STUDIA INFORMATICA 2003
Volume 24 Number 1 (52)

Aleksandra WERNER
Politechnika Śląska, Instytut Informatyki

STROJENIE BAZY DANYCH W SYSTEMIE ORACLE

Streszczenie. Artykuł zawiera informacje na temat poziomów strojenia związanych ze środowiskiem Oracle: strojenia na poziomie systemu operacyjnego, instancji bazy danych Oracle oraz strojenia na poziomie poleceń SQL. Każdy z poziomów został uzupełniony o praktyczne wskazówki dla administratorów danych, jak również o przykłady poleceń SQL, pozyskujących potrzebne dane ze słownika bazy.

Słowa kluczowe: strojenie, wydajność, baza danych, SZRBD Oracle, struktura fizyczna i logiczna bazy danych Oracle, monitorowanie awarii.

ORACLE DATABASE TUNING

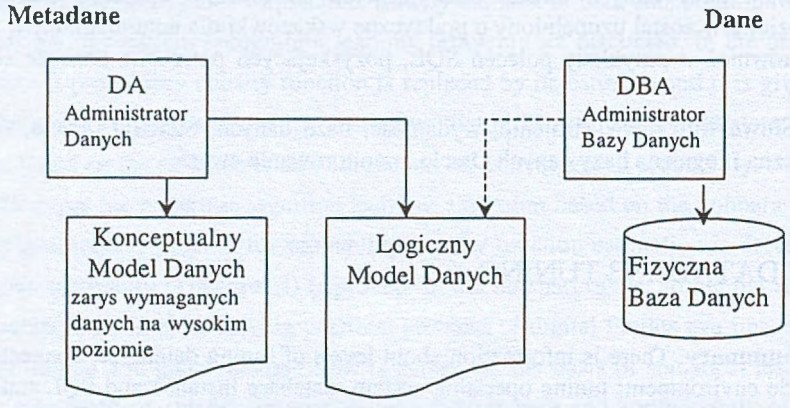
Summary. There is information about levels of tuning databases, connected with Oracle environment: tuning operating system, database instance and SQL statements, in this article. Each level is completed by useful directions for database administrators and by SQL statement examples, exploring adequate data from database dictionary.

Keywords: tuning, performance, database, DBMS Oracle, physical and logical structure of Oracle database, damage monitoring

1. Wprowadzenie

System zarządzania bazą danych jest oprogramowaniem, pozwalającym użytkownikom (np. końcowym lub programistom) współdzielić dane i zarządzać nimi, oraz dostarczającym procedur do tworzenia, aktualizacji, zapisywania informacji w repozytorium (tzn. w bazie danych). System zarządzania jest również odpowiedzialny za integralność i bezpieczeństwo danych, kontrolę dostępu do danych, optymalizację, odtwarzanie i wycofywanie transakcji.

Biorąc pod uwagę fakt, że dane należą do najcenniejszych aktywów przedsiębiorstwa, niezbędne jest, aby na wyższym szczeblu zarządzania była osoba właściwie interpretująca dane, a jednocześnie znająca potrzeby przedsiębiorstwa w tym zakresie. Osobą tą jest administrator danych (ang. DA – Data Administrator). Administracja danymi rozgranicza aspekt biznesowy zarządzania źródłami danych od technologii wykorzystywanej do zarządzania danymi. Do zadań administratora danych należy przede wszystkim podejmowanie decyzji o tym, które dane powinny być przechowywane, a następnie określenie zasad utrzymywania danych i postępowania z przechowywanymi danymi. DA jest odpowiedzialny za zrozumienie słownika biznesowego i przetłumaczenie go na model logiczny (dostarczenie szczegółów o typach danych, długościach, powiązaniach, itp.). Reasumując, DA zajmuje się metadanymi, dostarczającymi kontekstu, w którym dane zapisywane w bazie mogą być zrozumiane (rys. 1).



Rys. 1. Wpływ DBA i DA na kształt bazy danych
Fig. 1. Database affected by DBA and DA

Osoba technicznie odpowiedzialna za realizację decyzji administratora danych nazywana jest administratorem bazy danych (ang. DBA – DataBase Administrator). Zadaniem DBA jest tworzenie rzeczywistej bazy danych i implementacja technicznych sposobów kontroli, potrzebnych do realizacji różnych decyzji podejmowanych przez administratora danych. Ponadto, DBA odpowiada za zapewnienie odpowiedniej wydajności systemu i realizację szeregu innych usług technicznych [3, 1].

DBA mając centralną kontrolę nad bazą danych może również sprawić, by reprezentacja danych spełniała wszystkie wymagane standardy (np. przemysłowe lub międzynarodowe). Standaryzacja reprezentacji danych jest szczególnie pożądana, gdyż jest pomocna w wymianie danych lub przenoszeniu danych między systemami (co występuje np. w przetwarzaniu rozproszonym).

DBA ma zwykle do dyspozycji zespół programistów systemowych i asystentów technicznych, jednak w celu uproszczenia terminologii wygodnie jest przyjąć, że DBA to jedna osoba.

Zakres obowiązków administratorów: danych i baz danych przedstawia tabela 1.

Tabela 1

Zakres obowiązków DBA i DA

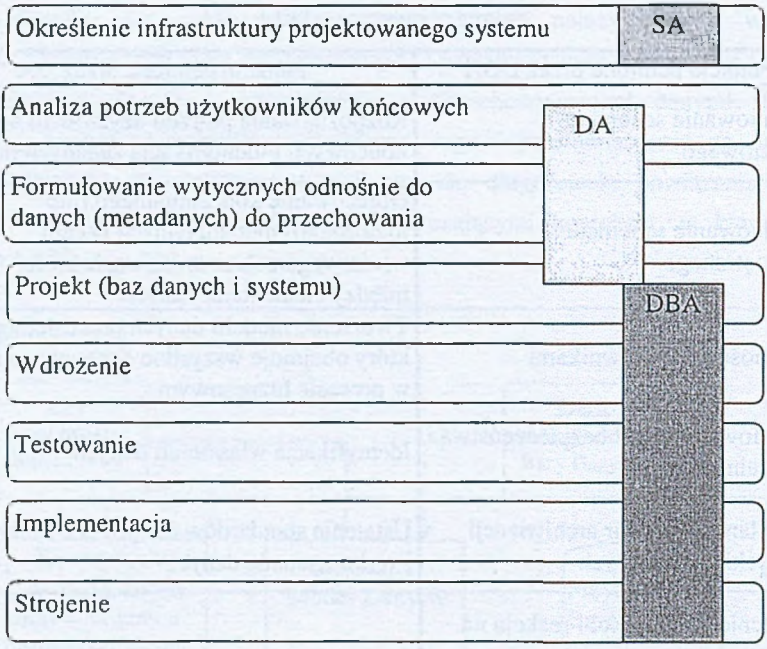
Funkcje pełnione przez DBA	Funkcje pełnione przez DA
Definiowanie schematu pojęciowego	Rozpoznawanie potrzeb użytkowników końcowych i identyfikacja żądanych danych
Definiowanie schematu wewnętrznego	Opracowanie konceptualnego (lub logicznego) modelu danych w celu właściwego zobrazowania zależności między elementami danych
Łączność z użytkownikami	Tworzenie modelu danych przedsiębiorstwa, który obejmuje wszystkie dane używane w procesie biznesowym
Definiowanie reguł bezpieczeństwa i integralności	Identyfikacja właścicieli danych
Określanie procedur archiwizacji i odzyskiwania danych	Ustalenie standardów dla potrzeb kontroli i wykorzystania danych
Śledzenie wydajności i reakcja na zmiany	
Migracja nowej bazy ze środowiska testowego do końcowego	

Często uszczegóławia się sposób podejmowania poszczególnych działań przez administratorów baz danych wyróżniając: DBA biernych (ang. reactive) – rozwiązujących problemy wtedy, gdy się pojawią, i czynnych – wdrażających pewne procedury w celu uniknięcia problemów przed ich wystąpieniem (tzn. opracowujących strategiczny plan ochrony baz danych w przedsiębiorstwie, obejmujący wszystkie fazy cyklu życia systemu informatycznego).

W niektórych przedsiębiorstwach z procesu administrowania bazą danych wyodrębnia się etap implementacji Systemu Zarządzania Bazami Danych (ang. DBMS – DataBase Management System), powierzając jego realizację administratorowi systemu (ang. SA – System Administrator). SA jest wówczas odpowiedzialny za instalację, dostrojenie (ang. setup), zabezpieczanie i utrzymywanie DBMS, czyli musi mieć rozległą wiedzę o tych

komponentach systemu informatycznego, które w sposób bezpośredni wiążą się z instalacją przyjętego oprogramowania (np. system operacyjny, protokoły sieciowe, sprzęt, itp.).

Obecność wyspecjalizowanych (grup) użytkowników na poszczególnych etapach cyklu życia systemu informatycznego przedstawia rysunek 2.



Rys. 2. Zakres obowiązków DBA, DA oraz SA
Fig. 2. DBA, DA and SA responsibilities

2. Struktura bazy danych Oracle

2.1. Struktura fizyczna bazy danych

Na fizyczną strukturę bazy danych Oracle'a składają się następujące pliki:

- Plik inicjalizacyjny (i konfiguracyjny),
- Pliki dziennika powtórzeń (co najmniej 2),
- Pliki kontrolne,
- Pliki z danymi.

2.1.1. Plik inicjalizacyjny

Plik inicjalizacyjny umożliwia:

- optymalizację wydajności przez dostosowanie struktury pamięci (np. liczba buforów danych w pamięci),
- ustawienie wartości domyślnych dla całej bazy w momencie tworzenia,
- ustawienie limitów bazy (np. maksymalna liczba użytkowników bazy),
- wyspecyfikowanie plików bazy.

2.1.2. Plik kontrolny

Plik kontrolny zawiera informacje o fizycznej strukturze bazy danych oraz pewne informacje kontrolne potrzebne przy odtwarzaniu. W związku z tym, że dane zawarte w pliku kontrolnym są niezbędne do otwarcia i pracy bazy danych oraz do odtwarzania, bardzo często powiela się plik kontrolny na innym dysku.

2.1.3. Pliki dziennika powtórzeń

Zawierają informacje o wszystkich zmianach w bazie danych (co zostało zmienione w wierszu, co zostało zapisane w segmencie wycofania, czy transakcja została potwierdzona, czy był realizowany punkt kontrolny, czy jest transakcja rozproszona, itd.) i są wykorzystywane tylko podczas odtwarzania.

2.2. Struktura logiczna bazy danych

Na strukturę logiczną bazy danych składają się następujące elementy (rys. 3):

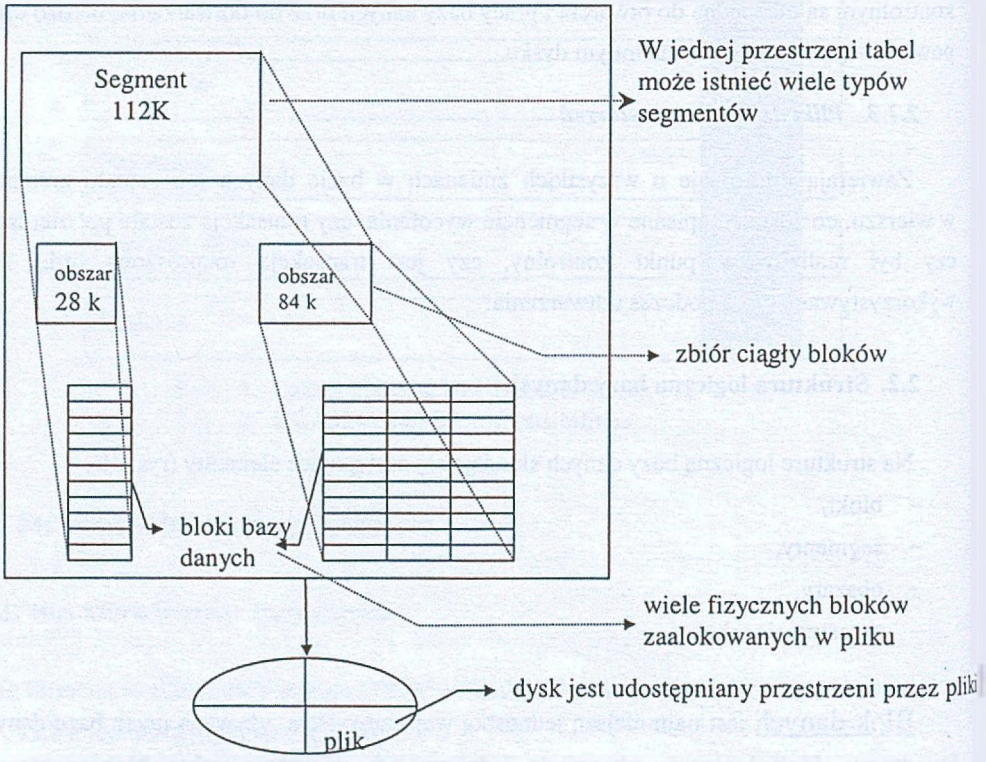
- bloki,
- segmenty,
- obszary,
- przestrzenie tabel.

Blok danych jest najmniejszą jednostką wejścia/wyjścia, używaną przez bazę danych. Pojedynczy blok logiczny odpowiada jednemu lub większej liczbie bloków systemu operacyjnego (zwykle, rozmiar bloku Oracle'a jest zależny od rozmiaru bloku systemu operacyjnego).

Obszar (ang. extent) jest zbiorem kolejnych bloków zaalokowanych przez segment. Pierwszy obszar zwany jest obszarem inicjalnym (ang. initial), natomiast następne – przyrostowymi (ang. incremental). Oracle uważa bloki z następującymi po sobie numerami

identyfikacyjnymi jako ciągłe, chociaż nie oznacza to, że są one ciągłe na dysku. Obiekt alokuje nowy obszar tylko wtedy, gdy aktualnie zaalokowane są użyte. Częsta dealokacja obszarów może doprowadzić do fragmentacji wolnej przestrzeni w plikach danych.

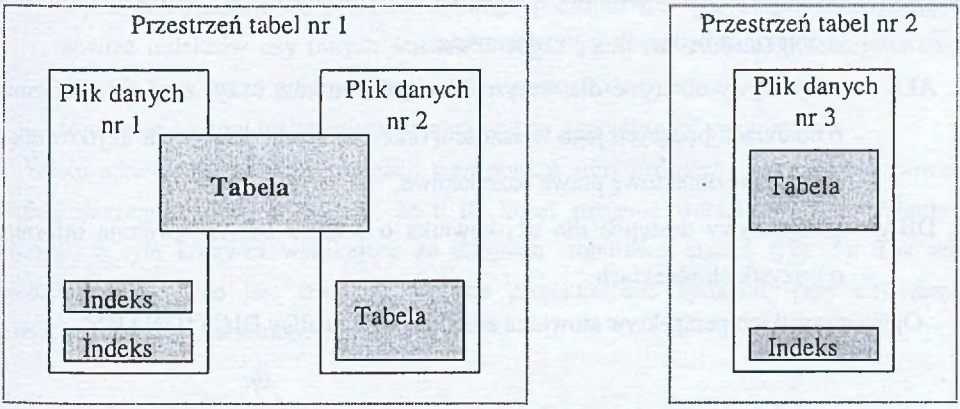
Segment jest zbiorem jednego lub więcej obszarów, w których zawarte są dane określonego typu logicznej struktury pamięci w przestrzeni tabel. Można wyróżnić następujące typy segmentów: danych, indeksu, tymczasowy, wycofania[1], otwarcia. Każdy segment w bazie jest tworzony z co najmniej jednym obszarem dla danych, jednakże segment wycofania zawsze ma co najmniej dwa obszary. Segmenty mogą powodować fragmentację wolnej przestrzeni (segment słownika danych nie powoduje fragmentacji, segmenty danych powodują niewielką fragmentację, segmenty wycofania powodują średnią fragmentację, segmenty tymczasowe powodują wysoką fragmentację).



Rys. 3. Powiązanie segmentów i obszarów
Fig. 3. Dependences between segments and extents

[1] Jest to zbiór obszarów, w których znajdują się dane modyfikowane. Służy do zapewnienia możliwości wycofania transakcji (każda transakcja jest przypisana do jednego segmentu wycofania), spójności czytania i do odtwarzania.

Przestrzeń tabel jest logicznym obszarem, w którym baza przechowuje zapisane w niej dane. Każda przestrzeń składa się z jednego lub więcej plików systemu operacyjnego, które dla serwera Oracle'a tworzą integralną całość. Użytkownik bazy nie ma wpływu na rozmieszczenie danych w plikach przestrzeni tablic, natomiast może zadecydować w poleceniu tworzenia obiektu (*CREATE*), w której przestrzeni obiekt ma powstać (rys.4). Pojedynczy obiekt nie może zostać umiejscowiony w kilku przestrzeniach tabel. Obiekty umiejscowione w danej przestrzeni nigdy nie mogą zaalokować obszarów poza nią.



Rys. 4. Przykładowe rozmieszczenie obiektów w przestrzeniach tabel
Fig. 4. Objects' locating example in tablespaces

W momencie tworzenia bazy danych automatycznie tworzona jest przestrzeń SYSTEM, zawierająca m.in.:

- słownik danych,
- definicje wyzwalaczy bazy,
- definicje procedur wbudowanych,
- segment wycofania SYSTEM,

ale zalecane jest (w zastosowaniach produkcyjnych) podzielenie bazy Oracle'a na kilka przestrzeni.

Nasuwa się pytanie: *Po co używać różnych przestrzeni tabel?*

- żeby oddzielić dane użytkowników od danych słownika,
- żeby oddzielić dane jednej aplikacji od drugiej,
- żeby zachować różne zbiory danych przestrzeni tabel na różnych dyskach,
- żeby oddzielić dane użytkowników od danych segmentu wycofania (zabezpieczenie od całkowitej utraty danych w przypadku zepsucia się pojedynczego dysku).

2.3. Słownik bazy danych

Słownik jest to szereg tablic dostępnych dla użytkowników tylko do czytania, zawierających opis stanu bazy danych. Tablice słownika powstają w czasie tworzenia bazy danych (do ich zakładania służy skrypt SQL.BSD) i są niemodyfikowalne przez cały czas życia bazy.

Istnieją 3 grupy perspektyw, ułatwiających korzystanie ze słownika:

1. USER – perspektywy dostępne dla wszystkich użytkowników bazy; zawierają informacje o obiektach będących jego własnością,
2. ALL – perspektywy dostępne dla wszystkich użytkowników bazy; zawierają informacje o obiektach będących jego własnością oraz obiektach, do których użytkownik ten ma nadane obiektowe prawa bazodanowe,
3. DBA – perspektywy dostępne dla użytkownika o statusie DBA; zawierają informacje o wszystkich obiektach.

Opisy wszystkich perspektyw słownika znajdują się w tablicy **DICTIONARY**.

3. Podstawy strojenia bazy danych

Wydajność systemu można znacząco poprawić poprzez strojenie aplikacji, systemu operacyjnego i samej bazy. W literaturze przedmiotu wyróżnia się dwa główne poziomy procesu strojenia bazy danych Oracle [11]:

1. strojenie systemu operacyjnego, które ogranicza się do odpowiedniej konfiguracji fizycznych zasobów systemu: pamięci operacyjnej, wolumenów dyskowych oraz odpowiedniego ustawienia semaforów systemu operacyjnego (dla systemu UNIX),
2. strojenie instancji bazy danych, polegające na zapewnieniu jej odpowiedniej ilości zasobów systemu operacyjnego.

3.1. Strojenie instancji bazy danych

Twórcy systemu Oracle zalecają następująca metodologię strojenia bazy danych:

1. Strojenie poleceń SQL i aplikacji

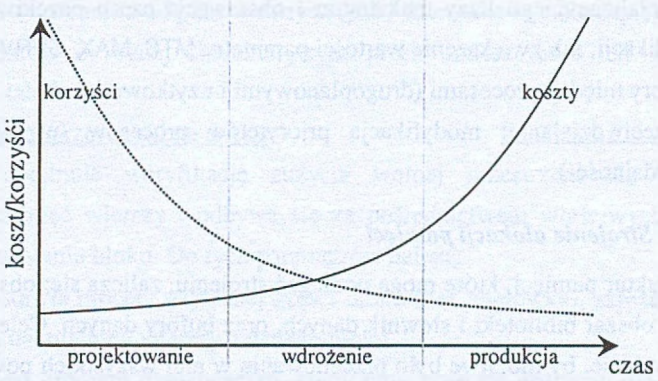
Działanie administratora: uzyskiwanie statystyk dla wbudowanego w system Oracle8 optymalizatora kosztu (wykonywane zwykle przez twórców aplikacji zamiast DBA), którego ogólnym zadaniem jest wybór wydajnej strategii obliczania danego wyrażenia relacyjnego. Zbieranie statystyk, dotyczących obiektów tworzonych przez użytkowników dla

optymalizatora, powinno się odbywać z częstością zależną od częstości modyfikacji tych obiektów.

Wyróżnia się cztery zasadnicze etapy procesu optymalizacji:

- sformułowanie zapytania w jakiejś wewnętrznej postaci (jako wewnętrzną postać zapytania wybiera się zwykle pewien rodzaj tzw. abstrakcyjnego drzewa składni lub drzewa zapytania),
- przekształcenie do postaci kanonicznej,
- wybór kandydatów do procedur niskiego poziomu (na tym etapie rozpoczyna się analiza indeksów czy innych ścieżek dostępu, rozkładu wartości przechowywanych danych, fizycznego grupowania przechowywanych danych, itp.),
- tworzenie planów realizacji zapytania i wybór „najtańszego" rozwiązania.

Warto odnotować, że przy realizacji tego punktu administratorzy bazy danych powinni zwrócić szczególną uwagę na fakt, że o ile koszt strojenia wzrasta wraz z rozbudową aplikacji, o tyle korzyści wynikające ze strojenia stopniowo maleją (rys. 5). Tak więc najbardziej efektywne jest strojenie w fazie projektowania aplikacji: przy najniższych kosztach osiąga się maksimum korzyści.



Rys. 5. Koszty i korzyści wynikające ze strojenia bazy danych w poszczególnych fazach życia aplikacji
Fig. 5. Cost and benefits of tuning during the life of an application

2. *Strojenie alokacji pamięci*

Działanie administratora: zmiana obszaru wspólnego, na który składa się obszar biblioteki i słownik danych, oraz buforów danych.

3. *Strojenie wejścia/wyjścia*

Działanie administratora: rozdział plików na różne dyski w celu zmniejszenia sporów o dysk, właściwe zaalokowanie przestrzeni dla tabel, co wpływa na zmniejszenie liczby operacji dyskowych oraz unikanie dynamicznego zarządzania przestrzenią przez alokację wystarczająco dużych obszarów (ang. extents).

4. *Strojenie sporów*

Spory o dostęp do dysku występują wtedy, gdy wiele procesów jednocześnie sięga do tego samego zasobu:

- do segmentu wycofania
  przeciwdziałanie: utworzenie większej liczby segmentów i przydzielenie segmentów wycofania poszczególnym transakcjom poleceniem SET TRANSACTION; szacunkowo przyjmuje się, że jeden segment wycofania może obsłużyć do czterech konkurencyjnych transakcji,
- do buforów dziennika powtórzeń
  przeciwdziałanie: sprawdzanie zapisanej w tabeli V$WAITSTAT statystyki sporów o bloki, porównanie oczekiwań z całkowitą liczbą żądań danych za określony przedział czasu i – jeżeli liczba oczekiwań jest większa od 1% wszystkich żądań – utworzenie kolejnych segmentów wycofania[1],
- spory o procesy serwera wielokanałowego
  przeciwdziałanie: sprawdzanie w tabeli V$SQUEUE, czy liczba wspólnych procesów serwera osiągnęła określone parametrem MTS_MAX_SERVERS pliku inicjalizacyjnego bazy maksimum i obserwacja czasu oczekiwania w trakcie pracy aplikacji, lub zwiększenie wartości parametru MTS_MAX_SERVERS,
- spory między procesami (drugoplanowymi i użytkowników)
  przeciwdziałanie: modyfikacja priorytetów procesów (uwaga: może zmniejszyć wydajność!).

3.1.1. Strojenie alokacji pamięci

Do struktur pamięci, które mogą podlegać strojeniu, zalicza się: obszar wspólny, na który składa się obszar biblioteki i słownik danych, oraz bufory danych. Celem strojenia biblioteki jest zapewnienie, by możliwe było przechowanie w niej wszystkich powszechnie używanych poleceń SQL, co spowoduje w miarę możliwości użycie takich poleceń SQL, na których dokonano już rozbioru (ang. parse). Podobnie, obszar słownika danych musi być wystarczająco duży, by wszystkie często używane dane o bazie, jej strukturach i użytkownikach mogły się tam zmieścić.

Sprawdzanie wydajności obszaru biblioteki i obszaru słownika odbywa się poprzez określenie stosunku liczby chybień do trafień odczytanych ze stosownych perspektyw słownika bazy danych:

```
select sum(pins) „wykonania", sum(reloads) „chybienia poczas wykonania"
from v$librarycache;
```

[1] Bufory dziennika powtórzeń mogą zawierać nagłówki bloków segmentów wycofania.

Jeżeli wynik porównania jest większy niż 1%, administrator powinien zwiększyć parametr pliku inicjalizacyjnego o nazwie: SHARED_POOL_SIZE.

```
select sum(gets) „pobrania ze slownika", sum(getmisses) „chybienia"
from v$rowcache;
```

Jeżeli wynik porównania jest większy niż 10%-15%, jest to wskazówka dla administratora, że powinien zwiększyć parametr SHARED_POOL_SIZE pliku inicjalizacyjnego.

3.1.2. Strojenie wejścia/wyjścia

1. Minimalizacja obciążenia dysku

Aby maksymalnie zminimalizować obciążenie dysku, konieczna jest:

- separacja segmentów słownika od innych segmentów (korzystanie z danych powoduje szereg odwołań do słownika),
- separacja segmentów wycofania od innych segmentów (kiedy w segmentach dokonuje się wiele zmian, obserwuje się wzmożoną aktywność segmentów wycofania),
- separacja segmentów danych od stowarzyszonych indeksów,
- separacja plików dziennika od plików danych (pliki dziennika są stale zapisywane),
- separacja obiektów o różnej charakterystyce przez umieszczenie ich w różnych przestrzeniach tabel.

2. Kontrola zajętości przestrzeni na dysku

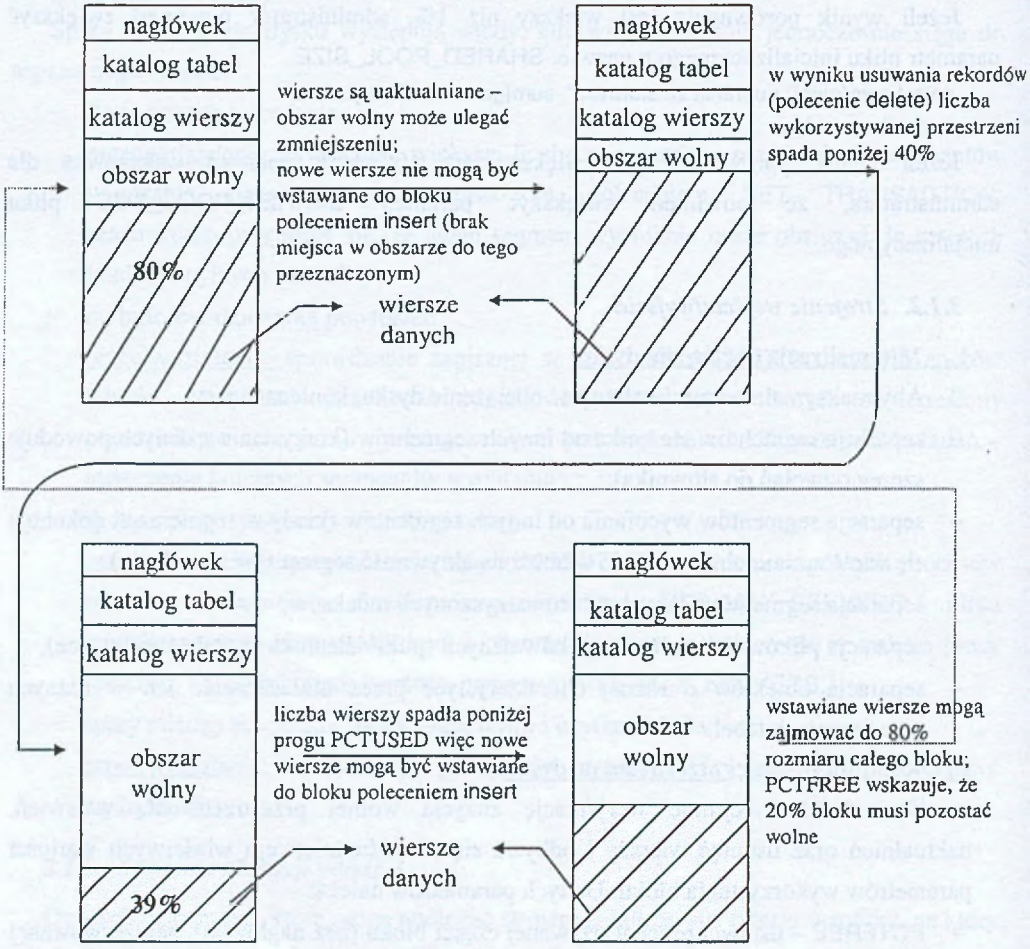
Kontrola ta obejmuje weryfikację zużycia wolnej przestrzeni dla wstawień, uaktualnień oraz usunięć wierszy i odbywa się za pośrednictwem właściwych wartości parametrów wykorzystania bloku. Do tych parametrów należą:

- PCTFREE – ustawia procent używanej części bloku (bez nagłówka), zarezerwowanej dla możliwych uaktualnień wierszy z danego bloku,
- PCTUSED – jest progiem określającym, kiedy blok staje się dostępny dla wstawień nowych wierszy,
- INITRANS – określa liczbę transakcji, dla których istnieje zarezerwowane miejsce w nagłówku bloku,
- MAXTRANS – określa maksymalną liczbę konkurencyjnych transakcji, które mogą być obsłużone przez blok.

Przykładową zajętość przestrzeni na dysku przedstawia rysunek 6.

Zmniejszenie wartości parametru PCTUSED wpływa na:

- zmniejszenie kosztów przetwarzania, ponieważ bloki rzadko są wolne,
- zwiększenie ilości nieużywanej przestrzeni.



Rys. 6. Zajętość przestrzeni na dysku dla przykładowych wartości PCTUSED = 40 oraz PCTFREE = 20

Fig. 6. Data blocks space usage for parameters: PCTUSED =40 and PCTFREE = 20

Niższe PCTUSED decyduje o:

- trzymaniu bloku mniej pełnego niż wyższy PCTUSED,
- redukcji kosztów przetwarzania,
- wzroście ilości nieużywanej przestrzeni w bazie danych.

Wyższe PCTUSED oddziałuje na:

- trzymanie bloków pełniejszych niż niższy PCTUSED,
- poprawę wydajności przestrzeni,
- wzrost kosztów przetwarzania podczas wykonywania poleceń UPDATE i INSERT.

Zmniejszenie wartości parametru PCTFREE może rzutować na:

- wzrost częstości migracji[1] wierszy,
- zwiększenie kosztów przetwarzania z powodu częstej reorganizacji bloków przez Oracle,
- zmniejszenie liczby bloków potrzebnych do przechowywania danych.

Zwiększenie wartości parametru PCTFREE powoduje:

- zwiększenie kosztów przetwarzania, ponieważ bloki mogą często stawać się wolne,
- mniejsze prawdopodobieństwo łączenia[2] wierszy (jest więcej miejsca na ewentualne powiększenie wiersza),
- gorsze wykorzystanie przestrzeni, gdyż w każdym bloku zostaje więcej wolnego miejsca, być może nigdy nie wykorzystanego.

W celu sprawdzenia, czy zmiany w tablicy użytkownika spowodowały migrację wierszy w tej tabeli, administrator powinien w pierwszej kolejności uzyskać statystyki dla optymalizatora kosztu, za pomocą polecenia analyze, a następnie zbadać powstałą w poprzednim kroku tabelę wiersze_łączone:

```
analyze table nazwa_analizowanej_tabeli
list chained rows
into wiersze_łączone;
select head_rowid
from wiersze_łączone
where table_name =' nazwa_analizowanej_tabeli;
```

3. Kontrola alokacji obszarów

Alokacja obszarów jest kontrolowana przez odpowiednie ustawienie następujących parametrów:

- INITIAL – rozmiar w [B] pierwszego zaalokowanego obszaru (domyślnie rozmiar 5 bloków logicznych Oracle'a),
- NEXT – rozmiar w [B] następnego zaalokowanego obszaru (domyślnie rozmiar 5 bloków logicznych Oracle'a),
- MAXEXTENTS – maksymalna ilość obszarów, które mogą być zaalokowane przez segment; wartość zależy od wielkości bloku (domyślnie 121 dla bloku 2[k]),
- MINEXTENTS – minimalna ilość obszarów, które są zaalokowane przez segment wycofania w momencie tworzenia (domyślnie 1),

[1] Migracja występuje w przypadku takiej aktualizacji wiersza, że jego długość zwiększa się powyżej ilości wolnej przestrzeni w bloku.

[2] Występuje w przypadku wierszy zawierających długie kolumny, gdy wszystkie dane wiersza nie mieszczą się w tym samym bloku.

- PCTINCREASE – procent, o jaki każdy następny alokowany obszar jest większy od poprzedniego (domyślnie 50 dla segmentu danych i indeksu, a 0 dla segmentu wycofania),
- OPTIMAL — określa optymalny rozmiar w [B] segmentu wycofania (domyślnie *null*),
- FREELIST – liczba list wolnych bloków dla wstawień.

Przeglądnięcie perspektyw USER_TABLES i DBA_FREE_SPACE słownika bazy przedstawi administratorowi te tabele, które nie mogą zaalokować następnego obszaru:

```
select tablespace_name, next_extent
from user_tables t1
where not exists (
        select *
        from dba_free_space t2
        where t1.tablespace_name = t2.tablespace_name and bytes >= next_extent);
```

Z kolei odczytanie ze słownika obszarów wolnej przestrzeni w każdej przestrzeni tabel pozwoli na podjęcie decyzji o ewentualnym dodaniu do niej pliku w celu zachowania możliwości dynamicznego alokowania obszarów przez obiekty bazy danych:

```
select dfs.tablespace_name, block_id, dfs.bytes, dfs.blocks, df.file_name
from dba_free_space dfs, dba_data_files df
where dfs.file_id = df.file_id
order by 1, 2;
```

4. Kontrola fragmentacji

Tabela 2

Własności obiektów przestrzeni tabel

Obiekty	Własności
Segmenty słownika danych	Nie powodują fragmentacji wolnej przestrzeni
Segmenty danych	Powodują niewielką fragmentację
Segmenty wycofania	Powodują średnią fragmentację
Segmenty tymczasowe	Powodują wysoką fragmentację

W związku z tym, że fragmentacja, będąca zgrupowaniem nieciągłych obiektów bazy danych, pociąga za sobą niepotrzebne zużycie zasobów (np. nadmierna liczba operacji wejścia/wyjścia, wydłużony czas odczytu danych z dysku), konieczne jest przeciwdziałanie powstawaniu niewykorzystanych obszarów pomiędzy blokami danych przez zwiększenie wartości parametru PCTUSED, oraz separację grup obiektów o różnej charakterystyce w różnych przestrzeniach tabel (tabela 2).

W przypadku zaobserwowania (w perspektywie DBA_FREE_SPACE) wystąpienia fragmentacji problem może zostać rozwiązany przez eksport, usunięcie i import obiektu.

3.1.3. Strojenie sporów

1. Spory o dostęp do buforów dziennika powtórzeń

Odpowiedź na pytanie, czy rozmiar buforów dziennika powtórzeń jest odpowiedni, zawarta jest w dynamicznej perspektywie słownika V$SYSSTAT:

```
select name, value
from v$sysstat
where name ='redo log space requests';
```

Odczytana przez administratora wartość powinna być bliska 0. W przeciwnym przypadku należy zwiększyć parametr LOG_BUFFERS pliku inicjalizacyjnego bazy.

2. Spory o segment wycofania

Perspektywa V$WAITSTAT zawiera przydatne dla administratora informacje o ilości oczekiwań na bufory segmentów wycofania:

```
select class, count
from v$waitstat
where class in ('system undo headers',
                'system undo block',
                'undo header',
                'undo block');
```

- 'system undo headers' — liczba oczekiwań na bufory zawierające nagłówki bloków segmentu wycofania
- 'system undo block' — liczba oczekiwań na bufory zawierające nienagłówkowe bloki segmentu wycofania
- 'undo header' — liczba oczekiwań na bufory zawierające nagłówki bloków segmentu wycofania innych niż *SYSTEM*

3. Spory o dostęp do list wolnych bloków

Spory występują, gdy wiele procesów Oracle wstawia wiersze do tej samej tabeli w tym samym czasie. Statystyki dotyczące sporów sprawdza się w tabeli V$WAITSTAT:

```
select class, count
from v$waitstat
where class = 'free list';
```

Liczba oczekiwań na wolne bloki, porównuje się następnie z całkowitą ilością żądań za ten sam okres czasu, znajdowaną w tablicy V$SYSSTAT:

```
select sum(value)
from v$sysstat
where name in ('db block gets', 'consistent gets', 'physical reads');
```

Stosunek oczekiwań na wolne bloki do całkowitej liczby pobrań powinien być mniejszy od 1%. Jeżeli jest on większy, należy rozważyć zwiększenie liczby list wolnych bloków parametrem FREELISTS opcji STORAGE tabeli, do wartości równej liczby użytkowników, którzy jednocześnie wstawiają dane do tabeli. Z tabeli X$KCBRBH można uzyskać dane o zysku wydajności z dodania buforów.

Przykładowo, następujące polecenie SQL:

```
select sum(count) 'dodatkowe trafienia'
from x$kcbrbh
where indx < 10;
```

pokaże, ile więcej trafień wystąpi po dodaniu 10 buforów.

Na podstawie uzyskanych z tabeli V$SYSSTAT informacji może być również obliczony współczynnik trafień *hit ratio* wg wzoru (1), którego mniejsza od 60% wartość informuje administratora o konieczności zwiększenia parametru DB_BLOCK_BUFFERS pliku inicjalizacyjnego.

$$hit\ ratio = 1 - \frac{physical\ reads}{(db\ block\ gets + consistent\ gets)} \quad (1)$$

4. Strojenie na poziomie projektu bazy danych

Podstawowym czynnikiem, pozwalającym administratorowi prawidłowo zaplanować bazę danych, jest znajomość danych, które będą ją wypełniać. Dokonany logiczny podział danych ma następnie wpływ na tworzenie przestrzeni tabel.

Na tym poziomie strojenia ważną zasadą jest rozmieszczanie obiektów konkurujących o dostęp do dysku, na różnych dyskach. Również wskazane jest, aby oddzielać dane testowe od rzeczywistych, umieszczając je w różnych bazach danych.

W systemie Oracle8 możliwy jest podział na partycje takich obiektów, jak tabele i indeksy, co pociąga za sobą możliwość dystrybucji tych obiektów pomiędzy wiele przestrzeni tabel. Rozdział danych jest szczególnie przydatny przy optymalizacji przypadkowych dostępów do tabel z wieloma wierszami.

Tabele i indeksy o znacznym rozmiarze mogą być podzielone na mniejsze segmenty zwane partycjami. Utworzone w momencie tworzenia tabeli lub indeksu partycje są całkowicie niewidoczne dla aplikacji i użytkowników, jednak użytkownicy w klauzuli FROM formułowanego zapytania SQL mogą nie brać pod uwagę pojedynczej, dużej tabeli, lecz jej mniejsze segmenty. W związku z tym, że czas odpowiedzi zależy od ilości zgromadzonych w tabeli danych, takie podejście zwiększa wydajność przetwarzania. Rozdzielone tabele i indeksy mogą być także równolegle dostępne, co jeszcze bardziej skraca czas przetwarzania. Podział tabeli na partycje niesie za sobą jeszcze inne korzyści [2, 11]:

- zmniejszenie czasu przestoju (reperacja dotyczy tylko wadliwej części tabeli),
- zwiększenie niezawodności bazy danych (wyłączenie partycji nie uniemożliwia pracy aplikacji, która korzysta z innych partycji),

- ułatwienie konserwacji (łatwiej zarządzać potrzebnymi operacjami eksportu, importu danych, odbudowy indeksu wykonywanymi niezależnie na każdej partycji zamiast na jednej dużej tabeli),
- zrównoważenie operacji wejścia/wyjścia (różne partycje można mapować jako oddzielne dyski, aby wyrównać obciążenie wejścia/wyjścia),
- skrócenie czasu odzyskiwania danych,
- zwiększenie dostępności danych.

Przykład: Tworzenie tabeli warehouse podzielonej zakresowo, w oparciu o wartości danych w kolumnie w_id, na trzy partycje, z których każda zaalokowana będzie w innej przestrzeni tabel:

```
CREATE TABLE warehouse (w_id number, w_name varchar2(10), w_address varchar2(40))
PARTITION BY RANGE (w_id)
(PARTITION ware_P1 VALUES LESS THAN (50) TABLESPACE TS1,
 PARTITION ware_P2 VALUES LESS THAN (100) TABLESPACE TS2,
 PARTITION ware_P1 VALUES LESS THAN (150) TABLESPACE TS3);
```

Korzystanie z pojedynczej partycji ogranicza się w późniejszej pracy do uzupełnienia zdania SQL identyfikatorem partycji, np.:

```
select w_name, w_address
from warehouse partition ware_P2;
```

4.1. Wyszukiwanie i usuwanie awarii

DBA poprzez proaktywne monitorowanie systemu może zapobiec powstawaniu wielu problemów i potencjalnych awarii baz danych, które można podzielić na następujące kategorie 10:

1. Błąd zdania SQL
- przyczyny: błędy logiczne zawarte w procedurach aplikacji, próby wstawiania błędnych danych do tabel, brak (fizycznego) miejsca w bazie do zapisania nowych danych, próba wykonania operacji przekraczającej uprawnienia użytkownika,
- objawy: efekty wykonania błędnego zdania są automatycznie wycofywane, a sterowanie jest przekazywane do programu użytkownika,
- przeciwdziałanie: eliminacja błędów zawartych w aplikacjach pracujących z bazą danych, modyfikacja zdania SQL, nadanie użytkownikom odpowiednich uprawnień, kontrola ilości wolnej, dostępnej do zapisu danych, przestrzeni tabel w bazie.

2. Awaria procesu użytkownika
- przyczyny: niestandardowe odłączenie się użytkownika od bazy, przerwanie sesji bez wylogowania się z bazy, program użytkownika spowodował wyjątek, kończący sesję,
- objawy: przejściowy brak łączności z bazą,

- przeciwdziałanie: brak; awarie tego typu obsługiwane są przez proces drugoplanowy PMON, który wykrywa nienormalnie zakończone procesy użytkownika, wycofuje transakcje przerwanych procesów użytkownika, zwalnia zasoby i blokady danego procesu.

3. Awaria instancji
- przyczyny: zanik zasilania serwera bazy danych, awaria systemu operacyjnego, uszkodzenie pamięci, błędne zadziałanie jednego z procesów drugoplanowych wchodzących w skład instancji (SMON, PMON, DBWR, LGWR),
- objawy: przerwanie pracy bazy danych,
- przeciwdziałanie: brak; w momencie startowania instancji (po zamontowaniu bazy) jest wykonywane jej automatyczne odtwarzanie.

4. Błąd użytkownika
- przyczyny: usunięcie wszystkich danych zapisanych w tabeli, zatwierdzenie wprowadzenia niepoprawnych danych, usunięcie z bazy całej tabeli,
- objawy: błędne wyniki lub nieprawidłowe działanie aplikacji,
- przeciwdziałanie: tworzenie kopii zapasowych lub plików eksportu oraz tworzenie szczelnego systemu praw dostępu do obiektów bazy (odtworzenie utraconych zasobów może wymagać dwukrotnego przebiegu: wgrania kopii bazy, odtworzenia stanu bazy do stanu sprzed awarii, wyeksportowania danych, które zostały zniszczone, ponownego wgrania kopii bazy, odtworzenia jej do stanu bieżącego oraz zaimportowania odzyskanych danych).

5. Awaria nośnika
- przyczyna: uszkodzenie napędu dysku, niepoprawne bloki na dysku, skasowanie plików danych lub uszkodzenie systemu plików,
- objawy: przerwanie pracy bazy danych,
- przeciwdziałanie: tworzenie kopii zapasowych.

5. Podsumowanie

Pomimo że serwer Oracle8 ma ulepszone procedury optymalizacji, wciąż jeszcze najlepsze działanie systemu w dużym stopniu zależy od jego administratora. Problem strojenia jest w praktyce procesem eksperymentowania, dlatego w artykule zawarto jedynie wskazówki, pozwalające na wybór najlepszej opcji dostosowania środowiska spośród wielu dostępnych. Rozwiązanie problemów związanych ze strojeniem, a co za tym idzie – również

wydajnością bazy danych, wymaga od administratora całościowego spojrzenia na system dla zidentyfikowania obszaru odpowiedzialnego za problemy.

Często źródłem kłopotów może być jednocześnie kilka obszarów:

- konfiguracja sprzętu,
- konfiguracja oprogramowania,
- architektura aplikacji,

przy czym podejmowane przez administratora działania powinny obejmować następujące etapy:

- ogólną analizę funkcji systemu,
- analizę serwera – zebranie statystyk dla systemu operacyjnego i serwera bazy danych,
- analizę zapytań – plany wykonania zapytań.

LITERATURA

1. Ullman J. D.: File and knowledge–base systems, Computer Science Press, 1988.
2. Austin D.: Poznaj Oracle 8, MIKOM 1999.
3. Date C.J.: Wprowadzenie do systemów baz danych, WNT, Warszawa 2000.
4. Rodgers U.: ORACLE przewodnik projektanta baz danych, WNT, Warszawa 1995.
5. Beynon-Davies P.: Systemy baz danych, WNT, Warszawa 1998.
6. ORACLE7 Server Concepts Manual. Oracle Corporation.
7. ORACLE7 Server Administrator's Guide. Oracle Corporation.
8. Database security, adres strony internetowej: http://www.cs.kau.se/~hansh/davc17/forelas/f8-s4.pdf.
9. Ochrona systemów teleinformatycznych w przedsiębiorstwach z wykorzystaniem backup'u, adres strony internetowej: http://www.software.com.pl/konferencje/sdp2000/Wyklady/CCS.asp
10. Gnybek J.: Oracle łatwiejszy niż przypuszczasz, Helion, 1996.
11. Greene J. „Oracle8 Server", Helion, 2000.

Recenzent: Dr inż. Maciej Bargielski

Wpłynęło do Redakcji 3 grudnia 2002 r.

Abstract

Database performance can be defined as the optimization of resource usage, to increase throughput and minimize contention (chapter 4.1). Many performance management tasks must be shared between the DBA and other technicians (Table 1, Fig. 2). To ensure efficient access to databases, DBA ought to do a lot of tasks and solve many problems, connected with it. For example, in order to guarantee optimal database design, DBA should permanently monitor system performance (chapter 3.1.1), tune Input/Output (chapter 3.1.2), SQL (chapter) and applications – ie DBA should prepare tuning strategy for the database. Some of the DBA's reaction abilities include specifying large enough buffers and caches and setting system parameters.

Adresy

Aleksandra WERNER: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-101 Gliwice, Polska, ola@ares.iinf.polsl.gliwice.pl .