_helptext` ani przez `sys.sql_modules` — silnik świadomie ukrywa treść, co bywa zaskoczeniem przy pierwszym natrafieniu na taki obiekt.

Szukanie procedury w złej bazie. `INFORMATION_SCHEMA.ROUTINES` i `sp_helptext` działają w kontekście bazy, na której akurat jesteśmy — procedura o tej samej nazwie może istnieć w kilku bazach naraz z zupełnie inną treścią, dlatego zawsze warto zacząć od jawnego `USE NazwaBazy`.

Powiązane: Rozpoznanie instancji i baz · Indeksy: status, fragmentacja, przebudowa · SQL Agent: zadania i harmonogramy

Źródła: dokumentacja Microsoft Learn, sekcja „sp_helptext" i „INFORMATION_SCHEMA.ROUTINES"

Sesje, zapytania i blokady

Poziom: średni **Dotyczy: SQL Server 2016-2022** **Czas: ~20 min**

Sprawdzenie, co aktualnie dzieje się na instancji — jakie zapytania są w toku, kto kogo blokuje i jak bezpiecznie zakończyć sesję, która się zacięła.

Skróty użyte na tej stronie (3)

Skrót	Co znaczy
CPU	Central Processing Unit: procesor
SQL	Structured Query Language: język zapytań do relacyjnych baz danych
SSMS	SQL Server Management Studio: graficzna konsola MSSQL

Fiszka

TYLKO ODCZYT

Sprawdzenie aktualnych zapytań na bazie danych: sys.dm_exec_requests

```
SELECT session_id, command, text
FROM sys.dm_exec_requests
CROSS APPLY sys.dm_exec_sql_text(sql_handle);
```

Sprawdzamy wszystkie zapytania aktualnie wykonywane przez silnik, razem z ich pełną treścią — `sys.dm_exec_requests` sam w sobie trzyma tylko uchwyt (`sql_handle`) do tekstu zapytania, dlatego doklejamy `CROSS APPLY sys.dm_exec_sql_text`, żeby zamienić uchwyt na czytelny tekst SQL.

Przykład:

```
session_id   command   text
------------ --------- --------------------------------------------------
```

```
62          SELECT    SELECT * FROM Zamowienia WHERE KlientId = 4021
71          UPDATE    UPDATE PozycjeZamowien SET Ilosc = 5 WHERE Id = 918

(2 rows affected)
```

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

TYLKO ODCZYT

Sprawdzenie zapytań pochodzących z konkretnego joba SQL Agent

```
SELECT session_id, program_name, host_name
FROM sys.dm_exec_sessions
WHERE program_name LIKE 'SQLAgent - TSQL JobStep (Job %';
```

Sprawdzamy, które sesje na instancji pochodzą z uruchomionych kroków zadań SQL Agent — nazwa programu dla takich sesji zawsze zaczyna się od `SQLAgent - TSQL JobStep`, co pozwala odróżnić je od połączeń użytkowników i aplikacji.

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

TYLKO ODCZYT

Sprawdzenie wszystkich aktywnych sesji: sp_who2

```
EXEC sp_who2;
```

Sprawdzamy skróconą listę wszystkich sesji na instancji — starsza, ale wciąż wygodna procedura systemowa pokazująca `SPID`, status, login, bazę, czas CPU i informację, czy dana sesja kogoś blokuje w kolumnie `BlkBy`.

Przykład:

```
SPID   Status    Login       HostName   BlkBy   DBName    Command   CPUTime
------ --------- ----------- ---------- ------- --------- --------- --------
62     RUNNABLE  nazwa_usera SRV-APP01  .       CDN_Zielona    SELECT    1230
71     SUSPENDED nazwa_usera SRV-APP01  62      CDN_Zielona    UPDATE    0

(2 rows affected)
```

SPID = Server Process ID: numer identyfikujący pojedynczą sesję w MSSQL, używany na przykład do zakończenia konkretnego połączenia poleceniem KILL · **CPU** = Central Processing Unit: procesor · **exec** = zastąp bieżący proces powłoki wskazanym programem, bez tworzenia nowego procesu

TYLKO ODCZYT

Sprawdzenie, co blokuje przełączanie trybu bazy: sys.dm_tran_locks

```
SELECT
    l.resource_type,
    l.request_mode,
    l.request_status,
    l.request_session_id,
    r.command,
    r.status,
    r.blocking_session_id,
    r.wait_type,
    r.wait_time,
    r.wait_resource,
    st.text AS RequestSqlText,
    s.program_name,
    stc.text AS MostRecentSqlText
FROM sys.dm_tran_locks l
LEFT JOIN sys.dm_exec_requests r ON l.request_session_id = r.session_id
LEFT JOIN sys.dm_exec_sessions s ON l.request_session_id = s.session_id
LEFT JOIN sys.dm_exec_connections c ON s.session_id = c.session_id
OUTER APPLY sys.dm_exec_sql_text(r.sql_handle) st
OUTER APPLY sys.dm_exec_sql_text(c.most_recent_sql_handle) stc
WHERE l.resource_database_id = DB_ID(N'NazwaBazyDanych')
ORDER BY request_session_id;
```

Sprawdzamy, które sesje trzymają blokady na wskazanej bazie i jakie zapytanie dokładnie wykonały jako ostatnie — kluczowe zapytanie w sytuacji, gdy `ALTER DATABASE ... SET SINGLE_USER` albo `... SET OFFLINE` wisi w nieskończoność, bo jakaś sesja nie chce zwolnić bazy. Kolumna `wait_resource` mówi, na jaki dokładnie zasób (stronę, wiersz, tabelę) czeka blokująca sesja, a `blocking_session_id` w połączonym `sys.dm_exec_requests` wskazuje, kto jest źródłem blokady.

Przykład:

```
resource_type    request_mode    request_status    request_session_id    command
blocking_session_id    wait_type
--------------- --------------- ---------------- --------------------- ---------
--------------------- -----------
DATABASE         S               GRANT             54                     SELECT
NULL                   NULL
DATABASE         S               GRANT             62                     SELECT
NULL                   NULL

(2 rows affected)
```

Jeżeli widzimy sesje, które nie chcą się zamknąć — jak wyżej — kończymy je poleceniem `KILL`, podając numer `SPID` z kolumny `request_session_id` albo `session_id`.

Przykład wyniku w siatce SSMS, gdy kilka sesji tej samej aplikacji trzyma blokadę współdzieloną na bazie:

Results Messages

	resource_type	request_mode	request_status	request_session_id	command	status	blocking_session_id	wait_type	wait_time	wait_resource	request_sql_text	program_name	most_recent_
1	DATABASE	S	GRANT	62	NULL	NULL	NULL	NULL	NULL	NULL	NULL	Microsoft Dynamics NAV Server	NULL
2	DATABASE	S	GRANT	62	NULL	NULL	NULL	NULL	NULL	NULL	NULL	Microsoft Dynamics NAV Server	NULL
3	DATABASE	S	GRANT	62	NULL	NULL	NULL	NULL	NULL	NULL	NULL	Microsoft Dynamics NAV Server	(@maxChang
4	DATABASE	S	GRANT	79	NULL	NULL	NULL	NULL	NULL	NULL	NULL	Microsoft Dynamics NAV Server	NULL
5	DATABASE	S	GRANT	79	NULL	NULL	NULL	NULL	NULL	NULL	NULL	Microsoft Dynamics NAV Server	NULL
6	DATABASE	S	GRANT	79	NULL	NULL	NULL	NULL	NULL	NULL	NULL	Microsoft Dynamics NAV Server	(@maxChang
7	DATABASE	S	GRANT	81	NULL	NULL	NULL	NULL	NULL	NULL	NULL	Microsoft Dynamics NAV Server	NULL
8	DATABASE	S	GRANT	81	NULL	NULL	NULL	NULL	NULL	NULL	NULL	Microsoft Dynamics NAV Server	NULL
9	DATABASE	S	GRANT	81	NULL	NULL	NULL	NULL	NULL	NULL	NULL	Microsoft Dynamics NAV Server	(@maxChang

Wynik zapytania do sys.dm_tran_locks i sys.dm_exec_requests — trzy sesje aplikacji z blokadą typu DATABASE w trybie S. (kliknięcie otwiera pełny rozmiar)

SET = Switch Embedded Teaming: zespół kart sieciowych zbudowany w przełączniku wirtualnym Hyper-V; łączy dwie karty w jedno łącze bez ustawień po stronie przełącznika fizycznego, więc bez LACP · **SPID** = Server Process ID: numer identyfikujący pojedynczą sesję w MSSQL, używany na przykład do zakończenia konkretnego połączenia poleceniem KILL · **SSMS** = SQL Server Management Studio: graficzna konsola MSSQL

ZMIENIA SYSTEM

Zakończenie sesji: KILL

```
KILL 62;
```

(gdzie `62` to numer `SPID` sesji do zakończenia, odczytany z zapytań powyżej — po zakończeniu silnik wycofuje wszystkie niezatwierdzone zmiany tej sesji, co przy długiej transakcji może chwilę potrwać)

Uważaj: `KILL` jest operacją nieodwracalną i natychmiastową z punktu widzenia sesji — jeśli sesja miała otwartą długą transakcję modyfikującą dane, rollback tej transakcji może trwać równie długo jak sama transakcja, a w tym czasie baza pozostaje częściowo zablokowana.

□□ **SPID** = Server Process ID: numer identyfikujący pojedynczą sesję w MSSQL, używany na przykład do zakończenia konkretnego połączenia poleceniem KILL · **kill** = wyślij sygnał do procesu, domyślnie SIGTERM proszący o zamknięcie

Z życia wzięte

□□ Z ŻYCIA WZIĘTE

W piątek o 15:40, tuż przed zamknięciem miesiąca, pani Halina zgłasza, że Optima wisi w środku zapisywania faktury. Sprawdzamy sys.dm_exec_requests i widzimy sesję w stanie SUSPENDED, zablokowaną przez inną sesję z tej samej aplikacji, która od godziny trzyma otwartą transakcję. Zamiast od razu użyć KILL, sprawdzamy najpierw sys.dm_tran_locks, żeby zobaczyć, czy blokująca sesja faktycznie nic już nie robi. Po zakończeniu tej sesji poleceniem KILL Optima wraca do normalnej pracy, a pani Halina kończy fakturę z piętnastominutowym opóźnieniem.

Po co to istnieje

Pojęcie: blokada (lock). Mechanizm, przez który silnik chroni spójność danych, gdy więcej niż jedna sesja próbuje jednocześnie odczytać albo zmienić ten sam fragment danych. Blokady mogą być współdzielone (kilka sesji czyta naraz) albo wyłączne (jedna sesja modyfikuje, wszystkie inne czekają) — pełny mechanizm i eskalacja blokad opisane są na stronie Blokady: współdzielone, wyłączne, eskalacja.

Sesja, która trzyma blokadę na całej bazie (np. w trakcie długiej transakcji na dużej tabeli), skutecznie blokuje operacje wymagające wyłącznego dostępu do bazy jako całości — `ALTER DATABASE ... SET SINGLE_USER`, `... SET OFFLINE`, `... SET READ_ONLY`. Zanim uznamy, że polecenie „się zawiesiło", sprawdzamy, czy nie czeka właśnie na zwolnienie takiej blokady przez inną,

aktywną sesję.

Częste błędy

Zabijanie pierwszej lepszej sesji bez sprawdzenia, co ona robi. `KILL` przerywa transakcję i wymusza rollback — na sesji wykonującej duży `INSERT` czy `UPDATE` rollback bywa równie kosztowny czasowo jak sama operacja, więc czasem szybciej jest po prostu poczekać na jej naturalne zakończenie.

Mylenie `wait_type` z przyczyną blokady. Typ oczekiwania (`LCK_M_X`, `PAGEIOLATCH_SH`, `ASYNC_NETWORK_IO`) mówi, na co sesja czeka technicznie, ale nie zawsze wprost wskazuje przyczynę biznesową — do pełnego obrazu potrzebne jest zestawienie z `sys.dm_os_waiting_tasks` i z treścią zapytania blokującej sesji.

Powiązane: Tryby pracy bazy: read-only, single user, offline · Blokady: współdzielone, wyłączne, eskalacja · Zakleszczenie (deadlock): jak powstaje i jak je czytać z logu

Źródła: dokumentacja Microsoft Learn, sekcja „sys.dm_exec_requests", „sys.dm_tran_locks" i „KILL (Transact-SQL)"

Indeksy: status, fragmentacja, przebudowa

Poziom: średni **Dotyczy: SQL Server 2016-2022** **Czas: ~20 min**

Sprawdzenie fragmentacji indeksów, decyzja między przebudową a reorganizacją i weryfikacja, czy zaplanowana reindeksacja w ogóle się wykonała.

Skróty użyte na tej stronie (1)

Skrót	Co znaczy
SQL	Structured Query Language: język zapytań do relacyjnych baz danych

Fiszka

ZMIENIA SYSTEM

Sprawdzenie fragmentacji wszystkich indeksów w bazie: sys.dm_db_index_physical_stats

```
SELECT
    DB_NAME(database_id) AS NazwaBazy,
    OBJECT_NAME(object_id, database_id) AS NazwaTabeli,
    index_type_desc AS TypIndeksu,
    avg_fragmentation_in_percent AS ProcentFragmentacji,
    page_count AS LiczbaStron
FROM sys.dm_db_index_physical_stats(DB_ID(N'NazwaBazyDanych'), NULL, NULL, NULL,
'SAMPLED')
WHERE page_count > 100
ORDER BY avg_fragmentation_in_percent DESC;
```

Sprawdzamy poziom fragmentacji każdego indeksu w bazie, pomijając te złożone z mniej niż stu stron — indeksy tak małe fragmentują się i defragmentują błyskawicznie, a ich fragmentacja nie ma praktycznego wpływu na wydajność. Tryb `'SAMPLED'` liczy statystykę na próbce stron zamiast przechodzić przez cały indeks, co jest wystarczająco dokładne do podjęcia decyzji i wielokrotnie szybsze niż tryb `'DETAILED'` na dużej tabeli.

Przykład:

```
NazwaBazy   NazwaTabeli        TypIndeksu       ProcentFragmentacji   LiczbaStron
----------- ------------------ ---------------- --------------------- -------------
CDN_Zielona      Zamowienia         CLUSTERED INDEX  38.42                 268160
CDN_Zielona      PozycjeZamowien    NONCLUSTERED     12.75                 94208
CDN_Zielona      Klienci            CLUSTERED INDEX  2.10                  8192

(3 rows affected)
```

> (gdzie `NULL, NULL, NULL` na miejscu drugiego, trzeciego i czwartego argumentu funkcji oznacza „wszystkie obiekty, wszystkie indeksy, wszystkie partycje" — podanie konkretnego `object_id` zawęża wynik do jednej tabeli)

Z życia wzięte

Z ŻYCIA WZIĘTE

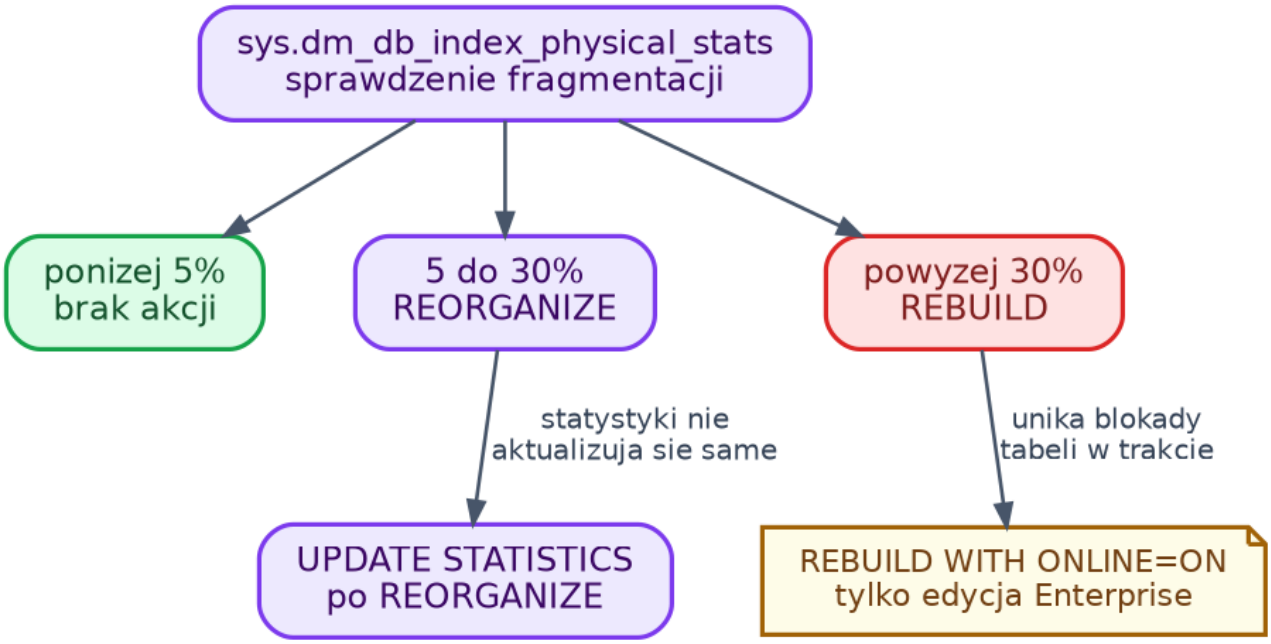
W poniedziałek po zamknięciu miesiąca raporty księgowe z CDN_Zielona zaczynają działać zauważalnie wolniej niż zwykle, mimo że ilość danych rośnie stopniowo od miesięcy. Sprawdzamy fragmentację indeksów zapytaniem do sys.dm_db_index_physical_stats i widzimy indeks klastrowany tabeli Zamowienia fragmentowany w prawie 40 procentach. Ponieważ instancja stoi na edycji Standard, REBUILD musi poczekać do wieczora, żeby nie blokować pracy księgowości w ciągu dnia. Po nocnym REBUILD i aktualizacji statystyk raporty wracają do wcześniejszego czasu wykonania.

Po co to istnieje

Pojęcie: fragmentacja indeksu. Rozjazd między logicznym porządkiem stron indeksu (kolejnością kluczy) a ich fizycznym rozłożeniem na dysku. Indeks buforowany na bieżąco wstawianiami i usunięciami z czasem przestaje mieć strony ułożone po kolei — silnik musi wtedy wykonywać więcej losowych odczytów zamiast jednego sekwencyjnego przebiegu, co najbardziej boli przy skanach dużych zakresów, mniej przy pojedynczych lookupach po kluczu głównym.

Jak to działa

Pojęcie: próg decyzyjny 5/30 procent. Microsoft rekomenduje dwa progi fragmentacji, od których zależy dobór operacji naprawczej.



Sprawdzenie fragmentacji indeksu i wybor miedzy brakiem akcji, REORGANIZE i REBUILD (kliknięcie otwiera pełny rozmiar)

Fragmentacja	Zalecana operacja
poniżej 5%	brak akcji — fragmentacja na tym poziomie nie wpływa zauważalnie na wydajność
5–30%	`REORGANIZE`

Fragmentacja	Zalecana operacja
powyżej 30%	`REBUILD`

> **Pojęcie: REORGANIZE.** Operacja defragmentująca liście indeksu przez przestawianie stron w miejscu, bez zwalniania i ponownego alokowania całej struktury. Działa zawsze online, jest mniej inwazyjna, ale wolniejsza przy wysokiej fragmentacji i nie aktualizuje statystyk — statystyki po `REORGANIZE` trzeba odświeżyć osobno.

```
ALTER INDEX NazwaIndeksu ON NazwaTabeli REORGANIZE;
```

> **Pojęcie: REBUILD.** Operacja budująca indeks od nowa w całości, w nowym, nieposzarpanym układzie stron. Domyślnie blokuje tabelę na czas trwania operacji (`ONLINE = OFF`) — w edycji Enterprise dostępna jest opcja `ONLINE = ON`, pozwalająca przebudować indeks bez blokowania odczytów i zapisów na tabeli, kosztem większego zużycia zasobów i dłuższego czasu wykonania.

```
ALTER INDEX NazwaIndeksu ON NazwaTabeli REBUILD WITH (ONLINE = ON);
```

Edycja silnika	ONLINE = ON dostępne
Express	nie
Standard	nie (od SQL Server 2019 dostępne częściowo dla wybranych operacji)
Enterprise	tak
Developer	tak

> **Pojęcie: statystyki.** Metadane opisujące rozkład wartości w kolumnach, na podstawie których optymalizator zapytań szacuje liczbę wierszy pasujących do warunku i dobiera plan wykonania. `REBUILD` aktualizuje statystyki jako efekt uboczny, `REORGANIZE` — nie, dlatego po reorganizacji warto dodatkowo wykonać:

```
UPDATE STATISTICS NazwaTabeli WITH FULLSCAN;
```

Weryfikacja statusu reindeksacji: master.dbo.CommandLog

Środowiska korzystające z popularnego zestawu skryptów konserwacyjnych (procedura `IndexOptimize` autorstwa Ola Hallengren, wspomniana na stronie Rozpoznanie procedur składowanych) zapisują historię wykonanych operacji do tabeli `CommandLog` w bazie `master`.

```
SELECT TOP (100)
    DatabaseName, SchemaName, ObjectName, IndexName,
    Command, StartTime, EndTime, ErrorNumber, ErrorMessage
FROM master.dbo.CommandLog
ORDER BY StartTime DESC;
```

Sprawdzamy, czy zaplanowana reindeksacja faktycznie wykonała się dla wszystkich tabel i czy któraś z operacji nie zakończyła się błędem — `ErrorNumber` różny od zera oznacza operację nieudaną, wymagającą powtórzenia ręcznie.

Przykład:

```
DatabaseName   ObjectName        IndexName          Command
StartTime               ErrorNumber
-------------- ----------------- ----------------- ------------------------------
----------------------- ------------
CDN_Zielona         Zamowienia        PK_Zamowienia      ALTER INDEX [PK_Zamowienia]... 2026-
09-21 01:15:03.000  0
CDN_Zielona         PozycjeZamowien   IX_PozZam_ZamId    ALTER INDEX [IX_PozZam_Zam...  2026-
09-21 01:22:41.000  0

(2 rows affected)
```

Częste błędy

Rebuild wszystkich indeksów niezależnie od ich rozmiaru i fragmentacji. Przebudowa indeksu liczącego kilkaset stron nie daje praktycznie żadnej korzyści, a mnoży czas trwania okna serwisowego — filtr `page_count > 100` w zapytaniu diagnostycznym istnieje właśnie po to, żeby odrzucić takie przypadki.

REBUILD bez ONLINE na produkcji w godzinach szczytu. Bez edycji Enterprise albo poza oknem serwisowym operacja blokuje tabelę na czas trwania rebuild — na dużej tabeli oznacza to realny przestój aplikacji, nie tylko spowolnienie.

Zapomnienie o aktualizacji statystyk po REORGANIZE. Sam brak fragmentacji nie oznacza dobrego planu wykonania — nieaktualne statystyki po samej reorganizacji nadal mogą prowadzić

optymalizator do złych decyzji.

Powiązane: Rozmiary tabel i indeksów · Fragmentacja i przebudowa indeksów · Statystyki i optymalizator kosztowy · Rozpoznanie procedur składowanych

Źródła: dokumentacja Microsoft Learn, sekcja „sys.dm_db_index_physical_stats" i „ALTER INDEX"