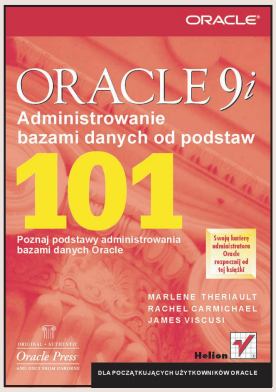
Oracle9i. Administrowanie bazami danych od podstaw



Autorzy: Marlene Theriault,
Rachel Carmichael, James Viscusi
Tłumaczenie: Michał Żyliński (rozdziały 0 – 8),
Leszek Mosingiewicz (rozdziały 9 – 13, dodatki)
ISBN: 83-7361-062-6
Format: B5, stron: około 480

Książka „Oracle9i. Administrowanie bazami danych od podstaw" krok po kroku wyjaśnia sposoby efektywnego administrowania bazą danych Oracle. Poznasz dzięki niej najważniejsze nowe funkcje baz danych Oracle9, zaznajomisz się z zadaniami czekającymi administratora baz danych, a także poznasz wiele cennych wskazówek, ułatwiających codzienną pracę z Oraclem. Jeśli jeszcze nie miałeś styczności z zaawansowanymi systemami bazodanowymi, znalazłeś właśnie doskonałe źródło informacji, dzięki któremu zdobędziesz wiele poszukiwanych na rynku pracy umiejętności.

Książka jest idealnym źródłem wiedzy dla początkujących administratorów na temat:

- Instalacji i konfiguracji wydajnej bazy danych Oracle
- Kontroli działania bazy danych za pomocą perspektyw DBA_ i V$
- Wykorzystania różnych poziomów ochrony danych
- Monitorowania i dostrajania bazy danych
- Stosowania narzędzi i programów Oracle
- Utrzymywania niezawodności i stałego dostępu do bazy danych
- Sposobów zapewnienia maksymalnej wydajności serwera bazodanowego
- Wykonywania kopii zapasowych i odtwarzania bazy danych

Autorzy:

Marlene Theriault od 19 lat jest administratorką baz danych i pracuje z produktami firmy Oracle od ukazania się wersji 2.0. Jest autorką wielu poczytnych książek na ich temat. Występowała jako prelegent i prowadziła warsztaty na organizowanych na całym świecie konferencjach użytkowników Oracle.

Rachel Carmichael od 10 lat jest administratorką baz danych. Pełni rolę przewodniczącej grupy tematycznej administratorów baz danych (DBA Special Interest Group) oraz koordynuje spotkania użytkowników oprogramowania Oracle w Nowym Yorku.

James Viscusi od 12 lat pracuje z relacyjnymi bazami danych, a od 8 lat ma styczność z produktami Oracle. Obecnie jest pracownikiem firmy Oracle, gdzie zajmuje się zagadnieniami wysokiej dostępności systemów.

Wydawnictwo Helion
ul. Chopina 6
44-100 Gliwice
tel. (32)230-98-63
e-mail: helion@helion.pl

Spis treści

Rozdział 9.

Codzienne czynności

Pomyślmy o czynnościach wykonywanych w codziennym życiu. Codziennie wstajemy o tej samej godzinie, myjemy zęby, bierzemy prysznic, jemy śniadanie. W drodze do pracy zatrzymujemy się i kupujemy gazetę i filiżankę kawy. Jeździmy tym samym pociągiem lub autobusem. Jeśli chodzimy do szkoły, zajęcia odbywają się zgodnie z ustalonym planem. Raz w tygodniu robimy większe zakupy i pranie. Raz w miesiącu możemy pójść do fryzjera. Co roku odwiedzamy dentystę i kontrolujemy stan techniczny samochodu. Być może twoje życie nie jest aż tak uporządkowane, ale ludzie ulegają przyzwyczajeniom — jeżeli masz stałą pracę, to działasz zgodnie z pewnymi zasadami i procedurami.

Uważamy, że podobnie jak w codziennym życiu, w pracy administratora bazy danych obowiązują pewne zasady działania. Poniższe procedury i procesy informują o potencjalnych problemach, tak aby możliwe było podjęcie działań, zanim sprawy przybiorą zły obrót. Wspaniale jest być profesjonalnym DBA. Nie tylko zwiększa się szansa posiadania płynnie pracującej bazy danych, lecz maleje również liczba weekendów spędzanych w pracy.

Jednocześnie zyskujemy pewność, że drobnostki nie urosną do rangi poważnych problemów.

W niniejszym rozdziale pokazujemy skrypty pomagające w implementacji nowych zwyczajów. Wykonanie każdego z nich wymaga podłączenia do bazy danych z uprawnieniami użytkownika uprzywilejowanego. Należy wykorzystać konta `SYS` lub `SYSTEM` z dostarczoną przez Oracle rolą DBA albo utworzyć użytkownika z rolą DBA lub jego przywilejami. Zalecamy utworzenie oddzielnego konta użytkownika do monitorowania bazy danych.

Obserwacja pracy bazy danych

Dlaczego mamy obserwować zachowanie bazy danych w regularnych odstępach czasu? Czy nie prościej jest odbierać powiadomienia o ewentualnych błędach? Codzienna kontrola pozwala właściwie określić:

- Jakie elementy bazy funkcjonują prawidłowo.
- Wzrost zajmowanej przestrzeni.

- Częstotliwość przełączania dzienników.
- Liczbę przyłączonych użytkowników.

Jeżeli wiadomo, jak powinna funkcjonować baza danych, to natychmiast można zauważyć każdą nieprawidłowość w jej pracy i skorygować ją, zanim jeszcze pojawi się problem. Niewątpliwie lepiej jest poinformować kierownictwo o fakcie dodania jeszcze jednego pliku danych z powodu braku miejsca w głównych tabelach aplikacji, niż doprowadzić do zatrzymania pracy całego działu, kiedy wolna przestrzeń, niezbędna do wzrostu tabel zostanie wyczerpana.

Przejdźmy do omówienia elementów, które należy monitorować.

Dzienniki ostrzeżeń

W czasie tworzenia bazy danych Oracle tworzy również dziennik, znany jako dziennik ostrzeżeń (alert log), w którym zapisuje informacje dotyczące:

- Każdego uruchomienia bazy danych.
- Odzyskiwania przeprowadzonego w czasie startu bazy.
- Wyłączeń bazy danych.
- Momentów przełączenia dzienników (określanych jako przełączenia wątków).
- Wykorzystania podczas startu bazy parametrów (z pliku parametrów inicjujących) o wartościach innych niż domyślne.
- Wszystkich poleceń DDL zmieniających strukturę bazy, takich jak np. `ALTER TABLESPACE PAYROLL_TS ADD plik_danych`.
- Pojawiających się błędów alokacji przestrzeni (`ORA-1650` do `ORA-1659`).
- Występujących błędów wraz z położeniem i nazwą pliku błędu (określanego jako plik śladu z powodu rozszerzenia *.trc*). ORACLE tworzy te pliki w celu pełnej dokumentacji błędów i dostarczenia pomocy technicznej przy lokalizacji źródła błędów.

Jak widać, dziennik błędów zawiera sporo cennych, może nawet nadmiarowych informacji. Jego rozmiar nigdy nie zmniejsza się automatycznie, a nazwa w różnych systemach budowana jest w następujący sposób:

- `alert<SID>.log` w systemach uniksowych,
- `<SID>alrt.log` w systemie Windows 2000.

Inaczej mówiąc, informacje są stale dodawane do utworzonego razem z bazą danych dziennika za każdym razem, gdy wystąpi jedno z wymienionych wyżej zdarzeń. Jak można sobie wyobrazić, rozmiar tego piku, dla pracującej przez dłuższy czas bazy, może być bardzo duży. Jeżeli zamierzamy sprawdzić wpisy z ostatniego tygodnia, a baza danych pracowała powiedzmy przez rok, zadanie to może się okazać kłopotliwe. Co zatem zrobić w takiej sytuacji?

Otóż istnieje pewna prawidłowość: Oracle utworzy nowy plik dziennika z identyczną nazwą w wypadku, gdy w czasie próby zapisu nie odnajdzie pliku. Oznacza to, że można codziennie zmieniać nazwę pliku dziennika i przenosić go do archiwum. Takie działanie wiąże się z dwoma korzyściami: bieżący plik dziennika błędów ma relatywnie mały rozmiar (zatem nie trzeba się martwić o wyczerpanie przestrzeni dyskowej), a jednocześnie poprzednie dzienniki są dostępne do przeglądania, np. w poszukiwaniu pewnych trendów. Mamy przygotowane zadanie wsadowe, uruchamiane co noc, które zmienia nazwy plików dzienników wszystkich baz danych w ten sposób, że dodaje aktualną datę jako rozszerzenie pliku. Kolejne zadania jego procedur polegają na przesłaniu dzienników na nasze konta pocztowe, co umożliwia ich szybki przegląd w poszukiwaniu błędów takich jak problemy z przestrzenią (błąd 600) lub zbyt szybkie przełączanie dzienników. Błędy Oracle w zakresie 600 są określane jako błędy `ORA-600`. Pomimo że nie zawsze sygnalizują one uszkodzenie bazy danych, mogą być przejawem poważnych błędów wewnętrznych i powinny zostać potraktowane poważnie. Jeżeli nie wiemy, z jakim typem błędu mamy do czynienia, możemy to sprawdzić, wykorzystując stronę Metalink.

W przypadku administrowania wieloma bazami danych analiza wszystkich dzienników błędów może zajmować zbyt wiele czasu. Nie zawsze można sobie pozwolić na analizę problemów następnego dnia. Niektórzy administratorzy wypracowali procedury poszukujące błędów i wysyłające powiadomienia tylko w wypadku ich stwierdzenia. Jeżeli pracujemy w środowisku, w którym błędy i przestoje są niedopuszczalne, można stworzyć lub kupić program monitorujący występowanie błędów i wysyłający powiadomienia. Operacja sprawdzenia dziennika bazy danych zajmuje jedynie kilka minut dziennie. Niektóry z nas wykorzystują skrypty przesyłające dzienniki do prywatnych kont pocztowych. Jeżeli codziennie przesyłasz do siebie dzienniki błędów, to przejrzenie ich umożliwi:

- Szybkie zlokalizowanie dowolnych problemów.
- Manualne sprawdzenie plików w celu upewnienia się, czy nie są one przełączane zbyt często.
- Upewnienie się, czy proces zapisujący do plików dziennika powtórzeń przebiega bez zakłóceń (które mogłyby wskazywać na zbyt małe rozmiary lub zbyt małą ilość dzienników powtórzeń w systemie).

Oracle Enterprise Manager (OEM) posiada usługę Event Monitoring Service, która również pomaga w monitorowaniu bazy danych. Więcej informacji o OEM, jego możliwościach i funkcjach, zawarto w rozdziale 13.

Jeżeli parametr inicjujący `log_checkpoints_to_alert` ma wartość `TRUE`, Oracle doda więcej informacji o przełączaniu dzienników zmian do pliku dziennika ostrzeżeń. Zwiększając jego rozmiar, umieści w nim jednocześnie więcej informacji o tym, co dzieje się w bazie danych. Poniżej pokazujemy fragment dziennika ostrzeżeń bazy danych Oracle9i, w której opcja `log_checkpoints_to_alert` została ustawiona na `TRUE`.

```
Sat Jan 18 11:37:22 2002
Begining log switch checkpoint up to RBA [0x33.2.10],SCN: 0x0000.000f9792
Thread 1 advanced to log sequence 51
  Current log# 3 seq# 51 mem# 0: C:\ORACLE\ORADATA\BB901\REDO03.LOG
Sat Jan 18 11:46:18 2002
ARC0: Begining to archive log 2 thread 1 sequence 50
ARC0: Begining to archive log 2 thread 1 sequence 50
```

```
Sat Jan 18 11:50:18 2002
Begining log switch checkpoint up to RBA [0x34.2.10],SCN: 0x0000.000f9793
Thread 1 advanced to log sequence 52
  Current log# 1 seq# 52 mem# 0: C:\ORACLE\ORADATA\DB901\REDO01.LOG
Sat Jan 18 11:52:18 2002
ARC0: Begining to archive log 3 thread 1 sequence 51
ARC0: Begining to archive log 3 thread 1 sequence 51
Sat Jan 18 11:57:51 2002
Completed checkpoint up to RBA [0x34.2.10],SCN: 0x0000.000f9793
Completed checkpoint up to RBA [0x33.2.10],SCN: 0x0000.000f9792
Sat Jan 18 11:59:51 2002
Begining log switch checkpoint up to RBA [0x35.2.10],SCN: 0x0000.000f9795
Thread 1 advanced to log sequence 53
  Current log# 2 seq# 53 mem# 0: C:\ORACLE\ORADATA\DB901\REDO02.LOG
Sat Jan 18 12:11:10 2003
ARC0: Begining to archive log 1 thread 1 sequence 52
ARC0: Begining to archive log 1 thread 1 sequence 52
Sat Jan 18 12:11:30 2002
Completed checkpoint up to RBA [0x35.2.10],SCN: 0x0000.000f9795
Sat Jan 18 12:11:50 2002
Begining global checkpoint up to RBA [0x35.4.10],SCN: 0x0000.000f9798
Completed checkpoint up to RBA [0x35.4.10],SCN: 0x0000.000f9798
Sat Jan 18 12:14:17 2002
Sat Jan 18 20:57:17 2002
ORA-1653: unable to extend table RACHEL.LRGTABLE by 640 in tablespace LRGDATA
```

Wygląda na to, że baza danych DB901 w sobotę 18 stycznia 2002 o godzinie 20:27 miała problem z przestrzenią. Zakładając, że o tak późnej porze w systemie nie pracowali już użytkownicy, możemy przypuszczać, że to nietypowe zadanie wsadowe próbowało załadować tabelę LRGTABLE. Niestety, jej obszar tabel zawiera za mało ciągłej przestrzeni dla alokacji obszaru. Do dziennika ostrzeżeń został zapisany błąd, a proces próbujący załadować tabelę zakończył się niepomyślnie. Jeżeli istniałby proces, monitorujący w sposób ciągły dziennik ostrzeżeń, który powiadamia o błędach, moglibyśmy zareagować natychmiast, dodając niezbędną przestrzeń i prosząc użytkownika o wznowienie zadania lub przesłać wiadomość o rozwiązaniu problemu do wszystkich zainteresowanych (w tym oczywiście do szefa, który powinien zostać o takim fakcie poinformowany).

W dalszej części tego rozdziału umieszczony jest skrypt pozwalający na sprawdzenie, czy któryś z obiektów bazy danych nie będzie w stanie pobrać wolnego obszaru z powodu braku wolnej przestrzeni. Umożliwia to dodanie przestrzeni jeszcze przed wystąpieniem problemu w czasie wykonywania zaplanowanych zadań którejś z aplikacji. Jeżeli zaobserwujemy częste błędy wykonania pewnego zadania wsadowego, to możemy wykorzystać nowy tryb alokacji przestrzeni, zatrzymać zadanie do czasu wykonania odpowiedniej akcji, a następnie wznowić je od miejsca, w którym zostało zatrzymane. O tym sposobie powiemy więcej w rozdziale 11.

LISTENER.LOG

Proces nasłuchu Oracle Net Listener (we wszystkich wersjach Oracle8 nazywany Net8, a SQL*Net we wcześniejszych) tworzy w czasie uruchamiania dziennik, do którego w zależności od ustawionego poziomu logowania zapisuje duże lub minimalne ilości danych. W opcji minimalnej zapisywane są następujące informacje:

- Czas uruchomieniu procesu nasłuchu.
- Lista portów, na których jest on prowadzony.
- Komputer, na którym nasłuch jest uruchomiony.
- Informacje o połączeniach zrealizowanych przez proces.

Przykładowy wpis wygląda następująco:

```
TNSLSNR for 32-bit Windows: Version 9.0.1.1.1 - Production on
13-STY-2002 18:59:09

Copyright (c) 1991, 2001, Oracle Corporation.  All rights reserved.

Plik parametrów systemowych jest c:\Oracle\Ora90\network\admin\listener.ora
Komunikaty dziennika zapisano do c:\Oracle\Ora90\network\log\listener.log
Informacja o śladzie zapisana do c:\Oracle\Ora90\network\trace\listener.trc
Obecny poziom śladu jest 0

Uruchomiono z pid=11192
Nasłuch na: (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=DRAGONFY_LAPTOP)(PORT=1521)))
Nasłuch na: (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)PIPENAME=\\.\pipe\EXTPROC0ipc))
TIMESTAMP * CONNECT DATA [* PROTOCOL INFO] * EVENT [* SID] * RETURN CODE
```

Jeżeli zostanie włączone logowanie na wyższym poziomie, w dzienniku będzie zapisywane więcej informacji, lecz jednocześnie znajdą się tam dane mniej istotne. Zalecamy systematyczne, okresowe sprawdzanie dziennika `listener.log` w poszukiwaniu błędów. Jeżeli odnotuje on problemy z Oracle Net, należy prowadzić dokładniejsze obserwacje. Proszę zauważyć, że wcześniejsze wersje oprogramowania tworzą inne wpisy w tym dzienniku. Jedynym sposobem na usunięcie dziennika `listener.log`, kiedy osiągnie zbyt duży rozmiar, jest zatrzymanie procesu nasłuchu, usunięcie pliku dziennika i ponowne uruchomienie listenera. W przeciwieństwie do dziennika alertów, po zmianie jego nazwy ORACLE nadal zapisuje do niego informacje. Zatrzymanie nasłuchu nie pociąga za sobą utraty istniejących połączeń, lecz do chwili jego wznowienia niemożliwe jest tworzenie nowych. Z tego powodu uruchamiamy zazwyczaj dwa oddzielne procesy dla każdej instancji i umieszczamy oba porty we wszystkich plikach `TNSNAMES.ORA` naszych klientów. Kiedy Oracle Net podejmuje próbę podłączenia się do listenera od strony klienta, stara się wykorzystać najpierw pierwszy podany w `TNSNAMES.ORA` port. Jeżeli nie istnieje na nim aktywny proces nasłuchu, to sprawdzany jest następny port. Zatem wyłączenie nasłuchu, usunięcie plików dzienników i ponowne włączenie listenera nie powoduje zakłóceń w dostępie użytkowników do bazy danych.

Słabą stroną posiadania wielu listenerów jest konieczność zarządzania nimi oraz wzrost liczby aktywnych procesów serwera (jeden proces dla każdego nasłuchu).Kilka listenerów pozwala rozłożyć obciążenie sieci na większą liczbę kart interfejsów sieciowych (Network Interface Cards NIC) i w ten sposób zbalansować połączenia do serwera bazy danych. Szczegółowe informacje dotyczące konfiguracji sieciowej zawierają Net Services Administrator Guide i Net Services Reference Guide dołączone do dokumentacji Oracle.

Status Redo Log

W jakim celu należy sprawdzać status dzienników zmian? Być może z powodu zbyt częstego przełączania zamierzasz zmienić ich rozmiar. Jeżeli ich nazwy mają pozostać niezmienione, to konieczne jest ich usuniecie, a następnie odtworzenie z nowym rozmiarem. Przed usunięciem dziennika zmian administrator powinien wiedzieć, jaki jest aktualny status tego dziennika. Usunięcie bieżącego dziennika nie jest dobrą praktyką, jeżeli nie dotyczy testowania scenariuszy odzyskiwania!

Informacje o aktywnych dziennikach zmian są utrzymywane w dwóch wirtualnych perspektywach V$: V$LOG i V$LOGFILE. W celu pobrania pełnych informacji, perspektywy te mogą zostać złączone na kolumnie Group#. Dzienniki zmian mogą stanowić część grupy dzienników, pozwalającej na ich dublowanie (ang. *mirroring*) z wykorzystaniem programów narzędziowych Oracle i zapewnienie w ten sposób dodatkowego zabezpieczenia przed ich ewentualną utratą. Strata pojedynczego aktywnego dziennika wymaga odbudowania bazy danych, natomiast gdy dziennik należy do grupy, Oracle wpisuje informację do dziennika alertów i kontynuuje pracę, wykorzystując pozostałe dzienniki. Istnieje zatem możliwość usunięcia i odtworzenia dziennika zmian bez potrzeby wyłączania bazy danych.

Poniższy skrypt umożliwia wyświetlenie informacji o dziennikach zmian w bazie danych test. Do jego uruchomienia wymagane są uprawnienia konta SYSTEM.

```
col Member format a40
col Logstat format a10 heading 'Use status'
select  Member,
        a.Group#,
        b.status,
        a.status Logstat
  from V$LOGFILE a,V$LOG b
   where a.Group#=b.Group#
/
MEMBER                                  GROUP# STATUS           Use status
--------------------------------------- ------ ---------- ---------------- --
\ORADATA6\TEST\REDO01.RDO                    1 ACTIVE
\ORADATA7\TEST\REDO02.RDO                    2 CURRENT
\ORADATA6\TEST\REDO03.RDO                    3 INACTIVE
\ORADATA7\TEST\REDO04.RDO                    3 INACTIVE
```

Przeglądając wyniki wykonania tego dziennika, możemy powiedzieć kilka istotnych rzeczy o tej bazie danych:

- Administrator bazy danych (DBA) nie zdecydował się na wykorzystanie zalet dublowania dzienników, ponieważ w każdej grupie jest tylko jeden dziennik.
- Istnieją cztery dzienniki zmian i wszystkie są częścią bazy danych.
- Kolumna USE STATUS zawierająca status z perspektywy V$LOGFILE jest pusta, jeżeli dziennik jest wykorzystywany przez bazę danych.
- Dzienniki są umieszczone na dwóch różnych dyskach, a DBA próbował poprawić wydajność przez zamianę ich położenia na tych dyskach. Do czasu, gdy ORACLE będzie czytać poprzedni dziennik w celu archiwizacji, istnieje możliwość wystąpienia konfliktów na poziomie głowic dyskowych, jeżeli oba

dzienniki znajdują się na tym samym dysku. Umieszczenie ich na oddzielnych dyskach umożliwia niezakłócony odczyt jednego z nich w czasie zapisu do drugiego. Jeżeli tylko jest to możliwe, dzienniki zmian powinny być umieszczane na pustych dyskach lub w ostateczności na dyskach zawierających pliki rzadko wykorzystywane.

- Bieżącym wykorzystywanym plikiem dziennika jest `redo2` (wartość w kolumnie statusu `CURRENT`).
- Ostatnio wykorzystywanym dziennikiem był `redo1` (Status `ACTIVE`).
- Jest prawdopodobne, że plik `redo1` jest albo archiwizowany, albo wymagany do odzyskania bazy danych w przypadku awarii. Status `INACTIVE` oznacza, że dziennik zmian nie jest wymagany do odzyskiwania.

Inne możliwe wartości kolumny statusu perspektywy `V$LOG` można znaleźć w dokumentacji Oracle.

Fragmentacja

W jakim celu sprawdzamy poziom fragmentacji w naszej przestrzeni tabel? Pomyślmy o przestrzeni tabel jak o puzzlach, które próbujemy ułożyć. Im więcej klocków uda nam się dopasować, tym bardziej komplikuje się kształt wolnego miejsca. Dopasowanie kolejnego elementu wymaga dokładnego określenia jego rozmiaru.

Poszukiwanie wolnej przestrzeni w obszarze tabel dla kolejnego obszaru funkcjonuje w podobny sposób. Przy odrobinie szczęścia można utworzyć lub odziedziczyć bazę, której tabele w każdej przestrzeni tabel wykorzystują domyślne parametry przechowywania przestrzeni tabel. W ten sposób każdy obszar ma identyczny rozmiar i nie istnieją porcje wolnej przestrzeni, których wielkość uniemożliwia ich wykorzystanie. Niestety, nie zawsze to tak wygląda. W przestrzeni tabel może istnieć dużo wolnego miejsca, lecz żaden z jego fragmentów nie jest wystarczająco duży do pomieszczenia obszaru. Pomimo że obszary tworzące obiekt mogą być rozrzucone na dysku, bloki w obrębie obszaru muszą być umieszczone w sposób ciągły. Z tego powodu należy monitorować przestrzeń tabel i upewniać się, że istnieją porcje wolnych bloków na tyle duże, aby kolejne zadanie zostało wykonane.

Poniższy skrypt PL/SQL wyświetla dostępne w obrębie przestrzeni tabel obszary włącznie z ich rozmiarami. Przed uruchomieniem skryptu należy utworzyć tabelę do przechowywania wyników przejściowych, wykorzystując skrypt utworzenia tabeli. Do uruchomienia skryptu podającego fragmentację należy wykorzystać konto z przywilejami użytkownika `SYSTEM`.

```
Create table FREESP (
Fname VARCHAR(513),
Tspace VARCHAR(30),
First NUMBER(20),
Blocks NUMBER(10),
Last NUMBER(10))
/
```

Tabela FREESP jest tworzona tylko raz, a po opróżnieniu poleceniem truncate, wykorzystywana wielokrotnie. Kreowanie tabeli przy każdym wykonaniu skryptu zwiększałoby niepotrzebnie stopień fragmentacji przestrzeni.

```
set feedback off term off verify off pagesize 60 newpage 0 linesize 66
 truncate table FREESP;
  declare
   Fileid      NUMBER(9);
   FileName    VARCHAR(513);
   Tsname      VARCHAR(30);
   Cursor Tablespaces is
       select File_Name,File_ID, Tablespace_Name
          from DBA_DATA_FILES
        where Tablespace_Name =upper('&1');
begin
open Tablespaces;
loop
   fetch Tablespaces into Filename, Fileid, TsName;
   exit when Tablespaces%NOTFOUND;
declare
  First    NUMBER(10);
  Blocks   NUMBER(10);
  Last     NUMBER(10);
  Tfirst   NUMBER(10);
  Tblocks  NUMBER(10);
  Tlast    NUMBER(10);
  Cursor Free is
    select BLOCK_ID a, Blocks b, BLOCK_ID+Blocks c
       from DBA_FREE_SPACE
     where File_ID = Fileid
     order by Block_ID;
begin
  open Free;
  fetch Free into First,Blocks, Last;
  if Free%NOTFOUND
   then
       goto close_free;
  end if;
  loop
    fetch Free into Tfirst, Tblocks, Tlast;
    exit when Free%NOTFOUND;
    if Tfirst= Last
     then
        Blocks:=Blocks + Tblocks;
        Last :=Tlast;
     else
       insert into FREESP
         values (Filename, Tsname, First,Blocks, Last-1);
       commit;
       First :=Tfirst;
       Blocks:=Tblocks;
       Last:=Tlast;
   end if;
end loop;
    insert into FREESP
       values (Filename,Tsname, First,Blocks, Last-1);
 commit;
```

```
    /* zamknięcie Free */
<<CLOSE_FREE>>
 close Free;
end;
end loop;
commit;
close Tablespaces;
end;
/
set term off echo off
  ttitle center ' RAPORT FRAGMENTACJI PRZESTRZENI TABEL'
  col Tspace heading 'NAZWA|PRZESTRZENI'  format a16 trunc
  col Fname  heading 'PLIK'               format a30 trunc
  col First  heading 'BLOK|POCZ '         format 999,999
  col Blocks heading 'ROZMIAR(BLK)'       format 99,999,999
  break on report on Tspace skip 1 on Fname skip 1
   compute  sum of Blocks on Fname
   compute sum of Blocks on report
   spool fragmentacja.rpt
   select Tspace,Fname, First, Blocks
     from FREESP
  order by Tspace,Fname, First;
  spool off
```

Skrypt jest wykonywany dla pojedynczej przestrzeni tabel, której nazwa jest jego argumentem wejściowym.

Przyjrzyjmy się wynikom jego wykonania.

```
                    RAPORT FRAGMENTACJI PRZESTRZENI TABEL
NAZWA                                                 BLOK
PRZESTRZENI      PLIK                                 POCZ ROZMIAR(BLK)
---------------- ------------------------------ -------- ------------
USR1            /home01/oracle/user_1             3,215            3
.................................................4,065............40
.................................................5,586............75
.................................................8,475...........145
................................................10,080...........110
................................................10,545...........100
................................................21,832........3,215
................................................50,563............85
................................................61,054...........111
................................................85,045........1,420
................................................95,996.............7
...............................................108,526............70
...............................................121,503...........121
...............................................125,689...........275
...............................................127,250...........450
...............................................131,418............58
...............................................151,903...........895
...............................................171,876............50
...............................................270,022...........665
...............................................292,447............18
...............................................328,862............50
...............................................328,922............85
...............................................334,617...........110
...............................................334,614...........285
```

```
.....359,493.....670
.....360,328.....675
.....365,297.....125
.....366,197.....325
.....366,567.....435
.....369,117.....33
.....371,945.....13
.....372,033.....220
.....372,833.....155
.....375,651.....150
.....423,382.....195
.....425,657.....100
.....431,110.....72
.....434,692.....1,905
.....481,830.....40
          ******************************  -----------
          sum                                  13,881


****************                          -----------
sum                                            13,881
```

Przestrzeń tabel jest bardzo niejednorodna. Czterdzieści obszarów w 13 881 wolnych blokach, średnio mniej niż 350 bloków na obszar. Jeżeli jedyną rzeczą, którą zrobiliśmy, było zwiększenie liczby dostępnych dla tej przestrzeni tabel bloków w perspektywie DBA_FREE_SPACE, to wydaje się, że posiada ona mnóstwo wolnej przestrzeni. Zbyt wielu niedoświadczonych administratorów wpada w pułapkę polegającą na sprawdzeniu jedynie całkowitej dostępnej przestrzeni i wnioskowaniu na tej podstawie o wystarczającej ilości ciągłego miejsca do alokowania kolejnego obszaru. Analiza przestrzeni tabel z powyższego raportu ujawnia, że większość obszarów ma rozmiar mniejszy niż 500 bloków. Dopóki przestrzeń tabel jest wykorzystywana w trybie tylko do odczytu lub zawiera tabele sporadycznie aktualizowane, nie ma problemów z przydzielaniem nowego miejsca.

Oracle nie wykonuje scalania ciągłych wolnych obszarów, chyba że nie istnieje inny sposób uzyskania wolnej przestrzeni. Jej fragmentacja wzrasta, w miarę jak Oracle przydziela ją z największego wolnego obszaru, przed powtórnym wykorzystaniem i połączeniem mniejszych wolnych obszarów, które pozostały po usuniętych obiektach. Rysunek 9.1 ilustruje obszary dostępne w przestrzeni tabel przed zażądaniem alokacji kolejnego obszaru, a następnie po wykonaniu tego polecenia.

Rysunek 9.1. *Łączenie wolnych obszarów: (a) fragmentacja z dwoma ciągłymi obszarami przed żądaniem alokacji obszaru złożonego z 25 bloków (b) fragmentacja po scaleniu i alokowaniu obszaru*

Perspektywa słownika danych DBA_FREE_SPACE zawiera informacje o każdym wolnym obszarze w plikach danych tworzących bazę. Oracle nie sprawdza tej perspektywy pod kątem ciągłości obszarów. Zatem dwa sąsiednie obszary (jak na rysunku 9.1a) przedstawione są w dwóch różnych wierszach. Nie jest to szczególnie efektowne i użyteczne rozwiązanie, ponieważ nie przedstawia rzeczywistego obrazu wolnej przestrzeni w plikach danych!

Skrypt fragment.sql czyta perspektywę DBA_FREE_SPACE, poszukuje ciągłych obszarów, sumuje ich liczbę bloków (lub podaje ich liczbę, jeżeli obszary nie są ciągłe), przechowuje wyniki w tabeli przejściowej i drukuje raport o poziomie fragmentacji plików danych w przestrzeni tabel. Przechowanie poprzednich raportów umożliwia analizę wykorzystania przestrzeni tabel i częstotliwości występowania jej fragmentacji.

Jeżeli wykorzystujesz przestrzeń tabel zarządzaną lokalnie, to nie należy się martwic o łączenie wolnych obszarów. W wypadku zarządzania przez słownik, operacje tę można wykonać manualnie, wydając polecenie:

```
Alter tablespace <tablespace name> coalesce;
```

Segmenty wycofania

Jak napisać wiersz? Zaczynamy od pomysłu i być może od pierwszego zdania. Zapisujemy kilkanaście linijek i decydujemy, że jest on po prostu okropny i rozpoczynamy pisanie od początku. W połowie pisania możemy stwierdzić, że większość tekstu jest wspaniała, lecz kilka linijek wymaga jednak zmiany. Ostatecznie, kończąc pisanie, decydujemy, że usunięte fragmenty powinny się w nim znaleźć. Wymaga to pamiętania treści tych wersów (uważamy, że nadawały one utworowi wyjątkowy charakter i koniecznie muszą zostać odtworzone). Jeżeli każda linia była pisana ostrożnie, a to co chcieliśmy zachować, było kopiowane przed każdą zmianą całego wiersza, to zachowany został jego wygląd na każdym etapie pisania. Z łatwością możemy stwierdzić, jak wyglądała ta wspaniała linijka. Po zakończeniu kolejnej wersji wiersza i wykonaniu jej kopii „na czysto" (czyli po jej ostatecznym zatwierdzeniu), możemy udostępnić ją komuś do przeczytania, podczas gdy sami zajmiemy się analizą jej treści. Oracle wykorzystuje segmenty wycofania dokładnie tak, jak my swoją kopię, do przechowania obrazu danych przed wprowadzeniem zmian, określanego też jako *obraz danych przed zmianą*.

Jaki cel ma przechowywanie obrazu danych przed zmianą? Oracle zapewnia, że „odczyt danych nie będzie blokował ich zapisu, a zapis nie będzie blokował odczytu". Co to dokładnie znaczy? Załóżmy, że otwieramy tabelę płac i wykonujemy raport o zarobkach. W tym samym czasie ktoś inny zmienia wartości kilku pozycji w tej tabeli. Bez segmentu wycofania, raport zawierałby niespójne dane, część z nich dotyczyłaby tabeli przed zmianami, a część po ich wprowadzeniu. Oracle zapewnia, że sytuacja taka jest niemożliwa, segment wycofania zawierający obraz danych przed zmianą zapewnia utrzymanie ich spójności nawet wtedy, gdy jednocześnie wprowadzane są modyfikacje.

W wersji Oracle9i wprowadzono koncepcję automatycznego zarządzania kasowaniem zmian (ang. *undo managment*). Zasadniczo oznacza to, że jeżeli zostanie utworzona specjalna przestrzeń określana jako przestrzeń wycofania (ang. *Undo Tablespace*) i ustawiony parametr inicjujący undo_managment na AUTO, Oracle automatycznie przejmie nad nim kontrolę, przydzielając i zwalniając w miarę potrzeby segmenty wycofania.

Kiedy użytkownik rozpoczyna pracę, generując raport, czy też modyfikując wiersze tabeli, do związanego z nim procesu jest przypisany segment wycofania. W rozdziale 2., mówiliśmy o sposobie alokowania segmentów wycofania i zwiększania ich rozmiarów. Co jednak stanie się w sytuacji, gdy w takim segmencie zabraknie miejsca? Jeżeli duża transakcja wymaga więcej miejsca niż aktualnie posiada segment wycofania, segment ten zwiększy swój rozmiar, zajmując część pozostałej wolnej przestrzeni tabel, czyli zachowując się dokładnie tak samo jak tabela w czasie dodawania nowych wierszy. Jeżeli jednak wymagana dla niego ilość miejsca przekracza rozmiar wolnej przestrzeni tabel, to transakcja nie zostanie wykonana i możesz spodziewać się telefonu od zdenerwowanego użytkownika! Jak zatem zdobyć pewność, że w segmencie wycofania istnieje dostateczna ilość miejsca dla wykonywanych transakcji? Należy kontrolować liczbę powiększeń rozmiarów segmentów wycofania, związaną z wykonywaniem transakcji.

Jeżeli obszary wewnątrz segmentu wycofania są zbyt małe, transakcja będzie się „zawijać" od jednego do innego obszaru w obrębie segmentu. Każda taka operacja wymaga od Oracle wykonania wewnętrznych zadań związanych z przydziałem przestrzeni. Zadania te pochłaniają czas i zasoby systemu. Zatem należy dążyć do sytuacji, w której rozmiar segmentu wycofania jest wystarczający do pomieszczenia całej transakcji i kosztowne wewnętrzne zarządzanie przestrzenią zostaje wyeliminowane.

Można monitorować ilość zawinięć segmentu wycofania pomiędzy obszarami. W jaki sposób to zrobić? Omówimy to w następnym podrozdziale.

Monitorowanie rozmiaru segmentu wycofania

Segmenty wycofania przypisane są do procesów w sposób cykliczny i wszystkie transakcje niezależnie od swoich rozmiarów rywalizują w dostępie do tych samych dostępnych segmentów. Jeżeli po każdym zatwierdzeniu transakcji nie zostanie wydane polecenie `set transaction use rollback segment`, nakazujące kolejnej wykorzystanie specyficznego segmentu, to nie mamy wpływu na przydzielony transakcji segment wycofania. Polecenie `set transaction use rollback segment` musi wystąpić w pierwszej linii transakcji, a `commit` lub `rollback` w ostatniej.

Jeżeli wykorzystywane jest polecenie `set transaction use rollback segment` dla dużej transakcji, powinieneś utworzyć segment wycofania przeznaczony specjalnie dla niej. Jeżeli nie wykorzystujemy `set transaction use rollback segment`, segment wycofania zostanie przydzielony w sposób przypadkowy. Ponieważ większość przypisań segmentów wycofania do transakcji ma charakter losowy, należy wykorzystywać standardowy rozmiar dla wszystkich segmentów wycofania (z wyjątkiem specyficznych, wspomagających duże transakcje). Nie ma żadnej gwarancji, że duży segment wycofania, stworzony specjalnie dla wielkiej transakcji, nie zostanie alokowany dla innego procesu, zanim transakcja ta zostanie uruchomiona. Wiele transakcji może korzystać i korzysta ze wspólnych segmentów wycofania, jeżeli zatem w systemie pracują inni użytkownicy, to nie można mieć stuprocentowej pewności, że to właśnie nasza transakcja jest jedyną, która korzysta z bardzo dużego segmentu.

Jak już wspominaliśmy, pojedynczy segment wycofania może przechowywać dane wielu różnych transakcji. Każda z transakcji wchodzących do segmentu wycofania może wymusić zwiększenie jego rozmiaru. Zatem jeżeli znajduje się w nim wiele małych transakcji, jego rozmiar może się zwiększyć dokładnie tak samo jak w przypadku jednej o dużym rozmiarze.

Segment wycofania, optymalny parametr

Załóżmy, że segment wycofania rozszerzał się i osiągnął punkt, w którym wykorzystuje już całą dostępną przestrzeń. Co się stanie w takim wypadku? Transakcja, która podejmie próbę zwiększenia jego rozmiarów, nie zostanie wykonana. Podobnie będą zachowywać się transakcje wymagające zwiększenia rozmiarów innych segmentów. Jakie możliwości działania mamy w takim wypadku? Istnieje tylko jedna — dodać nowy plik danych do przestrzeni tabel segmentów wycofania i w ten sposób zwiększyć ilość dostępnej przestrzeni.

Mimo to jednak, zanim jeszcze sprawy przybiorą taki obrót, jest coś, co można zrobić w celu zapewnienia dostatecznej ilości miejsca w przestrzeni tabel segmentów wycofania. Kontrolując zwiększanie się segmentów, można je przebudować (pojedynczo, jeżeli to konieczne) i ustalić optymalny rozmiar dla każdego z nich. Możesz wykorzystać parametr optymalny klauzuli `storage` do nadania mu najlepszej wartości w czasie wykonywania poleceń `create rollback segment` lub `alter rollback segment`. Wartość optymalnego parametru segmentów wycofania oraz statystyki dotyczące tych segmentów można znaleźć w perspektywie `V$ROLLSTAT`, który gromadzi informacje o wykorzystaniu wszystkich segmentów wycofania od czasu uruchomienia bazy danych, jak zostało to wyjaśnione w rozdziale 6.

Segmenty wycofania, którym nadano rozmiar optymalny mogą zostać skrócone (ang. *shrink*) po zwiększeniu swojego rozmiaru. Optymalna wielkość segmentu powinna być dobrana tak, by liczba jego rozszerzeń i skróceń, związana z obsługą transakcji, była minimalna. Każde zmniejszenie rozmiaru segmentu wycofania do poziomu jego wielkości optymalnej powoduje spadek wydajności, zatem najlepiej jest wybrać ich wielkość w taki sposób, by powiększanie nie było już konieczne. Mimo to jednak, dla sporadycznych, dużych transakcji wymuszających rozszerzenia, wartość optymalna umożliwia odzyskanie przestrzeni.

Poszukiwanie wartości optymalnych, skróceń i rozszerzeń

Poniższy skrypt wykonuje zapytanie na `V$ROLLSTAT` i `V$ROLLNAME`. Perspektywa `V$ROLLNAME` zawiera nazwy aktywnych segmentów wycofania, podczas gdy w `V$ROLLSTAT` są one identyfikowane jedynie przez swój USN (ang. *Undo Segment Number* — numer segmentu powrotu). Wykonanie kwerendy z tego listingu wyświetli optymalny rozmiar, liczbę skróceń, średni rozmiar na skrócenie i ilość rozszerzeń na segment wycofania.

```
REM Rozszerzanie segmentu wycofania
REM
Column Name format A20
Select Name, Optsize,Shrinks, Aveshrink, Extends
  from V$ROLLSTAT,V$ROLLNAME
 where V$ROLLSTAT.Usn= V$ROLLNAME.Usn;
```

Przykładowe wyniki dla zapytania na `V$ROLLSTAT` pokazano w listingu poniżej:

```
NAME                 OPTSIZE    SHRINKS  AVESHRINK    EXTENDS
-------------------- ---------- ---------- ---------- ----------
SYSTEM                                   0          0          0
RO1                    10485760         4   41943040         32
RO2                    10485760         2   44564480         17
```

Popatrzmy, co dokładnie zawiera ten raport. W bazie danych istnieją trzy aktywne segmenty wycofania. Segment SYSTEM jest wykorzystywany do transakcji słownika danych. Transakcje użytkowników są związane z segmentami R01 albo R02. Każdy z nich posiada rozmiar optymalny 10 MB (10 485 760 bajtów), rozszerzał się powyżej tej wartości i był ponownie do niej skracany, dwa lub więcej razy. Każdorazowo segmenty były skracane przeciętnie o 40 MB. Uważna analiza raportu pokazuje zatem, że optymalny rozmiar segmentów nie został dobrany właściwie dla transakcji, które je wykorzystują. Zbyt często zwiększają one swój rozmiar (49 razy od ostatniego startu bazy danych), a ich średni przyrost przekracza pięciokrotnie rozmiar przyjęty jako optymalny. W takiej sytuacji Oracle wykonuje mnóstwo niepotrzebnej pracy związanej z zarządzaniem przestrzenią. W celu zmniejszenia ilości rozszerzeń segmentów, wartość optymalna powinna zostać zwiększona co najmniej do 40 MB. Systematyczne wykonywanie skryptu dostarczy nam informacji, jaki skutek wywołała ta zmiana. Można również zdecydować się na dodanie do bazy danych dodatkowych segmentów wycofania, co poprawi wykonywanie transakcji.

Kiedy segment wycofania po raz pierwszy zwiększa swój rozmiar ponad wartość optymalną, nie zostaje skrócony. Za drugim razem, gdy wielkość segmentu wzrasta powyżej wartości optymalnej, będzie on skrócony — zapewniając następnej transakcji wymuszenie alokacji nowego obszaru. Pomimo że ustalenie wielkości optymalnej segmentów nie uchroni nas zupełnie przed administrowaniem nimi, pomoże niewątpliwie w ograniczeniu liczby i rodzajów tych czynności.

Z powodu specyfiki, zgodnie z którą zmienia się rozmiar segmentów wycofania, ilość „powiększeń" widoczna w wynikach wykonania poprzedniego skryptu będzie zawsze większa niż ilość „zmniejszeń". Dla przykładu, jeżeli początkowy rozmiar segmentu wycofania jest mniejszy niż przewidziana dla niego wartość optymalna, żądanie nowego obszaru powoduje zwiększenie statystyki powiększeń (ang. *Extends*), lecz skrócenie (*Shrink*) nie jest konieczne. Kiedy segment wycofania zwiększy swój rozmiar powyżej wartości optymalnej, może być skracany jedynie po zakończeniu transakcji, która jest w stanie wymusić jego wielokrotne rozszerzenie.

Do skracania segmentów wycofania do wybranego początkowo rozmiaru można wykorzystać polecenie `shrink rollback segment`. Segment wycofania musi zawierać co najmniej dwa obszary. Jeżeli nie zostanie określony rozmiar, do jakiego ma on zostać zredukowany, domyślnie przyjmowana jest wielkość optymalna. Poniższy listing ilustruje zredukowanie rozmiaru segmentu do wartości optymalnej.

```
Alter rollback segment R01 shrink;
```

Monitorowanie rozmiaru obszaru segmentu wycofania

W celu uproszczenia zarządzania wieloma pozycjami w segmencie wycofania, należy tak skonfigurować rozmiar segmentu, by każdy znajdujący się wewnątrz niego obszarów był wystarczająco duży do wsparcia typowej transakcji.

Kiedy segment wycofania transakcji nie może być przechowany w pojedynczym obszarze, próbuje wykorzystać kolejny obszar segmentu. Obszary segmentu wycofania są przydzielane cyklicznie, zatem transakcja może zostać przewinięta od ostatniego do pierwszego

obszaru segmentu wycofania, jeżeli tylko nie zawiera on aktywnej transakcji, w przeciwnym razie wykonywane jest powiększenie segmentu.

Zapytanie wykonane dla perspektywy V$ROLLSTAT umożliwia ilość okrążeń segmentu wycofania, które zaszły od ostatniego uruchomienia bazy danych. Jeżeli jest ona równa zero, oznacza to, że wielkość obszarów segmentu została dobrana prawidłowo dla wykorzystujących je transakcji. Wartość większa od zera sugeruje konieczność ponownego utworzenia segmentu wycofania z większymi obszarami. Poniżej pokazujemy dwa różne raporty uzyskane po wykonaniu zapytania i przedstawiamy sposób ich interpretacji. Najpierw zapytanie:

```
REM Sprawdzenie przewijania segmentów wycofania
REM
Column Name format A20
Select Name, Optsize,Shrinks, Aveshrink,Wraps, Extends
  From V$ROLLSTAT,V$ROLLNAME
 Where V$ROLLSTAT.Usn= V$ROLLNAME.Usn;
```

Podobnie jak zapytanie dla rozszerzeń segmentu wycofania, skrypt wykorzystuje statystyki umieszczone w V$ROLLSTAT, a nazwy segmentów pobiera z V$ROLLNAME. Pierwszy analizowany raport wygląda tak:

```
NAME                 OPTSIZE    SHRINKS    AVESHRINK  WRAPS  EXTENDS
-------------------- ---------- ---------- ---------- ------ -------
SYSTEM                                  0          0      0       0
RO1                    10485760         4    4194304      0      41      32
RO2                    10485760         2    4456448      0      26      17
```

Jak widać, od uruchomienia bazy zaszło 67 przewinięć (WRAPS). W porównaniu z ilością rozszerzeń, jej wartość nie wydaje się zaskakująca (rozszerzenie zazwyczaj jest poprzedzone przewinięciem). Występowanie rozszerzeń sygnalizuje, że segment wycofania obsługuje transakcje o większym rozmiarze niż te, dla których został on zaprojektowany. Jeżeli segment nie jest w stanie pomieścić informacji związanych z transakcją, to nie zrobi też tego jego pojedynczy obszar. Zatem segment zwiększający swój rozmiar często będzie wykazywać znaczną liczbę przewinięć.

Drugi raport pokazany jest w poniższym listingu i różni się on nieco od pierwszego. Proszę zwrócić uwagę na to, że nie wykazuje on rozszerzeń (EXTENDS), a jedynie przewinięcia.

```
NAME                 OPTSIZE    SHRINKS    AVESHRINK  WRAPS  EXTENDS
-------------------- ---------- ---------- ---------- ------ -------
SYSTEM                                  0          0      0       0
RO1                    10485760         0    4194304      0      41       0
RO2                    10485760         0    4456448      0      26       0
```

W takiej sytuacji optymalny rozmiar segmentu wycofania został dobrany prawidłowo, lecz wielkość jego obszarów jest zbyt mała. Pomimo że segment wycofania jest w stanie pomieścić transakcje, ich pozycje wymagają wielu obszarów wewnątrz segmentu. W takiej sytuacji należy ponownie utworzyć segmenty wycofania z identycznymi wartościami optymalnymi, ale ze zwiększonym rozmiarem obszarów.

Monitorowanie obszarów

W aktualnej wersji oprogramowania maksymalna ilość obszarów jest ograniczona jedynie możliwościami systemu operacyjnego. Możliwe jest utworzenie tabeli z wykorzystaniem opcji `maxextens unlimited` (lub przy użyciu dowolnej liczby takich opcji). Wielkość `maxextens unlimited` nie jest jednak zupełnie nieograniczona. Jeżeli sprawdzimy wartość kolumny `Max_Extens` tabeli `DBA_TABLES` dla tabeli utworzonej z wykorzystaniem opcji `maxextens unlimited`, to okaże się, że znajdująca się tam liczba to 2 147 483 645, co w praktyce oznacza wielkość nieograniczoną. Jeżeli dla przestrzeni tabel nie została zdefiniowana maksymalna liczba obszarów i w czasie tworzenia tabeli w klauzuli `storage` nie zostanie podana maksymalna liczba obszarów, przyjęty zostanie maksymalny domyślny limit o wielkości opartej na rozmiarze bloku bazy danych. Dla przykładu utwórzmy przestrzeń tabel w bazie danych o rozmiarze bloku 8 KB:

```
Create tablespace MY_TS
Datafile 'mydisk1:[Oracle.mydb]my_ts01.dbf' size 10M
default storage(initial 500k
                next 500k
                minextents 1)
 online;
Przestrzeń tabel została utworzona.
```

Nie podaliśmy maksymalnej ilości obszarów. Utwórzmy teraz tabelę i sprawdźmy, jaka będzie dla niej domyślna wartość `maxextens`.

```
create table MY_TAB (Enanme varchar2(20), Empno number);
Tabela została utworzona.
```

Sprawdźmy teraz nadaną jej wartość `maxextens` w tabeli `USER_TABLES`.

```
select Table_Name, Initial_Extent, Max_Extents
  from USER_TABLES
where Table_Name = 'MY_TAB';
TABLE_NAME                     INITIAL_EXTENT MAX_EXTENTS
------------------------------ -------------- -----------
MY_TAB                               524288         505
```

Jak widać, maksymalna ilość obszarów odpowiada wartości domyślnej dla bazy danych o długości bloku 8K.

Do monitorowania wzrostu tabel w bazie danych można wykorzystać poniższy skrypt:

```
prompt SPRAWDZANIE FRAGMENTACJI OBIEKTÓW BAZY DANYCH;
prompt
column Owner noprint new_value Owner_Var
column Segment_Name format a30 heading 'Object Name '
column Segment_Type format a9 heading 'Table/Indx '
column Bytes format 999,999,999 heading 'Bytes used '
column Extents format 999 heading 'No. '
break on Owner skip page 2
ttitle center ' Raport fragmentacji tabel' skip 2 - left 'Utworzył: ' Owner_Var skip 2
select Owner,
       Segment_Name,
       Segment_Type,
       Bytes,
```

```
       Max_Extents,
       Extents
 from DBA_SEGMENTS
where  Extents>50
  and Segment_Type = 'TABLE'
order by Owner, Segment_Name, Segment_Type, Max_Extents
/
ttitle center 'Raport fragmentacji indeksów ' skip 2 - left 'Utworzył : ' Owner_Var skip 2
select Owner ,
       Segment_Name,
       Segment_Type,
       Bytes,
       Max_Extents,
       Extents
 from DBA_SEGMENTS
where Extents >50
  and Segment_Type = 'INDEX'
  and Owner not in ('SYS', 'SYSTEM')
order by Owner,Segment_Name,Segment_Type,Max_Extents
/
```

Skrypt sprawdza dowolną tabelę lub indeks, dla których ilość obszarów przekracza 50. Oczywiście poziom ten można zmienić tak, aby był sensowny dla innego środowiska. Pomysł polega na kontrolowaniu tych tabel, które ze względu na wzrost mogą wymagać większej ilości obszarów, niż została im przydzielona. Wynik wykonania skryptu wygląda tak:

```
SPRAWDZANIE FRAGMENTACJI OBIEKTÓW BAZY DANYCH;

                                    Raport fragmentacji tabel
Utworzył: TEST
Object Name                    Table/Ind  Bytes used  MAX_EXTENTS No.
------------------------------ --------- ------------ ----------- ----
REG_IN                         TABLE       105,881,600         1000  584
REG_OUT                        TABLE        38,666,240         1000  211
                                    Raport fragmentacji indeksów

Utworzył : TEST

Object Name                    Table/Ind  Bytes used  MAX_EXTENTS No.
------------------------------ --------- ------------ ----------- ----
IDX_REB_IN                     INDEX        2,867,200          505   67
IDX_REG_OUT                    INDEX        1,925,120          505   43
IDX_REG_OUT_2                  INDEX        3,072,000          505   71
SYS_C005595                    INDEX        1,269,760          505   30
SYS_C005598                    INDEX        1,720,320          505   40
```

Jak widać, maksymalna ilość obszarów dla tabel `REG_IN` i `REG_OUT` jest większa od wartości domyślnej. W części raportu dotyczącej indeksów widzimy dwa indeksy, których nazwy rozpoczynają się od `SYS_C`. Jeżeli zostanie utworzony indeks bez nazwy, Oracle automatycznie przypisze mu identyfikator `SYS_C` zakończony numerem. Pomimo że nie istnieją żadne ograniczenia w tworzeniu takich indeksów, trudno jest uchwycić ich związek z tabelami, na których zostały zbudowane. Zalecamy jawne nazywanie indeksów w czasie ich tworzenia i wykorzystywanie konwencji nazw, pozwalającej na szybką identyfikację odpowiadających im tabel.

Jako alternatywę możemy uruchomić taki skrypt:

```
set feedback off ;
set term off;
set pagesize 60;
set newpage 0;
set linesize 80;
col Db_Name new_value instance
select 'INSTANCE NAME'  Description, Value Db_Name from V$PARAMETER
  where upper(Name) = 'DB_NAME'
/
ttitle center instance 'RAPORT PRZESTRZENI/TABEL/OBSZARACH'
column Tablespace_Name format a15 heading 'TABLESPACE|NAME'
column Segment_Name format a15 heading 'TABLE|NAME'
column Extents format 99999 heading 'EXTENTS'
column Max_Extents format 99999 heading 'MAXEXTENTS'
column Owner format a10
column Segment_Type  format a7 heading 'TABLE|TYPE'
break on Tablespace_Name skip 2 on report
spool check_extents.log
select Tablespace_Name, Segment_Name,Segment_Type, Owner, Extents, Max_Extents
  from DBA_SEGMENTS
 where Segment_Type not  in ('CACHE', 'ROLLBACK', 'DEFERRED ROLLBACK' )
  and (( Extents/decode(Max_Extents,0,1,Max_Extents))*100>25
   or (Extents >250 and Max_Extents = 2147483645) )
order by Tablespace_Name,Owner, Segment_Name
/
spool off
exit
```

Ta wersja skryptu tworzy raport o wszystkich segmentach (tabelach, indeksach, segmentach chwilowych), które mają już zaalokowane więcej niż 25 procent wartości maksymalnej obszarów (`maxetents`) lub, jeżeli wartość maksymalna jest nieograniczona, w których wypełnionych jest więcej niż 250 obszarów. Polecenie `decode` w klauzuli `where` zabezpiecza wyrażenie przed ewentualnym dzieleniem przez zero. Wyniki wykonania raportu mogą wyglądać tak:

```
                    RAPORT PRZESTRZENI/TABEL/OBSZARACH
TABLESPACE      TABLE           TABLE
NAME            NAME            TYPE   OWNER  EXTENTS - MAXEXTENTS
--------------- --------------- ------- -------------  -----------
USERS           REG_IN          TABLE   TEST       533        1000
DICTMGMT        OVERMAX         TABLE   RACHEL     278  2147483645
```

Niezależnie od monitorowania obiektów, których rozmiar zbliża się do maksymalnej ilości zdefiniowanych dla nich obszarów, powinniśmy wiedzieć, które z nich nie są w stanie w ogóle zwiększyć swoich rozmiarów. Kolejny skrypt umożliwia identyfikację obiektów, dla których następny obszar przekracza ilość wolnego miejsca w przestrzeni tabel.

```
column "OBJECT" format a20
column "TABLESPACE" format a20
column "TYPE" format a6
column "NEXT (M) " format 99999
column "EXTENTS" format  999
set pages 60
ttitle " RAPORT PRZEKROCZENIA KOLEJNYCH OBSZARÓW "
select  Segment_Name "OBLECT", Segment_Type  "TYPE", Extents "EXTENTS",
```

```
       round (Next_Extent/1024/1024)  "NEXT (M)",
       ds.Tablespace_Name "TABLESPACE", round(df.Lrgstext/1024/1024)  "LRGSTEXT"
  from DBA_SEGMENTS ds,
          (select Tablespace_Name "TABLESPACE", max(Bytes)  "LRGSTEXT"
              from DBA_FREE_SPACE
              group by  Tablespace_Name) df
 where  ds.Tablespace_Name  = df."TABLESPACE"
    and Next_Extent > "LRGSTEXT"
    and  ds.Segment_Type not in ('ROLLBACK', 'DEFERRED ROLBACK', 'TEMPORARY')
/
Pią Sty 17                                                          strona    1
                      RAPORT PRZEKROCZENIA KOLEJNYCH OBSZARÓW
OBLECT                  TYPE   EXTENTS   NEXT (M) TABLESPACE            LRGSTEXT
------ ------- ---------- ------------------- ------------------------------------
TAXRETURN                TABLE       1         5    SMALLTBL                    4
```

Raport pokazuje, że następny obszar dla tabeli TAXRETURN ma rozmiar 5 MB, podczas gdy największy dostępny w przestrzeni tabel obszar ma wielkość 4 MB. Jeżeli tabela będzie musiała zwiększyć swoją wielkość, to transakcja, która tego zażąda nie zostanie wykonana z powodu braku wystarczającej ilości wolnego miejsca w przestrzeni tabel. Jeżeli liczba aktualnie alokowanych obszarów jest znaczna, można rozważyć ewentualną przebudowę tabeli z większymi obszarami początkowym i kolejnym. W wypadku gdy w bazie danych istnieją przestrzenie z małymi, średnimi i dużymi rozmiarami obszarów, w grę wchodzi tez zmiana przestrzeni tabel, do której należy tabela.

Wiemy już, kiedy próba rozszerzenia tabeli zakończy się negatywnie, ale jak określić, kiedy będzie ona zwiększać swój rozmiar? Chcemy teraz odpowiedzieć na to pytanie.

Pozostała przestrzeń

Pomimo że skrypty w poprzednim podrozdziale dostarczają informacji o poziomie fragmentacji tabel i indeksów, umożliwiając tym samym porównywanie wyników na przestrzeni wielu dni lub tygodni i obserwowanie częstotliwości powiększeń obiektów, nie dostarczają żadnej informacji o tym, kiedy należy spodziewać się zwiększenia ich rozmiarów.

Dlaczego powinniśmy wiedzieć, jaką przestrzeń zajmują obiekty bazy danych? Otóż wyobraźmy sobie sytuację, w której wszystkie obiekty jednocześnie próbują zwiększyć swoje wymiary. Czy w bazie danych istnieje tyle wolnego miejsca, by sprostać takiej sytuacji, czy też nagle go zabraknie, a kolejni zniecierpliwieni użytkownicy będą sygnalizować brak przestrzeni? Czy w naszej przestrzeni tabel istnieje wystarczająco duży niezajęty obszar zdolny pomieścić największy kolejno alokowany obszar?

Jedynym sposobem na utrzymanie kontroli nad taką sytuacją jest utworzenie tabeli przeznaczonej do przechowywania informacji o historii obiektów i charakterystyk ich przechowywania. Można ją wykorzystać do analizowania trendów i tworzyć na jej postawie dzienne raporty ostrzegające o obiektach, którym brakuje miejsca do rozszerzenia. Pokazany poniżej skrypt wykorzystujemy do utworzenia tabeli informacji historycznych. Wypełniamy tę tabelę informacjami początkowymi o największych i najczęściej aktualizowanych obiektach. Pozostałe skrypty tego podrozdziału służą do wpisywania dziennych statystyk dotyczących obiektów, których zachowanie analizujemy oraz do

raportowania wyników. Oracle dostarcza pakiet DBMS_SPACE, który można znaleźć w podkatalogu */rdbms/admin*, *ORACLE_HOME*. Pakiet ten zawiera procedurę UNUSED_SPACE dostarczającą raport o rozmiarze alokowanej przestrzeni, która nie jest wykorzystywana dla danych. Przed wykorzystaniem tego pakietu trzeba zalogować się do bazy danych i zainstalować go w przedstawiony niżej sposób (przykład dotyczy systemu Unix):

```
sqlplus "/ as sysdba"
@$ORACLE_HOME/rdbms/admin/dbmsutil.sql
@$ORACLE_HOME/rdbms/admin/prvtutil.plb
```

Można teraz wykorzystać pakiet DBMS_SPACE.

```
create table EXTGROW
 (Segname        VARCHAR(81)  NOT NULL
 ,Segown         VARCHAR(30)  NOT NULL
 ,Growdate       DATE         NOT NULL
 ,Segtype        VARCHAR2(17)
 ,Segtbs         VARCHAR2(30)
 ,Exts           NUMBER
 ,Nextext        NUMBER
 ,Tbsfree        NUMBER
 ,Hwm            NUMBER
 ,Totbytes       NUMBER
 ,Usedbytes      NUMBER
 ,Contigfree     NUMBER
 ,Freepct        NUMBER
 ,Tbsexts        NUMBER
)
tablespace TOOLS
 storage (initial     819200
          next        819200
          minextents  10
          maxextents  505
          pctincrease 0
)
/
alter table EXTGROW add constraint EXTGROW_PK
      primary key (Segname,Segown,Growdate)
      using index tablespace TOOLS_IDX
      storage (initial     819200
               next        819200
               minextents  10
               maxextents  505
               pctincrease 0);
```

Stopień wzrostu tej tabeli zależy od ilości analizowanych tabel i indeksów. Część informacji dotyczy przestrzeni tabel, w których znajdują się obiekty, pozostałe to dane na temat samych obiektów.

Do wstępnego wypełnienia tej tabeli można wykorzystać pokazany niżej skrypt. Po uruchomieniu prosi on o nazwę użytkownika tabeli i indeksu. Można go uruchamiać wielokrotnie, dla każdego użytkownika bazy danych, którego obiekty zamierzamy kontrolować.

```
Insert into EXTGROW(Segname,Segown,Growdate,Segtype,Segtbs,Exts,
                    Nextext,Tbsfree,Hwm,Totbytes,Usedbytes,
                    Contigfree,Freepct, Tbsexts)
```

```
    select Segment_Name,Owner,
           to_date(to_char(sysdate-1,'YYYYMMDD'),'YYYYMMDD'),
           Segment_Type,Tablespace_Name,0,0,0,0,0,0,0,0,0
     from DBA_SEGMENTS
    where Owner= upper('&s1')
     and Segment_Type in ('TABLE','INDEX')
/
```

Codzienne uzupełnianie danych umożliwia kolejny skrypt, wymagający daty w formacie YYYMMDD jako parametru wejściowego. Należy podać tu aktualną datę, procedura zakłada, że odpowiada ona pozycjom w tabeli EXTGROW z poprzedniego dnia.

```
declare
  Total_Blocks     NUMBER;
  Total_Bytes      NUMBER;
  Unused_Blocks    NUMBER;
  Unused_Bytes     NUMBER;
  Lue_File_Id      NUMBER;
  Lue_Block_Id     NUMBER;
  Last_Used_Block  NUMBER;
  T_Name           EXTGROW.Segname%TYPE;
  Ts_Name          EXTGROW.Segtbs%TYPE;
  Hwm              NUMBER;
  T_Exts           NUMBER;
  T_Own            EXTGROW.Segown%TYPE;
  N_Exts           NUMBER;
  T_Free           NUMBER;
  N_Ext_Bytes      NUMBER;
  L_Chunk          NUMBER;
  Db_Blk_Size      NUMBER;

  cursor TABSPACE is
     select Table_Name,
            dt.Owner,
            dt.Tablespace_Name,
            dt.Next_Extent,
            Extents,
            dt.Blocks
       from DBA_TABLES dt, EXTGROW eg, DBA_SEGMENTS ds
     where  Table_Name = segname
        and dt.Owner = Segown
        and ds.Owner = Segown
        and Segment_Name = Segname
        and Growdate = to_date('&1','YYYYMMDD')-1 ;

  cursor INDSPACE is
      select  Index_Name,
              di.Owner,
              di.Tablespace_Name,
              di.Next_Extent,
              Extents
        from DBA_INDEXES di, EXTGROW eg, DBA_SEGMENTS ds
       where  Index_Name =Segname
          and di.Owner =Segown
          and ds.Owner =Segown
          and Segment_Name =Segname
          and Growdate = to_date('&1','YYYYMMDD')-1 ;
```

```
begin
 select Value
      into Db_Blk_Size
      from SYS.V_$PARAMETER
    where upper(Name) = 'DB_BLOCK_SIZE' ;
open TABSPACE;
 loop
    fetch TABSPACE
     into T_Name,
          T_Own,
          Ts_Name,
          N_Ext_Bytes,
          T_Exts,
          Hwm;
   exit when TABSPACE%NOTFOUND;

  DBMS_SPACE.UNUSED_SPACE(T_Own,T_Name,'TABLE',Total_Blocks,Total_Bytes,
                     Unused_Blocks,Unused_Bytes,Lue_File_Id,Lue_Block_Id,
                     Last_Used_Block);
 Hwm:=Hwm*Db_Blk_Size;
 select count(*),
        sum(Bytes),
        max(Bytes)
  into N_Exts,T_Free,L_Chunk
   from DBA_FREE_SPACE
 where Tablespace_Name= Ts_Name;

insert into EXTGROW (Segname,Segown,Growdate,SegType,Segtbs,Exts,
                     Nextext,TbsFree,Hwm,Totbytes,Usedbytes,Contigfree,
                     Freepct,Tbsexts)
            values (T_Name,T_Own,to_date('&1','YYYYMMDD'),'TABLE',Ts_Name,
                    T_Exts,N_Ext_Bytes,T_Free,Hwm,Total_Bytes,
                    (Total_Bytes-Unused_Bytes),L_Chunk,
                    (Unused_Bytes/Total_Bytes)*100,N_Exts);
   commit;
end loop;
 close TABSPACE;

 open INDSPACE;
 loop
    fetch INDSPACE
     into T_Name,
          T_Own,
          Ts_Name,
          N_Ext_Bytes,
          T_Exts;
    exit when INDSPACE%NOTFOUND;
     DBMS_SPACE.UNUSED_SPACE(T_Own,T_Name,'INDEX',Total_Blocks,Total_Bytes,
                     Unused_Blocks,Unused_Bytes,Lue_File_Id,Lue_Block_Id,
                     Last_Used_Block);
    select count(*),
           sum(Bytes),
           max(Bytes)
     into N_Exts,T_Free,L_Chunk
        from DBA_FREE_SPACE
      where Tablespace_Name= Ts_Name;
```

```
        insert into EXTGROW (Segname,Segown,Growdate,SegType,Segtbs,Exts,
                             Nextext,TbsFree,Hwm,Totbytes,Usedbytes,Contigfree,
                             Freepct,Tbsexts)
                     values (T_Name,T_Own,to_date('&1','YYYYMMDD'),'INDEX',Ts_Name,
                             T_Exts,N_Ext_Bytes,T_Free,Hwm,Total_Bytes,
                             (Total_Bytes-Unused_Bytes),L_Chunk,
                             (Unused_Bytes/Total_Bytes)*100,N_Exts);
      commit;
  end loop;
   close INDSPACE;
  end;
 /
```

Skrypt ten jest wykorzystywany do wypełnienia tabeli EXTGROW informacjami o przestrzeni dostępnej i wykorzystanej przez obiekty, których nazwy zostały w niej umieszczone dla przekazywanej jako parametr daty. Kod PL/SQL, który nie jest przechowywany w bazie danych jako procedura lub pakiet, nazywany jest anonimowym blokiem PL/SQL i jest kompilowany przed każdym wykonaniem. Jeżeli zamierzamy uruchamiać go częściej niż raz dziennie, wskazane jest przekształcenie go do postaci procedury składowanej, tak by Oracle mógł wykonywać go szybciej, pomijając etap kompilacji.

Posiadając wszystkie potrzebne informacje, potrzebujemy przedstawić je w jasnej i czytelnej formie. Jest to zadanie kolejnego skryptu, który pobiera i formatuje wiersze tabeli w oparciu o przekazane mu parametry. Wyniki jego działania są zapisywane do pliku spacerpt_.sql.

```
set pagesize 50 trimspool on linesize 250 verify off feedback off
set echo off term off
col Segname      format a20           heading   'OBJECT'
col Segown       format a10           heading   'OWNER'
col Segtype      format a5            heding    'TYPE'
col Segtbs       format a10           heading   'TABLESPACE'
col Exts         format 999           heading   'OBJEXTS'
col Nextext      format 9,999,999,999 heading   'NEXT EXTENT'
col Tbsfree      format 9,999,999,999 heading   'TOTAL TBS FREE'
col Hwm          format 9,999,999,999 heading   'HIGHWATER MARK'
col Totbytes     format 9,999,999,999 heading   'TOT ALOC SPC'
col Usedbytes    format 9,999,999,999 heading   'USED SPACE'
col Contigfree   format 9,999,999,999 heading   'LARGEST FREE'
col Frepct       format 999,9         heading   '% FREE'
col Tbsexts      format 9999          heading   'TBSEXTS'
compute sum of Nextext on Segtbs
break on report on Segtbs skip 1 on contigfree on Tbsexts on Segown
select  Segtbs,
        Tbsfree,
        Contigfree,
        Tbsexts,
        Segown,
        Segname,
        Segtype,
        Hwm,
        Totbytes,
        Usedbytes,Exts,
        Nextext,
        Freepct
   from EXTGROW
```

```
  where Freepct<&1 and Growdate = to_date('&2','YYYYMMDD')
     and Segtbs like upper('%&3%')
 order by Segtbs,SegOwn,SegName

   set concat +
   spool spacerpt_&2.log
/
exit
```

Po uruchomieniu skrypt wymaga trzech parametrów wejściowych: procentowej ilości wolnej przestrzeni, którą należy wziąć pod uwagę, daty analizy w formacie YYYYMMDD i nazwy przestrzeni tabel. Jeżeli zamierzasz zobaczyć dane dla wszystkich obiektów, którym pozostało 10 procent lub mniej wolnej przestrzeni 10 marca 2002 roku, to należy uruchomić skrypt z argumentami:

```
@@spacerpt  10  20020310 "%"
```

Przykładowy wynik jego działania pokazano poniżej. Ilość uzyskiwanych danych jest znaczna i listing zawija się w kilku liniach.

```
TABLESPACE TOTAL TBS FREE   LARGEST FREE TBSEXTS OWNER      OBJECT               SEGTYPE
HIGHWATER MARK   TOT ALOC SPC     USED SPACE OBJEXTS    NEXT EXTENT % FREE
---------- -------------- -------------- ------- ---------- -------------------- --------
--------- -------------- -------------- -------------- ------- -------------- ------
SYSTEM        126,087,168    126,087,168       1 SYS        ACCESS$              TABLE
1,445,888      1,445,888      15        102,400      0
              126,087,168                                   ARGUMENT$            TABLE
2,265,088      2,265,088      23        102,400      0
              126,087,168                                   COM$                 TABLE
421,888        401,408       5         102,400      5
              126,087,168                                   CON$                 TABLE
45,056         45,056       3          32,768       0
              126,087,168                                   DEPENDENCY$          TABLE
2,674,688      2,633,728      27        102,400      2
              126,087,168                                   DUAL                 TABLE
**********               ************** ******* **********
-------------
sum
3,956,736
```

Polecenie SELECT wykorzystuje zmienne w miejscu wpisywanych na stałe wartości, tak więc nie trzeba po zmianie daty lub innych danych wejściowych, wykonywać jego poprawek. Jesteśmy zdecydowanymi zwolennikami takiego sposobu kodowania. Skrypt tworzony jest raz, bez potrzeby wracania do niego i wykonywania drobnych poprawek.

Po podstawieniu wartości 100 do pierwszego parametru (&1) można zobaczyć każdą tabelę i indeks dla daty podanej jako drugi parametr. Skrypt oczekuje jej w formacie YYYMMDD i dołącza do dziennika wyjściowego. Wykorzystanie takiego formatu umożliwia sortownie plików raportu pod względem daty. Ponieważ Oracle nie oznacza końca wczytanego parametru, skrypt zmienia znak łączenia na + przed wywołaniem zmiennej włączonej do nazwy pliku raportu.

Znak % wykorzystywany jest przez Oracle jako wieloznacznik (ang. *wildcard*), dopuszczający dowolną wartość. Parametr &3 ujęty pomiędzy % informuje, że wybrane mają być wszystkie przestrzenie tabel posiadające w swojej nazwie podany pomiędzy znakiem %

fragment. Przekazanie parametru % jako wartości trzeciego parametru powoduje wybranie wszystkich przestrzeni tabel.

Przeanalizujmy teraz nieco dokładniej informacje uzyskane w wyniku wykonania polecenia SELECT. Każda z linii raportu zawiera ich znaczną ilość i na wydruku jest przedstawiona w dwóch kolejnych liniach tekstu. Bez dodatkowych wskazówek interpretacja wyników może okazać się trudna.

```
TABLESPACE TOTAL TBS FREE   LARGEST FREE TBSEXTS OWNER      OBJECT              SEGTYPE
HIGHWATER MARK   TOT ALOC SPC     USED SPACE OBJEXTS    NEXT EXTENT % FREE
---------- ------------- ------------- ------- ---------- ------------------- --------
--------- ------------- ------------- ------------- ------- ------------- ------
USERS        603,635,712    66,314,240      327 TEST          IDX_REG_IN         INDEX
               327,680       327,680        8              40,960    .00
                                                           IDX_REG_OUT        INDEX
             2,621,440     2,261,440       60              40,960
  44,998,656 14,049,280    14,049,280      332              40,960
                                                             REG_OUT
TABLE
  15,351,808 52,510,720     52,428,800      224           1,048,576     .16
************ ************   ************   ********   ***************
                                                      -----------------
  sum                                                    1,171,456
```

Co możemy stwierdzić na podstawie tego wydruku? Przestrzeń tabel o nazwie USERS ma 603,635,712 wolnych bajtów z największym ciągłym obszarem o rozmiarze 66,314,240. Przestrzeń tabel jest znacznie pofragmentowana i całkowita wolna przestrzeń dostępna jest w 327 obszarach. Informacje te są wydrukowane tylko w pierwszej linii z powodu wydanego w raporcie polecenia break, łatwiej jest w ten sposób określić, kiedy pojawia się zmiana.

Żaden z obiektów pokazanych na listingu nie ma kolejnego obszaru większego niż największy wolny w przestrzeni tabel. Suma następnych obszarów dla procentowej ilości pozostałego wolnego miejsca, wynosi 1,171,456 bajtów, które mogą zostać łatwo umieszczone w całkowitej wolnej przestrzeni tabel lub w jej największym niezajętym obszarze. Wszystkie obiekty wykazują mały procent wolnego miejsca, co sygnalizuje, że wkrótce może nastąpić ich rozszerzenie. Lista może, ale nie musi, zawierać wszystkie obiekty przestrzeni tabel. Jedynym sposobem zdobycia absolutnej pewności, że dysponujemy wystarczającą ilością wolnego miejsca, jest uruchomienie raportu z wartością jego pierwszego parametru równą 100, co umożliwia obserwację wszystkich obiektów ze stuprocentową lub mniejszą wartością pozostałej wolnej przestrzeni.

Pliki śladu

Kiedy w czasie sesji użytkownika lub podczas działającego w tle procesu wystąpi błąd wewnętrzny, Oracle tworzy plik śladu (ang. *trace file*), który ułatwia debugowanie i rozpoznanie jego rodzaju. Administrator bazy danych (DBA) może zdefiniować miejsce jego utworzenia i maksymalny rozmiar, wykorzystując poniższe parametry inicjujące:

- MAX_DUMP_FILE_SIZE — maksymalna, możliwa do osiągnięcia, wielkość pliku śladu.

- BACKGROUND_DUMP_DEST — lokalizacja pliku śladu tworzonego przez pracujące w tle procesy Oracle.
- USER_DUMP_DEST — położenie plików śladów tworzonych poza sesjami użytkowników.

Pliki śladu utworzone przez pracujące w tle procesy otrzymują nazwy zawierające określenie tworzącego je procesu, nazwy plików generowanych przez procesy użytkowników uwzględniają ich identyfikatory (ID).

Dobrym zwyczajem jest przeniesienie katalogów dla plików śladów ze standardowej lokalizacji *ORACLE_BASE* na odrębny dysk. Pliki śladów nie są automatycznie kasowane i ich liczba może znacznie wzrosnąć. W większości będą to prawdopodobnie stare, aktualnie bezwartościowe pliki. Z tego powodu należy zastanowić się nad czasem ich utrzymywania i częstotliwością opróżniania katalogu. Jedyną zaleta wynikającą z długiego utrzymywania plików śladu jest szansa uchwycenia momentu pojawienia się problemu.

Wystąpienie błędu wewnętrznego, niezależnie od utworzenia pliku śladu, powoduje zapis do dziennika ostrzeżeń. Umieszczana tam linia zawiera numer błędu Oracle, a także nazwę i lokalizację pliku śladu. Jeżeli po sprawdzeniu numeru błędu okaże się, że nie jest to błąd programowy lub błąd użytkownika, zalecane jest wywołanie strony pomocy technicznej Oracle (MetaLink) i zalogowanie się tam jako iTAR. Użytkownik zostanie poproszony o przesłanie pliku śladu do analizy.

Poniżej umieściliśmy dwa przykładowe pliki śladów. Pierwszy z nich pochodzi z katalogu plików śladów użytkownika, który prawdopodobnie włączył śledzenie wykonywania wydawanych przez siebie poleceń SQL. Jeżeli występuje tu jakiś problem, to z całą pewnością jego usunięcie nie należy do pomocy technicznej.

```
Dump file c:\Oracle\admin\bd901\udump\ORA00784.TRC
Thu Jan 16 20:16:14 2002
ORACLE V9.0.1.2.1 - Production vsnsta=0
vsnsql=10 vsnxtr=3
Windows 2000 V5.0, Service Pack 2, CPU type 586
Oracle9i Enterprice Edition Release 9.0.1.2.1 - Production
JServer Release 9.0.1.2.0 - Production
Windows 2000 V5.0, Service Pack 2, CPU type 586
Instance name: db901

Redo thread mounted by this instance: 1

Oracle process number: 13

Windows thread id: 784, image: ORACLE.EXE
*** 2002-01-16 20:16:14.000
*** SESSION ID:(8.65) 2002-01-16 20:16:14.000
APPNAME mod='SQL*Plus' mh=3669949024 act='' ah=4029777240
=====================
PARSING IN CURSOR #1 len=32 dep=0 uid=29 oct=42 lid=29 tim=3302036408 hv=1197935484
ad='b31134b0'
alter session set sql_trace=true
END OF STMT
EXEC #1:c=0,e=0,p=0,cr=0,cu=0,mis=1,r=0,dep=0,og=4,tim=3302026408
```

```
*** 2002-01-16 20:42:16.000
=====================
PARSING IN CURSOR #2 len=43 dep=1 uid=0 oct=3 lid=0 tim=3329481408 hv=2454029093
ad='b3fb7058'
select user#,type# from user$ where name=:1
END OF STMT
=====================
PARSING IN CURSOR #11 len=92 dep=1 uid=0 oct=3 lid=0 tim=3329786408 hv=2407357363
ad='b314884c'
select distinct(-privilege#),nvl(option$,0) from sysauth$ where grantee#=:1 and
privilege#<0
END OF STMT
PARSE #11:c=0,e=1000,p=0,cr=0,cu=0,mis=1,r=0,dep=1,og=0,tim=3329786408
EXEC #11:c=0,e=5000,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=4,tim=3329791408
FETCH #11:c=0,e=0,p=0,cr=2,cu=0,mis=0,r=0,dep=1,og=4,tim=3329791408
STAT #11 id=1 cnt=0 pid=0 pos=0 obj=0 op='SORT UNIQUE '
STAT #11 id=2 cnt=0 pid=1 pos=1 obj=91 op='TABLE ACCESS BY INDEX ROWID
```

Drugi przykład zawiera niewielki fragment pliku śladu umieszczonego w katalogu śladów procesów. Ilustruje on zapis błędu ORA-600, po którego wystąpieniu należy bezwzględnie zawiadomić pomoc techniczną.

```
/u03/home/oracle/product/8.0.5/rdbms/prod_ora_10760.trc
   Oracle8 Enterprise Edition Release 8.0.5.0.0 -Production
   PL/SQL Release 8.0.5.0.0 -Production
   ORACLE_HOME =/u03/home/oracle/product/8.0.5
   System name:      SunOS
   Node name:  x2
   Release:    5.7
   Version     Generic_106541-08
   Machine:    sun4u
   Instance name:   PROD
   Redo thread mounted by this instance: 0 <none>
   Oracle process number:  0
   Unix process pid   10760,  image: oralePROD

   *** 2002.01.23.23.32.22.000
   ksedmp: internal or fatal error
   ORA-00600: internal error code, arguments: [SKGMBUSY], [1], [0], [0], [0], [], [], []
   Current SQL information unavailable - no session.
```

Jak widzieliśmy w pierwszym przykładzie, możliwe jest wygenerowanie pliku śladu przeznaczonego specjalnie do celów dostrajania zapytań. Umożliwia to opcja `sql_trace = TRUE`, ustawiona w pliku parametrów inicjujących lub w czasie sesji użytkownika. Pliki śladów tego typu są wykorzystywane przez TKPROF do analizy wyrażeń SQL. Jeżeli jednak opcja ta jest włączona w większej ilości wypadków niż sesja pojedynczego użytkownika, pociąga za sobą obniżenie wydajności. Zalecamy zatem jej wykorzystywanie w przypadku specyficznej kontrolowanej sesji albo ustawienie jej na TRUE dla całej bazy danych w ograniczonym i kontrolowanym przedziale czasu. Druga ewentualność wymaga zatrzymania i powtórnego uruchomienia bazy danych, zatem spadek wydajności jest podwójny. Po pierwsze wiąże się z ustawieniem opcji na TRUE (kiedy nadajemy jej wartość TRUE), po drugie z wykonanymi dwa razy procesami zatrzymania i uruchomienia bazy danych (wyłączenie i włączenie przy zmianie wartości na FALSE). Z całą pewnością można stwierdzić, że nie jest wymagana dłuższa praca bazy danych z wartością TRUE. Dokładniejszy opis TKPROF zawiera rozdział 13.

Status sesji użytkownika

Perspektywa V$SESSION umożliwia sprawdzenie statusu sesji każdego z aktualnych użytkowników bazy danych. Poniższy skrypt zwraca ilość sesji zgrupowanych ze względu na ich status w bazie danych. Dlaczego należy kontrolować tę wartość? Administrator, przypisując wartość parametrowi sessions w pliku inicjującym, jawnie lub niejawnie, ustala limit sesji dla bazy danych.

Jeżeli w bazie danych istnieje znaczna liczba sesji określonych jako KILLED lub SNIPED, która nie maleje z upływem czasu, oznacza to, że liczba sesji możliwych do utworzenia maleje. Jedynym sposobem usunięcia takich sesji jest zamknięcie i powtórne uruchomienie bazy danych.

```
    select Status,Count(*)
     from V$SESSION
        group by Status
/
```

Zwracana jest lista:

```
STATUS     COUNT(*)
-------- ----------
ACTIVE           26
INACTIVE        160
KILLED           30
CACHED            5
SNIPED            3
```

Wartości kolumny Status mogą powodować nieporozumienia. Wartość INACTIVE niekoniecznie oznacza, że proces użytkownika nie jest aktywny. Stwierdza on tylko, że w chwili wykonania zapytania proces nie realizował polecenia SQL. Sesje oznaczone jako ACTIVE wykonują aktualnie zapytania SQL, KILLED są przeznaczone do usunięcia z bazy danych, SNIPPED są aktualnie nieaktywne i oczekują na działanie klienta. Sesje CACHED są obsługiwane przez Oracle*XA, zewnętrzny interfejs umożliwiający obsługę globalnych transakcji przez manager transakcji inny niż Oracle. Jeżeli liczba sesji typów KILLED, SNIPPED i CACHED nie zmienia się lub systematycznie się zwiększa, sygnalizuje to problem w bazie danych, a kolejne sesje są niepotrzebnie zajmowane. W takiej sytuacji pozostaje jedynie zatrzymanie i ponowne uruchomienie bazy danych.

Monitorowanie modyfikacji obiektów

Sprawdzenie zmian wprowadzonych do obiektów bazy danych umożliwia perspektywa słownika danych USER_OBJECTS lub DBA_OBJECTS. Dlaczego powinny nas interesować terminy ostatnich zmian w obiektach? Jeżeli system bazy danych jest środowiskiem eksploatacyjnym, zapewne chcesz wiedzieć, kto i kiedy wykonuje takie zmiany. Można w tym celu wykorzystać obserwację uprawnień (ang. *privilege auditing*), która jednak nie informuje, jaki obiekt został zmieniony. Perspektywy DBA_OBJECTS i USER_OBJECTS zawierają kolumnę o nazwie Last_DDL_Time, zawierającą czas ostatniej modyfikacji obiektu.

Ponieważ jednak Oracle uważa dodanie uprawnień lub indeksu za modyfikacje obiektu, pole to może nie zawierać czasu zmiany struktury samego obiektu. W kolumnie znacznika czasowego `Timestamp` obu perspektyw znajduje się czas modyfikacji zapisany jako `VARCHAR(75)`, który należy przekształcić na datę. Wskazane jest stworzenie uruchamianego raz dziennie zadania wsadowego wykonującego poniższy skrypt SQL:

```
col Owner        format a15
col Object_Name  format a20
col Timestamp    format a19
col Status       format a8
col Object_Type  format a10
select Owner, Object_Name, Object_Type,Status, Timestamp
   from DBA_OBJECTS
    where SUBSTR(Timestamp,1,10)= TO_CHAR(sysdate-1,'YYYY-MM-DD')
    order by Owner,Object_Name
/
```

Uruchomienie tego skryptu umożliwia pokazanie wszystkich obiektów, zmienionych w poprzednim dniu, uporządkowanych według właścicieli. Raport zawiera kolumnę `Status`, ponieważ procedury, funkcje i pakiety są obiektami, których zmiana może je unieważnić. Przykładowe wyniki wykonania skryptu wyglądają następująco:

```
OWNER           OBJECT_NAME          OBJECT_TYP   STATUS      TIMESTAMP
--------------- -------------------- ---------- -------- -------------------
SYS             DBASE_TEMP           TABLE         VALID    2002-01-17:10:36:45
SYS             TSPACE_TEMP          TABLE         VALID    2002-01-17:20:49:14
SYS             IDX_REG_IN           INDEX         VALID    2002-01-17:20:49:16
SYS             TEMP_IN              PACKAGE       VALID    2002-01-17:20:49:16
SYS             TEMP_IN              PACKAGE_BODY  INVALID  2002-01-17:20:49:15
```

Raport sygnalizuje, że albo skompilowana postać pakietu `TEMP_IN` zawiera błędy, albo uległa zmianie jedna z tabel, do których się odwołuje, ponieważ status ma wartość `INVALID`. W takiej sytuacji pierwszy krok polega na próbie rekompilacji pakietu:

```
alter package temp_in recompile;
```

Jeżeli po tej operacji nadal występują błędy, należy skontaktować się z odpowiedzialnym za pakiet programistą i powierzyć mu usunięcie problemu.