Apress

TWOJA OSTATNIA DESKA RATUNKU PRZY PROBLEMACH Z BAZĄ!

Oracle Database 12c

Problemy i rozwiązania

Sam Alapati, Darl Kuhn, Bill Padfield

Helion

Tytuł oryginału: Oracle Database 12c Performance Tuning Recipes: A Problem-Solution Approach

Tłumaczenie: Andrzej Watrak

ISBN: 978-83-246-9801-1

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *http://helion.pl* (księgarnia internetowa, katalog książek)

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:
ftp://ftp.helion.pl/przyklady/or12pr.zip

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/or12pr
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

Spis treści

ROZDZIAŁ 1

Optymalizacja wydajności tabel

W tym rozdziale zostały szczegółowo opisane funkcjonalności bazy danych, które wpływają na wydajność zapisywania i odczytywania danych z tabel. Wydajność tabel jest częściowo zdeterminowana przez charakterystykę bazy danych jeszcze przed ich utworzeniem. Na przykład cechy fizycznej pamięci masowej, zaimplementowane podczas tworzenia bazy danych, i związanych z nią przestrzeni tabel wpływają później na wydajność tabel. Podobnie wpływ na ich wydajność ma Twój wybór początkowych cech fizycznych bazy, takich jak typy tabel i typy danych. Dlatego implementacja praktycznych standardów dotyczących tworzenia baz danych, przestrzeni tabel i samych tabel (pod kątem wydajności) tworzy podstawy optymalizacji dostępu do danych i skalowalności bazy.

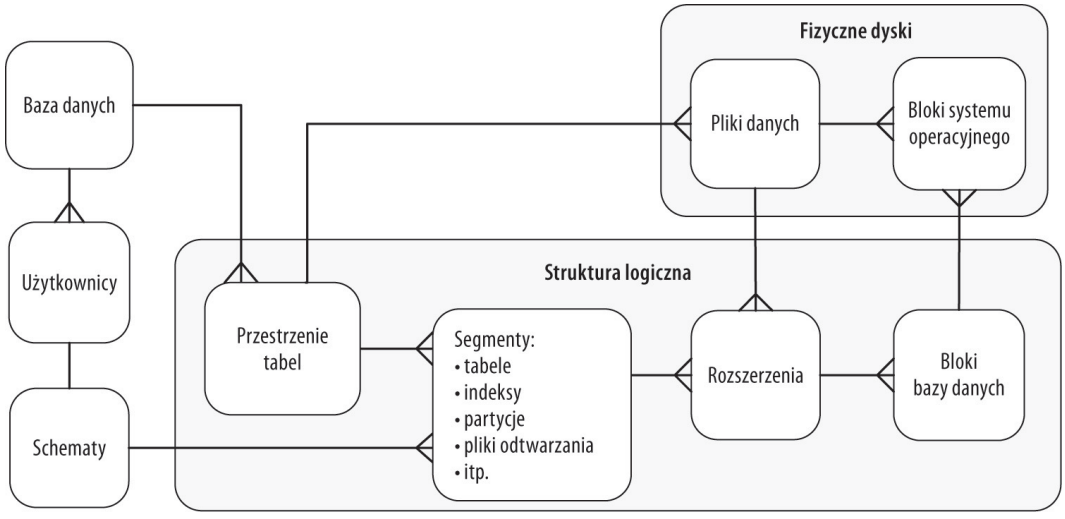
Baza danych Oracle składa się z różnych struktur fizycznych, służących do przechowywania, zabezpieczania i odczytywania danych oraz zarządzania nimi. Istnieje kilka funkcjonalności związanych z wydajnością, które możesz zaimplementować podczas tworzenia bazy danych, na przykład początkowy układ plików i rodzaj zarządzania przestrzeniami tabel. Decyzje architektoniczne podejmowane na tym etapie mają długotrwałe konsekwencje.

Wskazówka. Instancja bazy Oracle jest zdefiniowana jako struktura danych w pamięci i procesy wykonywane w tle systemu operacyjnego. Natomiast baza danych Oracle składa się z fizycznych plików, czyli plików danych, plików sterujących i plików dziennika powtórzeń.

Jak pokazuje rysunek 1.1, przestrzeń tabel jest strukturą logiczną umożliwiającą zarządzanie grupą plików danych. Pliki danych są to fizyczne pliki zapisane na dysku. Podczas konfiguracji przestrzeni tabel należy pamiętać o jej kilku cechach, które mogą skutkować długotrwałymi konsekwencjami. Przestrzenie tabel dzielą się na zarządzane lokalnie i przestrzenie z automatycznie zarządzanymi segmentami (ang. *automatic segment storage managed*, ASSM). Jeżeli zaimplementujesz te cechy w przemyślany sposób, będziesz w stanie osiągnąć w przyszłości wysoką wydajność tabel.

Tabela jest to obiekt przechowujący dane. Miernikiem wydajności bazy danych jest szybkość, z jaką aplikacja może zapisywać, aktualizować, usuwać i odczytywać dane. Dlatego dobrze będzie zacząć tę książkę od przedstawienia porad zawierających rozwiązania problemów z wydajnością tabel.

Zaczniemy od opisania aspektów tworzenia baz danych i przestrzeni tabel, mających wpływ na wydajność tabel. Następnie omówimy takie zagadnienia jak wybór typów tabel i danych spełniających wymogi wydajnościowe stawiane przez biznes. Kolejne tematy obejmują zarządzanie fizyczną implementacją przestrzeni tabel. Szczegółowo omówione są takie zagadnienia jak wykrywanie fragmentacji tabel, zarządzanie wolnym miejscem poniżej wskaźnika zajętości, migracja i łańcuchowanie wierszy oraz kompresja danych. Opisana jest również aplikacja Segment Advisor. To przydatne narzędzie pomoże Ci zautomatyzować wykrywanie i rozwiązywanie problemów z fragmentacją tabel i niewykorzystanym miejscem.



Rysunek 1.1. *Zależności pomiędzy logiczną i fizyczną strukturą bazy danych*

1.1. Tworzenie maksymalnie wydajnej bazy danych

Problem

Podczas tworzenia bazy danych zauważyłeś, że niektóre funkcjonalności (jeżeli zostały włączone) powodują długotrwałe konsekwencje mające wpływ na wydajność i dostępność tabel. Podczas tworzenia bazy danych powinieneś w szczególności:

- Wymusić, aby wszystkie przestrzenie tabel tworzone w bazie były zarządzane lokalnie. Tego typu przestrzenie charakteryzują się większą wydajnością niż przestrzenie wykorzystujące przestarzałą technologię zarządzania słownikowego.
- Zagwarantować, że użytkownicy będą automatycznie przypisywani do domyślnej trwałej przestrzeni tabel. Dzięki temu tworzeni użytkownicy będą przypisywani do domyślnej przestrzeni tabel innej niż `SYSTEM`. Gdy użyta jest funkcjonalność odroczonego tworzenia segmentów (omówiona później), a użytkownik posiada uprawnienia `CREATE TABLE`, wówczas będzie mógł on tworzyć obiekty w przestrzeni `SYSTEM` nawet bez określonego limitu zajmowanego miejsca. To jest niepożądany efekt. Wprawdzie użytkownik nie będzie mógł zapisywać danych w tabelach, nie mając odpowiedniego limitu, ale będzie mógł tworzyć obiekty i w ten sposób nieodwracalnie zapełniać przestrzeń `SYSTEM`.
- Zagwarantować, że użytkownicy będą automatycznie przypisywani do domyślnej tymczasowej przestrzeni tabel. Dzięki temu tworzeni użytkownicy będą przypisywani do właściwej tymczasowej przestrzeni tabel, jeżeli domyślna przestrzeń nie będzie jawnie określona.

Rozwiązanie

Dostępne są dwa narzędzia, których możesz użyć do utworzenia bazy danych:

- SQL*Plus i instrukcja `CREATE DATABASE`.
- Database Configuration Assistant (*dbca*).

Oba narzędzia opisane są w poniższych podrozdziałach.

Narzędzie SQL*Plus

Aby utworzyć bazę danych zgodnie ze sprawdzonymi standardami stanowiącymi podstawę wydajnych baz, użyj poniższego skryptu:

```
CREATE DATABASE O12C
  MAXLOGFILES 16
  MAXLOGMEMBERS 4
  MAXDATAFILES 1024
  MAXINSTANCES 1
  MAXLOGHISTORY 680
  CHARACTER SET AL32UTF8
DATAFILE
  '/u01/dbfile/O12C/system01.dbf'
  SIZE 500M REUSE
  EXTENT MANAGEMENT LOCAL
UNDO TABLESPACE undotbs1 DATAFILE
  '/u02/dbfile/O12C/undotbs01.dbf'
  SIZE 800M
SYSAUX DATAFILE
  '/u01/dbfile/O12C/sysaux01.dbf'
  SIZE 500M
DEFAULT TEMPORARY TABLESPACE TEMP TEMPFILE
  '/u02/dbfile/O12C/temp01.dbf'
  SIZE 500M
DEFAULT TABLESPACE USERS DATAFILE
  '/u01/dbfile/O12C/users01.dbf'
  SIZE 50M
LOGFILE
  GROUP 1
    ('/u01/oraredo/O12C/redo01a.rdo',
     '/u02/oraredo/O12C/redo01b.rdo') SIZE 200M,
  GROUP 2
    ('/u01/oraredo/O12C/redo02a.rdo',
     '/u02/oraredo/O12C/redo02b.rdo') SIZE 200M,
  GROUP 3
    ('/u01/oraredo/O12C/redo03a.rdo',
     '/u02/oraredo/O12C/redo03b.rdo') SIZE 200M
USER sys IDENTIFIED BY f0obar
USER system IDENTIFIED BY f0obar;
```

Powyższy skrypt `CREATE DATABASE` umożliwia utworzenie solidnej podstawy dla wydajnej bazy danych, ponieważ uaktywnia następujące funkcjonalności:

- Za pomocą klauzuli `EXTENT MANAGEMENT LOCAL` definiuje lokalnie zarządzaną przestrzeń tabel `SYSTEM`. Ten sposób gwarantuje, że wszystkie tworzone przestrzenie tabel będą zarządzane lokalnie. Począwszy od wersji Oracle Database 12c, przestrzeń `SYSTEM` jest zawsze tworzona jako zarządzana lokalnie.
- Definiuje domyślną przestrzeń tabel `USERS` dla wszystkich użytkowników tworzonych bez jawnie wskazanej domyślnej przestrzeni tabel. W ten sposób zapobiega się domyślnemu przypisywaniu użytkowników do przestrzeni `SYSTEM`.
- Dla wszystkich użytkowników definiuje domyślną tymczasową przestrzeń tabel o nazwie `TEMP`. Dzięki temu zapobiega się przypisywaniu użytkowników do przestrzeni `SYSTEM` jako domyślnej tymczasowej przestrzeni. Użytkownicy utworzeni w domyślnej tymczasowej przestrzeni `SYSTEM` mogą mieć niekorzystny wpływ na wydajność bazy, ponieważ mogą rywalizować o zasoby w tej przestrzeni.

Droga do wysokiej wydajności zaczyna się od poprawnie skonfigurowanej bazy danych. Powyższe zalecenia pomogą Ci stworzyć niezawodną infrastrukturę dla Twoich tabel.

Narzędzie dbca

Narzędzie *dbca* posiada interfejs graficzny i z wierszem poleceń, za pomocą których możesz konfigurować i tworzyć bazy danych. W trybie graficznym narzędzie jest proste w użyciu i ma bardzo intuicyjny interfejs użytkownika. Aby użyć go w trybie graficznym w środowisku Linux/Unix, musisz zainstalować odpowiednie oprogramowanie X, a następnie wprowadzić polecenie `xhost +` i odpowiednio ustawić zmienną `DISPLAY`, na przykład:

```
$ xhost +
$ echo $DISPLAY
:0.0

$ xhost +
$ echo $DISPLAY
:0.0
```

Narzędzie *dbca* jest otwierane w systemie operacyjnym w następujący sposób:

```
$ dbca
```

Pojawi się seria okien, które umożliwią Ci wybranie sposobu konfiguracji. Możesz wybrać opcję *Advanced Mode* (tryb zaawansowany), która da Ci większą kontrolę nad takimi parametrami jak umiejscowienie plików i multipleksacja logów powtórzeń.

Domyślnie *dbca* tworzy bazy danych o następujących cechach:

- Definiuje przestrzeń tabel `SYSTEM` zarządzaną lokalnie.
- Definiuje domyślną przestrzeń tabel o nazwie `USERS` dla wszystkich użytkowników tworzonych bez jawnie określonej domyślnej przestrzeni tabel.
- Definiuje domyślną tymczasową przestrzeń tabel `TEMP` dla wszystkich użytkowników.

W narzędziu *dbca*, podobnie jak SQL*Plus, dostępne są wszystkie niezbędne opcje stanowiące solidną podstawę do tworzenia aplikacji.

Narzędzie *dbca* umożliwia również tworzenie baz danych w trybie cichym, tj. bez interfejsu graficznego. Użycie tego trybu wraz z plikiem odpowiedzi jest skutecznym, spójnym i powtarzalnym sposobem tworzenia baz danych. Ta metoda sprawdza się również podczas instalacji baz na zdalnych serwerach, podłączonych wolnymi łączami lub bez zainstalowanego odpowiedniego oprogramowania X.

Narzędzie *dbca* możesz uruchomić w trybie cichym razem z plikiem odpowiedzi. W niektórych sytuacjach użycie trybu graficznego nie jest możliwe, na przykład z powodu wolnego łącza lub braku oprogramowania X. Aby utworzyć bazę danych za pomocą *dbca* w trybie cichym, wykonaj poniższe kroki:

1. Zlokalizuj plik *dbca.rsp*.
2. Wykonaj kopię pliku *dbca.rsp*.
3. Zmodyfikuj kopię pliku *dbca.rsp* odpowiednio do Twojego środowiska.
4. Otwórz narzędzie *dbca* w trybie cichym.

Najpierw przejdź do katalogu, w którym umieściłeś oprogramowanie instalacyjne bazy Oracle, a następnie użyj polecenia `find`, aby zlokalizować plik *dbca.rsp*:

```
$ find . -name dbca.rsp
./12.1.0.1/database/response/dbca.rsp
```

Skopiuj ten plik, aby nie modyfikować oryginału (dzięki temu zawsze będziesz miał poprawny, oryginalny plik).

```
$ cp dbca.rsp mydb.rsp
```

Teraz zmień plik *mydb.rsp*. Musisz w nim zmodyfikować przynajmniej następujące parametry: `GDBNAME`, `SID`, `SYSPASSWORD`, `SYSTEMPASSWORD`, `SYSMANPASSWORD`, `DBSNMPPASSWORD`, `DATAFILEDESTINATION`, `STORAGETYPE`, `CHARACTERSET` oraz `NATIONALCHARACTERSET`. Poniżej jest przedstawiony przykład zmodyfikowanych parametrów w pliku *mydb.rsp*:

```
[CREATEDATABASE]
GDBNAME = "O12C"
SID = "O12C"
TEMPLATENAME = "General_Purpose.dbc"
SYSPASSWORD = "fOObar"
SYSTEMPASSWORD = "fOObar"
SYSMANPASSWORD = "fOObar"
DBSNMPPASSWORD = "fOObar"
DATAFILEDESTINATION ="/u01/dbfile"
STORAGETYPE="FS"
CHARACTERSET = "AL32UTF8"
NATIONALCHARACTERSET= "UTF8"
```

Następnie uruchom narzędzie *dbca* w trybie cichym razem z plikiem odpowiedzi:

```
$ dbca -silent -responseFile /home/oracle/orainst/mydb.rsp
```

Na ekranie pojawią się poniższe informacje:

```
Copying database files                          // kopiowanie bazy danych
1% complete
...
Creating and starting Oracle instance           // tworzenie i uruchamianie instancji Oracle
...
62% complete
Completing Database Creation                    // zakończenie tworzenia bazy danych
...
100% complete
Look at the log file ... for further details.   // sprawdź szczegóły w logu ...
```

Jeżeli zajrzysz do plików logów, zauważysz, że narzędzie *dbca* korzysta z narzędzia *rman* do odtwarzania plików danych użytych w bazie. Następnie tworzy instancję bazy i wykonuje czynności poinstalacyjne. Na serwerze z systemem Linux w pliku */etc/oratab* powinien znaleźć się zapis dotyczący Twojej nowej bazy danych.

Wielu administratorów uruchamia narzędzie *dbca* i konfiguruje bazę danych w trybie graficznym, ale tylko nieliczni korzystają z opcji dostępnych w pliku odpowiedzi. Efektywnie używając pliku odpowiedzi, możesz w pełni zautomatyzować proces tworzenia baz danych. Możesz zmodyfikować plik tak, aby tworzyć bazy ASM, a nawet RAC. Ponadto za pomocą pliku odpowiedzi możesz kontrolować niemal każdy aspekt instalacji, podobnie jak w trybie graficznym.

■ **Wskazówka**. Wszystkie opcje narzędzia *dbca* możesz wyświetlić, używając parametru pomocy: `dbca –help`.

Jak to działa

Poprawne utworzenie i skonfigurowanie bazy danych gwarantuje, że będzie ona działać prawidłowo. Faktem jest, że po utworzeniu bazy możesz zmieniać jej cechy. Jednak niepoprawny skrypt `CREATE DATABASE` będzie powodował nieustanne problemy z wydajnością. W środowisku produkcyjnym czasami trudno jest wyłączyć bazę w celu skorygowania jej niepoprawnej konfiguracji. W miarę możliwości na każdym etapie tworzenia środowiska, począwszy od tworzenia bazy danych, miej na uwadze wydajność.

Podczas tworzenia bazy danych rozważ również użycie funkcjonalności wpływających na jej obsługę. Trwała baza danych ma długi okres działania, który jest częścią ogólnego równania wydajnościowego. Instrukcja `CREATE DATABASE` podana w części „Rozwiązanie” uwzględnia również następujące zagadnienia dotyczące trwałości bazy danych:

- Tworzy automatyczną przestrzeń tabel `UNDO` (automatyczne zarządzanie odwoływaniem transakcji włączone za pomocą parametrów inicjalizacyjnych `UNDO_MANAGEMENT` oraz `UNDO_TABLESPACE`). Dzięki temu baza Oracle może automatycznie zarządzać segmentami odwoływania transakcji i zwalnia Cię z ich ciągłego kontrolowania i korygowania.

- Umieszcza pliki danych w katalogach zgodnych ze standardami danego środowiska. Dzięki temu ułatwiona jest ich obsługa i zarządzanie, co skutkuje większą dostępnością bazy w dłuższym zakresie czasu, jak również większą wydajnością.
- Ustawia hasła użytkowników-administratorów na wartości inne niż domyślne. Dzięki temu baza jest bezpieczniejsza, co w dłuższym zakresie czasu również ma wpływ na jej wydajność (np. jeżeli ktoś niedoświadczony będzie manipulował przy bazie i usunie dane, będzie to miało negatywny wpływ na wydajność bazy).
- Określa trzy grupy logów powtórzeń po dwóch członków w każdym, o wielkości odpowiadającej ilości transakcji. Wielkość logów powtórzeń ma bezpośredni wpływ na częstotliwość ich przełączania. Jeżeli logi przełączane są zbyt często, wówczas spada wydajność. Pamiętaj, że tworząc bazę danych, możesz nie znać wielkości logów, dlatego będziesz musiał je później skorygować.

Poświęć chwilę czasu na sprawdzenie, czy każda tworzona przez Ciebie baza spełnia ogólnie przyjęte standardy, które pozwolą utworzyć solidny fundament wysokiej wydajności.

Jeżeli przejąłeś od kogoś administrowanie bazą danych i chcesz zweryfikować trwałe ustawienia przestrzeni tabel, użyj na przykład poniższego zapytania:

```
SELECT *
FROM database_properties
WHERE property_name = 'DEFAULT_PERMANENT_TABLESPACE';
```

Jeżeli chcesz zmodyfikować domyślną trwałą przestrzeń tabel, wykonaj zapytanie:

```
SQL> alter database default tablespace users;
```

Aby sprawdzić ustawienia domyślnej trwałej przestrzeni tabel, użyj zapytania:

```
SELECT *
FROM database_properties
WHERE property_name = 'DEFAULT_TEMP_TABLESPACE';
```

W celu zmiany ustawień tymczasowej przestrzeni tabel wykonaj zapytanie:

```
SQL> alter database default temporary tablespace temp;
```

Za pomocą poniższego zapytania możesz zweryfikować ustawienia przestrzeni `UNDO`:

```
SELECT name, value
FROM v$parameter
WHERE name IN ('undo_management','undo_tablespace');
```

Jeżeli chcesz zmienić przestrzeń `UNDO`, utwórz najpierw nową przestrzeń `UNDO`, a następnie wykonaj instrukcję `ALTER SYSTEM SET UNDO_TABLESPACE`.

1.2. Tworzenie przestrzeni tabel i maksymalizacja wydajności

Problem

Jak Ci wiadomo, przestrzenie nazw są logicznymi kontenerami obiektów bazy danych, takich jak tabele i indeksy. Ponadto wiesz już, że jeżeli podczas tworzenia obiektów nie określisz atrybutów dotyczących zajmowanego miejsca na dysku, wówczas tabele i indeksy automatycznie odziedziczą odpowiednie atrybuty po przestrzeni tabel (w której jest tworzona tabela lub indeks). Dlatego przestrzenie tabel musisz tworzyć w taki sposób, aby zapewnić maksymalną wydajność tabel i efektywność zarządzania nimi.

Rozwiązanie

Radzimy Ci tworzyć przestrzenie tabel zarządzanych lokalnie z włączoną funkcjonalnością automatycznego zarządzania przestrzenią segmentu (ang. *automatic segment space management*, ASSM). Od wersji Oracle 12c jest to domyślna metoda:

```
create tablespace tools
datafile '/u01/dbfile/O12C/tools01.dbf' size 100m;
```

Za pomocą poniższego zapytania możesz sprawdzić, czy utworzona przestrzeń jest zarządzana lokalnie i wykorzystuje funkcjonalność ASSM:

```
select tablespace_name, extent_management, segment_space_management
from dba_tablespaces
where tablespace_name='TOOLS';
```

Poniżej przedstawiony jest przykładowy wynik:

```
TABLESPACE_NAME  EXTENT_MANAGEMENT       SEGMENT_SPACE_MANAGEMENT
---------------- ---------------------- ------------------------
TOOLS            LOCAL                  AUTO
```

Jak to działa

Dla jasności: ta porada dotyczy dwóch osobnych zalecanych cech przestrzeni tabel:

- lokalnego zarządzania,
- automatycznego zarządzania przestrzenią segmentu (ASSM).

Począwszy od wersji Oracle 12c, wszystkie przestrzenie tabel są tworzone jako zarządzane lokalnie. W poprzednich wersjach mogłeś wybrać, czy przestrzeń ma być zarządzana lokalnie, czy słownikowo. Od tej chwili wszystkie tworzone przez Ciebie przestrzenie powinny być zarządzane lokalnie.

Funkcjonalność zarządzania przestrzenią segmentu może być ustawiona na `AUTO` (domyślne ustawienie) lub `MANUAL`. Oracle bardzo zaleca stosowanie ustawienia `AUTO` (czyli ASSM). W ten sposób baza może automatycznie zarządzać cechami wielu fizycznych przestrzeni, które wcześniej administrator musiał dopasowywać ręcznie. W większości przypadków przestrzeń z zarządzaniem ASSM bardziej efektywnie przetwarza transakcje niż przestrzeń z ustawieniem `MANUAL`. Jest kilka wyjątków, których ta zasada nie dotyczy. Zalecamy stosowanie zarządzania ASSM, o ile testy nie potwierdzą, że ustawienie `MMANUAL` jest lepsze.

Uwaga. Nie możesz utworzyć przestrzeni `SYSTEM` z zarządzaniem ASSM. Ponadto ASSM dotyczy tylko trwałych, lokalnie zarządzanych przestrzeni.

Jeżeli podczas tworzenia przestrzeni tabel nie określisz stałej wielkości rozszerzenia, baza automatycznie będzie ją rozszerzać o 64 kB, 1 MB, 8 MB lub 64 MB. Jeżeli obiekty w przestrzeni będą różnej wielkości, użyj funkcjonalności automatycznego rozszerzania. Za pomocą klauzuli `EXTENT MANAGEMENT LOCAL AUTOALLOCATE` możesz jawnie skonfigurować bazę, aby automatycznie określała wielkość rozszerzenia.

Za pomocą klauzuli `UNIFORM SIZE` możesz skonfigurować rozszerzanie przestrzeni za każdym razem o tę samą wielkość. Poniższy przykład ustala stałą wielkość rozszerzenia na 128 kB:

```
create tablespace tools
  datafile '/u01/dbfile/O12C/tools01.dbf' size 100m
  extent management local uniform size 128k;
```

Jeżeli masz ważne powody, aby ustawić stałą wielkość rozszerzenia, zdecydowanie tak zrób. Jednak jeżeli nie jesteś w stanie oszacować wielkości, stosuj domyślne ustawienie `AUTOALLOCATE`.

Możesz również określić, czy plik danych ma być po zapełnieniu automatycznie powiększany. Konfiguruje się to za pomocą klauzuli `AUTOEXTEND ON`. Jeżeli używasz tej funkcjonalności, zalecamy, abyś ustawił absolutną, maksymalną wielkość pliku danych. W ten sposób zapobiegniesz przypadkowemu wyczerpaniu całego miejsca

na dysku przez bardzo długie lub błędne zapytania SQL (wyobraź sobie, co mogłoby się stać z usługą w chmurze, która automatycznie dodawałaby przestrzeń dyskową na każde żądanie bazy). Poniżej jest przedstawiony przykład:

```
create tablespace tools
  datafile '/u01/dbfile/O12C/tools01.dbf' size 100m
  autoextend on maxsize 10G;
```

1.3. Dobór typów tabel do wymagań biznesowych

Problem

Jesteś początkującym administratorem bazy Oracle i przeczytałeś o różnych dostępnych typach tabel, na przykład stertowych, indeksowanych itp. Chcesz utworzyć aplikację opartą na bazie danych i musisz zadecydować, jakiego typu tabel użyjesz.

Rozwiązanie

Baza Oracle oferuje szeroki wybór typów tabel. Domyślnym typem jest tabela stertowa. W przypadku większości aplikacji tabela stertowa stanowi strukturę do efektywnego zapisywania i odczytywania danych. Są jednak inne typy, o których musisz pamiętać, i musisz znać przypadki, w których powinny być stosowane tabele tych typów. Tabela 1.1 zawiera opisy wszystkich typów tabel i ich odpowiednie zastosowanie.

Tabela 1.1. *Typy tabel Oracle i ich typowe zastosowanie*

Typ tabeli/cecha	Opis	Korzyść/zastosowanie
Stertowa	Domyślny i najczęściej stosowany typ tabeli.	Stosuj ten typ, jeżeli nie masz ważnego powodu, aby zastosować inny.
Tymczasowa	Prywatne dane sesji, zapisywane na czas trwania sesji lub transakcji. Przestrzeń jest przydzielana w tymczasowych segmentach.	Aplikacja wymaga struktury tymczasowej tabeli w celu zapisania lub modyfikacji danych. Po zakończeniu sesji dane nie są już potrzebne.
Indeksowana	Dane zapisane w indeksowanej strukturze B-drzewa, posortowanej według klucza podstawowego.	Tabela jest odpytywana głównie według kolumny z kluczem podstawowym. Dobrze sprawdza się w szerokim przeszukiwaniu, zapewnia szybki i swobodny dostęp do danych.
Partycjonowana	Tabela logiczna składająca się z osobnych fizycznych segmentów.	Typ stosowany w przypadku dużych tabel zawierających dziesiątki milionów wierszy. Silnie wpływa na skalowalność wydajności dużych tabel i indeksów.
Zewnętrzna	Tabela wykorzystująca dane w plikach systemu operacyjnego, zapisanych poza bazą danych.	Ten typ umożliwia efektywny dostęp do danych w plikach zapisanych poza bazą danych (np. CSV lub tekstowych). Tabele zewnętrzne oferują również skuteczny mechanizm przenoszenia danych pomiędzy bazami.
Widok zmaterializowany	Tabela zawierająca wynik zapytania SQL. Jest okresowo odświeżana, gdy trzeba zaktualizować dane bieżącymi wynikami zapytania SQL.	Agregowanie danych w celu ich szybszego zwrócenia lub zreplikowania i zmniejszenia obciążenia bazy.

Tabela 1.1. *Typy tabel Oracle i ich typowe zastosowanie — ciąg dalszy*

Typ tabeli/cecha	Opis	Korzyść/zastosowanie
Klastrowa	Grupa tabel mających wspólne bloki danych.	Typ stosowany w celu zmniejszenia ilości operacji wejścia-wyjścia wykonywanych często na tej samej kolumnie. Rzadko stosowany.
Zagnieżdżona	Tabela z kolumną danych zawierającą inną tabelę.	Typ rzadko stosowany.
Obiektowa	Tabela z kolumną danych typu obiektowego.	Typ rzadko stosowany.

Jak to działa

W większości przypadków tabele stertowe spełnią Twoje wymagania. Ten typ tabeli jest strukturą sprawdzoną w wielu środowiskach bazodanowych. Jeżeli poprawnie zaprojektujesz bazę (której struktura będzie zgodna ze standardami) i zastosujesz w niej odpowiednie indeksy i ograniczenia, otrzymasz sprawnie działający i efektywnie zarządzany system.

Większość Twoich tabel będzie typu stertowego. Jeżeli jednak zechcesz wykorzystać cechy innych typów (i jesteś pewien ich zalet), zdecydowanie tak zrób. Na przykład partycjonowanie jest skalowalnym sposobem budowania bardzo dużych tabel i indeksów. Zmaterializowane widoki stanowią skuteczną metodę agregowania i replikowania danych. Tabele indeksowane to wydajne struktury stosowane w sytuacjach, gdy ich kolumny są częściami klucza podstawowego (na przykład tabele będące wynikiem użycia relacji wiele-do-wielu) i tak dalej.

> **Ostrzeżenie.** Nie powinieneś wybierać określonego typu tabeli tylko dlatego, że wydaje Ci się on ciekawą cechą, o której ostatnio słyszałeś. Niekiedy administratorzy, gdy dowiedzą się o jakiejś funkcjonalności, implementują ją, nie wiedząc, jaki będzie jej wpływ na wydajność i koszty zarządzania bazą. Zawsze powinieneś najpierw sprawdzić i upewnić się, że dana funkcjonalność przyniesie wymierne korzyści wydajnościowe.

1.4. Dobór cech tabel zwiększających wydajność

Problem

Podczas tworzenia tabel musisz zaimplementować odpowiednie funkcjonalności, które zmaksymalizują ich wydajność, skalowalność i skuteczność zarządzania nimi.

Rozwiązanie

Jest wiele zagadnień związanych z wydajnością i trwałością tabel, które powinieneś rozważyć podczas ich tworzenia. Tabela 1.2 opisuje aspekty dotyczące wydajności tabel.

Jak to działa

W części „Rozwiązanie” zostały opisane aspekty związane z wydajnością tabel. Podczas tworzenia tabel powinieneś również wziąć pod uwagę funkcjonalności poprawiające skalowalność i dostępność bazy danych. Administratorzy i programiści często nie traktują ich jako metod zwiększających wydajność bazy. Jednak tworzenie stabilnej i skutecznie zarządzanej bazy danych idzie w parze z wysoką wydajnością. Tabela 1.3 opisuje dobre praktyki ułatwiające zarządzanie tabelami.

Tabela1.2. *Cechy tabel wpływające na ich wydajność*

Zalecenie	Uzasadnienie
Jeżeli tabela na początku posiada wiersze zawierające wartości `null`, które później będą zamieniane na większe wartości, rozważ ustawienie atrybutu `PCTFREE` na wartość większą niż domyślne 10%. Jeżeli nie będzie żadnych aktualizacji danych, nadaj temu parametrowi mniejszą wartość.	Ponieważ baza Oracle zapisuje rekordy danych w blokach, więc atrybut `PCTFREE` określa, jaka część bloku ma być zarezerwowana (wolna) na przyszłe aktualizacje danych. Właściwie ustawiony atrybut pozwala zapobiec migracji/łańcuchowaniu wierszy, co w przypadku dużego odsetka wierszy poddawanych tej operacji wpływa na wydajność operacji wejścia-wyjścia.
Wszystkie tabele powinny być tworzone z kluczem podstawowym (z wyjątkiem tabel zawierających np. logi).	Spełnione wymagania biznesowe i możliwa jednoznaczna identyfikacja każdego wiersza. Indeks jest tworzony dla kolumny z kluczem podstawowym i umożliwia wydajny dostęp do podstawowych wartości w tabeli.
Jeżeli rzeczywisty klucz podstawowy dotyczy kolumny zawierającej długie ciągi znaków lub obejmuje wiele kolumn, rozważ utworzenie numerycznego klucza zastępczego, będącego kluczem podstawowym.	Łączenie tabel jest łatwiejsze (w przypadku łączenia tylko jednej kolumny), a jeden klucz numeryczny pozwala tworzyć szybsze łączenia tabel niż klucz oparty na kolumnach znakowych lub zestawie kolumn.
Rozważ użycie automatycznej inkrementacji wartości do wypełniania kolumn z kluczem podstawowym.	Brak konieczności ręcznego tworzenia kodu, wyzwalaczy i sekwencji wypełniających kolumny z kluczami podstawowymi lub obcymi. Jednak jednym z potencjalnym problemów może być kolizja jednoczesnych zapisów do tabeli.
Utwórz unikatowy klucz dla logicznego klucza biznesowego, tj. dla określonej kombinacji kolumn tworzących unikatowy identyfikator wiersza.	Spełnione wymagania biznesowe i bardziej czytelne dane. Wydajne pobieranie danych z kolumn tworzących klucz logiczny i często wykorzystywanych w klauzuli `WHERE`. Jeżeli klucz podstawowy jest kluczem zastępczym, wówczas zazwyczaj jest przynajmniej jeden unikatowy klucz identyfikujący logiczny klucz biznesowy.
Zdefiniuj klucze obce tam, gdzie jest to uzasadnione.	Spełnione wymagania biznesowe i bardziej czytelne dane. Umożliwienie optymalizatorowi wyboru wydajnej ścieżki dostępu do danych.
Rozważ utworzenie indeksów na kolumnach z obcymi kluczami.	Przyspieszenie wykonywania zapytań, które często łączą kolumny z kluczami obcymi lub podstawowymi. Uniknięcie niektórych problemów związanych z blokowaniem danych.
Rozważ zastosowanie specjalnych funkcjonalności, takich jak kolumny wirtualne lub niewidzialne (12c), dane tylko do odczytu, równoległe polecenia, kompresja, wyłączenie logowania itp.	Funkcjonalności takie jak równoległe polecenia DML, kompresja lub wyłączenie logowania mogą wpływać na wydajność odczytu i zapisu danych.

Tabela 1.3. *Cechy tabel wpływające na ich skalowalność i skuteczność zarządzania*

Zalecenie	Uzasadnienie
Podczas określania nazw tabel, kolumn, widoków, ograniczeń, wyzwalaczy, indeksów itp. stosuj określone standardy.	Łatwiejsze dokumentowanie aplikacji i zarządzanie.
Stosuj osobne przestrzenie tabel dla osobnych schematów.	Dodatkowa elastyczność spełniająca wymagania dotyczące tworzenia kopii zapasowych i dostępności różnych danych.
Umożliwiaj tabelom i indeksom dziedziczenie po przestrzeniach tabel atrybutów pamięci dyskowej (szczególnie gdy używasz przestrzeni z zarządzaniem ASSM).	Łatwiejsza administracja i obsługa bazy.
Twórz ograniczenia podstawowego klucza jako ograniczenie tabeli.	Większa elastyczność podczas tworzenia klucza podstawowego, szczególnie w sytuacji, gdy klucz obejmuje kilka kolumn.
Stosuj ograniczenia sprawdzające tam, gdzie jest to uzasadnione.	Spełnione wymagania biznesowe i bardziej czytelne dane. Stosuj tę funkcjonalność, aby narzucić krótką i statyczną listę wartości kolumn.
Jeżeli kolumna musi zawsze zawierać wartość, określ ograniczenie `NOT NULL`.	Spełnione wymagania biznesowe i bardziej czytelne dane.
Twórz komentarze do tabel i kolumn.	Łatwiejsze dokumentowanie i utrzymywanie aplikacji.
Jeżeli używasz obiektów LOB w wersji 11g lub wyższej, stosuj nową architekturę *SecureFiles*.	*SecureFiles* jest zalecaną architekturą dla obiektów LOB. Umożliwia stosowanie takich funkcjonalności jak kompresja, szyfrowanie i deduplikacja danych.

1.5. Właściwy dobór typów danych

Problem

Podczas tworzenia tabel musisz zaimplementować odpowiednie typy danych, umożliwiające osiągnięcie maksymalnej wydajności, skalowalności i łatwości obsługi bazy.

Rozwiązanie

Istnieje wiele zagadnień dotyczących wydajności i trwałości bazy danych, które powinieneś rozważyć podczas dobierania typów danych w tabelach. Tabela 1.4 opisuje cechy typów wpływających na wydajność.

■ **Uwaga.** W wersjach bazy Oracle wcześniejszych niż 12c maksymalna długość typów `VARCHAR2` oraz `NVARCHAR2` była równa 4000, natomiast typów `RAW` — 2000. Począwszy od wersji 12c, oba typy zostały rozszerzone do długości równej 32767.

Jak to działa

Podczas tworzenia tabeli musisz określić nazwy kolumn i odpowiednie typy danych. Jako programista lub administrator musisz znać właściwe zastosowanie każdego typu. Spotkaliśmy się z wieloma problemami (dotyczącymi wydajności i dokładności przetwarzania danych) spowodowanymi doborem niewłaściwych

Tabela 1.4. *Cechy typów danych wpływających na wydajność*

Zalecenie	Uzasadnienie
Jeżeli kolumna zawsze będzie zawierać dane liczbowe i może być wykorzystywana do obliczeń, wybierz liczbowy typ danych. Pamiętaj, że niektórych kolumn nie musisz definiować jako liczbowe tylko dlatego, że zawierają cyfry (np. kody pocztowe lub numery PESEL).	Spełnione wymagania biznesowe i największa elastyczność, wydajność i spójność wyników w zapytaniach SQL wykorzystujących funkcje matematyczne (które mogą inaczej działać w przypadku znaku o kodzie „01” i liczby 1). Odpowiednie typy danych zapobiegają ich niepotrzebnej konwersji.
Jeżeli zasada biznesowa określa długość i precyzję pola liczbowego, wówczas użyj tych parametrów, np. `NUMBER(7,2)`. Jeżeli nie ma narzuconych zasad, użyj typu `NUMBER`.	Spełnione wymagania biznesowe i bardziej czytelne dane. Liczby o zdefiniowanej precyzji nie będą przechowywały niepotrzebnych cyfr dziesiętnych. Może to mieć wpływ na długość wiersza i w efekcie wydajność operacji wejścia-wyjścia.
W przypadku większości danych tekstowych używaj typu `VARCHAR2` (nie `CHAR`).	Typ danych `VARCHAR2` oferuje większą elastyczność i efektywniej wykorzystuje przestrzeń niż typ `CHAR`. Dlatego możesz używać typu `CHAR` w przypadku danych o stałej długości, na przykład kodów ISO nazw państw.
Jeżeli zasada biznesowa określa maksymalną długość danych w kolumnie, wówczas zastosuj tę długość i nie określaj typu `VARCHAR2(4000)` dla wszystkich kolumn.	Spełnione wymagania biznesowe i bardziej czytelne dane.
Stosuj odpowiednie typy danych zawierających datę i czas, tj. `DATE`, `TIMESTAMP` oraz `INTERVAL`.	Spełnione wymagania biznesowe, zapewnienie odpowiedniego formatu danych i uzyskanie największej elastyczności i wydajności zapytań SQL wykorzystujących funkcje wykonujące operacje na datach.
W miarę możliwości unikaj typu `LOB` (ang. *large object*, duży obiekt).	Uniknięcie problemów utrzymaniowych związanych z kolumnami typu `LOB`, takich jak nieoczekiwany rozrost bazy, niewydajne kopiowanie itp.

typów danych. Na przykład jeżeli zamiast typu daty jest użyty typ znakowy, wówczas przeprowadzana jest niepotrzebna konwersja danych, a próba wykonania obliczeń i raportowania danych przyprawia o ból głowy. Co gorsza, po zaimplementowaniu w środowisku produkcyjnym niewłaściwego typu danych jego modyfikacja może być bardzo trudna, ponieważ wprowadzane zmiany mogą spowodować ryzyko zakłócenia działania istniejącego kodu. Gdy popełnisz błąd, będzie Ci niezwykle trudno wycofać się z niego i wybrać właściwy kierunek. Prawdopodobnie lepiej wtedy będzie zaniechać ciągłego manipulowania w bazie i zmuszania błędnie wybranych typów danych do wykonywania zadań, do których nigdy nie były przeznaczone.

Dlatego baza Oracle obsługuje następujące grupy typów danych:

- znakowe,
- liczbowe,
- daty/czasu,
- `RAW`,
- `ROWID`,
- `LOB`.

■ **Wskazówka**. Typy `LONG` i `LONG RAW` są przestarzałe i nie powinny być stosowane.

Typ znakowy

W bazie Oracle są dostępne cztery typy znakowe: VARCHAR2, CHAR, NVARCHAR2 i NCHAR. W większości przypadków do przechowywania danych tekstowych powinieneś używać typu VARCHAR2. Ten typ zajmuje ilość miejsca proporcjonalną do ilości znaków w ciągu. Jeżeli w kolumnie typu VARCHAR2(30) zapiszesz tylko jeden znak, wówczas baza przydzieli miejsce tylko na ten jeden znak.

Podczas definiowania typu VARCHAR2 musisz określić długość danych. Można to zrobić na dwa sposoby: używając słów BYTE lub CHAR. Słowo BYTE określa maksymalną długość ciągu w bajtach, natomiast CHAR — maksymalną liczbę znaków. Na przykład aby określić ciąg znaków zajmujący najwyżej 30 bajtów, zdefiniuj go w następujący sposób:

```
VARCHAR2(30 BYTE)
```

Aby określić ciąg znaków składający się z co najwyżej 30 znaków, zdefiniuj go w następujący sposób:

```
VARCHAR2(30 CHAR)
```

Niemal w każdym przypadku bezpieczniej jest określać długość, używając słowa CHAR. Jeżeli używasz zestawu znaków wielobajtowych i określisz długość jako VARCHAR2(30 BYTE), wówczas możesz otrzymać nieoczekiwane efekty, ponieważ niektóre znaki są zapisywane za pomocą więcej niż jednego bajtu. Natomiast jeżeli określisz typ VARCHAR2(30 CHAR), wówczas zawsze będziesz mógł zapisać w ciągu 30 znaków, niezależnie od tego, czy niektóre z nich będą wymagały więcej niż jednego bajtu.

Typy NVARCHAR2 i NCHAR są przydatne wówczas, gdy masz bazę utworzoną wcześniej, przygotowaną do przechowywania danych wykorzystujących zestaw znaków jednobajtowych, w której później będziesz chciał zapisywać dane złożone z zestawu znaków wielobajtowych.

Wskazówka. Baza Oracle oferuje jeszcze inny typ danych o nazwie VARCHAR. Obecnie Oracle definiuje ten typ jako synonim VARCHAR2, jednak bardzo zaleca stosowanie typu VARCHAR2 (a nie VARCHAR), ponieważ — jak podaje dokumentacja — typ VARCHAR może być w przyszłości przeznaczony do innych celów.

Typ liczbowy

Typ liczbowy stosuj do przechowywania danych, które będziesz wykorzystywał w funkcjach matematycznych, na przykład SUM, AVG, MAX i MIN. Nigdy nie przechowuj informacji liczbowej, używając typów znakowych. Jeżeli do przechowywania danych z natury liczbowych użyjesz typu VARCHAR2, wówczas w przyszłości pojawią się błędy w systemie. Ponadto jeżeli będziesz chciał raportować lub wykonywać obliczenia na danych liczbowych, których typ nie będzie określony jako liczbowy, wówczas otrzymasz nieprzewidziane i często błędne wyniki.

Baza Oracle obsługuje następujące typy liczbowe:

- NUMBER,
- BINARY_DOUBLE,
- BINARY_FLOAT.

W większości przypadków do zapisywania wszelkich danych liczbowych używaj typu NUMBER. Składnia typu jest następująca:

```
NUMBER(skala, precyzja)
```

Gdzie *skala* oznacza całkowitą liczbę cyfr, a *precyzja* oznacza liczbę cyfr po przecinku. Na przykład w liczbie zdefiniowanej jako NUMBER(5, 2) możesz zapisywać wartości od –999,99 do +999,99. Składają się one w sumie z pięciu cyfr, w tym z dwóch po przecinku.

Wskazówka. Baza Oracle umożliwia określenie 38 cyfr w typie NUMBER. Jest to ilość wystarczająca niemal dla każdej aplikacji wykorzystującej liczby.

Dla programistów i administratorów może być niejasne tworzenie tabel zawierających kolumny typu INT, INTEGER, REAL, DECIMAL itp. Wszystkie te typy są zaimplementowane jako jeden typ NUMBER. Na przykład kolumna typu INTEGER jest zaimplementowana jako NUMBER(38).

Typy BINARY_DOUBLE i BINARY_FLOAT są wykorzystywane do obliczeń naukowych. Odpowiadają one typom DOUBLE i FLOAT w języku Java. Jeżeli aplikacja nie wykonuje zaawansowanych obliczeń naukowych, dla wszystkich swoich danych liczbowych stosuj typ NUMBER.

■ **Uwaga.** Typ BINARY może powodować błędy zaokrągleń, które nie występują w typie NUMBER. Ponadto jego interpretacja może być różna w zależności od użytego systemu operacyjnego i sprzętu.

Typ daty/czasu

Do zapisywania i raportowania danych określających datę i czas zawsze używaj typów DATE lub TIMESTAMP (a nie VARCHAR2 lub NUMBER). Użycie właściwego typu danych umożliwi bazie Oracle wykonywanie dokładnych operacji na datach, agregację danych i właściwe ich sortowanie podczas raportowania. Jeżeli użyjesz typu VARCHAR2 dla kolumn zawierających informacje o datach, wówczas niemal na pewno pojawią się w przyszłości niespójności danych i konieczność zastosowania zbędnych funkcji konwersji (na przykład TO_DATE lub TO_CHAR).

Typ DATE zawiera komponent daty oraz komponent czasu z dokładnością sekundową. Jeżeli podczas zapisywania danych nie określisz komponentu czasu, wówczas zostanie przyjęta jego domyślna wartość oznaczająca północ (0 godzin, 0 minut i 0 sekund). Jeżeli chcesz zapisywać czas z większą dokładnością niż jednosekundowa, używaj typu TIMESTAMP. W przeciwnym wypadku zawsze używaj typu DATE.

Typ TIMESTAMP zawiera komponent daty i komponent czasu z dokładnością ułamka sekundy. Definiując kolumnę typu TIMESTAMP, możesz określić komponent czasu będący ułamkiem sekundy. Na przykład jeżeli potrzebna jest dokładność pięciu cyfr po przecinku, możesz ją określić w następujący sposób:

```
TIMESTAMP(5)
```

Maksymalna liczba cyfr po przecinku jest równa 9, domyślna 6. Jeżeli określisz dokładność 0, uzyskasz odpowiednik typu DATE.

Typ RAW

Typ RAW umożliwia przechowywanie w kolumnie danych binarnych. Ten typ jest niekiedy wykorzystywany do przechowywania globalnie unikatowych identyfikatorów lub niewielkich ilości zaszyfrowanych danych. Jeżeli potrzebujesz przechowywać duże ilości danych binarnych (ponad 2000 bajtów), użyj typu BLOB.

Narzędzie SQL*Plus podczas odczytywania danych z kolumny typu RAW niejawnie stosuje wbudowaną funkcję RAWTOHEX. Dane są wyświetlane w formacie szesnastkowym za pomocą znaków 0 – 9 i A – F. Podczas zapisywania danych do kolumny typu RAW niejawnie jest stosowana funkcja HEXTORAW. Jest to ważna informacja, ponieważ jeżeli utworzysz kolumnę typu RAW, wówczas optymalizator może pominąć indeks, ponieważ narzędzie SQL*Plus niejawnie stosuje powyższe funkcje tam, gdzie w zapytaniu SQL występuje odwołanie do takiej kolumny. Zwykły indeks może nie być przydatny, natomiast indeks oparty na funkcjach wykorzystujący funkcję RAWTOHEX może znacznie poprawić wydajność zapytania.

Typ ROWID

Często zdarza się, że programiści i administratorzy, słysząc słowo ROWID (ang. *row identifier*, identyfikator wiersza), myślą, że jest to pseudokolumna zawarta w każdej tabeli, zawierająca informacje o fizycznym miejscu zapisu każdego wiersza na dysku. Jest to prawda. Jednak niewielu wie o tym, że Oracle obsługuje typ ROWID, co oznacza, że można tworzyć tabele zawierające kolumnę tego typu.

Jest kilka praktycznych przypadków zastosowania typu ROWID, na przykład kiedy będziesz miał problem z ustaleniem ograniczenia dotyczącego spójności odwołań i będziesz potrzebował określić identyfikator ROWID wierszy, które naruszają to ograniczenie. W takim przypadku możesz utworzyć tabelę z kolumną typu ROWID i zapisać w niej identyfikatory wadliwych wierszy tabeli. Jest to skuteczny sposób identyfikowania i rozwiązywania problemów z niepoprawnymi danymi.

Nigdy nie staraj się używać typu ROWID i związanych z nim identyfikatorów wierszy tabeli do tworzenia klucza podstawowego, ponieważ identyfikator ten może się zmieniać. Na przykład polecenie ALTER TABLE...MOVE może zmienić identyfikator każdego wiersza tabeli. Wartości klucza podstawowego dla wierszy w tabeli nie powinny nigdy się zmieniać. Dlatego zamiast użyć kolumny typu ROWID jako klucza podstawowego, zastosuj do wypełnienia kolumny z kluczem podstawowym nic nie znaczące liczby generowane przez sekwencję (lub w wersji 12c automatycznie zwiększaną wartość kolumny).

Typ LOB

Oracle umożliwia przechowywanie dużej ilości danych w kolumnie typu LOB. Dostępne są następujące typy:

- CLOB,
- NCLOB,
- BLOB,
- BFILE.

Jeżeli masz dane tekstowe, które nie mieszczą się w granicach wyznaczonych przez typ VARCHAR2, wówczas możesz użyć do ich przechowywania typu CLOB. Typ CLOB jest przydatny do przechowywania dużych ilości danych znakowych, takich jak treści artykułów (wpisy na blogach) i logi. Typ NCLOB jest podobny do CLOB, ale umożliwia przechowywanie informacji zapisanej za pomocą zestawu znaków danego języka, używanego w bazie danych.

Typ BLOB służy do przechowywania dużych ilości danych binarnych, które zazwyczaj nie są czytelne dla człowieka. Tego typu dane to zazwyczaj obrazy, pliki dźwiękowe, dokumenty, pliki PDF, arkusze kalkulacyjne i pliki wideo.

Typy CLOB, NCLOB i BLOB zwane są wewnętrznymi typami LOB, ponieważ dane są zapisane wewnątrz bazy Oracle w plikach skojarzonych z bazą.

Typ BFILE zwany jest zewnętrznym typem LOB. Kolumny tego typu zawierają wskaźniki do plików zapisanych w systemie operacyjnym poza bazą danych. Jeżeli przechowywanie dużych ilości danych binarnych w bazie nie jest możliwe, wówczas stosuj typ BFILE. Dane tego typu nie są przetwarzane przez transakcje realizowane w bazie, jak również nie dotyczą ich zasady bezpieczeństwa oraz tworzenia i odzyskiwania kopii zapasowych. Jeżeli potrzebujesz tych funkcjonalności, wówczas zamiast typu BFILE użyj typu BLOB.

1.6. Zapobieganie opóźnieniom przydzielania rozszerzeń podczas tworzenia tabel

Problem

Instalujesz aplikację wykorzystującą tysiące tabel i indeksów. Każda tabela i indeks są skonfigurowane tak, że na początku zajmują rozszerzenie o wielkości 10 MB. Podczas wykonywania zapytań instalacyjnych DDL w środowisku produkcyjnym chcesz jak najszybciej zainstalować obiekty bazy danych. Wiesz jednak, że wykonanie zapytań DDL zajmie trochę czasu, jeżeli podczas tworzenia każdego obiektu jest rezerwowanych dla niego 10 MB miejsca na dysku. Chcesz wiedzieć, czy możesz w jakiś sposób poinstruować bazę Oracle, aby odroczyła przydzielanie rozszerzenia dla każdego obiektu do chwili, gdy do tabel będą zapisywane rzeczywiste dane.

Rozwiązanie

Jedynym sposobem odroczenia tworzenia początkowych segmentów jest użycie wersji bazy Enterprise Edition Oracle Database 11g R2 lub nowszej. W wersji Enterprise Edition przydzielanie fizycznego miejsca dla rozszerzenia tabeli (i związanych z nią indeksów) jest odraczane do chwili umieszczenia w niej pierwszego wpisu. Prosty przykład pozwoli zilustrować to zagadnienie. Najpierw jest tworzona tabela:

```
create table pracownicy(
  id_pracownika number,
  imie varchar2(30),
  nazwisko varchar2(30)
);
```

Następnie można sprawdzić tabele USER_SEGMENTS oraz USER_EXTENTS, aby przekonać się, że nie zostało przydzielone fizyczne miejsce:

```
SQL> select count(*) from user_segments where segment_name='PRACOWNICY';
COUNT(*)
----------
         0

SQL> select count(*) from user_extents where segment_name='PRACOWNICY';
COUNT(*)
----------
         0
```

Następnie zapisywany jest jeden rekord danych i ponownie są wykonywane powyższe zapytania:

```
SQL> insert into pracownicy values(1,'Jan','Nowak');
1 row created.

SQL> select count(*) from user_segments where segment_name='PRACOWNICY';
COUNT(*)
----------
         1

SQL> select count(*) from user_extents where segment_name='PRACOWNICY';
COUNT(*)
----------
         1
```

Opisane zachowanie bazy jest nieco inne niż w poprzednich wersjach, w których podczas tworzenia tabeli przydzielane było miejsce na segment i związane z nim rozszerzenie.

■ **Uwaga.** Odroczone tworzenie segmentów dotyczy również tabel partycjonowanych i indeksów. Miejsce dla rozszerzenia nie będzie przydzielone do chwili zapisania pierwszego rekordu w danym segmencie.

Jak to działa

Począwszy od wersji Enterprise Edition bazy Oracle 11g R2 (i żadnej innej, jak np. Standard Edition), w przypadku niepartycjonowanych tabel stertowych tworzonych w przestrzeniach zarządzanych lokalnie tworzenie segmentów jest odraczane do momentu umieszczenia rekordu w tabeli. O tej funkcjonalności powinieneś pamiętać z kilku powodów:

- Umożliwia ona szybszą instalację aplikacji składających się z dużej liczby tabel i indeksów. Przyspiesza instalację szczególnie w przypadkach, gdy są tworzone tysiące obiektów.
- Jako administratora mogą Cię niepokoić raporty dotyczące wykorzystania miejsca, pokazujące, że dla obiektów nie zostało przydzielone żadne miejsce.
- Tworzenie pierwszego wiersza zajmie nieco więcej czasu niż w poprzednich wersjach bazy (ponieważ teraz baza Oracle przydziela miejsce dla pierwszego rozszerzenia dopiero podczas tworzenia pierwszego wiersza). W przypadku większości aplikacji takie zmniejszenie wydajności jest niezauważalne.
- Mogą pojawić się nieprzewidziane efekty użycia tej funkcjonalności (więcej na ten temat w jednym z poniższych akapitów).

Zauważyliśmy, że jedynym „rozwiązaniem" umożliwiającym zastosowanie tej funkcjonalności jest aktualizacja bazy do wersji 11g R2 (Enterprise Edition), co nie zawsze jest możliwe. Uznaliśmy jednak, że warto omówić tę funkcjonalność, ponieważ ostatecznie spotkasz się z opisaną wyżej charakterystyką.

Możesz wyłączyć funkcjonalność odroczonego tworzenia segmentów, ustawiając parametr inicjalizacyjny `DEFERRED_SEGMENT_CREATION` na `FALSE`. Jego domyślną wartością jest `TRUE`. Możesz również sterować realizacją tej funkcjonalności podczas tworzenia tabel. Instrukcja `CREATE TABLE` składa się z dwóch klauzul: `SEGMENT CREATION IMMEDIATE` oraz `SEGMENT CREATION DEFERRED`, na przykład:

```
create table pracownicy(
  id_pracownika number,
  imie varchar2(30),
  nazwisko varchar2(30)
)
segment creation immediate;
```

Należy zaznaczyć, że mogą pojawić się pewne potencjalne skutki uboczne odroczonego tworzenia segmentów. Na przykład uwaga nr 1050193.1 w serwisie MOS (My Oracle Support — „moje wsparcie Oracle") opisuje potencjalną możliwość zdefiniowania sekwencji, która powinna odliczać od 1, a w rzeczywistości zaczyna od 2.

Ponadto ponieważ funkcjonalność odroczonego tworzenia segmentów jest dostępna tylko w wersji Enterprise Edition bazy Oracle, więc gdy wyeksportujesz obiekty (dla których nie zostały jeszcze utworzone segmenty) i zaimportujesz je do wersji Standard Edition, wówczas może pojawić się błąd `ORA-00439: feature not enabled` (funkcjonalność niedostępna). Potencjalne rozwiązanie tego problemu polega na zmianie ustawienia `ALTER SYSTEM SET DEFERRED_SEGMENT_CREATION=FALSE` lub utworzeniu tabeli z ustawieniem `SEGMENT CREATION IMMEDIATE`. Szczegółowe informacje zawarte są w notatce nr 1087325.1 w serwisie MOS.

■ **Uwaga**. Przed użyciem klauzuli `SEGMENT CREATION DEFERRED` parametr inicjalizacyjny `OMPATIBLE` musi być ustawiony na wartość `11.2.0.0.0` lub większą.

1.7. Maksymalizacja prędkości ładowania danych

Problem

Ładujesz do tabeli dużą ilość danych i chcesz jak najszybciej zapisać nowe rekordy.

Rozwiązanie

Najpierw ustaw atrybut logowania operacji wykonywanych na tabeli na wartość `NOLOGGING`. W ten sposób zminimalizujesz tworzenie logów powtórzeń dla operacji wykorzystujących bezpośrednią ścieżkę do pliku (ta funkcjonalność nie dotyczy zwykłych operacji DML). Następnie użyj funkcjonalności ładowania danych z wykorzystaniem bezpośredniej ścieżki do pliku, na przykład:

- `INSERT /*+ APPEND */` w zapytaniach wykorzystujących podzapytania określające umieszczane rekordy.
- `INSERT /*+ APPEND_VALUES */` w zapytaniach wykorzystujących klauzulę `VALUES`.
- `CREATE TABLE...AS SELECT`.

Poniżej przedstawiony jest przykład użycia atrybutu `NOLOGGING` i bezpośredniej ścieżki ładowania. Najpierw uruchom poniższe zapytanie, aby sprawdzić status logowania operacji wykonywanych na tabeli:

```
select table_name, logging
from user_tables
where table_name = 'PRACOWNICY';
```

Przykładowy wynik:

```
TABLE_NAME LOG
---------- ---
PRACOWNICY YES
```

Powyższy wynik potwierdza, że tabela została utworzona z atrybutem LOGGING (domyślne ustawienie). Aby ustawić atrybut NOLOGGING, użyj następującej instrukcji ALTER TABLE:

```
SQL> alter table pracownicy nologging;
```

Ponieważ teraz ustawiony jest atrybut NOLOGGING, podczas operacji z bezpośrednią ścieżką powinna być generowana minimalna ilość rekordów powtórzeń.

Poniższy przykład wykorzystuje instrukcję INSERT z bezpośrednią ścieżką do ładowania danych do tabeli:

```
SQL> insert /*+APPEND */ into pracownicy(imie) select username from all_users;
```

Powyższa instrukcja jest skuteczną metodą ładowania danych, ponieważ operacje wykorzystujące bezpośrednią ścieżkę, takie jak INSERT /*+APPEND */, w połączeniu z ustawionym atrybutem NOLOGGING generują minimalną ilość rekordów powtórzeń.

Upewnij się również, że zatwierdziłeś dane załadowane za pomocą bezpośredniej ścieżki, gdyż w przeciwnym razie nie będziesz mógł ich zobaczyć, a baza Oracle zgłosi błąd ORA-12838 informujący, że dane załadowane za pomocą bezpośredniej ścieżki muszą być zatwierdzone przed ich odczytaniem.

■ **Uwaga**. Podczas ładowania danych do tabeli za pomocą bezpośredniej ścieżki baza Oracle umieszcza nowe wiersze powyżej wskaźnika zajętości miejsca. Nawet jeżeli za pomocą instrukcji DELETE zostanie zwolniona duża ilość miejsca, baza Oracle zawsze będzie umieszczać nowe dane powyżej wskaźnika, przez co tabela może zajmować dużo miejsca na dysku, lecz niekoniecznie będzie zawierać dużo danych.

Jak to działa

Umieszczanie danych z wykorzystaniem bezpośredniej ścieżki do pliku ma w porównaniu ze zwykłymi poleceniami dwie zalety związane z wydajnością:

- Jeżeli ustawiony jest atrybut NOLOGGING, wówczas generowana jest minimalna ilość rekordów powtórzeń.
- Pomijany jest bufor i dane są ładowane bezpośrednio do plików. W ten sposób wydajność operacji ładowania danych może być znacznie zwiększona.

Atrybut NOLOGGING minimalizuje ilość generowanych rekordów powtórzeń tylko dla operacji z bezpośrednią ścieżką dostępu do danych. W przypadku umieszczania danych opcja NOLOGGING może znacznie zwiększyć szybkość ładowania danych. Panuje przekonanie, że atrybut NOLOGGING wyłącza generowanie rekordów powtórzeń dla wszystkich operacji DML. Nie jest to prawdą. Funkcjonalność ta w żaden sposób nie wpływa na generowanie rekordów dla zwykłych instrukcji INSERT, UPDATE, MERGE i DELETE.

Jednym z ubocznych skutków redukcji ilości generowanych rekordów powtórzeń jest brak możliwości odtworzenia danych utworzonych z opcją NOLOGGING w przypadku wystąpienia błędu po załadowaniu danych (a przed utworzeniem kopii zapasowej tabeli). Jeżeli dopuszczasz pewne ryzyko utraty danych, stosuj atrybut NOLOGGING, ale utwórz kopię zapasową tabeli zaraz po jej załadowaniu. Jeżeli dane są szczególnie ważne, nie stosuj atrybutu NOLOGGING podczas operacji z użyciem bezpośredniej ścieżki. Jeżeli dane mogą być łatwo wygenerowane ponownie, to użycie NOLOGGING w celu zwiększenia wydajności operacji ładowania jest uzasadnione.

Co się stanie, jeżeli po załadowaniu danych do tabeli w trybie NOLOGGING (a przed utworzeniem jej kopii zapasowej) wystąpi problem z dyskiem? Po wykonaniu operacji odzyskania danych tabela będzie wyglądała na odtworzoną:

```
SQL> desc pracownicy;
```

Jednak podczas wykonywania zapytania odczytującego każdy blok tabeli zostanie zgłoszony błąd.

```
SQL> select * from pracownicy;
```

Oznacza on, że plik danych jest uszkodzony:

```
ORA-01578: ORACLE data block corrupted (file # 4, block # 147)
ORA-01110: data file 4: '/u01/dbfile/O12C/users01.dbf'
ORA-26040: Data block was loaded using the NOLOGGING option
// Uszkodzony blok danych (plik nr 4, blok nr 147)
// Plik danych nr 4: '/u01/dbfile/O12C/users01.dbf'
// Blok danych został załadowany z opcją NOLOGGING
```

Zgodnie z powyższą informacją danych w tabeli nie da się odzyskać. Stosuj atrybut `NOLOGGING` tylko w sytuacji, gdy dane nie są krytycznie ważne lub gdy możesz wykonać ich kopię zapasową wkrótce po utworzeniu.

Wskazówka. Jeżeli do tworzenia zapasowej kopii bazy danych używasz narzędzia *RMAN*, możesz wyświetlić nienaprawialne pliki danych za pomocą polecenia `REPORT UNRECOVERABLE`.

Opcja `NOLOGGING` ma kilka haczyków, które wymagają omówienia. Możesz określić tryb logowania operacji na poziomie bazy danych, przestrzeni tabel i obiektów. Jeżeli Twoja baza wymusza logowanie, wówczas opcja ta ma znaczenie nadrzędne w stosunku do atrybutu `NOLOGGING` ustawionego dla tabeli. Jeżeli użyjesz klauzuli logowania na poziomie przestrzeni tabel, wówczas ustawienie to będzie domyślne dla wszystkich poleceń `CREATE TABLE` nie stosujących w sposób jawny klauzuli logowania.

Tryb logowania na poziomie bazy danych możesz sprawdzić w następujący sposób:

```
SQL> select name, log_mode, force_logging from v$database;
```

Poniższa instrukcja służy do sprawdzenia trybu logowania dla przestrzeni tabel:

```
SQL> select tablespace_name, logging from dba_tablespaces;
```

Natomiast poniższa weryfikuje tryb logowania dla tabeli:

```
SQL> select owner, table_name, logging from dba_tables where logging = 'NO';
```

Jak możesz sprawdzić, czy baza Oracle zapisała rekord powtórzeń dla danej operacji? Jednym ze sposobów jest odczyt liczby rekordów wygenerowanych dla danej operacji z włączonym logowaniem i porównanie z trybem `NOLOGGING`. Jeżeli posiadasz środowisko testowe, możesz śledzić częstotliwość przełączania logów powtórzeń podczas realizacji transakcji. Innym prostym sposobem jest pomiar czasu wykonywania operacji z logowaniem i bez logowania. Operacja w trybie `NOLOGGING` powinna być wykonywana krócej, ponieważ ilość rekordów powtórzeń wygenerowanych podczas ładowania danych jest minimalna.

1.8. Wydajne usuwanie danych z tabel

Problem

Podczas usuwania danych z tabeli pojawiły się problemy wydajnościowe. Chcesz usunąć dane tak szybko, jak to jest możliwe.

Rozwiązanie

W celu usunięcia rekordów z tabeli możesz użyć instrukcji `TRUNCATE` lub `DELETE`. Instrukcja `TRUNCATE` jest zazwyczaj bardziej wydajna, ale powoduje pewne skutki uboczne, o których musisz pamiętać. Na przykład `TRUNCATE` jest instrukcją języka DDL. Oznacza to, że baza Oracle automatycznie zatwierdza tę instrukcję (i bieżącą transakcję) zaraz po jej wykonaniu i nie ma sposobu na jej odwołanie. Ponadto w jednej transakcji nie możesz usunąć danych z dwóch oddzielnych tabel.

W poniższym przykładzie instrukcji TRUNCATE użyto do usunięcia wszystkich danych z tabeli:

```
SQL> truncate table pracownicy
```

Domyślnie podczas usuwania danych z tabeli zwalniana jest cała przestrzeń z wyjątkiem ilości określonej za pomocą parametru MINEXTENTS. Jeżeli nie chcesz, aby instrukcja TRUNCATE zwolniła aktualnie zajęte rozszerzenia, możesz użyć klauzuli REUSE STORAGE:

```
SQL> truncate table pracownicy reuse storage;
```

Aby sprawdzić, czy rozszerzenia zostały (lub nie zostały) zwolnione, możesz odczytać widoki DBA/ALL/USER_EXTENTS, na przykład:

```
SQL> select count(*) from user_extents where segment_name = 'PRACOWNICY';
```

Warto w tym miejscu wspomnieć, że instrukcja TRUNCATE jest często stosowana w przypadku tabel partycjonowanych, szczególnie podczas archiwizacji przestarzałych danych, których nie trzeba już przechowywać w określonej partycji. Na przykład poniższa instrukcja skutecznie usuwa dane z zadanej partycji, nie wpływając na inne partycje danej tabeli:

```
SQL> alter table f_sales truncate partition p_2012;
```

Jak to działa

Jeżeli chcesz mieć możliwość odwołania (zamiast potwierdzenia) usunięcia danych, używaj instrukcji DELETE. Jednak ta instrukcja ma tę wadę, że generuje dużą ilość rekordów powtórzeń i odwołań. Dlatego w przypadku dużych tabel instrukcja TRUNCATE jest zazwyczaj najskuteczniejszą metodą usuwania danych.

Inną cechą instrukcji TRUNCATE jest ustawianie **wskaźnika zajętości miejsca** tabeli na zero. Baza Oracle definiuje ten wskaźnik jako granicę między zajętym i wolnym miejscem w segmencie. Podczas tworzenia tabeli Oracle przydziela jej pewną liczbę rozszerzeń, określoną przez parametr MINEXTENTS. Każde rozszerzenie zawiera pewną liczbę bloków. Zanim dane zostaną umieszczone w tabeli, żaden blok nie jest zajęty i wskaźnik jest ustawiony na zero. Podczas dodawania danych do tabeli wskaźnik granicy jest przesuwany.

Jeżeli do usuwania danych z tabeli użyjesz instrukcji DELETE, wówczas wskaźnik nie będzie zmieniany. Jedna z zalet instrukcji TRUNCATE i resetowania wskaźnika polega na tym, że zapytania wykonujące pełne skanowanie tabeli przeszukują tylko wiersze zapisane poniżej wskaźnika. Ma to znaczny wpływ na wydajność zapytań wykonujących pełne skanowanie tabel.

Innym efektem ubocznym instrukcji TRUNCATE jest brak możliwości usunięcia danych z tabeli nadrzędnej ze zdefiniowanym kluczem podstawowym, do którego odwołuje się ograniczenie odblokowanego klucza obcego zdefiniowane w tabeli podrzędnej, nawet jeżeli tabela podrzędna nie zawiera żadnych wierszy. W takim przypadku podczas próby usunięcia danych z tabeli nadrzędnej baza Oracle zgłasza poniższy błąd:

```
ORA-02266: unique/primary keys in table referenced by enabled foreign keys
// Do unikatowych/podstawowych kluczy w tabeli odwołują się odblokowane klucze obce
```

Baza uniemożliwi Ci usunięcie danych z tabeli nadrzędnej, ponieważ może się zdarzyć, że w okresie pomiędzy usunięciem przez Ciebie danych z tabeli podrzędnej a później z tabeli nadrzędnej tabela podrzędna zostanie wypełniona wierszami w innej sesji. W takim przypadku musisz tymczasowo zablokować ograniczenia odwołujące się do klucza obcego w tabeli podrzędnej, wykonać instrukcję TRUNCATE, a następnie odblokować ograniczenia.

Porównaj działanie instrukcji TRUNCATE i DELETE. Baza Oracle umożliwia Ci użycie dwóch instrukcji DELETE w celu usunięcia wierszy w tabeli nadrzędnej, gdy włączone są ograniczenia odwołujące się do tabeli podrzędnej (przy założeniu, że nie ma w niej żadnych wierszy). Jest tak, ponieważ instrukcja DELETE generuje rekordy powtórzeń, zapewnia spójność odczytywanych danych i może być odwołana. Tabela 1.5 zawiera podsumowanie różnic pomiędzy instrukcjami TRUNCATE i DELETE.

Tabela 1.5. *Porównanie instrukcji* `TRUNCATE` *i* `DELETE`

	DELETE	TRUNCATE
Umożliwia zatwierdzanie i odwoływanie zmian	Tak	Nie (język DDL zawsze zatwierdza transakcje po wykonaniu instrukcji)
Generuje rekordy odwołań	Tak	Nie
Ustawia wskaźnik zajętości miejsca na zero	Nie	Tak
Uzależniona od włączonego ograniczenia klucza obcego	Nie	Tak
Działa wydajnie w przypadku dużych ilości danych	Nie	Tak

Jeżeli potrzebujesz użyć instrukcji `DELETE`, musisz użyć również instrukcji `COMMIT` lub `ROLLBACK`, aby zakończyć transakcję. Zatwierdzenie instrukcji `DELETE` sprawia, że zmiany wprowadzone w danych są trwałe:

```
SQL> delete from pracownicy;
SQL> commit;
```

> **Uwaga**. Inne (niekiedy niezbyt oczywiste) sposoby zatwierdzania transakcji polegają na wykonywaniu serii instrukcji DDL (które niejawnie zatwierdzają aktywne transakcje w sesji) lub po prostu na zamknięciu narzędzia klienckiego (np. SQL*Plus).

Jeżeli zamiast instrukcji `COMMIT` wprowadzisz instrukcję `ROLLBACK`, wówczas tabela będzie zawierała dane sprzed wykonania instrukcji `DELETE`.

Podczas korzystania z instrukcji DML możesz sprawdzić szczegóły transakcji, odpytując widok `V$TRANSACTION`. Załóżmy na przykład, że właśnie umieściłeś dane w tabeli. Zanim wprowadzisz instrukcję `COMMIT` lub `ROLLBACK`, możesz sprawdzić informacje o aktywnych transakcjach bieżącej sesji w następujący sposób:

```
SQL> insert into pracownicy values(1, 'Jan', 'Nowak');
SQL> select xidusn, xidsqn from v$transaction;
    XIDUSN     XIDSQN
---------- ----------
         9        369
SQL> commit;
SQL> select xidusn, xidsqn from v$transaction;
no rows selected
```

> **Uwaga**. Innym sposobem usuwania danych z tabeli jest usunięcie całej tabeli i ponowne jej utworzenie. Oznacza to jednak, że musisz ponownie utworzyć wszystkie indeksy, uprawnienia i wyzwalacze dotyczące tej tabeli. Ponadto tabela po usunięciu będzie niedostępna do momentu jej ponownego utworzenia i wprowadzenia wymaganych uprawnień. Zazwyczaj usuwanie i ponowne tworzenie tabel jest dopuszczalne tylko w środowiskach programistycznych i testowych.

1.9. Wyświetlanie automatycznych zaleceń narzędzia Segment Advisor

Problem

Znalazłeś niewydajne zapytanie odczytujące dane z tabeli. Po dokładniejszym przyjrzeniu się mu odkryłeś, że tabela zawiera tylko kilka wierszy. Jesteś ciekaw, dlaczego zapytanie wykonuje się tak długo, skoro jest tylko kilka wierszy. Chciałbyś sprawdzić informacje zwracane przez narzędzie Segment Advisor, aby dowiedzieć się, czy są jakieś zalecenia, które mogłyby pomóc w tym przypadku, dotyczące przydzielania miejsca tabelom.

Rozwiązanie

Stosuj narzędzie Segment Advisor do wyświetlania informacji dotyczących tabel, którym zostało przydzielone miejsce (wykorzystane wcześniej), ale które teraz jest puste (po usunięciu dużej liczby wierszy). Tabele z dużą ilością niewykorzystanego miejsca mogą powodować, że wydajność zapytań wykonujących pełne skanowanie będzie bardzo słaba. Jest tak dlatego, że baza Oracle skanuje każdy blok poniżej wskaźnika zajętego miejsca, niezależnie od tego, czy blok zawiera dane, czy nie.

Rozwiązanie problemu polega na wyświetleniu zalecenia narzędzia Segment Advisor za pomocą pakietu `DBMS_SPACE` języka PL/SQL. Ten pakiet pobiera informacje wygenerowane przez narzędzie Segment Advisor dotyczące segmentów, które mogą nadawać się do zmniejszenia, przesunięcia lub skompresowania. Jednym z prostych i skutecznych sposobów użycia pakietu `DBMS_SPACE` (w celu uzyskania zalecenia narzędzia Segment Advisor) jest odwołanie się do niego w zapytaniu SQL, na przykład:

```
SELECT
  'Segment Advice --------------------------'|| chr(10) ||
  'TABLESPACE_NAME  : ' || tablespace_name   || chr(10) ||
  'SEGMENT_OWNER    : ' || segment_owner     || chr(10) ||
  'SEGMENT_NAME     : ' || segment_name      || chr(10) ||
  'ALLOCATED_SPACE  : ' || allocated_space   || chr(10) ||
  'RECLAIMABLE_SPACE: ' || reclaimable_space || chr(10) ||
  'ZALECENIA        : ' || recommendations   || chr(10) ||
  'ROZWIAZANIE 1    : ' || c1                || chr(10) ||
  'ROZWIAZANIE 2    : ' || c2                || chr(10) ||
  'ROZWIAZANIE 3    : ' || c3 Porada
FROM
TABLE(dbms_space.asa_recommendations('FALSE', 'FALSE', 'FALSE'));
```

Poniżej jest przedstawiony przykładowy wynik:

```
PORADA
--------------------------------------------------------------------------------
Segment Advice --------------------------
TABLESPACE_NAME  : USERS
SEGMENT_OWNER    : SERWIS
SEGMENT_NAME     : PRACOWNICY
ALLOCATED_SPACE  : 50331648
RECLAIMABLE_SPACE: 40801189
ZALECENIA        : Enable row movement of the table SERWIS.PRACOWNICY and perform shrink, estimated
savings is 40801189 bytes.
// Włącz przenoszenie wierszy w tabeli SERWIS.PRACOWNICY i zmniejsz ją. Szacowana oszczędność miejsca 40801189 bajtów
ROZWIAZANIE 1   : alter table "SERWIS"."PRACOWNICY" shrink space
ROZWIAZANIE 2   : alter table "SERWIS"."PRACOWNICY" shrink space COMPACT
ROZWIAZANIE 3   : alter table "SERWIS"."PRACOWNICY" enable row movement
```

Zgodnie z powyższymi informacjami tabela `PRACOWNICY` jest kandydatem do zwolnienia miejsca w wyniku wykonania takich operacji jak jej zmniejszenie lub reorganizacja.

Jak to działa

Baza Oracle w wersjach 10g R2 i nowszych automatycznie planuje i uruchamia narzędzie Segment Advisor. To zadanie analizuje segmenty bazy i zapisuje wnioski w wewnętrznych tabelach. Informacje zwrócone przez narzędzie Segment Advisor zawierają wnioski (problemy, które należy rozwiązać) oraz zalecenia (czynności prowadzące do rozwiązania problemów). Narzędzie Segment Advisor podaje następujące informacje:

- Segmenty nadające się do operacji zmniejszenia.
- Segmenty zawierające znaczną liczbę zmigrowanych/łańcuchowanych wierszy.
- Segmenty, które mogą zyskać na zaawansowanej kompresji danych.

Podczas przeglądania wniosków i zaleceń narzędzia Segment Advisor ważna jest znajomość kilku jego cech. Po pierwsze, Segment Advisor regularnie wykonuje analizę za pomocą automatycznie planowanego zadania `DBMS_SCHEDULER`. Czas wykonania ostatniego automatycznego zadania możesz sprawdzić, odpytując widok `DBA_AUTO_SEGADV_SUMMARY`:

```
select segments_processed, end_time
from dba_auto_segadv_summary
order by end_time;
```

Możesz porównać datę `END_TIME` z bieżącą datą i sprawdzić, czy Segment Advisor jest uruchamiany w regularnych odstępach czasu.

Uwaga. Oprócz automatycznego generowania zaleceń masz możliwość ręcznego uruchomienia narzędzia Segment Advisor i wygenerowania zaleceń dotyczących określonych przestrzeni, tabel i indeksów (szczegółowe informacje zawarte są w poradzie 1.10).

Narzędzie Segment Advisor korzysta w trakcie swego działania z repozytorium AWR (Automatic Workload Repository, repozytorium automatycznego obciążenia) jako źródła informacji do analizy. Na przykład sprawdza statystyki wykorzystania i wzrostu bazy zapisane w repozytorium AWR do generowania zaleceń dotyczących segmentów. Segment Advisor w trakcie działania generuje zalecenia i zapisuje wyniki w wewnętrznych tabelach. Wnioski i zalecenia mogą być przeglądane za pomocą widoków słownikowych, takich jak:

- `DBA_ADVISOR_EXECUTIONS`,
- `DBA_ADVISOR_FINDINGS`,
- `DBA_ADVISOR_OBJECTS`.

Uwaga. Widoki `DBA_ADVSIOR_*` są częścią pakietu Diagnostics Pack, wymagającego bazy Oracle Enterprise Edition i dodatkowej licencji.

Są trzy różne sposoby odczytywania informacji zebranych przez narzędzie Segment Advisor:

- Wykonanie procedury `DBMS_SPACE.ASA_RECOMMENDATIONS`.
- Ręczne odpytanie widoków `DBA_ADVISOR_*`.
- Użycie interfejsu graficznego narzędzia Enterprise Manager.

W części „Rozwiązanie” opisaliśmy, jak korzystać z procedury `DBMS_SPACE.ASA_RECOMMENDATIONS` w celu odczytania zaleceń narzędzia Segment Advisor. Informacje zwrócone przez procedurę `ASA_RECOMMENDATIONS` mogą być modyfikowane za pomocą trzech parametrów opisanych w tabeli 1.6. Na przykład można skonfigurować procedurę tak, aby pokazywała tylko informacje zebrane podczas ręcznego uruchomienia narzędzia Segment Advisor.

Tabela 1.6. *Opis parametrów wejściowych procedury ASA_RECOMMENDATIONS*

Parametr	Opis
`all_runs`	Wartość `TRUE` spowoduje, że procedura zwróci wnioski zebrane podczas wszystkich jej wywołań, natomiast `FALSE` spowoduje, że procedura zwróci wnioski tylko z jej ostatniego wywołania.
`show_manual`	Wartość `TRUE` konfiguruje procedurę tak, aby zwróciła wyniki zebrane podczas ręcznego uruchomienia narzędzia Segment Advisor.
`show_findings`	Wartość `FALSE` konfiguruje procedurę tak, aby zwróciła wyniki zebrane podczas automatycznego uruchomienia narzędzia Segment Advisor. Pokazywane są tylko wnioski, bez zaleceń.

Aby przejrzeć zalecenia zebrane przez narzędzie Segment Advisor, możesz również bezpośrednio odpytać widoki słownikowe. Poniżej przedstawione jest zapytanie zwracające zalecenia wygenerowane przez narzędzie w ciągu ostatniego dnia:

```
SELECT
  'Nazwa zadania     : ' || f.task_name || chr(10) ||
  'Czas uruchomienia : ' || TO_CHAR(execution_start, 'dd-mon-yy hh24:mi') || chr (10) ||
  'Nazwa segmentu    : ' || o.attr2     || chr(10) ||
  'Typ segmentu      : ' || o.type      || chr(10) ||
  'Nazwa partycji    : ' || o.attr3     || chr(10) ||
  'Komunikat         : ' || f.message   || chr(10) ||
  'Więcej informacji : ' || f.more_info || chr(10) ||
  '--------------------------------------------------' Porada
FROM
  dba_advisor_findings f,
  dba_advisor_objects o,
  dba_advisor_executions e
WHERE
  o.task_id = f.task_id
  AND o.object_id = f.object_id
  AND f.task_id = e.task_id
  AND e. execution_start > sysdate - 1
  AND e.advisor_name = 'Segment Advisor'
ORDER BY f.task_name;
```

■ **Uwaga**. Zalecenia zebrane przez narzędzie Segment Advisor możesz obejrzeć w narzędziu Enterprise Manager. W tym celu w zakładce *Performance* przejdź do strony *Advisors Home*, a następnie *Segment Advisor*. Na tej stronie możesz generować raporty narzędzia Segment Advisor.

1.10. Ręczne generowanie zaleceń narzędzia Segment Advisor

Problem

Masz tabelę, w której bardzo często są aktualizowane dane. Użytkownicy zgłaszają Ci, że zapytania na tej tabeli są wykonywane coraz dłużej. W ramach swojej analizy masz zamiar ręcznie uruchomić narzędzie Segment Advisor dla tej tabeli i sprawdzić, czy pojawiły się jakieś problemy dotyczące miejsca na dysku, na przykład migracja/łańcuchowanie wierszy lub niewykorzystane miejsce poniżej wskaźnika zajętości.

Rozwiązanie

Możesz ręcznie uruchomić narzędzie Segment Advisor i zlecić w nim analizę wszystkich segmentów wybranej przestrzeń tabel lub sprawdzić określony obiekt (na przykład tabelę lub indeks). Poniżej opisane są kroki operacji ręcznego uruchamiania narzędzia Segment Advisor:

1. Utwórz zadanie.
2. Przypisz obiekt do zadania.
3. Ustaw parametry zadania.
4. Uruchom zadanie.

Uwaga. Użytkownik bazy uruchamiający procedurę `DBMS_ADVISOR` musi posiadać uprawnienie systemowe `ADVISOR`. To uprawnienie jest nadawane za pomocą instrukcji `GRANT`.

Poniższy przykład przedstawia kod PL/SQL wykonujący cztery opisane wyżej kroki. Sprawdzana tabela nosi nazwę `PRACOWNICY`, a jej właścicielem jest użytkownik `SERWIS`:

```
DECLARE
  my_task_id number;
  obj_id number;
  my_task_name varchar2(100);
  my_task_desc varchar2(500);
BEGIN
  my_task_name := 'Informacje o tabeli PRACOWNICY';
  my_task_desc := 'Ręczne uruchomienie Segment Advisor';
  ---------
  -- Krok 1
  ---------
  dbms_advisor.create_task (
    advisor_name => 'Segment Advisor',
    task_id => my_task_id,
    task_name => my_task_name,
    task_desc => my_task_desc);
  ---------
  -- Krok 2
  ---------
  dbms_advisor.create_object (
    task_name => my_task_name,
    object_type => 'TABLE',
    attr1 => 'SERWIS',
    attr2 => 'PRACOWNICY',
    attr3 => NULL,
    attr4 => NULL,
    attr5 => NULL,
    object_id => obj_id);
  ---------
  -- Krok 3
  ---------
  dbms_advisor.set_task_parameter(
    task_name => my_task_name,
    parameter => 'recommend_all',
    value => 'TRUE');
  ---------
  -- Krok 4
  ---------
  dbms_advisor.execute_task(my_task_name);
END;
/
```

Teraz możesz uruchomić pakiet DBMS_SPACE, zlecić zebranie informacji poprzez ręczne uruchomienie narzędzia Segment Advisor (za pomocą parametrów wejściowych — patrz tabela 1.7 zawierająca szczegółowe informacje) i przejrzeć zebrane zalecenia, na przykład:

```
SELECT
  'Segment Advice --------------------------'|| chr(10) ||
  'TABLESPACE_NAME  : ' || tablespace_name   || chr(10) ||
  'SEGMENT_OWNER    : ' || segment_owner     || chr(10) ||
  'SEGMENT_NAME     : ' || segment_name      || chr(10) ||
  'ALLOCATED_SPACE  : ' || allocated_space   || chr(10) ||
  'RECLAIMABLE_SPACE: ' || reclaimable_space || chr(10) ||
  'ZALECENIA        : ' || recommendations   || chr(10) ||
  'ROZWIAZANIE 1    : ' || c1                || chr(10) ||
  'ROZWIAZANIE 2    : ' || c2                || chr(10) ||
  'ROZWIAZANIE 3    : ' || c3 Porada
FROM
TABLE(dbms_space.asa_recommendations('TRUE', 'TRUE', 'FALSE'));
```

Tabela 1.7. *Procedury pakietu DBMS_ADVISOR wykorzystujące narzędzie Segment Advisor*

Nazwa procedury	Opis
CREATE_TASK	Tworzy zadanie narzędzia Segment Advisor. Ustaw wartość parametru ADVISOR_NAME procedury CREATE_TASK na **Segment Advisor**. Odpytaj widok DBA_ADVISOR_DEFINITIONS, aby poznać listę wszystkich narzędzi.
CREATE_OBJECT	Określa obiekt będący przedmiotem analizy. Tabela 1.8 zawiera listę wszystkich typów obiektów i parametrów.
SET_TASK_PARAMETER	Określa typ informacji, jaka ma być zwrócona. Tabela 1.9 zawiera listę parametrów i ich wartości.
EXECUTE_TASK	Uruchamia zadania Segment Advisor.
DELETE_TASK	Usuwa zadanie.
CANCEL_TASK	Przerywa aktualnie wykonywane zadanie.

Informacje zebrane przez Segment Advisor możesz również pobrać, odpytując widok słownikowy, na przykład:

```
SELECT
  'Nazwa zadania     : ' || f.task_name || chr(10) ||
  'Nazwa segementu   : ' || o.attr2     || chr(10) ||
  'Typ segmentu      : ' || o.type      || chr(10) ||
  'Nazwa partycji    : ' || o.attr3     || chr(10) ||
  'Komunikat         : ' || f.message   || chr(10) ||
  'Więcej informacji : ' || f.more_info PORADA
FROM
  dba_advisor_findings f,
  dba_advisor_objects o
WHERE
  o.task_id = f.task_id
  AND o.object_id = f.object_id
  AND f.task_name like '&task_name'
ORDER BY f.task_name;
```

Jeżeli w tabeli zostaną znalezione problemy związane z miejscem na dysku, wówczas zwrócone informacje opiszą je w następujący sposób:

```
PORADA
--------------------------------------------------------------------------------
Nazwa zadania     : PRACOWNICY Advice
Nazwa segmentu    : PRACOWNICY
Typ segmentu      : TABLE
Nazwa partycji    :
Komunikat         : The object has chained rows that can be removed by re-org.
Więcej informacji : 22 percent chained rows can be removed by re-org.
```

// Komunikat: Obiekt zawiera łańcuchowane wiersze, które mogą być usunięte poprzez reorganizację tabeli
// Więcej informacji: 22 procent łańcuchowanych wierszy może być usuniętych poprzez reorganizację tabeli[GM4]

Jak to działa

Pakiet DBMS_ADVISOR jest wykorzystywany do ręcznego zlecania narzędziu Segment Advisor generowania zaleceń dotyczących określonych tabel. Pakiet zawiera procedury wykonujące takie operacje jak tworzenie i uruchamianie zadań. Tabela 1.7 zawiera listę procedur związanych z narzędziem Segment Advisor.

Narzędzie Segment Advisor może być uruchomione z różnymi stopniami dokładności. Na przykład może generować zalecenia dla wszystkich obiektów w przestrzeni tabel lub tylko dla określonej tabeli, indeksu lub partycji. Tabela 1.8 zawiera listę typów obiektów, dla których zalecenia mogą być generowane przez narzędzie Segment Advisor i odczytywane za pomocą procedury DBMS_ADVISOR.CREATE_TASK.

Tabela 1.8. *Typy obiektów dla procedury DBMS_ADVISOR.CREATE_TASK*

Typ obiektu	Atrybut 1	Atrybut 2	Atrybut 3	Atrybut 4
TABLESPACE	Nazwa przestrzeni tabel	NULL	NULL	NULL
TABLE	Nazwa użytkownika	Nazwa tabeli	NULL	NULL
INDEX	Nazwa użytkownika	Nazwa indeksu	NULL	NULL
TABLE PARTITION	Nazwa użytkownika	Nazwa tabeli	Nazwa partycji	NULL
INDEX PARTITION	Nazwa użytkownika	Nazwa indeksu	Nazwa partycji	NULL
TABLE SUBPARTITION	Nazwa użytkownika	Nazwa tabeli	Nazwa subpartycji	NULL
INDEX SUBPARTITION	Nazwa użytkownika	Nazwa indeksu	Nazwa subpartycji	NULL
LOB	Nazwa użytkownika	Nazwa segmentu	NULL	NULL
LOB PARTITION	Nazwa użytkownika	Nazwa segmentu	Nazwa partycji	NULL
LOB SUBPARTITION	Nazwa użytkownika	Nazwa segmentu	Nazwa subpartycji	NULL

Możesz również określić maksymalny czas, przez jaki może działać narzędzie Segment Advisor. Jest on określany za pomocą procedury SET_TASK_PARAMETER. Procedura ta określa również typ zbieranych zaleceń. Tabela 1.9 opisuje parametry wejściowe tej procedury.

Tabela 1.9. *Parametry wejściowe procedury DBMS_ADVISOR.SET_TASK_PARAMETER*

Parametr	Opis	Poprawne wartości
TIME_LIMIT	Limit czasu (w sekundach), przez który działa narzędzie.	N (liczba sekund) lub UNLIMITED (wartość domyślna).
RECOMMEND_ALL	Generuje wszystkie zalecenia lub tylko dotyczące zajmowanego miejsca.	TRUE (wartość domyślna) dla wszystkich zaleceń lub FALSE tylko dla dotyczących zajmowanego miejsca.

1.11. Automatyczne wysyłanie pocztą e-mail zaleceń narzędzia Segment Advisor

Problem

Wiesz już, że narzędzie Segment Advisor automatycznie generuje zlecenia, i chcesz je automatycznie wysyłać do siebie pocztą e-mail.

Rozwiązanie

Najpierw umieść w skrypcie powłoki systemu operacyjnego zapytanie SQL zwracające informacje z narzędzia Segment Advisor, na przykład w poniższy sposób:

```
#!/bin/bash
if [ $# -ne 1 ]; then
  echo "Użycie: $0 SID"
  exit 1
fi
# Odczytaj zmiennie środowiskowe dla bazy Oracle
. /etc/oraset $1
#
BOX=`uname -a | awk '{print$2}'`
#
sqlplus -s <<EOF
serwis/foo
spo $HOME/bin/log/seg.txt
set lines 80
set pages 100
SELECT
  'Segment Advice --------------------------'|| chr(10) ||
  'TABLESPACE_NAME  : ' || tablespace_name   || chr(10) ||
  'SEGMENT_OWNER    : ' || segment_owner     || chr(10) ||
  'SEGMENT_NAME     : ' || segment_name      || chr(10) ||
  'ALLOCATED_SPACE  : ' || allocated_space   || chr(10) ||
  'RECLAIMABLE_SPACE: ' || reclaimable_space || chr(10) ||
  'ZALECENIA        : ' || recommendations   || chr(10) ||
  'ROZWIAZANIE 1    : ' || c1                || chr(10) ||
  'ROZWIAZANIE 2    : ' || c2                || chr(10) ||
  'ROZWIAZANIE 3    : ' || c3 Porada
FROM
  TABLE(dbms_space.asa_recommendations('FALSE', 'FALSE', 'FALSE'));
EOF
cat $HOME/bin/log/seg.txt | mailx -s "Raport Segment Advisor z bazy: $1 $BOX" dkuhn@oracle.com
exit 0
```

Powyższy skrypt może być regularnie uruchamiany za pomocą np. narzędzia *cron* systemu Linux/Unix. Poniżej przedstawiona jest przykładowa konfiguracja narzędzia *cron*:

```
# Raport narzędzia Segment Advisor
16 11 * * * /orahome/oracle/bin/seg.bsh DWREP
```

W ten sposób będziesz mógł regularnie otrzymywać zalecenia narzędzia Segment Advisor i proaktywnie rozwiązywać problemy, zanim zaczną pogarszać wydajność bazy.

Jak to działa

Segment Advisor generuje zalecenia automatycznie i w regularnych odstępach czasu. Niekiedy przydatne jest zapobiegawcze wysyłanie zaleceń na własny adres e-mail. Dzięki temu możesz regularnie przeglądać informacje i implementować uzasadnione zalecenia.

Skrypt przedstawiony w części „Rozwiązanie" zawiera na początku wiersz, w którym poprzez uruchomienie skryptu *oraset* ustawiane są zmienne środowiskowe. Jest to osobny skrypt będący odpowiednikiem skryptu *oraenv* dostarczanego z bazą Oracle. Możesz go użyć do ustawienia zmiennych środowiskowych lub zakodować na stałe odpowiednie wiersze we własnym skrypcie. Wywoływanie innego skryptu ustawiającego zmienne środowiskowe zapewnia większą elastyczność i łatwość obsługi, ponieważ umożliwia podanie jako parametru wejściowego nazwy dowolnej bazy danych zawartej w pliku *oratab*.

1.12. Przebudowa wierszy obejmujących kilka bloków

Problem

Uruchomiłeś narzędzie Segment Advisor w następujący sposób:

```
SELECT
  'Nazwa zadania     : ' || f.task_name || chr(10) ||
  'Nazwa segmentu    : ' || o.attr2     || chr(10) ||
  'Typ segmentu      : ' || o.type      || chr(10) ||
  'Nazwa partycji    : ' || o.attr3     || chr(10) ||
  'Komunikat         : ' || f.message   || chr(10) ||
  'Więcej informacji : ' || f.more_info PORADA
FROM
  dba_advisor_findings f,
  dba_advisor_objects o
WHERE
  o.task_id = f.task_id
  AND o.object_id = f.object_id
  AND f.task_name like '&task_name'
ORDER BY f.task_name;
```

Zauważyłeś, że zwrócony wynik informuje o łańcuchowanych wierszach:

```
PORADA
--------------------------------------------------------------------------------
Nazwa zadania     : PRACOWNICY Advice
Nazwa segmentu    : PRACOWNICY
Typ segmentu      : TABLE
Nazwa partycji    :
Komunikat         : The object has chained rows that can be removed by re-org.
Więcej informacji : 22 percent chained rows can be removed by re-org.
// Komunikat: Obiekt zawiera łańcuchowane wiersze, które mogą być usunięte poprzez reorganizację tabeli
// Więcej informacji: 22 procent łańcuchowanych wierszy może być usuniętych poprzez reorganizację tabeli
```

Wiesz, że migracja/łańcuchowanie wierszy zwiększa częstotliwość wykonywanych operacji wejścia-wyjścia i może pogarszać wydajność bazy. Dlatego musisz wyeliminować migrację/łańcuchowanie wierszy w tej tabeli.

Rozwiązanie

Migracja/łańcuchowanie wierszy pojawia się w sytuacji, gdy w bloku nie ma miejsca na zapisanie wiersza i baza Oracle zapisuje go w więcej niż jednym bloku (więcej szczegółowych informacji znajduje się w części „Jak to działa" niniejszej porady). Jeżeli takich przypadków jest dużo, wówczas migracja/łańcuchowanie

wierszy może powodować wykonywanie nadmiernej ilości operacji wejścia-wyjścia podczas odczytu wiersza. Są trzy podstawowe sposoby rozwiązania tego problemu:

- Przeniesienie tabeli.
- Przeniesienie poszczególnych zmigrowanych/łańcuchowanych wierszy wewnątrz tabeli.
- Przebudowanie tabeli za pomocą narzędzia Data Pump (eksport/import).

Ta część porady skupia się na dwóch pierwszych sposobach. Użycie narzędzia Data Pump wymaga eksportu tabel, następnie ich usunięcia lub zmiany nazw i ponownego zaimportowania z pliku. Szczegółowe informacje dotyczące posługiwania się narzędziem Data Pump są zawarte w książce *Pro Oracle Database 12c Administration* (Apress) oraz w podręczniku *Oracle's Utility Guide* na stronie internetowej Oracle Technology Network.

Przeniesienie tabeli

Jednym ze sposobów rozwiązania problemu migracji/łańcuchowania wierszy w tabeli jest użycie instrukcji `MOVE` i przebudowanie tabeli wraz z jej wierszami z mniejszą wartością atrybutu `PCTFREE`. Pomysł z użyciem mniejszej wartości tego parametru polega na tym, że w bloku jest pozostawiane więcej miejsca do zapisu migrowanego/łańcuchowanego wiersza (ponieważ jest on przenoszony z bloku z większą wartością atrybutu `PCTFREE` do bloku z większą ilością miejsca dzięki mniejszej wartości tego atrybutu).

Załóżmy na przykład, że tabela została utworzona z wartością atrybutu `PCTFREE` równą 40%. Kolejna operacja przesunięcia spowoduje przebudowanie tabeli z wartością tego parametru równą 5%:

```
SQL> alter table pracownicy move pctfree 5;
```

Pamiętaj jednak, że jeżeli tak zrobisz, możesz spowodować jeszcze większy problem, ponieważ w bloku będzie mniej miejsca na przyszłe aktualizacje (co może skutkować większą liczbą migrowanych/łańcuchowanych wierszy).

■ **Uwaga**. Podczas przenoszenia tabeli baza Oracle wymaga zablokowania tabeli na wyłączność. Dlatego tę operację powinieneś wykonywać wtedy, gdy z przenoszoną tabelą nie są skojarzone żadne aktywne transakcje.

Ponadto instrukcja `MOVE` powoduje, że wierszom są przypisywane nowe identyfikatory `ROWID`. W ten sposób unieważniane są wszystkie indeksy związane z daną tabelą. Dlatego podczas przenoszenia tabeli musisz przebudować wszystkie związane z nią indeksy. Status indeksów możesz sprawdzić, odpytując widok `DBA/ALL/USER_INDEXES`:

```
select owner, index_name, status
from dba_indexes
where table_name='PRACOWNICY';
```

Przebudowa wszystkich indeksów sprawia, że mogą być znów wykorzystane:

```
SQL> alter index pracownicy_pk rebuild;
```

Możesz sprawdzić, czy problem z migracją/łańcuchowaniem wierszy został (lub nie został) rozwiązany, ręcznie uruchamiając narzędzie Segment Advisor (patrz porada 1.10) lub wykonując polecenie `ANALYZE TABLE ... COMPUTE STATISTICS` (więcej szczegółowych informacji znajdziesz w części „Jak to działa" niniejszej porady).

Przeniesienie poszczególnych migrowanych/łańcuchowanych wierszy

Za pomocą polecenia `ANALYZE TABLE ... LIST INTO` możesz wyświetlić identyfikatory `ROWID` zmigrowanych/łańcuchowanych wierszy. Najpierw musisz utworzyć tabelę do przechowania informacji zwróconych przez instrukcję `ANALYZE TABLE`. Oracle oferuje skrypt, który utworzy taką tabelę za Ciebie:

```
SQL> @?/rdbms/admin/utlchn1.sql
```

Powyższy skrypt tworzy tabelę o nazwie `CHAINED_ROWS`. Teraz możesz użyć instrukcji `ANALYZE`, aby wypełnić tabelę `CHAINED_ROWS` zmigrowanymi/łańcuchowanymi wierszami:

```
SQL> analyze table pracownicy list chained rows;
```

Następnie odczytaj liczbę wierszy w tabeli CHAINED_ROWS:

```
SQL> select count(*) from chained_rows where table_name='PRACOWNICY';
```

Zaletą wyszukiwania zmigrowanych/łańcuchowanych wierszy w powyższy sposób jest możliwość naprawienia takich wierszy bez naruszania pozostałych rekordów w tabeli. Wykonaj w tym celu następujące operacje:

1. Utwórz tymczasową tabelę przechowującą łańcuchowane wiersze.
2. Usuń zmigrowane/łańcuchowane wiersze z oryginalnej tabeli.
3. Wstaw wiersze z tymczasowej tabeli do oryginalnej tabeli.

Ponieważ powyższy sposób składa się z kilku kroków, dlatego zalecamy, abyś przetestował je najpierw w środowisku nieprodukcyjnym, a potem zastosował we właściwej bazie danych. Poniżej przedstawiony jest krótki przykład ilustrujący opisane wyżej kroki. Najpierw utwórz tymczasową tabelę zawierającą te wiersze z tabeli PRACOWNICY, dla których istnieją odpowiednie wiersze w tabeli CHAINED_ROWS:

```
create table pracownicy_tymcz as
  select *
  from pracownicy
  where rowid in
   (select head_rowid from chained_rows where table_name = 'PRACOWNICY');
```

Następnie usuń z tabeli PRACOWNICY te rekordy, dla których istnieją odpowiednie wiersze w tabeli CHAINED_ROWS:

```
delete from pracownicy
where rowid in
  (select head_rowid from chained_rows where table_name = 'PRACOWNICY');
```

Teraz umieść rekordy z tabeli CHAINED_ROWS w tabeli PRACOWNICY:

```
insert into pracownicy select * from pracownicy_tymcz;
```

Powyższy proces przenoszenia zmigrowanych/łańcuchowanych wierszy powinien rozwiązać problem migracji. Teraz możesz usunąć wiersze z tabeli CHAINED_ROWS, jak również tabelę tymczasową.

Jak możesz sprawdzić, czy zmigrowane/łańcuchowane wiersze zostały naprawione? Użyj ponownie polecenia ANALYZE TABLE ... LIST CHAINED ROWS. Jeżeli w tabeli CHAINED_ROWS pojawią się nowe wiersze, będzie to najprawdopodobniej oznaczać, że wiersze zostały złańcuchowane (ale nie zmigrowane) i mogą być naprawione przez przeniesienie tabeli i zmniejszenie wartości atrybutu PCTFREE albo przez przeniesienie tabeli do przestrzeni tabel zawierającej większe bloki (więcej szczegółowych informacji znajdziesz w części „Jak to działa").

Czym jest identyfikator ROWID?

Każdy wiersz w każdej tabeli posiada fizyczny adres. Adres wiersza jest tworzony jako kombinacja następujących informacji:

- numer pliku danych,
- numer bloku,
- pozycja wiersza w bloku,
- numer obiektu.

Możesz wyświetlić adresy wierszy w tabeli, odpytując pseudokolumnę ROWID, na przykład w taki sposób:

```
SQL> select rowid, id_pracownika from pracownicy;
```

Przykładowy wynik jest następujący:

```
ROWID              ID_PRACOWNIKA
------------------ -------------
AAAEtQAAEAAAACDAAA           100
AAAEtQAAEAAAACDAAB           101
```

Wartości pseudokolumny ROWID nie są fizycznie zapisywane w bazie. Baza Oracle generuje je podczas wykonywania zapytania. Wartości ROWID są wyświetlane w formacie Base64 i zawierają znaki A–Z, a–z, 0–9, + oraz /. Wartości te możesz przełożyć na zrozumiały format za pomocą pakietu DBMS_ROWID. Na przykład aby wyświetlić numer pliku, bloku i wiersza, wykonaj następujące zapytanie:

```
select
  id_pracownika,
  dbms_rowid.rowid_to_absolute_fno(rowid,schema_name=>'SERWIS',object_name=>'PRACOWNICY') file_num,
  dbms_rowid.rowid_block_number(rowid) block_num,
  dbms_rowid.rowid_row_number(rowid) row_num
from pracownicy;
```

Przykładowy wynik jest następujący:

```
ID_PRACOWNIKA   FILE_NUM  BLOCK_NUM    ROW_NUM
------------- ---------- ---------- ----------
          100          4        131          0
          101          4        131          1
```

Wartości ROWID możesz wykorzystać w klauzulach SELECT i WHERE w zapytaniach SQL. W większości przypadków wartość ROWID jednoznacznie identyfikuje wiersz. Jednak możliwy jest przypadek, że wiersze będą zapisane w tabelach umieszczonych w tym samym klastrze i będą miały te same identyfikatory ROWID (podobnie jest w przypadku tabeli klastrowej).

Jak to działa

Oracle definiuje **wiersz łańcuchowany** jako taki, który jest zbyt duży, aby go pomieścić w wolnym miejscu bloku, i dlatego w tym celu potrzebne są dwa lub więcej bloków. Łańcuchowanie wierszy może wystąpić w przypadku gdy:

- W tabeli został umieszczony wiersz zawierający wartości powodujące, że był zbyt długi, aby go pomieścić w którymś z dostępnych bloków. W przypadku pustego bloku jest to miejsce równe wielkości bloku pomniejszone o zarezerwowane miejsce określone atrybutem PCTFREE.
- W tabeli został umieszczony wiersz zawierający wartości na tyle małe, że można go było pomieścić w jednym bloku, ale został zaktualizowany wartościami powodującymi, że stał się zbyt długi, aby mógł się zmieścić w wolnym miejscu jednego bloku (atrybut PCTFREE określa procentową wielkość miejsca w bloku zarezerwowanego na aktualizacje danych).
- Tabela zawiera więcej niż 255 kolumn i wymaga do ich przechowania dwóch lub więcej bloków.

Baza Oracle obsługuje wskaźniki pomiędzy blokami zawierającymi łańcuchowane wiersze. Oznacza to, że za każdym razem, gdy wiersz jest odczytywany, wykonywanych jest kilka operacji wejścia-wyjścia. Również operacje aktualizacji i usuwania wierszy wymagają wykonania zapisów do wielu bloków. Duża liczba łańcuchowanych wierszy może negatywnie wpływać na wydajność bazy. Jeżeli więc wiersz jest zbyt długi, aby mógł się zmieścić w którymś z dostępnych bloków, wówczas jedynym sposobem rozwiązania problemu jest zmniejszenie długości wiersza lub zwiększenie rozmiaru bloku.

W przypadku tabel zawierających 255 lub mniej kolumn istnieje potencjalna możliwość rozwiązania problemu przez przeniesienie tabeli i jednoczesne zmniejszenie wartości atrybutu PCTFREE. Jeżeli to nie rozwiąże problemu, możesz utworzyć przestrzeń tabel z większymi blokami i przenieść do niej tabelę. Żadne inne rozwiązania nie pozwolą Ci uniknąć problemu łańcuchowania dużych wierszy.

Baza Oracle definiuje **zmigrowany wiersz** jako taki, który zawierał wartości na tyle małe, że został najpierw zapisany w wolnym miejscu określonego bloku. Wiersz ten zawierał kolumny, które zostały następnie zaktualizowane większymi wartościami (na przykład kolumnę zawierającą początkowo wartość NULL, do której została następnie wpisana jakaś określona wartość), co spowodowało, że wiersz nie zmieścił się w bieżącym bloku. Jednak wiersz był wciąż na tyle krótki, że mógł zmieścić się w wolnym miejscu innego bloku.

W takim przypadku baza Oracle przesunęła (zmigrowała) ten wiersz do innego bloku zawierającego odpowiednio dużo miejsca.

Podczas migracji baza Oracle umieszcza w oryginalnym bloku wskaźnik do bloku zawierającego zmigrowany wiersz. Migracja wierszy może pogarszać wydajność, ponieważ przy każdym odwołaniu do wiersza baza danych musi odczytywać/zapisywać dane z wielu bloków. Problem migracji może niemal w każdym przypadku być rozwiązany przez przeniesienie tabeli w sposób opisany w części „Rozwiązanie" niniejszej porady.

1.13. Wykrywanie łańcuchowania i migracji wierszy

Problem

Stwierdziłeś, że istnieje problem z wydajnością zapytań odczytujących dane z tabeli. Chcesz sprawdzić, czy potencjalną przyczyną jest migracja lub łańcuchowanie wierszy.

Rozwiązanie

Poniżej przedstawione są dostępne sposoby wykrywania migracji/łańcuchowania wierszy:

- Użycie narzędzia Segment Advisor.
- Wykonanie instrukcji `ANALYZE TABLE ... INTO CHAINED_ROWS`.
- Wykonanie instrukcji `ANALYZE TABLE ... COMPUTE STATISTICS`.
- Odpytanie widoku `V$SYSSTAT`.

Pierwsze dwa sposoby z powyższej listy zostały omówione w poradzie 1.12. W poniższych częściach opisane są dwie pozostałe metody.

Obliczanie statystyk

Dobrym sposobem wykrywania migracji/łańcuchowania wierszy jest użycie instrukcji `ANALYZE TABLE...COMPUTE STATISTICS`. Użycie jej w odniesieniu do określonej tabeli powoduje umieszczenie informacji w kolumnie `CHAINED_CNT` tabeli `DBA_TABLES`, na przykład:

```
SQL> analyze table pracownicy compute statistics;
```

Teraz uruchom poniższe zapytanie, które pokaże Ci odsetek łańcuchowanych wierszy tabeli:

```
select
  owner, chain_cnt,
  round(chain_cnt/num_rows*100,2) chain_pct,
  avg_row_len, pct_free
from dba_tables
where table_name = 'PRACOWNICY';
```

Kolumna `CHAINED_CNT` zawiera sumę zarówno zmigrowanych, jak i łańcuchowanych wierszy umieszczonych w tabeli. Jeżeli ich odsetek jest większy niż 15%, może to oznaczać potencjalny problem.

Odpytanie widoku V$SYSSTAT

Innym sposobem sprawdzenia, czy występuje problem z migracją wierszy, jest odpytanie widoku `V$SYSSTAT`. Po uruchomieniu instancji bazy danych przy każdorazowym odczycie zmigrowanego/łańcuchowanego wiersza zwiększana jest wartość statystyki `table fetch continued row`. Możesz ją odczytać w następujący sposób:

```
SQL> select name, value from v$sysstat where name = 'table fetch continued row';
```

Wartość sama w sobie nic nie mówi. Jeżeli zostanie zwrócona liczba 10 000, nie będzie wiadomo, czy oznacza ona jeden zmigrowany wiersz odczytany 10 000 razy, czy jest 10 000 tabel, w każdej po jednym zmigrowanym wierszu odczytanym tylko raz. Co więcej, bez porównania jej z całkowitą liczbą odczytów wierszy w całej bazie danych od chwili jej uruchomienia nie wiadomo, czy liczba ta oznacza problem, czy nie.

Aby mieć wyobrażenie, czy występuje problem z migracją/łańcuchowaniem wierszy, porównaj liczbę odczytów zmigrowanych/łańcuchowanych wierszy z całkowitą liczbą odczytów wszystkich wierszy w bazie danych:

```
with a as (select sum(value) total_rows_read
       from v$sysstat
       where name like '%table%'
       and name != 'table fetch continued row'),
     b as (select value total_mig_chain_rows
       from v$sysstat where name = 'table fetch continued row')
select a.total_rows_read, b.total_mig_chain_rows,
       b.total_mig_chain_rows/a.total_rows_read pct_rows_mig_or_chained
from a, b;
```

Jak to działa

Jednym z najlepszych sposobów na wykrycie łańcuchowania lub migracji wierszy jest sprawdzenie informacji zwracanych przez narzędzie Segment Advisor. Jeżeli nie posiadasz licencji, która pozwala użyć tego narzędzia, zastosuj inne metody uzyskania potrzebnych informacji, na przykład wykonanie instrukcji `ANALYZE TABLE` lub odpytanie widoku `V$SYSSTAT`.

Jeżeli chcesz przeanalizować wszystkie tabele w danym schemacie, możesz użyć poniższego zapytania SQL do wygenerowania innych zapytań SQL:

```
SQL> select 'analyze table ' || table_name || ' compute statistics;' from user_tables;
```

Powyższy skrypt generuje zapytania SQL dla wszystkich tabel aktualnie podłączonego użytkownika. Jeżeli masz duże tabele, pamiętaj, że ich analiza może zająć trochę czasu.

Drugi sposób opisany w części „Rozwiązanie” polega na odpytaniu widoku `V$SYSSTAT`. Umożliwi on bardziej dynamiczny wgląd w Twoją bazę danych. Na przykład po sprawdzeniu całkowitej liczby łańcuchowanych wierszy nie zawsze będziesz wiedział, czy należą one do aktywnie odpytywanej części tabeli. Odpytanie widoku `V$SYSSTAT` jest pomocne w tej sytuacji, ponieważ znajdują się w nim informacje o danych aktualnie przetwarzanych przez transakcje.

1.14. Odróżnienie migracji od łańcuchowania wierszy

Problem

Chcesz określić, czy w bazie ma miejsce migracja, czy łańcuchowanie wierszy. Ta informacja decyduje o strategii rozwiązania problemu. Na przykład jeżeli problemem jest łańcuchowanie wierszy spowodowane bardzo długimi rekordami zapisywanymi w wolnym miejscu bloku, wówczas niewiele można zrobić. Jednak problem z migracją wierszy może być rozwiązany poprzez przeniesienie tabeli.

Rozwiązanie

Są trzy podstawowe sposoby odróżnienia łańcuchowania wierszy od migracji:

- Reorganizacja tabeli (na przykład jej przeniesienie) zawsze rozwiązuje problem z migracją wierszy, ale może nie rozwiązać problemu z ich łańcuchowaniem. Dlatego rozróżnij migrację wierszy od łańcuchowania, zreorganizuj tabelę i po jej przeniesieniu oblicz statystyki. Jeżeli w kolumnie `CHAIN_CNT` pojawi się jakaś liczba, będzie ona oznaczać łańcuchowane wiersze, których nie można naprawić

przez przeniesienie tabeli. W takim wypadku rozważ nadanie mniejszej wartości atrybutowi PCTFREE podczas operacji przenoszenia tabeli.

- Przeanalizowanie tabeli i zapisanie identyfikatorów ROWID w tabeli CHAINED_ROWS, a następnie przeniesienie wybranych zmigrowanych/łańcuchowanych wierszy na podstawie ich identyfikatorów ROWID. Powtórz ten proces i jeżeli w tabeli CHAINED_ROWS znajdą się nowe wiersze, będą to wiersze łańcuchowane.
- Oblicz długość każdego wiersza. Jeżeli będzie większa niż wolne miejsce dostępne w pustym bloku, to prawdopodobnie problem dotyczy łańcuchowania wierszy. Jeżeli długość wiersza będzie mniejsza, wówczas najprawdopodobniej będzie oznaczać problem z migracją.

Pierwsze dwa sposoby zostały omówione w poradzie 1.12. W trzecim sposobie możesz ręcznie obliczyć długość wiersza, sumując długości wszystkich kolumn, na przykład:

```
SELECT NVL(vsize(id_pracownika),0) + NVL(vsize(imie),0) + NVL(vsize(nazwisko),0)
FROM pracownicy;
```

Wiersze, których długość jest większa niż dostępne wolne miejsce w pustym bloku, są najprawdopodobniej łańcuchowane, ale nie zmigrowane.

Jak to działa

Ręczne obliczanie długości wierszy jest jak na razie najdokładniejszym sposobem określania, czy problem dotyczy łańcuchowania, czy migracji wierszy. Jeżeli długość wiersza osiąga ilość wolnego miejsca w bloku, wówczas uniknięcie jego łańcuchowania jest prawie niemożliwe. Jeżeli długość wiersza jest znacznie mniejsza, wówczas najprawdopodobniej jest on zmigrowany i ten problem może być rozwiązany przez przeniesienie tabeli (patrz porada 1.12).

Wskazówka. Aby zmaksymalizować liczbę wierszy w bloku, rozważ użycie kompresji danych, dostępnej w bazie Oracle (szczegółowe informacje zawarte są w poradach 1.20 i 1.21).

1.15. Proaktywne zapobieganie migracji/łańcuchowaniu wierszy

Problem

Zauważyłeś, że w tabeli często ma miejsce migracja/łańcuchowanie wierszy. Zamierzasz proaktywnie zapobiec temu problemowi.

Rozwiązanie

Ilość zarezerwowanego wolnego miejsca w bloku jest określona za pomocą atrybutu PCTFREE tabeli. Jego domyślna wartość jest równa 10, co oznacza, że 10% miejsca w bloku jest zarezerwowane na aktualizacje danych, które skutkują zajęciem większego miejsca w bieżącym bloku wiersza. Jeżeli masz tabele, w których w kolumnach zostały umieszczone małe ilości danych lub wartości NULL, a później zostały one zaktualizowane dużymi wartościami, wówczas rozważ ustawienie większej wartości atrybutu PCTFREE, na przykład 40%. W ten sposób zapobiegniesz migracji wierszy. Pamiętaj jednak, że większe wartości atrybutu PCTFREE mogą skutkować zwiększoną liczbą łańcuchowanych wierszy.

Pomocne może być wyliczenie średniej długości wiersza po umieszczeniu danych, a następnie zaktualizowanie go i ponowne obliczenie jego długości. W ten sposób będziesz miał rozeznanie, jak bardzo wiersz się wydłużył (sposób obliczania długości wiersza został szczegółowo opisany w poradzie 1.14).

Jeżeli masz tabelę, w której wiersze nigdy nie zostały zaktualizowane po ich umieszczeniu, rozważ ustawienie atrybutu PCTFREE na wartość 0. W ten sposób zmaksymalizujesz liczbę wierszy w bloku, dzięki czemu będzie mniej operacji odczytu danych z dysku (i zwiększy się wydajność bazy) podczas odczytu danych z tabeli.

Jak to działa

Wartość atrybutu PCTFREE możesz sprawdzić, odpytując widok DBA/ALL/USER_TABLES, na przykład:

```
SQL> select table_name, pct_free from user_tables;
```

Przeniesienie tabeli niemal zawsze rozwiązuje problem migracji wierszy. Operacja przeniesienia usuwa każdy rekord z bloku i ponownie umieszcza go w nowym bloku. W przypadku migracji stare wiersze są usuwane i tworzone jako pojedyncze wiersze w nowym bloku.

Łańcuchowane wiersze mogą być naprawione tylko wtedy, gdy długość łańcuchowanego wiersza jest mniejsza niż ilość wolnego miejsca w pustym bloku. Jeżeli łańcuchowany wiersz jest dłuższy niż ilość wolnego miejsca, rozważ nadanie atrybutowi PCTFREE mniejszej wartości i przeniesienie tabeli albo utworzenie przestrzeni tabel z większymi blokami.

1.16. Wykrywanie niewykorzystanego miejsca w tabeli

Problem

Odpytujesz tabelę, w której nie ma wierszy, a mimo to zwrócenie przez zapytanie ich zerowej liczby zajmuje kilka minut. Wiesz z doświadczenia, że może to oznaczać problem z niewykorzystanym miejscem w tabeli. Przypuszczasz, że tabela na początku zawierała dużo wierszy, ale stopniowo były usuwane i w ten sposób pozostało niewykorzystane miejsce poniżej tak zwanego wskaźnika zajętości miejsca. Operacje takie jak pełne skanowanie tabeli mogą długo trwać, ponieważ baza Oracle przeszukuje bloki, które początkowo zawierały dane, ale teraz już ich nie zawierają. Dlatego chcesz sprawdzić, czy w tabeli nie ma niewykorzystanego miejsca.

Rozwiązanie

Tabele, w których występuje problem z niewykorzystanym miejscem, możesz wykryć, odpytując widok DBA/ALL/USER_EXTENTS. Jeżeli jakaś tabela zajmuje dużą liczbę rozszerzeń, ale nie zawiera żadnych wierszy, oznacza to, że zostało z niej usuniętych bardzo dużo danych. Na przykład:

```
SQL> select count(*) from user_extents where segment_name='PRACOWNICY';
  COUNT(*)
----------
        44
```

Następnie sprawdź liczbę wierszy w tabeli:

```
SQL> select count(*) from pracownicy;
  COUNT(*)
----------
         0
```

W powyższej tabeli prawdopodobnie zostały umieszczone wiersze, co spowodowało przydzielenie rozszerzeń. Następnie dane zostały usunięte, natomiast rozszerzenia pozostały.

Analogicznie możesz sprawdzić liczbę wierszy i porównać ją z liczbą bloków przydzielonych tabeli, na przykład:

```
SQL> select blocks from user_segments where segment_name='PRACOWNICY';
    BLOCKS
----------
      1024
```

Jeżeli liczba bloków jest duża, ale liczba wierszy mała, wówczas najprawdopodobniej poniżej wskaźnika zajętości znajduje się wolne miejsce.

Jak to działa

Oracle definiuje wskaźnik zajętości tabeli jako granicę pomiędzy używanym i nieużywanym miejscem w segmencie. Podczas tworzenia tabeli baza Oracle przydziela jej pewną liczbę rozszerzeń, określoną przez atrybut `MINEXTENTS`. Każde rozszerzenie zawiera pewną liczbę bloków. Zanim w tabeli zostaną umieszczone dane, żaden blok nie jest używany, a wskaźnik zajętości jest ustawiony na `0`. W miarę umieszczania danych w tabeli i przydzielania rozszerzeń przesuwany jest wskaźnik zajętości. Instrukcja `DELETE` nie resetuje wskaźnika, dlatego w tabeli powstaje niewykorzystane miejsce.

Musisz pamiętać o kilku zagadnieniach związanych z wydajnością i wskaźnikiem zajętości:

- o pełnym skanowaniu tabel przez zapytania SQL;
- o ładowaniu danych z użyciem bezpośredniej ścieżki dostępu do pliku.

Niekiedy podczas wykonywania zapytania baza Oracle musi przejrzeć każdy blok tabeli (poniżej wskaźnika zajętości). Operacja ta jest nazywana pełnym skanowaniem. Jeżeli z tabeli została usunięta znaczna ilość danych, wówczas wykonanie pełnego skanowania zajmuje dużo czasu, nawet jeżeli tabela nie zawiera żadnych wierszy.

Natomiast podczas ładowania danych z użyciem bezpośredniej ścieżki do pliku baza Oracle umieszcza dane powyżej wskaźnika zajętości. W tabeli, z której regularnie usuwane są dane i umieszczane są w niej nowe za pomocą bezpośredniej ścieżki, może powstać dużo niewykorzystanego miejsca.

Oprócz sposobu opisanego w części „Rozwiązanie" jest kilka innych metod wykrywania niewykorzystanego miejsca znajdującego się poniżej wskaźnika zajętości:

- narzędzie Autotrace,
- pakiet `DBMS_SPACE`,
- narzędzie Segment Advisor.

Szczegółowe informacje dotyczące śledzenia niewykorzystanego miejsca są opisane w poradzie 1.17, natomiast użycie pakietu `DBMS_SPACE` — w poradzie 1.18. W poradzie 1.10 zawarte są szczegóły dotyczące ręcznego uruchamiania narzędzia Segment Advisor.

1.17. Śledzenie i wykrywanie miejsca poniżej wskaźnika zajętości

Problem

Wykonałeś kroki opisane w poradzie 1.16 i podejrzewasz, że pojawił się problem z niewykorzystanym miejscem poniżej wskaźnika zajętości tabeli. Chcesz dokładniej zweryfikować wynik i sprawdzić informacje zwracane przez narzędzie Autotrace.

Rozwiązanie

Aby sprawdzić, czy występuje problem z niewykorzystanym miejscem poniżej wskaźnika zajętości tabeli, możesz wykonać prosty test:

1. `SQL> set autotrace trace statistics.`
2. Uruchom zapytanie wykonujące pełne skanowanie.
3. Porównaj liczbę przetworzonych wierszy z liczbą bloków odczytanych z pamięci.

Jeżeli liczba przetworzonych wierszy jest mała, a liczba bloków odczytanych z pamięci duża, może to oznaczać problem z dużą ilością wolnych bloków poniżej wskaźnika zajętości. Poniżej przedstawiony jest prosty przykład ilustrujący ten sposób:

```
SQL> set autotrace trace statistics
```

Poniższe zapytanie wykonuje pełne skanowanie tabeli `INV`:

```
SQL> select * from pracownicy;
```

Poniżej przedstawiony jest fragment informacji zwracanych przez narzędzie Autotrace:

```
no rows selected                  // brak zwróconych wierszy
Statistics                        // statystyka
----------------------------------------------------------
          4  recursive calls      // 4 wywołania rekursywne
          0  db block gets        // 0 pobrań bloków
       7371  consistent gets      // 7371 spójnych pobrań
```

Liczba zwróconych wierszy jest równa zero, natomiast miało miejsce 7371 spójnych pobrań (bloków odczytanych z bufora podręcznego), co oznacza, że poniżej wskaźnika zajętości jest niewykorzystane miejsce.

Następnie usuń dane z tabeli i ponownie uruchom zapytanie:

```
SQL> truncate table pracownicy;
SQL> select * from pracownicy;
```

Fragment informacji zwróconych przez Autotrace jest następujący:

```
no rows selected
Statistics
----------------------------------------------------------
          6  recursive calls
          0  db block gets
         12  consistent gets
```

Zwróć uwagę, że tym razem liczba odczytów z pamięci jest bardzo mała.

Jak to działa

Każdy użytkownik, któremu została przypisana rola `PLUSTRACE`, może używać narzędzia Autotrace dostępnego za pomocą aplikacji SQL*Plus. Dostarcza ono szczegółowych informacji dotyczących planów wykonania oraz statystyk pomyślnie zrealizowanych instrukcji `SELECT`, `INSERT`, `UPDATE` i `DELETE`. Statystyki są zapisywane wewnątrz bazy Oracle i zawierają szczegółowe informacje o zasobach systemu wykorzystywanych przez zapytanie SQL. Liczba spójnych pobrań informuje, ile bloków zostało odczytanych z pamięci w przypadku danego zapytania SQL. Jeżeli ta liczba jest duża, a liczba wierszy mała, wówczas najprawdopodobniej oznacza to problem z wolnym miejscem poniżej wskaźnika zajętości.

1.18. Zastosowanie pakietu DBMS_SPACE do wykrywania wolnego miejsca poniżej wskaźnika zajętości

Problem

Stosując metodę śledzenia (porada 1.17), sprawdziłeś, że poniżej wskaźnika zajętości jest wolne miejsce. Chcesz potwierdzić wynik za pomocą pakietu DBMS_SPACE.

Rozwiązanie

Poniżej przedstawiony jest anonimowy kod PL/SQL, wykrywający za pomocą pakietu DBMS_SPACE wolne miejsce poniżej wskaźnika zajętości. Kod możesz uruchomić w aplikacji SQL*Plus:

```
set serverout on size 1000000
declare
  p_fs1_bytes number;
  p_fs2_bytes number;
  p_fs3_bytes number;
  p_fs4_bytes number;
  p_fs1_blocks number;
  p_fs2_blocks number;
  p_fs3_blocks number;
  p_fs4_blocks number;
  p_full_bytes number;
  p_full_blocks number;
  p_unformatted_bytes number;
  p_unformatted_blocks number;
begin
  dbms_space.space_usage(
    segment_owner => user,
    segment_name => 'PRACOWNICY',
    segment_type => 'TABLE',
    fs1_bytes => p_fs1_bytes,
    fs1_blocks => p_fs1_blocks,
    fs2_bytes => p_fs2_bytes,
    fs2_blocks => p_fs2_blocks,
    fs3_bytes => p_fs3_bytes,
    fs3_blocks => p_fs3_blocks,
    fs4_bytes => p_fs4_bytes,
    fs4_blocks => p_fs4_blocks,
    full_bytes => p_full_bytes,
    full_blocks => p_full_blocks,
    unformatted_blocks => p_unformatted_blocks,
    unformatted_bytes => p_unformatted_bytes
  );
  dbms_output.put_line('FS1: liczba bloków = '||p_fs1_blocks);
  dbms_output.put_line('FS2: liczba bloków = '||p_fs2_blocks);
  dbms_output.put_line('FS3: liczba bloków = '||p_fs3_blocks);
  dbms_output.put_line('FS4: liczba bloków = '||p_fs4_blocks);
  dbms_output.put_line('Pełne bloki        = '||p_full_blocks);
end;
/
```

W tym przypadku sprawdzasz wolne miejsce poniżej wskaźnika zajętości w tabeli PRACOWNICY. Poniżej przedstawione są informacje zwracane przez powyższy kod PL/SQL:

```
FS1: liczba bloków = 0
FS2: liczba bloków = 0
FS3: liczba bloków = 0
FS4: liczba bloków = 3646
Pełne bloki        = 0
```

W powyższym wyniku wartość parametru FS1 wskazuje, że żaden blok nie ma wolnego miejsca zajmującego od 0% do 25% całości. Wartość parametru FS2 wskazuje, że żaden blok nie ma wolnego miejsca zajmującego od 25% do 50% całości. Wartość parametru FS3 wskazuje, że żaden blok nie ma wolnego miejsca zajmującego od 50% do 75% całości. Wartość parametru FS4 wskazuje, że jest 3646 bloków, które mają od 50% do 75% wolnego miejsca. I wreszcie nie ma żadnych pełnych bloków. Ponieważ nie ma pełnych bloków, a dużo jest bloków prawie pustych, więc można wysnuć wniosek, że poniżej wskaźnika zajętości jest wolne miejsce.

Jak to działa

Procedura SPACE_USAGE zawarta w pakiecie DBMS_SPACE oferuje alternatywny sposób sprawdzania, czy jest wolne miejsce poniżej wskaźnika zajętości. Procedura ta może być użyta tylko w przypadku tabel w przestrzeniach utworzonych za pomocą funkcjonalności Automatic Space Segment Management (szczegółowe informacje znajdują się w poradzie 1.2). Więcej szczegółowych informacji na temat użycia tej procedury jest zawartych w podręczniku *Oracle PL/SQL Packages and Types Reference Guide*.

1.19. Zwalnianie niewykorzystanego miejsca w tabelach

Problem

Przeanalizowałeś informacje zwrócone przez narzędzie Segment Advisor i znalazłeś tabelę, w której jest dużo wolnego miejsca poniżej wskaźnika zajętości. Chcesz je zwolnić, aby poprawić wydajność zapytań wykonujących pełne skanowanie tej tabeli.

Rozwiązanie

Aby zmniejszyć rozmiar tabeli i ponownie ustawić wskaźnik zajętości, wykonaj następujące operacje:

1. Włącz funkcjonalność przenoszenia wierszy tabeli.
2. Wykonaj instrukcję ALTER TABLE...SHRINK SPACE, aby zwolnić niewykorzystane miejsce.

■ **Uwaga**. Funkcjonalność zmniejszania tabeli wymaga, aby w przestrzeni tabel była włączona funkcjonalność automatycznego zarządzania przestrzenią segmentów (ASSM). W poradzie 1.2 znajdują się szczegółowe informacje na temat tworzenia przestrzeni z włączoną funkcjonalnością ASSM.

Operacja zmniejszania tabeli wymaga przenoszenia wierszy (jeżeli takie będą), a to z kolei wymaga włączenia odpowiedniej funkcjonalności:

```
SQL> alter table pracownicy enable row movement;
```

Teraz może być wykonana operacja zmniejszenia tabeli za pomocą instrukcji ALTER TABLE:

```
SQL> alter table pracownicy shrink space;
```

Za pomocą klauzuli CASCADE możesz również zmniejszyć miejsce przypisane do segmentów indeksów:

```
SQL> alter table pracownicy shrink space cascade;
```

Jak to działa

Podczas zmniejszania tabeli baza Oracle przestawia bloki w taki sposób, aby zajmowały jak najmniej miejsca. Ustawia również na nowo wskaźnik zajętości. Ta operacja ma wpływ na wydajność zapytań wykonujących pełne skanowanie tabeli. Podczas takiego skanowania baza Oracle przegląda każdy blok poniżej wskaźnika zajętości i sprawdza, czy są w nim dane spełniające zapytanie. Jeżeli zauważysz, że zwrócenie wyniku przez zapytanie zajmuje dużo czasu, a w tabeli jest niewiele wierszy, może to być sygnałem, że poniżej wskaźnika zajętości znajduje się dużo niewykorzystanych bloków (z powodu usunięcia danych).

Możesz również zlecić bazie, aby podczas zmniejszania tabeli nie ustawiała na nowo wskaźnika zajętości. Osiąga się to za pomocą klauzuli COMPACT, na przykład:

```
SQL> alter table pracownicy shrink space compact;
```

Gdy zostanie użyta klauzula COMPACT, baza Oracle będzie defragmentować tabelę, ale nie zmieni wskaźnika zajętości. Aby go zmienić, będziesz musiał użyć instrukcji ALTER TABLE...SHRINK SPACE. Możesz tak zrobić ze względu na długi czas wykonywania defragmentacji i ustawiania wskaźnika. W ten sposób można zmniejszyć tabelę za pomocą dwóch krótszych kroków zamiast jednego długiego.

Włączenie przenoszenia wierszy tabeli umożliwia bazie Oracle modyfikowanie identyfikatorów ROWID wszystkich wierszy, które muszą być przeniesione. Oznacza to, że baza Oracle musi również zmienić indeksy odwołujące się do tych identyfikatorów. Dlatego pamiętaj, że wydajność operacji zmniejszania tabeli zależy od liczby wierszy, które muszą być przeniesione, i indeksów, które muszą być uaktualnione.

Oprócz metody opisanej w części „Rozwiązanie" są jeszcze inne sposoby, które możesz wykorzystać do zwolnienia miejsca poniżej wskaźnika zajętości:

- przycięcie tabeli,
- przeniesienie tabeli,
- użycie narzędzia Data Pump do eksportu tabeli, usunięcia jej i ponownego importu.

Oczywiście instrukcji TRUNCATE używaj tylko wtedy, gdy chcesz trwale usunąć wszystkie dane z tabeli. Przycięcie tabeli może być dopuszczalne w przypadku, gdy nie ma w niej wierszy.

Wskaźnik zajętości możesz również obniżyć, przenosząc tabelę, na przykład:

```
SQL> alter table pracownicy move;
```

Przenosząc tabelę, musisz również przebudować indeksy, ponieważ operacja przenoszenia zmienia identyfikatory ROWID wierszy tabeli i unieważnia wszystkie indeksy.

Narzędzie Data Pump umożliwia eksport tabel, następnie ich usunięcie lub zmianę ich nazw i ponowny import z plików. Szczegółowe informacje na temat użycia narzędzia Data Pump można znaleźć w książce *Pro Oracle Database 12c Administration* (Apress) i podręczniku *Oracle's Utility Guide* dostępnym na stronie internetowej Oracle Technology Network.

1.20. Kompresja danych podczas ładowania za pomocą bezpośredniej ścieżki

Problem

Używasz bazy danych Oracle Enterprise Edition i obsługujesz system wspomagania decyzji. Chcesz poprawić wydajność związanej z nim aplikacji raportującej. Środowisko zawiera duże tabele, do których zostały kiedyś załadowane dane i które są często poddawane pełnemu skanowaniu. Chcesz skompresować załadowane dane,

ponieważ dzięki temu będą zajmować mniej bloków i wymagać wykonania mniejszej liczby operacji odczytu danych z dysku. Ponieważ skompresowane dane będą odczytywane z mniejszej liczby bloków, dlatego zwiększona zostanie wydajność pobierania danych.

Rozwiązanie

Do skompresowania danych załadowanych do tabeli stertowej z wykorzystaniem bezpośredniej ścieżki wykorzystaj podstawową funkcjonalność kompresji, którą włącza się w następujący sposób:

1. Zastosuj klauzulę `ROW STORE COMPRESS`, aby włączyć kompresję podczas tworzenia, zmieniania i przenoszenia istniejącej tabeli.
2. Załaduj dane z wykorzystaniem mechanizmu bezpośredniej ścieżki dostępu, na przykład `CREATE TABLE...AS SELECT` lub `INSERT /*+ APPEND */`.

■ **Uwaga**. W wersjach bazy wcześniejszych niż 11g R2 podstawowa kompresja nosi nazwę kompresji DSS i jest włączana za pomocą klauzuli `COMPRESS FOR DIRECT_LOAD OPERATION`. W wersji 11g R2 klauzula została zmieniona na `COMPRESS BASIC`. W wersji 12c została ponownie zmieniona na `ROW STORE COMPRESS`.

Poniżej przedstawiony jest przykład wykorzystujący instrukcję `CREATE TABLE...AS SELECT` do utworzenia tabeli z włączoną podstawową kompresją i bezpośrednim ładowaniem danych:

```
create table pracownicy_dss
compress
as
  select id_pracownika, nazwisko
  from pracownicy;
```

Powyższa instrukcja tworzy tabelę zawierającą skompresowane dane. Każda kolejna operacja bezpośredniego ładowania danych spowoduje zapisanie ich w skompresowanym formacie.

■ **Wskazówka**. Poniższe klauzule mają takie samo znaczenie: `COMPRESS`, `COMPRESS BASIC`, `ROW STORE COMPRESS` oraz `ROW STORE COMPRESS BASIC`.

Odpytując odpowiedni widok `DBA/ALL/USER_TABLES`, możesz sprawdzić, czy dla danej tabeli została włączona kompresja danych. Poniższy przykład zakłada, że jesteś podłączony do bazy jako właściciel tabeli:

```
select table_name, compression, compress_for
from user_tables
where table_name='PRACOWNICY_DSS';
```

Przykładowy wynik jest następujący:

```
TABLE_NAME                     COMPRESS COMPRESS_FOR
------------------------------ -------- ------------
PRACOWNICY_DSS                 ENABLED  BASIC
```

Powyższy wynik pokazuje, że dla tej tabeli została włączona kompresja w podstawowym trybie. Jeżeli pracujesz nad tabelą, która została już utworzona, możesz zmienić jej podstawowy sposób kompresji za pomocą instrukcji `ALTER TABLE`, na przykład:

```
SQL> alter table pracownicy_dss compress;
```

Zmiana właściwości tabeli w celu włączenia podstawowej kompresji nie wpływa na znajdujące się w niej dane, natomiast kompresowane będą tylko dane umieszczane w następnych operacjach bezpośredniego ładowania.

Jeżeli chcesz włączyć podstawową kompresję danych w istniejącej tabeli, użyj klauzuli `MOVE COMPRESS`:

```
SQL> alter table pracownicy_dss move compress;
```

Pamiętaj, że podczas przenoszenia tabeli unieważniane są wszystkie związane z nią indeksy, które będziesz musiał przebudować.

Jeżeli w tabeli włączyłeś podstawową kompresję danych, możesz ją wyłączyć za pomocą klauzuli NOCOMPRESS, na przykład:

```
SQL> alter table pracownicy_dss nocompress;
```

Jeżeli zmienisz właściwości tabeli, wyłączając kompresję, to istniejące dane nie zostaną rozpakowane. Ta operacja jedynie instruuje bazę Oracle, aby nie kompresowała danych podczas kolejnych operacji ich bezpośredniego ładowania. Jeżeli chcesz rozpakować istniejące dane, użyj klauzuli MOVE NOCOMPRESS:

```
SQL> alter table pracownicy_dss move nocompress;
```

Jak to działa

Funkcjonalność podstawowej kompresji danych jest dostępna w wersji bazy Oracle Enterprise Edition bez dodatkowych opłat. Do każdej tabeli stertowej utworzonej lub zmienionej tak, aby wykorzystywała podstawową kompresję, dane ładowane z wykorzystaniem bezpośredniej ścieżki będą zapisywane w skompresowanym formacie. Z kompresją danych wiąże się dodatkowe obciążenie procesora, ale w niektórych przypadkach możesz się przekonać, że to dodatkowe obciążenie jest kompensowane większą wydajnością bazy dzięki mniejszej liczbie wykonywanych operacji wejścia-wyjścia.

Z punktu widzenia wydajności główna zaleta kompresji polega na tym, że po załadowaniu skompresowanych danych kolejne operacje wejścia-wyjścia wykorzystują mniej zasobów, ponieważ do odczytu lub zapisu danych potrzeba mniej bloków. Sprawdź zysk na wydajności w swoim środowisku testowym. Na ogół tabele przechowujące duże ilości danych znakowych nadają się do kompresji, szczególnie w przypadkach, w których dane zostały bezpośrednio raz załadowane i później wielokrotnie odczytywane.

Pamiętaj, że funkcjonalność podstawowej kompresji w bazie Oracle nie dotyczy zwykłych instrukcji DML, na przykład INSERT, UPDATE, MERGE i DELETE. Jeżeli chcesz, aby kompresja była stosowana w zwykłych instrukcjach DML, rozważ użycie zaawansowanej kompresji danych (szczegółowe informacje znajdują się w poradzie 1.21).

Podstawową kompresję danych możesz również określić na poziomie partycji i przestrzeni tabel. Każda tabela utworzona z użyciem klauzuli COMPRESS będzie miała domyślnie włączoną podstawową kompresję. Poniżej przedstawiony jest przykład tworzenia przestrzeni tabel z użyciem klauzuli COMPRESS:

```
CREATE TABLESPACE dane_skompr
  DATAFILE '/u01/dbfile/O12C/dane_skompr01.dbf'
  SIZE 500M
  EXTENT MANAGEMENT LOCAL
  UNIFORM SIZE 512K
  SEGMENT SPACE MANAGEMENT AUTO
  DEFAULT COMPRESS;
```

Możesz również zmienić istniejącą przestrzeń, aby ustawić domyślny stopień kompresji:

```
SQL> alter tablespace dane_skompr default compress;
```

Aby sprawdzić, czy w przestrzeni tabel jest włączona podstawowa kompresja danych, uruchom poniższą instrukcję:

```
select tablespace_name, def_tab_compression, compress_for
from dba_tablespaces
where tablespace_name = 'DANE_SKOMPR';
```

Wskazówka. Nie możesz usunąć kolumny z tabeli utworzonej z włączoną podstawową kompresją. Możesz jednak zaznaczyć kolumnę jako nieużywaną.

1.21. Kompresja danych dla wszystkich instrukcji DML

Problem

Pracujesz w środowisku OLTP i zauważyłeś bardzo dużą liczbę operacji wejścia-wyjścia wykonywanych podczas odczytu danych z tabeli. Chcesz wiedzieć, czy możesz zwiększyć wydajność tych operacji za pomocą kompresji danych w tabeli. Rzecz w tym, że skompresowane dane zajmują fizycznie mniej miejsca na dysku i do ich odczytania wymagana jest mniejsza liczba operacji wejścia-wyjścia.

■ **Uwaga**. Funkcjonalność zaawansowanej kompresji danych wymaga wersji bazy Oracle Enterprise Edition i opcji *Advanced Compression* (dodatkowe licencje).

Rozwiązanie

Aby włączyć kompresję danych dla zwykłych instrukcji DML, podczas tworzenia tabeli użyj klauzuli `ROW STORE COMPRESS ADVANCED`. W poniższym przykładzie tworzona jest tabela z włączoną zaawansowaną kompresją:

```
create table regiony(
  id_regionu number,
  nazwa varchar2(2000)
) row store compress advanced;
```

■ **Uwaga**. W wersjach bazy wcześniejszych niż 11g R2 zaawansowaną kompresję danych włącza się za pomocą klauzuli `COMPRESS FOR ALL OPERATIONS`. W wersji 11g R2 włącza się ją klauzulą `COMPRESS FOR OLTP`, natomiast w wersji 12c klauzula została zmieniona na `ROW STORE COMPRESS ADVANCED`.

Aby sprawdzić, czy włączona została kompresja, możesz odpytać odpowiedni widok `DBA/ALL/USER_TABLES`. Poniższy przykład zakłada, że jesteś połączony z bazą jako właściciel tabeli:

```
select table_name, compression, compress_for
from user_tables
where table_name='REGIONY';
```

Aby włączyć kompresję dla istniejącej tabeli, możesz użyć instrukcji `ALTER TABLE`, na przykład:

```
SQL> alter table regiony row store compress advanced;
```

Zmiana trybu kompresji tabeli nie wpływa na dane, które już się w niej znajdują. Kolejne instrukcje DML będą powodowały zapisywanie danych w kompresowanym formacie.

Jeżeli chcesz włączyć zaawansowaną kompresję danych w istniejącej tabeli, użyj klauzuli `MOVE ROW STORE COMPRESS ADVANCED`:

```
SQL> alter table regiony move row store compress advanced;
```

Pamiętaj, że podczas przenoszenia tabeli unieważniane są wszystkie skojarzone z nią indeksy, które będziesz musiał przebudować.

Jeżeli włączyłeś w tabeli zaawansowaną kompresję danych, możesz ją wyłączyć za pomocą klauzuli `NOCOMPRESS`, na przykład:

```
SQL> alter table regiony nocompress;
```

Zmiana właściwości tabeli i wyłączenie zaawansowanej kompresji nie powoduje rozpakowania istniejących danych. Operacja ta zleca jedynie bazie Oracle, aby nie kompresowała danych podczas kolejnych operacji DML.

Jak to działa

Klauzula `ROW STORE COMPRESS ADVANCED` włącza kompresję dla wszystkich operacji DML. Włączenie tej funkcjonalności nie powoduje natychmiastowej kompresji danych podczas ich umieszczania lub aktualizowania w tabeli. Kompresja jest wykonywana w trybie wsadowym w momencie, gdy stopień zmian w bloku przekroczy zadany próg. Gdy próg zostanie osiągnięty, kompresowane są jednocześnie wszystkie nieskompresowane wiersze. Próg, po którego przekroczeniu wykonywana jest kompresja, jest określany przez wewnętrzny algorytm (nad którym nie masz kontroli).

Zaawansowaną kompresję danych możesz również skonfigurować na poziomie przestrzeni tabel. Każda tabela utworzona w przestrzeni z włączoną zaawansowaną kompresją będzie domyślnie dziedziczyć to ustawienie. Poniżej przedstawiony jest przykład skryptu tworzącego przestrzeń tabel i konfigurującego zaawansowaną kompresję:

```
CREATE TABLESPACE dane_skompr
  DATAFILE '/u01/dbfile/O12C/dane_skompr01.dbf'
  SIZE 500M
  EXTENT MANAGEMENT LOCAL
  UNIFORM SIZE 1M
  SEGMENT SPACE MANAGEMENT AUTO
  DEFAULT ROW STORE COMPRESS ADVANCED;
```

Możesz również zmienić istniejącą przestrzeń tabel i ustawić domyślny stopień kompresji:

```
SQL> alter tablespace dane_skompr default row store compress advanced;
```

Domyślną konfigurację kompresji możesz sprawdzić za pomocą następującego zapytania:

```
select tablespace_name, def_tab_compression, compress_for
from dba_tablespaces
where tablespace_name = 'DANE_SKOMPR';
```

1.22. Kompresja danych na poziomie kolumny

Problem

Korzystasz z maszyny Oracle Exadata i chcesz skutecznie kompresować dane. Stwierdziłeś, że kompresja znacznie poprawia efektywność operacji wejścia-wyjścia, szczególnie podczas odczytywania danych z dysku. Jest tak, ponieważ dzięki kompresji danych podczas wykonywania zapytań `SELECT` odczytywanych jest znacznie mniej bloków.

Rozwiązanie

Aby podczas tworzenia tabeli włączyć kompresję na poziomie kolumny (Oracle nazywa tę funkcjonalność hybrydową kompresją kolumnową), użyj klauzuli `COLUMN STORE COMPRESS FOR QUERY` lub `COLUMN STORE COMPRESS FOR ARCHIVE`, na przykład:

```
create table pracownicy(
  id_pracownika number,
  imie varchar2(30),
  nazwisko varchar2(30)
)
column store compress for query;
```

Możesz również określić niski (`LOW`) lub wysoki (`HIGH`) stopień kompresji:

```
create table pracownicy(
  id_pracownika number,
```

```
  imie varchar2(30),
  nazwisko varchar2(30)
)
column store compress for query high;
```

Domyślny poziom kompresji dla operacji QUERY jest ustawiany na HIGH, natomiast dla operacji ARCHIVE na LOW. Za pomocą poniższego zapytania możesz sprawdzić poziom włączonej kompresji:

```
select table_name, compression, compress_for
from user_tables
where table_name='PRACOWNICY';
```

Przy próbie użycia kompresji kolumnowej w środowisku innym niż Exadata pojawi się następujący błąd:

```
ORA-64307: Exadata Hybrid Columnar Compression is not supported for tablespaces on this storage type
// hybrydowa kompresja kolumnowa Exadata nie jest obsługiwana w przestrzeniach tabel w tym systemie dyskowym
```

Jak to działa

Exadata jest maszyną bazodanową o wysokiej wydajności. Służy do budowania wysokiej wydajności hurtowni danych i baz OLTP (ang. *Online Transaction Processing*, bieżące przetwarzanie transakcji). Maszyna Exadata obsługuje hybrydową kompresję kolumnową i jest dostępna od wersji bazy Oracle 11g R2.

■ **Uwaga**. W wersji 11g R2 kompresja kolumnowa jest włączana za pomocą klauzuli COMPRESS FOR QUERY. Począwszy od wersji 12c, klauzula została zmieniona na COLUMN STORE COMPRESS FOR QUERY.

Hybrydowa kompresja kolumnowa kompresuje dane na poziomie kolumn. Umożliwia ona osiągnięcie wyższego stopnia kompresji niż w przypadku kompresji podstawowej (patrz porada 1.15) i zaawansowanej (patrz porada 1.16). Dostępne są cztery poziomy hybrydowej kompresji kolumnowej, wymienione niżej w kolejności od najniższego do najwyższego:

- COLUMN STORE COMPRESS FOR QUERY LOW,
- COLUMN STORE COMPRESS FOR QUERY HIGH,
- COLUMN STORE COMPRESS FOR ARCHIVE LOW,
- COLUMN STORE COMPRESS FOR ARCHIVE HIGH.

Kompresję COLUMN STORE COMPRESS FOR QUERY należy stosować w przypadku operacji ładowania dużej ilości danych do tabel stertowych, które są rzadko aktualizowane. Tego typu kompresja jest zoptymalizowana pod kątem wydajności zapytań, dlatego lepiej sprawdza się w systemach wspomagania decyzji i hurtowniach danych. Natomiast kompresja COLUMN STORE COMPRESS FOR ARCHIVE maksymalizuje stopień kompresji i jest lepsza w przypadku danych, które są przechowywane przez długi czas i nie są aktualizowane.

■ **Uwaga**. Więcej informacji na temat hybrydowej kompresji kolumnowej znajdziesz w dokumentacji *Oracle Exadata Storage Server Software*.

Skorowidz

A

B

C

D

E

F

G

H

I

J

K

L

Ł

M

N

O

P

R

S

Ś

T

U

V

W

Z

BŁYSKAWICZNIE ROZWIĄŻ PROBLEMY Z BAZĄ DANYCH!

Oracle Database 12c
Problemy i rozwiązania

Oracle to jedna z najpopularniejszych baz danych na świecie i znajduje zastosowanie w wielu firmach. W jej tabelach przechowywane są gigantyczne ilości danych. Zasoby te są każdego dnia przetwarzane na różne sposoby, a szybkość dostępu do rezultatów istotnie wpływa na efektywność pracy wielu ludzi. Jeżeli dostęp do danych jest utrudniony, to administrator bazy musi błyskawicznie zlokalizować i rozwiązać problem.

Dzięki tej książce będziesz przygotowany na najbardziej stresujące sytuacje spotykane w codziennej pracy administratora. W trakcie lektury opanujesz techniki optymalizacji wykorzystania pamięci i dysków, czasu trwania zapytań SQL oraz wydajności. Znajdziesz tu również liczne opisy problemów z życia wziętych oraz najlepsze sposoby ich rozwiązywania. Ponadto nauczysz się monitorować pracę systemu i zidentyfikujesz problemy, zanim dotkną one jego użytkowników. Przekonasz się, jak istotną rolę odgrywają właściwie dobrane indeksy. Książka ta jest obowiązkową pozycją dla każdego administratora bazy danych!

Dzięki tej książce:

- zapoznasz się z możliwymi przyczynami problemów z bazą
- zoptymalizujesz zużycie pamięci RAM i dysków twardych
- stworzysz indeksy, które poprawią wydajność Twojej bazy
- skutecznie przeanalizujesz wolne zapytania SQL
- będziesz przygotowany na problemy z bazą

sięgnij po WIĘCEJ

KOD KORZYŚCI

Apress

Nr katalogowy: 26483

Księgarnia internetowa: http://helion.pl

Zamówienia telefoniczne: 0 801 339900 0 601 339900

helion.pl księgarnia internetowa

Sprawdź: najnowsze promocje: http://helion.pl/promocje
Książki najchętniej czytane: http://helion.pl/bestsellery
Zamów informacje o nowościach: http://helion.pl/nowosci

Helion

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
http://helion.pl

cena: 89,00 zł

ISBN 978-83-246-9801-1

9 788324 698011

Informatyka w najlepszym wydaniu