Spis treści

Wstęp

Od momentu wydania mojej ostatniej książki minęło już kilka lat. Przez ten czas świat się zmienił. Pojawiły się nowe wersje SQL Server. Produkt rozwinął się i obecnie oferuje wsparcie dla wielu systemów operacyjnych oraz zawiera opcje pozwalające na wykorzystanie chmury. Mimo tego uważałem, że nie nadszedł jeszcze właściwy moment, by opublikować nowe wydanie książki *Pro SQL Server Internals* (Apress).

Istnieje ku temu kilka powodów. Mimo że nowe funkcje są wspaniałe, nie zmieniły *w zasadzie* sposobu działania produktu. Większość wiedzy zaprezentowanej w moich poprzednich książkach można było zastosować do SQL Server 2017, SQL Server 2019, a nawet najnowszej wersji SQL Server 2022. Jednak co ważniejsze, chciałem napisać książkę w inny sposób.

Być może powinienem to dokładniej wyjaśnić. Zapewne niektórzy z Czytelników wiedzą, że przez wiele lat prowadziłem kursy dotyczące SQL Server. Książki, których byłem autorem, traktowałem jako materiały uzupełniające. W rzeczywistości zacząłem je pisać, ponieważ chciałem przedstawić wiedzę w bardziej uporządkowanym formacie – innym niż Power Point. Cieszę się, że książki spodobały się moim Czytelnikom, którzy uznali je za szczególnie przydatne.

Wszystkie kursy, które prowadziłem, dotyczyły wewnętrznych mechanizmów SQL Server. Zawsze wierzyłem, że profesjonalista musi znać narzędzia, aby odnieść sukces. Uczyłem słuchaczy, jak działa środowisko SQL Server, a dzięki temu pomagałem im wykorzystywać tę wiedzę i budować wydajne systemy. Z czasem jednak odkryłem, że najpopularniejszym tematem moich zajęć stało się rozwiązywanie problemów i poprawianie wydajności systemu – słuchaczom podoba się, gdy zaczynam od przedstawienia *problemu*, a następnie wyjaśniam, *dlaczego* się on pojawia.

Po zmianie stylu prowadzenia wykładów postanowiłem również zmodyfikować sposób pisania książek. Od tej pory minęło 18 miesięcy. Czytelnicy mogą teraz ocenić rezultat tej decyzji. Osobiście podoba mi się to, co stworzyłem. W książce wciąż poruszam

zagadnienia związane z wewnętrznymi mechanizmami SQL Server, chociaż w bardziej zwięzły i praktyczny sposób, niż w którejkolwiek z moich poprzednich prac. Dzięki niej Czytelnik uzyska wystarczającą wiedzę, aby wykrywać i usuwać główne problemy związane z systemem, a przy tym nie będzie przeciążony nadmierną ilością informacji. Uzyska również wskazówki, gdzie może zdobyć dodatkową wiedzę, jeśli będzie się chciał dowiedzieć więcej.

W książce zaprezentowano metodologię stosowaną przez wielu wysokiej klasy specjalistów związanych z SQL Server. Czytelnik dowie się, jak można gromadzić i analizować dane, a dzięki temu wykrywać ograniczenia i problemy z wydajnością. Co ważniejsze, nauczy się, jak należy traktować system *całościowo* i unikać rutyny.

Treść książki nie dotyczy określonej wersji SQL Server. Z kilkoma wyjątkami można ją zastosować do wszystkich produktów, poczynając od SQL Server 2005, a kończąc na najnowszym SQL Server 2022. Sprawdzi się również w przypadku zarządzanych usług SQL Server, działających w chmurze.

Dla kogo jest przeznaczona ta książka?

Gdy ktoś pyta mnie, jaka jest docelowa grupa czytelników moich książek, zawsze odpowiadam, że piszę dla *profesjonalistów zajmujących się bazami danych*. Tego określenia używam celowo. Uważam, że granica dzieląca administratorów baz danych, programistów baz danych, a nawet programistów aplikacji jest dość cienka. W dzisiejszych czasach nie możemy odnieść sukcesu w branży IT, jeśli nie będziemy rozszerzać zakresu wiedzy i obowiązków.

Jest to szczególnie ważne w przypadku metodologii DevOps. Zespoły stają się właścicielami swoich produktów, więc samodzielnie rozwijają i utrzymują rozwiązania. Projektanci często rozwiązują problemy z wydajnością, które mogły zostać spowodowane przez infrastrukturę lub nieefektywny kod bazy danych.

Ta pozycja jest przeznaczona dla Czytelników, którzy wykorzystują SQL Server w jakikolwiek sposób – zarówno w siedzibie firmy, jak i w chmurze. Mam nadzieję, że skorzystają z książki niezależnie od tego, jakie jest ich stanowisko pracy.

Jeszcze raz dziękuję za zaufanie, jakim zostałem obdarzony. Mam nadzieję, że lektura tej książki sprawi Czytelnikowi tyle samo radości, ile ja miałem podczas jej pisania!

Przegląd rozdziałów

Książka składa się z 16 rozdziałów zorganizowanych w następujący sposób:

- Rozdział 1. – „Instalowanie i konfigurowanie systemu bazy danych SQL Server”. W rozdziale zostanie zaprezentowanych szereg wskazówek i najlepszych technik umożliwiających wybór właściwego sprzętu i skonfigurowanie instancji SQL Server.

- Rozdział 2. – „Model wykonawczy SQL Server i statystyka oczekiwania". W rozdziale zostanie przeanalizowany ważny składnik produktu SQL Server – system operacyjny SQLOS. Oprócz tego Czytelnik pozna powszechnie stosowaną technikę rozwiązywania problemów, zwaną statystyką oczekiwania. Rozdział ten stanowi podstawę dla całej książki.
- Rozdział 3. – „Działanie podsystemu dyskowego". Czytelnik dowie się, w jaki sposób SQL Server współpracuje z podsystemem operacji wejścia i wyjścia, a także jak można analizować jego wydajność oraz usuwać istniejące problemy.
- Rozdział 4. – „Nieoptymalne zapytania". W rozdziale zostanie zaprezentowanych kilka metod, które pozwalają wykrywać nieoptymalne zapytania oraz wyszukiwać elementy, które powinny zostać dostrojone.
- Rozdział 5. – „Przechowywanie danych i dostrajanie zapytań". W rozdziale zostanie wyjaśnione, w jaki sposób SQL Server obsługuje bazy danych, a także zostanie podanych szereg wskazówek i metod związanych z dostrajaniem zapytań.
- Rozdział 6. – „Obciążenie procesora". Czytelnik pozna najczęstsze przyczyny skutkujące wysokim obciążeniem procesora oraz sposoby radzenia sobie z takimi problemami.
- Rozdział 7. – „Problemy związane z pamięcią". W rozdziale zostanie omówione konfigurowanie pamięci oraz zaprezentowane, w jaki sposób można monitorować jej zużycie. Dodatkowo zostanie wyjaśnione, jak można rozwiązywać problemy związane z pamięcią.
- Rozdział 8. – „Blokady, blokowanie i współbieżność". W tym rozdziale Czytelnik pozna model współbieżności wykorzystywany w środowisku SQL Server, a także dowie się, jak można uniknąć blokowania procesów i zakleszczeń.
- Rozdział 9. – „Użycie bazy tempdb i optymalizacja jej wydajności". W rozdziale zostaną zaprezentowane najlepsze metody umożliwiające konfigurowanie i używanie bazy *tempdb*. Zawiera on również kilka wskazówek dotyczących optymalnego wykorzystania obiektów tymczasowych i wyjaśnia, jak nie dopuszczać do powstawania problemów związanych z obsługą bazy *tempdb*.
- Rozdział 10. – „Zatrzaski". Rozdział zawiera informacje o zatrzaskach używanych w SQL Server. Prezentuje sytuacje, w których mogą się one stać problematyczne i udostępnia sposoby rozwiązywania problemów.
- Rozdział 11. – „Dziennik transakcji". W rozdziale zostanie wyjaśnione, w jaki sposób SQL Server wykorzystuje dziennik transakcji, a także jak można sobie radzić z najczęściej występującymi problemami i błędami.
- Rozdział 12. – „Grupy dostępności AlwaysOn". Czytelnik pozna najczęściej wykorzystywaną metodę wysokiej dostępności oraz problemy, jakie mogą się pojawiać.
- Rozdział 13. – „Inne ważne typy oczekiwania". W tym rozdziale zostaną zaprezentowane często spotykane typy oczekiwania, które nie zostały wcześniej omówione.

- Rozdział 14. – „Schemat bazy danych i analiza indeksów”. Czytelnik dowie się, jak można wykrywać problemy związane z nieoptymalnym projektem bazy danych, a także w jaki sposób oceniać poziom użycia oraz stan indeksów.
- Rozdział 15. – „SQL Server w środowisku wirtualnym”. W rozdziale zostaną przedstawione najlepsze sposoby konfigurowania wirtualnych instancji SQL Server, a także metody radzenia sobie z problemami.
- Rozdział 16. – „SQL Server w chmurze”. Z tego rozdziału Czytelnik dowie się, jak można konfigurować i wykorzystywać SQL Server w chmurowych środowiskach wirtualnych. Rozdział zawiera również przegląd zarządzanych usług SQL Server, dostępnych w chmurach Microsoft Azure, Amazon AWS i Google GCP.

Każdy rozdział kończy się podrozdziałem „Podsumowanie metod rozwiązywania problemów”, który zawiera najważniejsze sposoby radzenia sobie z problemami związanymi z zagadnieniami poruszanymi w danej części książki.

Warto też skorzystać z Dodatku A „Typy oczekiwania”, który jest przewodnikiem po standardowych typach oczekiwania i technikach rozwiązywania problemów.

Konwencje użyte w tej książce

W tej książce zostały użyte następujące konwencje typograficzne:

Czcionka pochyła
: Oznacza nowe terminy, adresy URL, adresy mailowe, nazwy plików i ich rozszerzenia.

`Czcionka o stałej szerokości`
: Używana w listingach, a także w akapitach w celu odwoływania się do elementów programu, takich jak nazwy zmiennych lub funkcji, bazy danych, typy danych, zmienne środowiskowe, instrukcje i słowa kluczowe.

`Pogrubiona czcionka o stałej szerokości`
: Stosowana do wyróżniania poleceń lub innych tekstów, które są wprowadzane przez użytkownika, a także szczególnie istotnych fragmentów kodu.

`Pochyła czcionka o stałej szerokości`
: Używana w przypadku tekstu, który powinien zostać zastąpiony wartościami podanymi przez użytkownika lub wynikającymi z kontekstu.

W ten sposób są oznaczone sztuczki i wskazówki.

W ten sposób są oznaczone ogólne uwagi.

W ten sposób są oznaczone ostrzeżenia.

Użycie przykładów kodu

Dodatkowe materiały (przykłady kodów, ćwiczenia itd.) można pobrać ze strony *https://github.com/aboutsqlserver/code*.

Folder *Troubleshooting Scripts* udostępnia zestaw notatników Azure Data Studio[1] włącznie ze skryptami umożliwiającymi rozwiązywanie problemów i diagnostykę, które wykorzystałem w tej książce. Przykładowe skrypty i aplikacje Czytelnik znajdzie również w folderze *Companion Materials (Books)*.

Jeśli jawnie nie zaznaczono, skrypty będą działać we wszystkich wersjach SQL Server, począwszy od wydania SQL Server 2005. Niektóre kolumny dynamicznego widoku zarządzania mogą jednak nie być obsługiwane w starszych wersjach systemu i należy je zakomentować.

Ponieważ zamierzam utrzymywać i rozszerzać bibliotekę skryptów diagnostycznych, warto co jakiś czas sprawdzać repozytorium.

W przypadku pytań technicznych lub problemów z wykorzystaniem przykładów kodów należy przesłać maila na adres *bookquestions@oreilly.com*.

Celem tej książki jest wsparcie wykonywanej pracy. Ogólnie rzecz ujmując, przykładowe kody dołączone do książki można używać w programach i dokumentacji. Nie trzeba się kontaktować z wydawnictwem w celu uzyskania pozwolenia, chyba że jest wykorzystywana znaczna część kodu. Na przykład napisanie programu, który używa kilka fragmentów kodu z tej książki, nie wymaga pozwolenia. Sprzedaż lub dystrybucja przykładów z książek wydawnictwa O'Reilly wymaga już pozwolenia. Odpowiedź na pytanie poprzez zacytowanie tej książki i wykorzystanie przykładowego kodu nie wymaga pozwolenia. Włączenie znacznej ilości przykładowego kodu z tej książki do dokumentacji produktu wymaga pozwolenia.

Doceniamy uznanie autorstwa, ale nie wymagamy go. Zazwyczaj obejmuje ono tytuł, autora, wydawcę i numer ISBN. Na przykład: „*SQL Server Advanced Troubleshooting and Performance Tuning*, autor Dmitri Korotkevitch (O'Reilly). Copyright 2022 Dmitri Korotkevitch, 978-1-098-10192-3".

1 Produkt Azure Data Studio można pobrać z witryny Microsoft (*https://oreil.ly/zwwCf*).

W przypadku, gdy wykorzystanie przykładów kodu może wykraczać poza dozwolony użytek lub powyżej podane zezwolenie, należy się skontaktować z wydawnictwem poprzez wysłanie maila na adres *permissions@oreilly.com*.

Kontakt z wydawnictwem

Komentarze i pytania dotyczące tej książki należy kierować do wydawcy:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (ze Stanów Zjednoczonych lub Kanady)
707-829-0515 (rozmowa międzynarodowa lub lokalna)
707-829-0104 (faks)

Strona internetowa książki z erratą, przykładami i dodatkowymi informacjami jest dostępna pod adresem *https://oreil.ly/sql-server-advanced*. W celu skomentowania książki lub zadania pytania technicznego należy wysłać maila na adres *bookquestions@oreilly.com*.

Najnowsze wiadomości i informacje o książkach oraz szkoleniach można znaleźć na stronie *http://oreilly.com*.

Jesteśmy na Facebooku: *http://facebook.com/oreilly*.

Można nas także śledzić na Twitterze: *http://twitter.com/oreillymedia*.

Nasz kanał na YouTubie: *http://youtube.com/oreillymedia*.

Kontakt z autorem

Czytelnicy, którzy mają pytania dotyczące książki lub ogólnie SQL Server, mogą wysłać maila na adres *dk@aboutsqlserver.com*. Chętnie udzielę wyjaśnień i odpowiedzi.

Można także odwiedzić mój blog dostępny pod adresem *https://aboutsqlserver.com*. Ponieważ książka została już opublikowana, obiecuję, że znów zacznę go częściej uaktualniać!

Podziękowania

Przede wszystkim chciałbym jak zawsze podziękować mojej rodzinie za okazaną mi pomoc i wsparcie.

Pisanie książki to doskonała wymówka, by uniknąć wykonywania codziennych obowiązków. Nadal nie rozumiem, dlaczego mi się to wciąż udaje!

Jestem ogromnie wdzięczny Erlandowi Sommarskogowi, Thomasowi Grohserowi i Uwe Rickenowi, którzy wykonali wspaniałą pracę będąc recenzentami mojej książki. Ich uwagi radykalnie poprawiły jej treść, dzięki czemu nabrała ona ostatecznego kształtu.

Erland Sommarskog (*https://www.sommarskog.se*) od trzydziestu lat zajmuje się SQL Server. Od 2001 jest posiadaczem tytułu Microsoft Data Platform MVP. Pracuje w Sztokholmie w Szwecji jako niezależny konsultant. Z pasją dzieli się ze społecznością swoją wiedzą i doświadczeniem. Gdy nie zajmuje się SQL Server, gra w brydża i podróżuje.

Thomas Grohser jest od ponad 35 lat specjalistą IT, a od 12 lat posiada tytuł Microsoft Data Platform MVP. Używa SQL Server od 1994 roku i specjalizuje się w architekturze oraz wdrażaniu wysoce bezpiecznych, dostępnych, odzyskiwalnych i wydajnych baz danych oraz leżącej u ich podstaw infrastruktury. W wolnym czasie Thomas uwielbia się dzielić zdobytą przez dziesięciolecia wiedzą ze społecznością użytkowników SQL Server i platformy danych, udzielając się na grupach i konferencjach organizowanych na całym świecie.

Uwe Ricken ma tytuły Microsoft Data Platform MVP i Microsoft Certified Master (SQL Server 2008), a pracuje we Frankfurcie w Niemczech. Uwe poznał SQL Server w 2007 roku i jest specjalistą od wewnętrznych mechanizmów bazy danych, indeksów, architektury i jej projektowania. Regularnie bierze udział w różnych konferencjach i wydarzeniach związanych z SQL Server, a także prowadzi blog pod adresem *http://www.sqlmaster.de*.

Bardzo wam dziękuję! Naprawdę podobała mi się ta podróż! Przekazuję wielkie podziękowania dla mojego kolegi Andre Fiano, jednego z najbardziej kompetentnych

inżynierów infrastruktury, jakiego kiedykolwiek poznałem. Wiele nauczyłem się od niego. Andre pomógł mi także przygotować kilka przykładów wykorzystanych w tej książce.

Moje podziękowania otrzymuje także cały zespół O'Reilly, a zwłaszcza Sarah Grey, Elizabeth Kelly, Kate Dullea, Kristen Brown i Audrey Doyle. Dziękuję za to, że mój angielski stał się zrozumiały, a także, że nauczyłem się rysować diagramy!

Ponieważ ta książka dotyczy SQL Server, chcę podziękować zespołowi inżynierów firmy Microsoft za ciężką pracę nad tym produktem. Z wielką niecierpliwością czekam na jego dalszy rozwój.

Na koniec chciałbym podziękować wszystkim moim przyjaciołom z #SQLFamily za wszelkie wsparcie i zachęty! To przyjemność pisać do tak wspaniałej społeczności!

Dziękuję wszystkim!

Rozdział 1

Instalowanie i konfigurowanie systemu bazy danych SQL Server

Serwery baz danych nigdy nie działają w odosobnieniu. Są wykorzystywane przez jedną lub więcej aplikacji używanych przez klientów. Bazy danych dla takich aplikacji są zarządzane przez instancje SQL Server, zainstalowane na urządzeniach fizycznych lub wirtualnych. Dane są przechowywane na dyskach, które zwykle są współdzielone z innymi programami i systemami baz danych. Oprócz tego wszystkie komponenty wykorzystują sieć do komunikacji i przesyłania danych.

Złożoność i wewnętrzne zależności systemów baz danych sprawiają, że rozwiązywanie problemów jest bardzo trudnym zadaniem. Z punktu widzenia klienta większość problemów polega na pogorszeniu wydajności – aplikacja może działać zbyt wolno i nie reagować na polecenia, czas oczekiwania na odpowiedź z bazy danych może być zbyt długi, a program może nawet się z nią nie połączyć. Pierwotną przyczyną tych problemów może być cokolwiek. Sprzęt może działać nieprawidłowo lub być źle skonfigurowany, schemat bazy danych, sposób użycia indeksów lub sam kod może być nieprawidłowy, SQL Server może być przeciążony, aplikacje klienckie mogą zawierać błędy lub wady projektowe. Oznacza to, że w celu zidentyfikowania i usunięcia problemu należy spojrzeć na system całościowo.

Niniejsza książka dotyczy rozwiązywania problemów związanych z SQL Server. Przed rozpoczęciem pracy zawsze należy odpowiednio przeanalizować środowisko aplikacji i systemu. W tym rozdziale zostaną przedstawione wskazówki, w jaki sposób można przeprowadzić taką walidację i wykryć najczęściej występujące błędy w konfiguracji SQL Server. Najpierw zostanie przeanalizowana konfiguracja sprzętu i systemu operacyjnego. Następnie zostanie zaprezentowana konfiguracja SQL Server i bazy danych. W tym rozdziale zostaną także omówione kwestie dotyczące konsolidacji środowiska SQL Server, a także dodatkowych kosztów, które mogą się pojawić po wdrożeniu monitoringu w systemie.

Rozważania dotyczące sprzętu i systemu operacyjnego

W większości przypadków rozwiązywanie problemów i poprawianie wydajności dotyczy systemów produkcyjnych, które obsługują duże ilości danych i pracują pod dużym obciążeniem. Problemy należy usuwać w środowiskach w trakcie ich pracy. Niemniej jednak nie da się całkowicie uniknąć dyskusji o sprzęcie, ponieważ może się okazać, że serwery nie są odpowiednio wydajne i trzeba je zmodernizować.

W tej książce nie polecamy określonych dostawców, urządzeń lub modeli – sprzęt komputerowy szybko się zmienia i każda taka porada byłaby już przestarzała w momencie wydania tej pozycji. Zamiast tego zaprezentujemy kilka zdroworozsądkowych uwag o ponadczasowym znaczeniu.

Procesor

Koszt licencji komercyjnego silnika bazy danych jest zdecydowanie najdroższym składnikiem systemu. SQL Server nie jest tu wyjątkiem – można zbudować przyzwoity serwer w cenie niższej, niż koszt czterech rdzeni w wersji Enterprise. Należy więc użyć najpotężniejszego procesora, na jaki pozwala budżet, zwłaszcza, jeśli są wykorzystywane wersje inne, niż Enterprise, które ograniczają maksymalną liczbę rdzeni.

Powinno się także zwrócić uwagę na model procesora. Każda generacja procesorów jest wydajniejsza od poprzednich. Dzięki wybraniu nowszego procesora można więc uzyskać wzrost wydajności na poziomie 10-15% nawet wtedy, jeśli taktowanie zegara jest takie samo, jak w przypadku starszego.

W niektórych sytuacjach, gdy koszt licencji nie jest problemem, może się pojawić konieczność wyboru pomiędzy wolniejszymi procesorami z większą liczbą rdzeni a szybszymi, lecz z mniejszą ich liczbą. W takim przypadku decyzja w dużej mierze zależy od obciążenia systemu. Ogólnie rzecz ujmując, systemy przetwarzania transakcji online (ang. *Online Transactional Processing*, w skrócie *OLTP*), a zwłaszcza wykorzystujące pamięć operacyjną (ang. *In-Memory OLTP*), skorzystają z wyższej wydajności pojedynczego rdzenia. Z kolei hurtownia danych i systemy analityczne mogą działać lepiej przy wyższym stopniu równoległości i większej liczbie rdzeni.

Pamięć

W społeczności użytkowników SQL Server krąży taki żart:

> Pytanie: *Ile pamięci zazwyczaj potrzebuje SQL Server?*
>
> Odpowiedź: *Więcej.*

Nie jest to do końca żart. SQL Server potrafi wykorzystywać duże ilości pamięci, co pozwala mu na buforowanie większych ilości danych. To z kolei zmniejsza liczbę dyskowych operacji wejścia i wyjścia oraz poprawia wydajność systemu. Dodanie większej ilości pamięci do serwera może więc być najtańszym i najszybszym sposobem rozwiązania niektórych problemów z wydajnością.

Załóżmy, że w systemie pojawiają się niezoptymalizowane zapytania. Można zmniejszyć ich wpływ poprzez dodanie większej ilości pamięci i wyeliminowanie niepotrzebnych odczytów z dysków, które powodują. Nie rozwiązuje to oczywiście głównej przyczyny problemu. Jest również niebezpieczne, ponieważ zbyt duża ilość danych może się nie zmieścić w pamięci podręcznej. W niektórych przypadkach takie tymczasowe rozwiązanie może być jednak dopuszczalne.

W wersji Enterprise SQL Server nie ma ograniczenia dla ilości pamięci, którą można wykorzystywać. Takie ograniczenie spotyka się jednak w innych. W przypadku SQL Server Standard Edition w wersji 2016 i nowszych można wykorzystywać do 128 GB pamięci RAM w puli buforowej, 32 GB pamięci RAM w trybie In-Memory OLTP oraz 32 GB pamięci RAM do przechowywania kolumnowych segmentów indeksów. Wersja Web pozwala na wykorzystanie maksymalnie połowy tego, co oferuje wydanie Standard. Te ograniczenia należy uwzględnić w przypadku zamiaru uaktualnienia wersji innej, niż Enterprise. Dodatkowe zasoby powinno się przydzielić także innym komponentom SQL Server, na przykład magazynowi planów wykonania i menedżerowi blokad.

Bez względu na powyższe wskazówki należy użyć tyle pamięci, ile jest możliwe. W dzisiejszych czasach jest ona tania. Nie powinno się jednak przesadzać z ilością pamięci, jeśli bazy danych są małe. W każdym z przypadków należy uwzględnić przyszły wzrost danych.

Podsystem dyskowy

Poprawnie działający, szybki podsystem dyskowy jest niezbędny w celu uzyskania wysokiej wydajności SQL Server. Jest on aplikacją bardzo intensywnie korzystającą z operacji wejścia i wyjścia – przez cały czas odczytuje dane z dysku i zapisuje je na niego.

Podsystem dyskowy wykorzystywany przez bazę danych SQL Server można zaprojektować na różne sposoby. Kluczową kwestią jest zapewnienie niskich opóźnień dla żądań operacji wejścia i wyjścia. W przypadku krytycznych systemów warstwy 1. nie powinno się przekraczać opóźnień rzędu 3-5 milisekund (ms) przy odczycie i zapisie danych oraz 1-2 ms podczas zapisywania dziennika transakcji. Na szczęście takie parametry można obecnie łatwo osiągnąć dzięki pamięci masowej opartej na technologii flash.

Jest jednak pewien warunek – podczas rozwiązywania problemów dotyczących wydajności operacji wejścia i wyjścia należy *na poziomie* SQL Server, a nie pamięci masowej analizować wskaźniki związane z opóźnieniami. Z powodu kolejek, które mogą się pojawić przy intensywnym wykorzystywaniu operacji wejścia i wyjścia, często można zaobserwować znacznie wyższe wartości wskaźników SQL Server, niż kluczowych wskaźników wydajności, odnoszących się do pamięci masowej (sposób odczytywania i analizowania danych dotyczących wydajności operacji wejścia i wyjścia przeanalizowano w rozdziale 3).

Jeśli podsystem pamięci masowej zapewnia wiele poziomów wydajności, na dysku najszybszym powinno się umieścić przede wszystkim bazę danych `tempdb`, a dopiero później zadbać o dziennik transakcji i pliki danych. Baza `tempdb` jest zasobem współdzielonym, dlatego ważne jest, by charakteryzowała się wysoką przepustowością operacji wejścia i wyjścia.

Zapisy w plikach dziennika transakcji są wykonywane synchroniczne, więc należy zapewnić, by charakteryzowały się niskim poziomem opóźnień. Zapisy do dziennika

transakcji są również sekwencyjne. Powinno się jednak brać pod uwagę, że umieszczenie wielu plików dziennika i (lub) danych na tym samym dysku doprowadzi do zwiększenia liczby operacji wejścia i wyjścia w wielu bazach danych.

Najlepszym rozwiązaniem jest umieszczenie plików danych i dziennika na różnych dyskach fizycznych ze względu na ich obsługę i odzyskiwanie. Należy jednak dokładniej przyjrzeć się konfiguracji pamięci masowej. Gdy macierz dyskowa nie ma wystarczającej liczby dysków, umieszczenie plików na wielu jednostkach LUN może czasem pogorszyć jej wydajność.

Nie jest zalecane umieszczanie indeksów klastrowych i nieklastrowych w oddzielnych grupach plików, czyli na różnych dyskach. Rzadko poprawia to wydajność operacji wejścia i wyjścia, chyba że dla grup plików można zastosować różne ścieżki pamięci masowej. Z drugiej strony, taka konfiguracja może znacząco skomplikować odzyskiwanie danych po awarii.

Wreszcie należy pamiętać, że niektóre technologie zastosowane w SQL Server nie wykorzystują sekwencyjnych operacji wejścia i wyjścia w celu zwiększenia wydajności. Na przykład technologia In-Memory OLTP w ogóle nie korzysta z operacji wejścia i wyjścia o dostępie swobodnym, a wydajność odczytów sekwencyjnych zwykle staje się czynnikiem ograniczającym uruchamianie i odzyskiwanie bazy danych. Procesy skanowania hurtowni danych również mogą odnieść korzyść z wydajności sekwencyjnych operacji wyjścia i wyjścia, gdy indeksy oparte na B-drzewie i kolumnowe nie są zbytnio pofragmentowane. Różnica między wydajnością operacji wejścia i wyjścia sekwencyjnych i o dostępie swobodnym nie jest duża w przypadku pamięci masowej opartej na technologii flash, może być jednak znaczącym czynnikiem w przypadku dysków magnetycznych.

Sieć

SQL Server łączy się z klientami i innymi serwerami za pośrednictwem sieci, która oczywiście musi zapewnić wystarczającą przepustowość, aby obsłużyć tę komunikację. W związku z tym pojawia się kilka zaleceń.

Po pierwsze, podczas rozwiązywania problemów dotyczących wydajności sieci należy przeanalizować całą jej topologię. Przepustowość sieci jest ograniczona przez jej najwolniejszy element. Na przykład serwer może mieć łącze 10 Gb/s, ale, jeśli w ścieżce sieciowej znajdzie się przełącznik 1 Gb/s, stanie się on czynnikiem ograniczającym. Jest to szczególnie ważne w przypadku sieciowych pamięci masowych. Powinno się więc zapewnić, by ścieżka sieciowa prowadząca do takich urządzeń miała jak najwyższą przepustowość.

Po drugie, w przypadku grup dostępności AlwaysOn oraz klastrów pracy awaryjnej AlwaysOn, w celu monitorowania aktywności węzłów klastra często używa się oddzielnej sieci. W niektórych sytuacjach można również rozważyć utworzenie osobnej sieci dla całego ruchu z grupy dostępności. Jest to właściwa metoda, które poprawia niezawodność klastra w prostych rozwiązaniach, gdy wszystkie węzły należą do tej samej podsieci i mogą wykorzystywać routing warstwy 2. W złożonych środowiskach z wieloma podsieciami mogą się jednak pojawiać problemy z routingiem. Należy wtedy zachować ostrożność i upewnić się, że sieci łączące węzły zostały prawidłowo skonfigurowane, zwłaszcza w przypadku środowisk wirtualnych, które zostaną przeanalizowane w rozdziale 15.

Wirtualizacja przyczynia się do zwiększenia poziomu złożoności. Rozważmy sytuację, w której istnieje zwirtualizowany klaster SQL Server z węzłami zainstalowanymi na różnych maszynach. Należy sprawdzić, czy maszyny te mogą obsługiwać ruch w sieci klastra niezależnie od ruchu wynikającego z obsługi klientów. Przesyłanie wszystkich danych z sieci wirtualnej przez pojedynczą fizyczną kartę sieciową zniweczyłoby cel utworzenia oddzielnej sieci monitorującej węzły klastrów.

Systemy operacyjne i aplikacje

Ogólnie rzecz ujmując, należy używać najnowszej wersji systemu operacyjnego, która wspiera daną wersję SQL Server. Zarówno system operacyjny, jak i SQL Server powinny mieć zainstalowane poprawki. Oprócz tego należy wdrożyć proces, który będzie je regularnie instalować.

Jeśli ma być wykorzystywana stara wersja SQL Server (wcześniejsza, niż 2016), powinno się użyć wariantu 64-bitowego. W większości przypadków wersja 64-bitowa ma wyższą wydajność od wersji 32-bitowej i lepiej dostosowuje się do zmiany sprzętu.

Od wersji SQL Server 2017 można używać Linuksa do udostępniania serwera bazy danych. Z punktu widzenia wydajności wersje SQL Server przeznaczone dla systemów Windows i Linux są do siebie bardzo podobne. Wybór systemu operacyjnego zależy od konfiguracji środowiska w przedsiębiorstwie i od tego, jakie rozwiązanie może być łatwiej zarządzane przez zespół IT. Należy pamiętać, że wdrożenia oparte na systemie Linux mogą wymagać nieco innej strategii wysokiej dostępności (HA), niż w przypadku konfiguracji z systemem Windows. Na przykład, w celu zautomatyzowania przejścia w tryb awaryjny może być potrzebne zainstalowanie oprogramowania Pacemaker zamiast Windows Server Failover Cluster (WSFC).

Jeśli to możliwe, należy używać oddzielnej maszyny przeznaczonej na instalację SQL Server. Łatwiej i taniej można skalować serwery aplikacyjne – nie marnujmy więc cennych zasobów na serwer bazy danych!

Nie należy również uruchamiać na serwerze procesów, które nie są niezbędne. Inżynierowie baz danych często uruchamiają środowisko SQL Server Management Studio (SSMS) w sesjach zdalnego pulpitu. Zawsze lepiej jest pracować zdalnie i nie zużywać zasobów serwera.

Poza tym, jeśli na serwerze jest konieczne uruchomienie oprogramowania antywirusowego, należy wykluczyć ze skanowania foldery baz danych.

Wirtualizacja i środowiska chmurowe

Nowoczesna infrastruktura IT w dużym stopniu zależy od wirtualizacji, która zapewnia wyższą uniwersalność, upraszcza zarządzanie i zmniejsza koszty sprzętu. W wyniku tego coraz częściej zarządza się zwirtualizowaną infrastrukturą SQL Server.

Nie ma w tym nic złego. Prawidłowo wdrożone środowisko wirtualizacji daje wiele korzyści przy akceptowalnym narzucie na wydajność. Dzięki rozwiązaniom VMware vSphere vMotion lub Hyper-V Live Migration pojawia się kolejne narzędzie wysokiej

dostępności. Pozwala to na bezproblemową aktualizację sprzętu i uproszczenie zarządzania bazą danych. Warto więc zwirtualizować SQL Server, chyba że mamy do czynienia z przypadkiem skrajnym, w którym należy wykorzystać maksymalną wydajność sprzętu.

Koszty obsługi wirtualizacji wzrastają w przypadku dużych serwerów z wieloma procesorami. Nadal jednak w wielu przypadkach mogą być akceptowalne.

Pamiętajmy jednak, że wirtualizacja bardziej komplikuje proces rozwiązywania problemów. Oprócz przeanalizowania wskaźników charakteryzujących maszynę wirtualną (VM) należy zwrócić uwagę na stan i obciążenie hosta. Co gorsza, jego wpływ na wydajność może nie być wyraźnie zauważalny w standardowych wskaźnikach dostępnych w systemie operacyjnym maszyny wirtualnej.

W rozdziale 15. zostanie zaprezentowanych kilka metod rozwiązywania problemów związanych z warstwą wirtualizacji. Na samym początku można jednak się skontaktować z inżynierami infrastruktury, aby potwierdzić, że host nie wykorzystuje zbyt dużej ilości zasobów. Należy zwrócić uwagę na liczbę procesorów fizycznych i przydzielonych procesorów wirtualnych (`vCPU`), a także na ilość fizycznej i przydzielonej pamięci. Krytyczne maszyny wirtualne z zainstalowanym SQL Server powinny mieć zarezerwowane zasoby, aby uniknąć problemów związanych z wydajnością.

Jeśli pominie się warstwę wirtualizacji, rozwiązywanie problemów w przypadku zwirtualizowanych instancji SQL Server niczym się nie różni od rozwiązywania problemów z instancjami fizycznymi. To samo dotyczy instalacji w chmurze, gdy SQL Server działa w ramach maszyn wirtualnych. W końcu chmura to tylko inne centrum danych zarządzane przez zewnętrznego dostawcę.

Konfigurowanie SQL Server

Domyślny proces konfigurowania SQL Server jest stosunkowo dobrze zaprojektowany i może być stosowany w przypadku środowisk prostych, a nawet średnio zaawansowanych. Kilka elementów trzeba jednak zatwierdzić i dopasować.

Wersja SQL Server oraz poziom wdrożenia poprawek

Najważniejszą instrukcją wykonywaną podczas sprawdzania stanu SQL Server jest `SELECT @@VERSION`. Są ku temu dwa powody. Po pierwsze, dzięki niej można poznać wersję aplikacji, co umożliwia wdrożenie ewentualnych ulepszeń. Po drugie, pomaga zidentyfikować znane problemy, które mogą istnieć w systemie.

Ten drugi powód jest bardzo ważny. Często klienci proszą o rozwiązywanie problemów, które zostały już naprawione w dodatkach Service Pack i aktualizacjach zbiorczych. Należy zawsze przeczytać informacje o wydaniu, aby sprawdzić, czy któryś z wymienionych

problemów wygląda znajomo. Być może pewne błędy, które negatywnie wpływają na działanie danego SQL Server, zostały usunięte.

W miarę możliwości warto rozważyć aktualizację do najnowszej wersji, by poprawić wydajność, funkcjonalność i skalowalność systemu. Jest to szczególnie ważne, gdy planujemy zainstalować wersję SQL Server 2016 (lub nowszą), a używamy starszej. SQL Server 2016 był wydaniem przełomowym, które zawierało wiele ulepszeń wydajności. Z doświadczeń autora wynika, że zwykłe przejście z wersji SQL Server 2012 na 2016 lub nowszą może poprawić wydajność od 20% do 40% bez podejmowania żadnych dodatkowych działań.

Warto również zauważyć, że począwszy od SQL Server 2016 SP1, wiele funkcji zawartych tylko w wersji Enterprise stało się dostępnych w niższych edycjach produktu. Niektóre z nich, takie jak kompresja danych, pozwalają przechowywać większe ilości danych w puli buforów i poprawiają wydajność systemu.

Przed aktualizacją należy oczywiście przetestować system – zawsze istnieje możliwość wystąpienia problemów regresyjnych. W przypadku drobnych poprawek ryzyko jest zazwyczaj niewielkie, jednak wzrasta przy większych aktualizacjach. W dalszej części tego rozdziału dowiemy się, że dzięki kilku opcjom dotyczącym bazy danych można zmniejszyć wpływ niektórych z zagrożeń.

Natychmiastowa inicjalizacja pliku

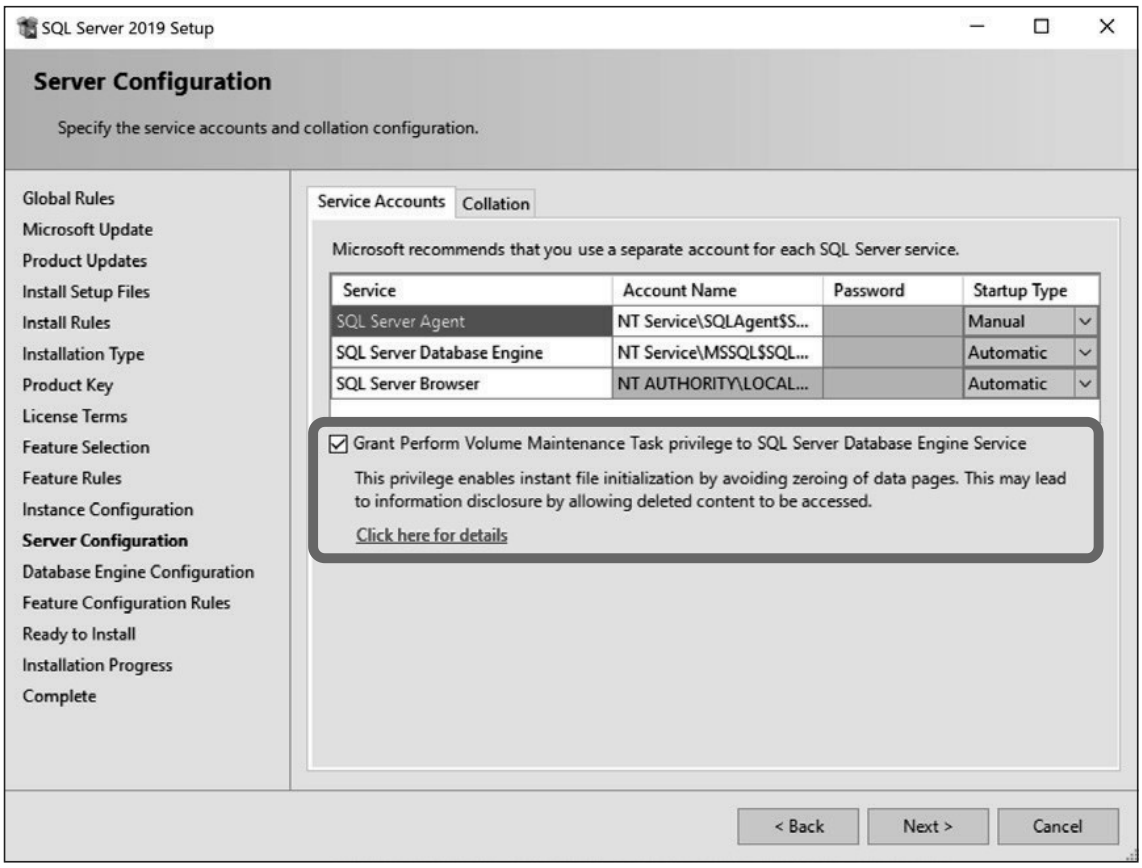
Za każdym razem, gdy SQL Server powiększa pliki danych i dziennika transakcji (automatycznie lub w reakcji na wykonanie polecenia `ALTER DATABASE`), wypełnia nowo przydzieloną część zerami. Proces ten blokuje wszystkie sesje, które próbują zapisywać do odpowiedniego pliku. W przypadku dziennika transakcji przestają być generowane jakiekolwiek wpisy. Może on również powodować skokowy wzrost liczby operacji zapisu.

Dla plików dziennika transakcji nie można zmienić tego zachowania – SQL Server zawsze je zeruje. Można to jednak zrobić dla plików danych poprzez włączenie opcji *natychmiastowej inicjalizacji pliku* (ang. *instant file initialization*, w skrócie *IFI*). Przyspiesza to operację rozszerzania plików danych i skraca czas tworzenia lub przywracania baz danych.

Natychmiastową inicjalizację pliku można włączyć poprzez przypisanie do konta startowego SQL Server uprawnienia `SA_MANAGE_VOLUME_NAME`, znanego również jako „wykonywanie zadań konserwacyjnych w woluminach" (ang. *Perform Volume Maintenance Task*). Można to zrobić w aplikacji zarządzającej *zasadami zabezpieczeń lokalnych* (*secpol.msc*). Aby zmiana zaczęła obowiązywać, trzeba ponownie uruchomić SQL Server.

Jak pokazano na rysunku 1.1, w wersjach SQL Server 2016 i nowszych to uprawnienie można również przyznać podczas procesu konfigurowania systemu.

Istnieje możliwość sprawdzenia, czy opcja natychmiastowej inicjalizacji pliku jest włączona. W tym celu należy wyświetlić kolumnę `instant_file_initialization_enabled` z dynamicznego widoku zarządzania (DMV) `sys.dm_server_services` (*https://oreil.ly/58Vd7*). Ta kolumna jest dostępna w wersjach SQL Server 2012 SP4, SQL Server 2016 SP1 i późniejszych. W przypadku starszych można uruchomić kod pokazany na listingu 1.1.



Rysunek 1.1 *Włączanie opcji natychmiastowej inicjalizacji pliku podczas procesu konfigurowania SQL Server*

Listing 1.1 *Sprawdzanie, czy jest włączona opcja natychmiastowej inicjalizacji pliku w starszych wersjach SQL Server*

```
DBCC TRACEON(3004,3605,-1);
GO
CREATE DATABASE Dummy;
GO
EXEC sp_readerrorlog 0,1,N'Dummy';
GO
DROP DATABASE Dummy;
GO
DBCC TRACEOFF(3004,3605,-1);
GO
```

Jeśli opcja natychmiastowej inicjalizacji pliku nie jest włączona, w dzienniku SQL Server pojawi sie wpis, że oprócz zerowania pliku dziennika *.ldf* następuje zerowanie pliku *.mdf* z danymi (rysunek 1.2). Gdy opcja jest włączona, w dzienniku pojawi się jedynie informacja o zerowaniu pliku dziennika *.ldf*.

	LogDate	ProcessInfo	Text
104	2020-10-26 15:57:45.370	spid32s	A connection timeout has occurred while attempting to establish a connection to availa…
105	2020-10-26 15:58:35.510	spid51	DBCC TRACEON 3004, server process ID (SPID) 51. This is an informational message only;…
106	2020-10-26 15:58:35.510	spid51	DBCC TRACEON 3605, server process ID (SPID) 51. This is an informational message only;…
107	2020-10-26 15:58:35.520	spid51	Zeroing C:\DB\Dummy.mdf from page 0 to 1024 (0x0 to 0x800000)
108	2020-10-26 15:58:35.530	spid51	Zeroing completed on C:\DB\Dummy.mdf (elapsed = 2 ms)
109	2020-10-26 15:58:35.530	spid51	Zeroing C:\DB\Dummy_log.ldf from page 0 to 1024 (0x0 to 0x800000)
110	2020-10-26 15:58:35.540	spid51	Zeroing completed on C:\DB\Dummy_log.ldf (elapsed = 2 ms)
111	2020-10-26 15:58:35.550	spid51	Starting up database 'Dummy'.

Rysunek 1.2 *Sprawdzanie, czy opcja natychmiastowej inicjalizacji pliku została włączona*

Z tym ustawieniem wiąże się niewielkie zagrożenie bezpieczeństwa. Gdy opcja natychmiastowej inicjalizacji pliku jest włączona, administratorzy bazy danych poprzez wyświetlenie nowo przydzielonych stron mogą odczytać niektóre informacje zawarte we wcześniej usuniętych plikach z systemu operacyjnego. Jest to dopuszczalne w przypadku większości systemów.

Konfigurowanie bazy tempdb

Środowisko SQL Server używa systemowej bazy danych `tempdb` w celu przechowywania obiektów tymczasowych, tworzonych przez użytkowników oraz przez samą aplikację. Ta baza jest bardzo aktywnie używana i często staje się źródłem konfliktów w systemie. Sposób rozwiązywania problemów związanych z bazą `tempdb` zostanie przedstawiony w rozdziale 9. W tym skoncentrujemy się na jej konfigurowaniu.

Jak już wspomniano, bazę `tempdb` należy umieścić na najszybszym dysku systemu. Ogólnie rzecz ujmując, dysk ten nie musi być nadmiarowy ani trwały. Baza danych jest zawsze ponownie tworzona podczas uruchamiania SQL Server, więc w tej roli dobrze sprawdzą się lokalny dysk SSD lub krótkotrwała pamięć masowa w chmurze. Należy jednak pamiętać, że SQL Server przestanie działać, jeśli baza `tempdb` będzie niedostępna.

Jeśli jest wykorzystywany SQL Server w wersji innej, niż Enterprise, a serwerowi przydzielono więcej pamięci, niż może zostać zużyta, bazę `tempdb` można umieścić na dysku RAM. Takiej operacji nie powinno się jednak wykonywać w przypadku wersji SQL Server Enterprise – lepsza wydajność zostanie zazwyczaj osiągnięta, jeśli dodatkową pamięć użyje się w puli buforów.

Aby uniknąć braku miejsca, należy wstępnie zaalokować plik `tempdb` o rozmiarze równym wielkości dysku RAM. Oprócz tego na innym dysku powinno się utworzyć dodatkowe, niewielkie pliki dla danych i dziennika. SQL Server nie będzie ich używał, dopóki pliki na dysku RAM nie zostaną zapełnione.

Baza danych `tempdb` powinna się zawsze składać z wielu plików. Niestety, domyślne ustawienia wprowadzane podczas konfigurowania SQL Server nie są optymalne – zwłaszcza w przypadku starszych wersji produktu. Sposób ustalania liczby plików dla bazy `tempdb`

zostanie wyjaśniony w rozdziale 9. Podczas początkowej konfiguracji można jednak zastosować następującą zasadę:

- Jeśli serwer ma osiem lub mniej rdzeni procesora, należy utworzyć taką samą liczbę plików danych.
- Jeśli serwer ma więcej, niż osiem rdzeni CPU, należy w zależności od tego, jaka wartość jest większa, użyć albo ośmiu plików, albo tyle, ile wynosi jedna czwarta z całkowitej liczby rdzeni zaokrąglona w górę, by uzyskać liczbę podzielną przez cztery. Na przykład powinno się użyć 8 plików w serwerze 24-rdzeniowym i 12 plików w serwerze 40-rdzeniowym.

Ważne jest, by wszystkie pliki danych z bazy `tempdb` miały ten sam rozmiar początkowy, a parametry automatycznego poszerzania były określone w megabajtach (MB), a nie w procentach. Dzięki temu środowisko SQL Server będzie mogło lepiej zrównoważyć wykorzystanie plików danych, zmniejszając przez to ewentualne obciążenia systemu.

Opcje śledzenia

SQL Server wykorzystuje opcje śledzenia w celu włączania niektórych funkcji produktu lub zmiany ich działania. Chociaż firma Microsoft wprowadza w kolejnych wersjach oprogramowania coraz więcej możliwości konfigurowania bazy danych i serwera, opcje śledzenia są nadal szeroko stosowane. Należy sprawdzić wszystkie, które są dostępne w systemie – być może niektóre z nich trzeba będzie włączyć.

Listę włączonych opcji śledzenia można uzyskać za pomocą polecenia `DBCC TRACESTATUS`. Można je włączać za pomocą narzędzia SQL Server Configuration Manager i (lub) przy użyciu parametru startowego `-T` SQL Server.

Oto kilka często stosowanych opcji śledzenia:

`T1118`

Ta opcja śledzenia zapobiega użyciu obszarów mieszanych (*https://oreil.ly/CnPxm*). W przypadku wersji SQL Server 2014 i wcześniejszych pozwala na poprawę szybkości działania bazy `tempdb` poprzez zmniejszenie liczby zmian, a tym samym współzawodniczenia o zasoby w katalogach systemowych tej bazy. Ta opcja nie jest wymagana w wersjach SQL Server 2016 i późniejszych, ponieważ w ich przypadku baza `tempdb` domyślnie nie używa obszarów mieszanych.

`T1117`

Dzięki tej opcji śledzenia SQL Server automatycznie rozszerza wszystkie pliki danych z grupy plików, gdy tylko w jednym z nich brakuje miejsca. Zapewnia to bardziej zrównoważone obciążenie plików operacjami wejścia i wyjścia. Tę opcję należy włączyć, aby poprawić wydajność bazy `tempdb` w starszych wersjach SQL Server. Wcześniej powinno się jednak sprawdzić, czy bazy danych użytkowników zawierają pliki z różnymi rozmiarami. Podobnie jak w przypadku `T1118`, ta opcja śledzenia nie jest wymagana w produkcie SQL Server 2016 i nowszych, ponieważ w ich przypadku baza `tempdb` domyślnie automatycznie rozszerza wszystkie pliki danych.

`T2371`

Domyślnie SQL Server automatycznie aktualizuje statystyki dopiero po stwierdzeniu, że w indeksie zmianie uległo co najmniej 20% danych. Oznacza to, że w przypadku dużych tabel statystyki rzadko są automatycznie aktualizowane. Opcja śledzenia `T2371` zmienia to zachowanie poprzez ustanowienie dynamicznego progu aktualizowania statystyk. Polega on na tym, że im większa tabela, tym mniejszy odsetek zmian będzie wymagany do uruchomienia aktualizacji. Począwszy od wersji SQL Server 2016 to zachowanie można również modyfikować poprzez zmianę poziomu zgodności bazy danych. Mimo tego włączenie tej opcji śledzenia jest wciąż zalecane, chyba że wszystkie bazy danych na serwerze mają poziom zgodności 130 lub wyższy.

`T3226`

Gdy ta opcja jest włączona, SQL Server nie zapisuje do dziennika błędów informacji o udanych operacjach tworzenia kopii zapasowych bazy danych. Dzięki temu można zmniejszyć rozmiar dzienników i sprawić, że będą łatwiejsze do zarządzania.

`T1222`

Opcja ta powoduje zapisywanie grafów zakleszczeń do dziennika błędów SQL Server. Opcja ta jest niegroźna, jednak utrudnia odczytywanie i analizowanie dzienników. Jest również zbędna – w razie potrzeby można uzyskać graf zakleszczeń z sesji `System_Health Extended Event`. Autor zazwyczaj wyłącza tę opcję.

`T4199`

Ta opcja śledzenia wraz z opcją bazy danych `QUERY_OPTIMIZER_HOTFIXES` (dostępna w wersji SQL Server 2016 i nowszych) sterują zachowaniem poprawek dla optymalizatora zapytań (Query Optimizer). Gdy opcja ta jest włączona, zostaną użyte poprawki dostępne w pakietach serwisowych i aktualizacjach skumulowanych. Może to umożliwić usunięcie niektórych błędów w optymalizatorze i poprawić wydajność zapytań. Po zaktualizowaniu narzędzia zwiększa jednak ryzyko pojawienia się błędów regresji. Autor zazwyczaj nie włącza tej opcji śledzenia w systemach produkcyjnych, chyba że przed wprowadzeniem poprawek jest możliwe przeprowadzenie dokładnych testów regresyjnych systemu.

`T7412`

Ta opcja śledzenia umożliwia minimalne profilowanie infrastruktury wykonawczej w SQL Server 2016 i 2017. Pozwala na gromadzenie planów wykonania i wielu wskaźników wykonania dotyczących zapytań w systemie, przy jednoczesnym niewielkim obciążeniu procesora. Bardziej szczegółowo zostanie ona przeanalizowana w rozdziale 5.

Podsumowując – w SQL Server 2014 i wcześniejszych należy włączyć opcje `T1118`, `T2371`, a być może także `T1117`. W wersjach SQL Server 2016 i późniejszych powinno się włączyć opcję `T2371`, chyba że wszystkie bazy danych mają poziom zgodności 130 lub wyższy. Po wykonaniu tych działań należy się przyjrzeć wszystkim innym opcjom śledzenia w systemie i zrozumieć ich działanie. Niektóre z nich mogą być nieumyślnie włączone przez narzędzia innych firm i negatywnie wpływać na wydajność serwera.

Opcje serwera

SQL Server udostępnia wiele ustawień konfiguracyjnych. Wiele z nich zostanie dokładnie przeanalizowanych w dalszej części książki. Istnieje jednak kilka ustawień, o których warto wspomnieć już teraz.

Optymalizacja obciążeń doraźnych

Pierwszym ustawieniem konfiguracyjnym jest *Optimize for Ad-hoc Workloads* (optymalizacja obciążeń doraźnych). Ta opcja pozwala modyfikować sposób, w jaki SQL Server buforuje plany wykonania zapytań doraźnych (niesparametryzowanych). Jeśli opcja jest wyłączona (stan domyślny), SQL Server buforuje pełne plany wykonania tych zapytań, co może znacznie zwiększyć zużycie pamięci podręcznej. Jeśli opcja jest włączona, SQL Server zaczyna od buforowania niewielkiej struktury (o wielkości zaledwie kilkuset bajtów), zwanej *zalążkiem planu*, a następnie zastępuje ją pełnym planem wykonania, jeśli zapytanie doraźne zostanie wykonane po raz drugi.

W większości przypadków zapytania doraźne nie są wykonywane wielokrotnie, dlatego w każdej wersji systemu warto włączyć ustawienie *Optimize for Ad-hoc Workloads*. Może to znacznie zmniejszyć zużycie pamięci podręcznej kosztem rzadko wykonywanych dodatkowych rekompilacji zapytań doraźnych. Oczywiście ustawienie to nie ma wpływu na działanie buforowania zapytań sparametryzowanych i kodu T-SQL bazy danych.

Począwszy od wersji SQL Server 2019 (a także w systemie Azure SQL Database) można zmieniać ustawienie *Optimize for Ad-hoc Workloads* na poziomie bazy danych za pomocą opcji konfiguracyjnej `OPTIMIZE_FOR_AD_HOC_WORKLOADS`.

Maksymalna ilość pamięci wykorzystywana przez serwer

Kolejnym ważnym ustawieniem jest *Max Server Memory* (maksymalna ilość pamięci wykorzystywana przez serwer), które określa, ile pamięci może zużywać SQL Server. Inżynierowie baz danych uwielbiają toczyć dyskusje o tym, jak prawidłowo skonfigurować to ustawienie. Wielu z nich ma własną metodę określania właściwej wartości. Niektórzy sugerują nawet, by pozostawić wartość domyślną, dzięki czemu SQL Server będzie mógł nią automatycznie zarządzać. Najlepszym rozwiązaniem jest jednak modyfikacja tego ustawienia. Ważne jest, by zrobić to poprawnie (więcej szczegółów na ten temat w rozdziale 7.). Nieprawidłowe ustawienie będzie miało większy wpływ na wydajność SQL Server, niż pozostawienie wartości domyślnej.

Poważnym problemem, który często spotyka się podczas kontroli stanu systemu, jest błędna wartość tego ustawienia. Czasem zapomina się o jego zmianie po aktualizacji sprzętu lub maszyny wirtualnej. Innym razem jest ono nieprawidłowo ustalone w środowiskach niededykowanych, w których SQL Server współdzieli serwer z innymi aplikacjami. W obu przypadkach można uzyskać natychmiastową poprawę poprzez zwiększenie

wartości ustawienia *Max Server Memory* lub nawet zastosowanie wartości domyślnej do czasu przeprowadzenia pełnej analizy w późniejszym czasie.

Maska koligacji

Jeśli SQL Server działa na sprzęcie z wieloma węzłami o niejednorodnym dostępie do pamięci (NUMA), należy sprawdzić koligację i ewentualnie ustawić maskę koligacji. W nowoczesnym sprzęcie każdy fizyczny procesor zazwyczaj staje się osobnym węzłem NUMA. Jeśli ograniczamy SQL Server możliwość korzystania z niektórych fizycznych rdzeni, musimy równomiernie przypisać procesory (lub zarządców zadań – patrz rozdział 2.) do węzłów NUMA.

Załóżmy przykładowo, że środowisko SQL Server działa na serwerze z dwoma 18-rdzeniowymi procesorami Xeon. Jeśli zezwolimy systemowi na wykorzystanie jedynie 24 rdzeni, będziemy musieli ustawić maskę koligacji tak, by używał po 12 rdzeni z każdego fizycznego CPU. Dzięki temu uzyskamy lepszą wydajność, niż gdyby SQL Server używał na przykład 18 rdzeni z pierwszego procesora i 6 rdzeni z drugiego.

Na listingu 1.2 przedstawiono kod, który pozwala na wyświetlenie informacji o przypisaniu zarządców zadań (procesorów) SQL Server do węzłów NUMA. W uzyskanym wyniku należy zwrócić uwagę na liczbę zarządców zadań dostępnych w każdej kolumnie `parent_node_id`.

Listing 1.2 *Wyświetlenie informacji o przypisaniu zarządców zadań (procesorów) do węzłów NUMA*

```
SELECT
  parent_node_id
  ,COUNT(*) as [Schedulers]
  ,SUM(current_tasks_count) as [Current]
  ,SUM(runnable_tasks_count) as [Runnable]
FROM sys.dm_os_schedulers
WHERE status = 'VISIBLE ONLINE'
GROUP BY parent_node_id;
```

Równoległość

Ważnym zadaniem jest sprawdzenie parametrów dotyczących równoległości. Ustawienia domyślne, takie jak `MAXDOP = 0` i `Cost Threshold for Parallelism = 5`, nie działają dobrze w nowoczesnych systemach. Podobnie jak w przypadku ustawienia *Max Server Memory*, lepiej jest precyzyjnie dostroić parametry w oparciu o obciążenie systemu (więcej na ten temat w rozdziale 6.). Podstawowa, ogólna zasada dotycząca ustawień jest jednak następująca:

- Należy ustawić wartość parametru `MAXDOP` na jedną czwartą liczby dostępnych procesorów w systemach OLTP, a połowę tej liczby w hurtowniach danych. W bardzo dużych serwerach OLTP wartość `MAXDOP` powinna wynosić 16 lub mniej. Nie należy przekraczać liczby zarządców zadań w węźle NUMA.

- Należy ustawić wartość parametru `Cost Threshold for Parallelism` na 50.

Począwszy od wersji SQL Server 2016 (a także w produkcie Azure SQL Server Database) można ustawić wartość `MAXDOP` na poziomie bazy danych za pomocą polecenia `ALTER DATABASE SCOPED CONFIGURATION SET MAXDOP`. Jest to przydatne, gdy instancja obsługuje bazy danych o różnych obciążeniach.

Ustawienia konfiguracyjne

Podobnie jak w przypadku opcji śledzenia, należy przeanalizować wszelkie zmiany w ustawieniach konfiguracyjnych, które zostały zastosowane w serwerze. Za pomocą widoku `sys.configurations` (*https://oreil.ly/nsLMW*) można sprawdzić bieżące opcje konfiguracyjne. Niestety, SQL Server nie udostępnia listy domyślnych wartości konfiguracyjnych, więc należy je zapisać w kodzie programu, jak pokazano na listingu 1.3. Aby zaoszczędzić miejsce, zamieszczono na nim tylko kilka ustawień konfiguracyjnych, jednak pełną wersję skryptu można pobrać z materiałów dostępnych w repozytorium książki.

Listing 1.3 *Wykrywanie zmian w ustawieniach konfiguracyjnych serwera*

```
DECLARE
    @defaults TABLE
    (
        name SYSNAME NOT NULL PRIMARY KEY,
        def_value SQL_VARIANT NOT NULL
    )

INSERT INTO @defaults(name,def_value)
VALUES('backup compression default',0);
INSERT INTO @defaults(name,def_value)
VALUES('cost threshold for parallelism',5);
INSERT INTO @defaults(name,def_value)
VALUES('max degree of parallelism',0);
INSERT INTO @defaults(name,def_value)
VALUES('max server memory (MB)',2147483647);
INSERT INTO @defaults(name,def_value)
VALUES('optimize for ad hoc workloads',0);
/* Pozostałe ustawienia pominięto na listingu opublikowanym w książce */

SELECT
    c.name, c.description, c.value_in_use, c.value
    ,d.def_value, c.is_dynamic, c.is_advanced
FROM
    sys.configurations c JOIN @defaults d ON
        c.name = d.name
WHERE
    c.value_in_use <> d.def_value OR
```

```
    c.value <> d.def_value
ORDER BY
    c.name;
```

Rysunek 1.3 przedstawia przykładowe dane wyjściowe uzyskane po uruchomieniu powyższego kodu. Rozbieżność między kolumnami `value` i `value_in_use` oznacza oczekujące zmiany konfiguracyjne, które wymagają ponownego uruchomienia, aby zostały uwzględnione. Kolumna `is_dynamic` informuje, czy opcja konfiguracyjna może zostać zmodyfikowana bez ponownego uruchomienia.

	name	description	value_in_use	value	def_value	is_dynamic	is_advanced
1	max degree of parallelism	maximum degree of paralle…	1	1	0	1	1
2	optimize for ad hoc workloads	When this option is set, …	1	1	0	1	1

Rysunek 1.3 *Niestandardowe opcje konfiguracji serwera*

Konfigurowanie baz danych

W kolejnym kroku należy zatwierdzić kilka ustawień i opcji konfiguracyjnych dotyczących bazy danych. Przyjrzyjmy się im.

Ustawienia dotyczące bazy danych

Środowisko SQL Server umożliwia modyfikowanie wielu ustawień bazy danych, aby dopasować jej działanie do obciążenia systemu i innych wymagań. Wiele z nich zostanie przeanalizowanych w dalszej części książki. Istnieje jednak kilka ustawień, które należy wyjaśnić już teraz.

Pierwszym z nich jest *Auto Shrink* (automatyczne zmniejszanie rozmiaru). Gdy ta opcja jest włączona, SQL Server okresowo zmniejsza rozmiar bazy danych i zwalnia nieużywane miejsce do systemu operacyjnego. Chociaż taka operacja wydaje się być atrakcyjna, ponieważ umożliwia zmniejszenie wykorzystania przestrzeni dyskowej, może jednak spowodować powstanie pewnych problemów.

Proces zmniejszania rozmiaru bazy danych działa na poziomie fizycznym. Lokalizuje niewykorzystaną przestrzeń na początku pliku i przenosi do niej zaalokowane obszary z końca, nie biorąc pod uwagę, kto jest ich właścicielem. Wprowadza to zauważalne obciążenie i powoduje fragmentację indeksów. Co więcej, w wielu przypadkach taka operacja jest bezużyteczna – pliki bazy po prostu ponownie się powiększają wraz ze wzrostem ilości danych. W każdym przypadku lepszym rozwiązaniem jest ręczne zarządzanie przestrzenią plików i wyłączenie opcji *Auto Shrink*.

Kolejna opcja bazy danych, *Auto Close* (automatyczne zamykanie), służy do określania sposobu, w jaki SQL Server buforuje dane z bazy. Jeśli opcja ta jest włączona, SQL Server usuwa strony danych z puli buforów oraz plany wykonania z magazynu planów, gdy baza danych nie posiada żadnych aktywnych połączeń. Ma to wpływ na wydajność nowych sesji, gdy dane muszą zostać zbuforowane, a zapytania ponownie skompilowane.

Poza nielicznymi wyjątkami należy zawsze wyłączać funkcję *Auto Close*. Jednym z nich może być sytuacja, gdy instancja zarządza dużą liczbą rzadko używanych baz danych. Nawet w takim przypadku warto jednak pozostawić tę opcję wyłączoną i pozwolić SQL Server na usunięcie zbuforowanych danych w normalny sposób.

Opcja *Page Verify* (weryfikacja strony) powinna zostać ustawiona na `CHECKSUM`. Pozwoli to skuteczniej wykrywać błędy związane ze spójnością i pomoże rozwiązać problemy wynikające z uszkodzenia bazy danych.

Należy zwrócić uwagę na typ *modelu odzyskiwania bazy danych*. Jeśli bazy danych używają trybu odzyskiwania `SIMPLE`, w przypadku awarii niemożliwe będzie odzyskanie danych sprzed ostatniej kopii zapasowej `FULL`. Gdy takie ustawienie zostanie wykryte, należy natychmiast poinformować o tym odpowiednich interesariuszy i upewnić się, że rozumieją ryzyko utraty danych.

Opcja *Database Compatibility Level* (poziom zgodności z bazą danych) określa kompatybilność i zachowanie SQL Server na poziomie bazy danych. Na przykład, jeśli jest używana wersja SQL Server 2019, a poziom zgodności bazy danych jest równy 130 (SQL Server 2016), system będzie się zachowywał tak, jakby baza danych została uruchomiona w środowisku SQL Server 2016. Stosowanie niższych poziomów zgodności upraszcza aktualizowanie programu SQL Server poprzez minimalizację prawdopodobieństwa pojawienia się problemów związanych z regresją. Z drugiej strony nie pozwala jednak na wykorzystanie niektórych nowych funkcji i ulepszeń.

Zalecaną zasadą jest uruchamianie bazy danych z najnowszym poziomem zgodności, który odpowiada bieżącej wersji SQL Server. Należy być ostrożnym przy zmianie poziomu – podobnie jak każda zmiana wersji, może ona powodować problemy regresyjne. Przed modyfikacją powinno się przetestować system i upewnić, że w razie konieczności można powrócić do poprzedniej wersji – zwłaszcza, jeśli baza danych ma poziom zgodności 110 (SQL Server 2012) lub niższy. Zwiększenie poziomu zgodności do 120 (SQL Server 2014) lub wyższego umożliwi zastosowanie nowego modelu szacowania liczności i może znacząco zmienić plany wykonania zapytań. System powinien zostać dokładnie przetestowany, by ustalić, co zmieniło się po wprowadzeniu zmiany.

Można jednak zmusić SQL Server do używania starszych modeli szacowania liczności z nowymi poziomami zgodności bazy danych. W przypadku wersji SQL Server 2016 lub nowszych należy opcję bazy danych `LEGACY_CARDINALITY_ESTIMATION` ustawić na `ON`, a w wersji SQL Server 2014 włączyć opcję śledzenia `T9481`. Takie rozwiązanie umożliwi przeprowadzanie aktualizacji lub modyfikowanie poziomu zgodności etapami, a przez to zmniejszy wpływ operacji na system (szacowanie liczności będzie dokładniej analizowane w rozdziale 5., w którym zostanie wyjaśnione, jak zmniejszyć ryzyko podczas aktualizacji programu SQL Server i zmiany poziomu zgodności bazy danych).

Ustawienia związane z dziennikiem transakcji

SQL Server podczas zapisywania w dzienniku transakcyjnym informacji o wszystkich zmianach w bazie danych wykorzystuje metodę rejestrowania z wyprzedzeniem (ang. *write-ahead logging*). Dzienniki transakcji są obsługiwane sekwencyjnie, w sposób cykliczny.

W większości przypadków nie ma potrzeby stosowania wielu plików dziennika – komplikują one zarządzanie bazą danych i rzadko poprawiają wydajność.

Wewnętrznie SQL Server dzieli dzienniki transakcji na części zwane wirtualnymi plikami dzienników (ang. *Virtual Log Files*, w skrócie *VLF*) i zarządza nimi jak pojedynczymi jednostkami. Na przykład SQL Server nie może zresetować rozmiaru pliku VLF i ponownie go użyć, jeśli zawiera nawet jeden aktywny rekord dziennika. Należy zwrócić uwagę na liczbę plików VLF w bazie danych. Zbyt mała liczba bardzo dużych plików VLF sprawi, że zarządzanie dziennikami i ich zmniejszanie będzie nieoptymalne. Z drugiej strony, zbyt duża liczba małych plików VLF spowoduje pogorszenie wydajności operacji w dzienniku transakcji. W środowisku produkcyjnym nie powinno się przekraczać liczby kilkuset plików VLF.

Liczba plików VLF, które SQL Server dodaje podczas powiększania dziennika, zależy od wersji systemu i wielkości przyrostu. W większości przypadków, gdy przyrost zawiera się pomiędzy 64 MB a 1 GB, system tworzy 8 plików VLF. Jeśli przyrost przekracza 1 GB, zostaje dodanych 16 plików VLF. W przypadku opcji automatycznego poszerzania dziennika nie należy używać metody opartej na procentach, ponieważ powoduje ona wygenerowanie wielu plików VLF o różnych wielkościach. Zamiast tego powinno się umożliwić rozszerzanie pliku za pomocą segmentów o określonej wielkości. Najczęściej są wykorzystywane segmenty o wielkości 1024 MB (chyba że jest wymagany bardzo duży dziennik transakcji), co powoduje wygenerowanie plików VLF o rozmiarze 128 MB.

W przypadku wersji SQL Server 2016 i późniejszych można za pomocą widoku `sys.dm_db_log_info` określić, ile plików VLF zawiera baza danych. W starszych wersjach taką informację można uzyskać za pomocą polecenia `DBCC LOGINFO`. Jeśli dziennik transakcji nie został poprawnie skonfigurowany, należy rozważyć jego przebudowę. Można to wykonać poprzez zmniejszenie dziennika do minimalnego rozmiaru, a następnie poszerzenie go przy użyciu segmentów o wielkości od 1024 MB do 4096 MB.

Nie należy umożliwiać automatycznego zmniejszania plików dziennika transakcji. Taka operacja nie zapobiegnie ich ponownemu poszerzaniu, co wpłynie na wydajność. Lepiej wstępnie zaalokować przestrzeń, a następnie ręcznie zarządzać rozmiarem pliku dziennika. Nie powinno się jednak ograniczać maksymalnego rozmiaru pliku oraz wyłączać opcji automatycznego poszerzania – chcemy przecież, aby dzienniki były automatycznie powiększane w nagłych przypadkach (rozwiązywanie problemów z dziennikami transakcji zostanie dokładniej omówione w rozdziale 11).

Pliki danych i grupy plików

SQL Server domyślnie tworzy nowe bazy danych wykorzystując w tym celu grupę plików `PRIMARY`, składającą się z jednego pliku, a także jeden plik dziennika transakcji. Niestety, taka konfiguracja jest nieoptymalna z punktu widzenia wydajności, sposobu zarządzania bazą danych oraz wymagań wysokiej dostępności.

Środowisko SQL Server monitoruje wykorzystanie miejsca w plikach danych poprzez strony systemowe zwane *mapami alokacji*. Mogą one być źródłem konfliktów w przypadku systemów z często zmieniającymi się danymi – SQL Server szereguje dostęp do map alokacji

podczas ich modyfikowania (więcej o tym w rozdziale 10.). Każdy plik danych zawiera własny zestaw stron z mapą alokacji. Problemy związane ze współzawodnictwem można zmniejszyć poprzez umieszczenie wielu plików w grupie plików z często modyfikowanymi danymi.

Należy się upewnić, że dane zostały równomiernie rozmieszczone w wielu plikach zawartych w tej samej grupie plików. SQL Server używa algorytmu wypełnienia proporcjonalnego, który pozwala na zapis większości danych do pliku z największą ilością wolnego miejsca. Zrównoważenie rozmiarów plików danych pomoże wyrównać poziomy zapisów, zmniejszając tym samym współzawodnictwo podczas dostępu do mapy alokacji. Ważne, by wszystkie pliki danych w grupie plików miały ten sam rozmiar i parametry automatycznego poszerzania, określone w megabajtach.

Można również włączyć opcję `AUTOGROW_ALL_FILES` związaną z grupami plików (dostępną w wersji SQL Server 2016 i nowszych), która jednocześnie włącza automatycznie poszerzanie dla wszystkich plików zawartych w grupie plików. W przypadku wcześniejszych wersji SQL Server można w tym celu wykorzystać opcję śledzenia `T1117`. Należy jednak pamiętać, że ta opcja jest ustawiana na poziomie serwera i będzie miała wpływ na wszystkie bazy danych i grupy plików w systemie.

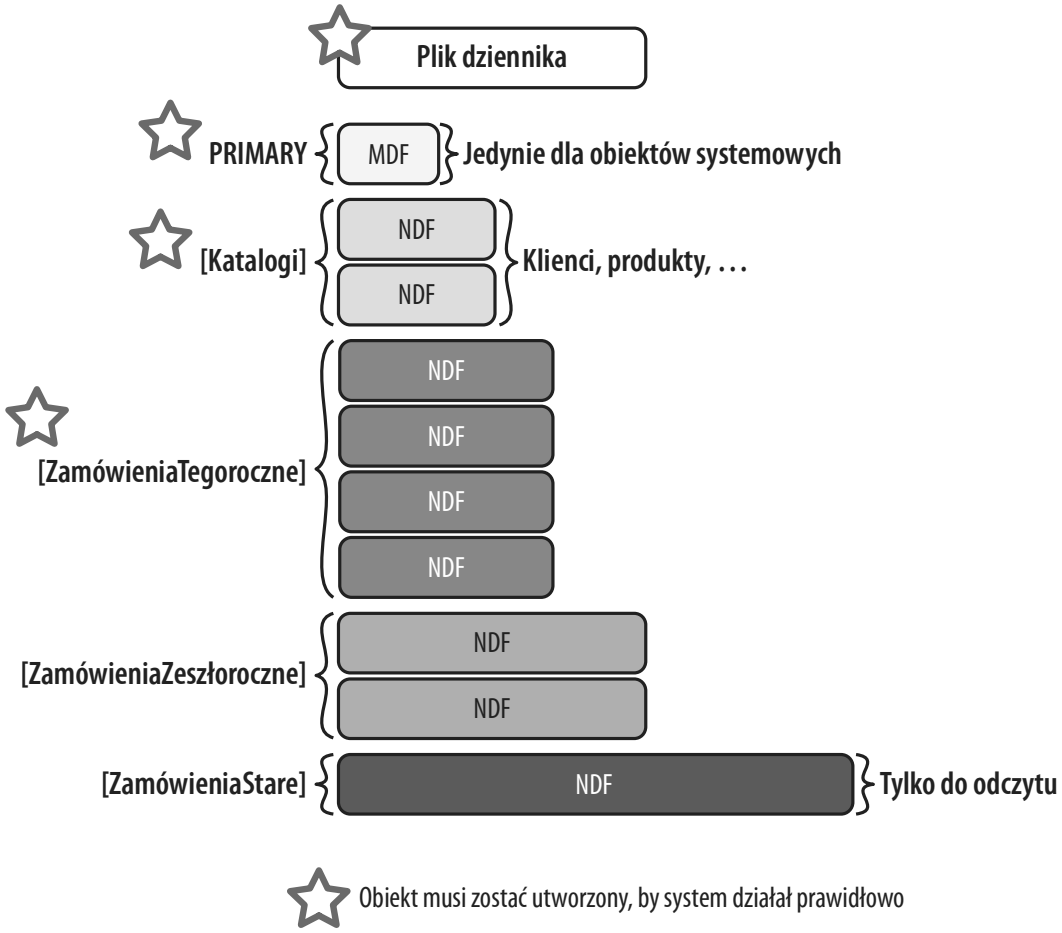
Zmiana układu istniejących baz danych jest często niepraktyczna lub niemożliwa. Może się jednak okazać, że podczas dostrajania wydajności trzeba będzie utworzyć nowe grupy plików i przenieść dane. Oto kilka sugestii, które pozwolą na wykonanie tego procesu w sposób efektywny:

- Należy utworzyć wiele plików w grupach plików z szybko zmieniającymi się danymi. Zwykle powinno się zaczynać od czterech plików, a następnie zwiększać ich liczbę, jeśli pojawiają się problemy z zatrzaskami (patrz rozdział 10.). Ważne, by wszystkie pliki danych miały ten sam rozmiar i parametry automatycznego wzrostu określone w megabajtach. Należy także włączyć opcję `AUTOGROW_ALL_FILES`. W przypadku grup plików z danymi tylko do odczytu zwykle wystarczy utworzenie jednego pliku.
- Nie należy umieszczać indeksów klastrowych, indeksów nieklastrowych lub danych typu LOB (ang. *large object*) w wielu grupach plików. Rzadko pozwala to zwiększyć wydajność, a w przypadku uszkodzenia bazy danych może spowodować problemy.
- Powiązane encje (na przykład `Orders` i `OrderLineItems`) powinno się przechowywać w tej samej grupie plików. Ułatwi to zarządzanie bazą danych oraz jej odtwarzanie po awarii.
- Jeśli to możliwe, grupa plików `PRIMARY` powinna być pusta.

Rysunek 1.4 przedstawia przykładową strukturę bazy danych dla hipotetycznego systemu handlu elektronicznego. Dane zostały podzielone na partycje i umieszczone w wielu grupach plików w celu zminimalizowania przestojów i wykorzystania częściowej dostępności bazy danych w przypadku awarii[1]. Pozwoliło to również ulepszyć strategię backupu po-

1 Czytelnikom, którzy chcą się dokładniej zapoznać z zagadnieniem partycjonowania danych i strategią ich odzyskiwania po awarii, polecamy książkę *Pro SQL Server Internals, Second Edition* (Apress, 2016).

przez wykonywanie częściowych kopii zapasowych bazy danych, a w przypadku pełnych kopii zapasowych wyłączenie danych tylko do odczytu.



Rysunek 1.4 *Struktura bazy danych dla systemu handlu elektronicznego*

Analizowanie dziennika błędów w SQL Server

Dziennik błędów SQL Server to kolejny element, który zazwyczaj należy od razu przeanalizować po pojawieniu się problemu. Informacje, które można w nim znaleźć, mogą wskazywać pewne obszary wymagające dalszych działań. Na przykład błędy 823 i 824 oznaczają problemy z podsystemem dyskowym i (lub) uszkodzenie bazy danych.

Zawartość dziennika błędów można odczytać w środowisku SQL Server Management Studio. Można ją również wyświetlić programowo za pomocą systemowej procedury składowanej `xp_readerrorlog`. W tym przypadku wyzwaniem jest ilość danych w dzienniku – nadmiar komunikatów informacyjnych może ukrywać użyteczne dane.

Ten problem można rozwiązać za pomocą kodu z listingu 1.4. Pozwala on odfiltrować niepotrzebne informacje i skoncentrować się na komunikatach o błędach. Działaniem kodu można sterować za pomocą następujących zmiennych:

`@StartDate` i `@EndDate`
: Określenie czasu analizy.

`@StartDate` *i* `@EndDate`
: Zakres czasowy analizy.

`@NumErrorLogs`
: Liczba plików dziennika do odczytania, jeśli SQL Server stosuje strategię dostępu cyklicznego.

`@ExcludeLogonErrors`
: Pomijanie komunikatów audytu logowania.

`@ShowSurroundingEvents` *i* `@ExcludeLogonSurroundingEvents`
: Umożliwienie pobrania komunikatów informacyjnych znajdujących się w pobliżu wpisów z błędami. Okno czasowe dla tych wiadomości jest określane przez zmienne `@Surrounding EventsBeforeSeconds` i `@SurroundingEventsAfterSeconds`.

Skrypt generuje dwa zestawy danych wyjściowych. Pierwszy z nich zawiera wpisy z dziennika błędów, które zawierają słowo *error*. Gdy jest używana opcja `@ShowSurroundingEvents`, w wynikach pojawiają się również wpisy z dziennika, które towarzyszą wierszom z błędami. Wpisy z dziennika zawierające słowo *error* można wykluczyć poprzez umieszczenie ich w tablicy `@ErrorsToIgnore`.

Listing 1.4 *Analizowanie dziennika błędów z SQL Server*

```
IF OBJECT_ID('tempdb..#Logs',N'U') IS NOT NULL DROP TABLE #Logs;
IF OBJECT_ID('tempdb..#Errors',N'U') IS NOT NULL DROP TABLE #Errors;
GO

CREATE TABLE #Errors
(
  LogNum INT NULL,
  LogDate DATETIME NULL,
  ID INT NOT NULL identity(1,1),
  ProcessInfo VARCHAR(50) NULL,
  [Text] NVARCHAR(MAX) NULL,
  PRIMARY KEY(ID)
);
CREATE TABLE #Logs
(
  [LogDate] DATETIME NULL,
  ProcessInfo VARCHAR(50) NULL,
  [Text] NVARCHAR(MAX) NULL
);

DECLARE
```