SKRYPTY DLA SZKÓŁ WYŻSZYCH

POLITECHNIKA ŁÓDZKA

Wanda Gryglewicz-Kacerka

Bohdan Szymczak

ADMINISTRACJA BAZĄ DANYCH

Część I - Administracja bazą danych ORACLE 8i
Część II - Administracja bazą danych MAGIC 8

ŁÓDŹ 2002

NAKŁADEM POLITECHNIKI ŁÓDZKIEJ

Skrypt jest materiałem pomocniczym do wykładów i ćwiczeń z Wybranych zagadnień z baz danych dla studentów Wydziału Fizyki Technicznej, Informatyki i Matematyki Stosowanej Politechniki Łódzkiej. Mogą ze skryptu korzystać także studenci innych wydziałów, którzy mają zajęcia z systemów informatycznych.

Opracowanie poszczególnych rozdziałów:

Część I

Administracja bazą danych ORACLE 8i – Bohdan Szymczak

Część II

Administracja bazą danych MAGIC 8 – Wanda Gryglewicz-Kacerka

Autorzy pragną złożyć szczególne podziękowania za udostępnienie wszelkich materiałów i fachową pomoc następującym firmom:

- ORACLE POLSKA
- WONLOK S.A.
- MSE
- KOMTECH

Spis treści:

Część I

Oracle 8i

Administracja bazą danych

1. Wstęp

Niniejsze opracowanie omawia zagadnienia związane z administracją bazy danych Oracle8i, a w szczególności:

- planowanie i tworzenie bazy danych
- czynności wykonywane przez administratora w ramach utrzymania bazy
- bezpieczeństwo bazy
- auditing i śledzenie zmian w bazie
- elementy strojenia

Przeznaczony jest dla osób posiadających wiedzę teoretyczną na temat funkcjonowania bazy Oracle w zakresie omówionym w części „Oracle8" skryptu „Wybrane zagadnienia z baz danych" tegoż autora. Opracowanie koncentruje się na problemach, z którymi najczęściej spotyka się administrator bazy Oracle wykorzystywanej jako platforma bazodanowa dla typowych aplikacji produkcyjnych. Dlatego duży nacisk położono na problemy bezpieczeństwa systemu i odzyskiwania danych po awarii, zaś całkowicie pominięto zarządzanie obiektami schematów, które należy do zagadnień rozwiązywanych zwykle przez twórców aplikacji.

Przykłady mające odniesienie do określonej platformy systemu operacyjnego zostały przedstawione dla systemów: Windows 2000 i Solaris 8. Diagramy składniowe komend SQL zostały zaczerpnięte z oryginalnej dokumentacji Oracle.

2. Tworzenie bazy danych

2.1 Planowanie bazy

Baza danych Oracle przechowuje informacje specyficzne dla używanych aplikacji. Projektując bazę należy uwzględnić, jakiego rodzaju aplikacja będzie z niej korzystała. Należy wziąć pod uwagę następujące cechy:

- Wielkości wierszy używanych tabel
- Język składowanych danych (strona kodowa)
- Charakter obciążenia bazy (liczba jednoczesnych połączeń, długość transakcji)
- Przewidywany rozmiar bazy, roczny przyrost danych
- Kryteria bezpieczeństwa bazy (waga informacji składowanych w bazie, akceptowalny czas odtworzenia po awarii, dopuszczalność utraty części ostatnio wprowadzonych danych)
- Możliwości sprzętowe serwera bazy danych

Powyższe cechy aplikacji wpływają na określenie następujących parametrów bazy:

- Wielkość bloku bazy
- Strona kodowa bazy
- Liczba i rozmiar plików dzienników powtórzeń
- Liczba plików dzienników powtórzeń w grupie
- Wielkość i rozkład plików bazy danych
- Umieszczenie przestrzeni tabel w systemie plików lub na surowych partycjach (raw device)

Po zainstalowaniu bazy danych możliwa jest późniejsza zmiana niektórych parametrów, np. w zakresie rozkładu plików danych. Często dopiero po pewnym czasie pracy bazy możliwe staje się prawidłowe określenie właściwych parametrów wynikających ze szczególnego użycia danej aplikacji w określonej instalacji produkcyjnej. Ta sama aplikacja np. system bankowy może bowiem pracować w odmienny sposób w różnych instalacjach. Przykładowo inaczej obciążona jest baza danych w banku prowadzącym dużą liczbę rachunków bieżących, charakteryzujących się licznymi operacjami na rachunkach, a inaczej w banku specjalizującym się w prowadzeniu lokat. Dlatego też normalną procedurą jest przyjęcie podstawowych założeń dla pracy bazy na etapie planowania, zaś w pierwszym etapie eksploatacji określenie optymalnych parametrów w procesie strojenia bazy (tuning).

Należy jednak zwrócić uwagę na fakt, iż zmiana takich parametrów jak rozmiar bloku, lub strona kodowa bazy wymaga wykonania procedury reinstalacji bazy połączonej z wykonaniem eksportu/importu danych i wiąże się zawsze z koniecznością zatrzymania pracy instalacji produkcyjnej. W przypadku niektórych systemów pracujących w trybie 24x7 taka operacja jest niemożliwa.

Jeżeli baza danych jest używana przez kilka różnych aplikacji, szczególnie różniących się istotnie charakterem np. system DSS (Decision Support Systems) i system OLTP (Online Transaction Processing) dobranie parametrów optymalnych dla każdej z nich jest na ogół niemożliwe. W takiej sytuacji należy rozważyć możliwość postawienia na jednej maszynie więcej niż jednej bazy danych lub przyjęcie pewnego kompromisu, zwykle polegającego na uznaniu priorytetu jednej z aplikacji. Dla podanego przykładu najczęściej przyjmuje się priorytet dla aplikacji OLTP, gdyż są one bardziej krytyczne czasowo.

Określenie wielkości bloku bazy

Domyślnie wielkość bloku bazy danych jest zależna od systemu operacyjnego maszyny będącej serwerem bazy i wynosi zwykle 2K lub 4K. Jeżeli wielkość bloku bazy nie jest równa wielkości bloku systemu operacyjnego, to powinna być jego wielokrotnością.

Z punktu widzenia efektywności pracy określonych aplikacji przyjmuje się, że małe wartości wielkości bloku bazy (2K, 4K) są odpowiednie dla systemów OLTP, zaś wartości duże (32K, 64K) typowe dla systemów DSS. Można przyjąć następującą charakterystykę bloków bazy z punktu widzenia ich wielkości:

- Bloki małe (2KB-4KB) – umożliwiają redukcję sporów o dostęp do bloków, są odpowiednie do składowania krótkich wierszy, efektywnie realizują dużą liczbę losowych dostępów. Jednocześnie mało efektywnie wykorzystywana jest przestrzeń dyskowa ze względu na relatywnie duży rozmiar nagłówka bloku. W bloku można zmieścić tylko niewiele wierszy, zaś wiersze długie muszą być dzielone pomiędzy wiele bloków.

- Bloki średnie (8KB) – umożliwiają wczytanie do buforów danych w SGA wielu wierszy podczas jednej operacji I/O, co jest korzystne podczas operacji czytania metodą table full scan. Jednocześnie dla operacji pobrania pojedynczego wiersza (random accesss) znaczny obszar w buforach danych jest marnowany na trzymanie niepotrzebnych w tym momencie danych.
- Bloki duże (16KB-32KB) – umożliwiają oszczędne gospodarowanie przestrzenią dyskową ze względu na relatywnie mały nagłówek bloku – większość przestrzeni używana jest na składowanie danych. Bardzo efektywnie realizowany jest sekwencyjny dostęp do danych oraz składowanie bardzo długich wierszy. Jednocześnie występują niekorzystne zjawiska związane z obsługą indeksów, które muszą korzystać z bloków o tym samym rozmiarze, przy masowym dostępie przez wielu użytkowników. Taki charakter pracy mają systemy OLTP; w tym przypadku należy liczyć się ze wzrostem sporów o bloki przechowujące liście indeksów.

Strona kodowa bazy

Oracle wykorzystuje wsparcie języków narodowych NLS (National Language Support) w celu składowania, przetwarzania i wyszukiwania danych w językach narodowych. NLS gwarantuje prawidłowe sortowanie, datę, czas, walutę, system liczbowy i kalendarz zgodny z ustawieniami języka narodowego i ustawień lokalizacyjnych. Dotyczy to nie tylko składowania danych w bazie, ale również języka komunikatów oraz innych narzędzi Oracle. Używając architektury klient/serwer możliwe jest niezależne stosowanie odmiennej lokalizacji po obu stronach; w takim przypadku Oracle gwarantuje poprawne przeprowadzenie konwersji znaków należących do odmiennych zbiorów w sposób automatyczny i przeźroczysty poprzez Net8.

Strona kodowa bazy ustalana jest w momencie generacji bazy i nie może być później zmieniona bez ponownej reinstalacji i przeprowadzenia operacji eksport/import. Jedynym wyjątkiem jest sytuacja zmiany strony kodowej na inną, która jest nadzbiorem strony pierwotnej. Strona kodowa nie może należeć do grupy mającej reprezentację znaków typu wielobajtowego (fixed-width multibyte character sets). Jeżeli występuje potrzeba przechowywania takich znaków np. należących do alfabetu japońskiego, należy określić pomocniczą stronę kodową (national character set), w której można składować dane w kolumnach typu: NCHAR, NCLOB i NVARCHAR2.

Dla składowania danych w języku polskim można stosować szereg stron kodowych zawierających polskie znaki np.:

- EE8ISO8859P2 - strona ISO 8859-2 East European
- EE8PC852 - strona IBM-PC Code Page 852 8-bit East European
- EE8MSWIN1250 - strona MS Windows Code Page 1250 8-bit East European

Obie strony używają jednobajtowego kodowania znaków i są nadzbiorem standardu ASCII.

Przestrzenie tabel

Zasadniczo przestrzenie tabel powinny być zaprojektowane przez twórców aplikacji. Do zadań administratora bazy należy w takim przypadku decyzja o ich prawidłowym rozmieszczeniu. Na ogół ze względów wydajnościowych tworzy się następujące przestrzenie tabel:

- Systemowa przestrzeń tabel
- Przestrzeń tabel przeznaczona na dane
- Przestrzeń tabel przeznaczona na indeksy
- Przestrzeń tabel przeznaczona na segmenty wycofania
- Przestrzeń tabel przeznaczona na segmenty tymczasowe

Jeżeli z analizy dostępu do danych wynika, że mogą wystąpić spory o dostęp do zasobów ulokowanych w ramach jednej przestrzeni tabel, zaś konfiguracja maszyny umożliwia większy rozkład, celowe może okazać się np. utworzenie dwóch przestrzeni tabel na dane i rozdzielenie ich pomiędzy różne dyski. Takie rozwiązania stosuje się w sytuacji, gdy składem danych są pojedyncze dyski. Dla rozwiązań macierzowych typu RAID0 lub podobnych rozkład podany powyżej jest na ogół odpowiedni, szczególnie, gdy każdą przestrzeń tabel można alokować na grupie dysków podłączonej do osobnego kanału kontrolera.

Gdy baza danych obsługuje kilka niezależnych aplikacji często stosuje się oddzielne przestrzenie tabel dla każdej z nich. Jednocześnie nie ma przeszkód, aby wszystkie aplikacje używały jednej przestrzeni dla segmentów tymczasowych i jednej dla segmentów wycofania. Obciążenie ich zależy wyłącznie od liczby jednoczesnych sesji użytkowników, nie zaś od liczby aplikacji korzystających z bazy. Rozkład przestrzeni tabel na aplikacje ma tutaj wyłącznie uzasadnienie administratorskie, lub wynika z udogodnień podczas instalacji.

Pliki dzienników powtórzeń

Planując pliki dzienników powtórzeń należy brać pod uwagę dwa kryteria:

- Bezpieczeństwo bazy
- Wydajność bazy

Ponieważ system bezpieczeństwa bazy danych opiera się w zasadzie wyłącznie o pliki dzienników powtórzeń (redo logs) zachowanie ich w sytuacji awarii gwarantuje możliwość odtworzenia bazy do momentu awarii. Jeżeli baza pracuje w trybie archiwizacji logów (archive log mode) to posiadając pełną archiwalną kopię bazy, wszystkie archiwalne pliki dzienników powstałe po backupie bazy oraz bieżące pliki dzienników można zawsze odtworzyć bazę po awarii do momentu ostatniego zapisu w logach, co jest równoznaczne z odtworzeniem ostatniej zatwierdzonej transakcji.

A zatem skoro pliki dzienników powtórzeń decydują o możliwości awaryjnego odtworzenia bazy, kluczowe znaczenie ma takie skonfigurowanie systemu, aby nawet w przypadku uszkodzenia dysków możliwe było ich odzyskanie. Oracle dostarcza mechanizmu obsługi grup plików dzienników powtórzeń składających się z kilku kopii lustrzanych. W tej konfiguracji proces LGWR jednocześnie zapisuje tę samą informację do identycznych plików logów.

Z punktu widzenia bezpieczeństwa bazy należy zatem utworzyć grupy plików składające się z co najmniej dwóch kopii i alokować każdą kopię na oddzielnym dysku. Pliki różnych grup można umieszczać na tym samym dysku.

Z punktu widzenia wydajności bazy istotne jest określenie wielkości, liczby i rozkładu plików dzienników powtórzeń.

Wielkość plików musi być skorelowana z przewidywanym szczytowym obciążeniem bazy, które generuje bardzo dużą liczbę zapisów do logów (redo entries). Mały rozmiar plików logów powoduje wystąpienie częstego ich przełączania i w konsekwencji spadek wydajności. Jeżeli jednocześnie liczba grup plików dzienników powtórzeń jest za mała, zaś baza pracuje w trybie archiwizacji logów, może wystąpić zjawisko zatrzymania przetwarzania ze względu na brak wolnego pliku logu do zapisu. Obserwuje się również niezakończone wykonanie punktów kontrolnych. Zwykle zakłada się, że przełączanie plików dzienników powtórzeń nie powinno odbywać się częściej, niż co 1.5-2 minuty. O właściwym wystrojeniu tego elementu systemu dowodzi również brak komunikatów w plikach *alrt.log* i plikach śladu (trace).

Prawidłowe zaplanowanie wielkości i liczby plików logów jest na ogół trudne dla nierozpoznanego działania określonej instalacji. Dlatego też zakłada się na ogół, iż właściwe parametry zostaną dobrane już po uruchomieniu systemu. Zmiana rozkładu, liczby i wielkości logów możliwa jest w trakcie pracy bazy i nie wymaga zatrzymania pracy.

W dużych i silnie obciążonych systemach może okazać się celowe rozłożenie poszczególnych plików logów z kolejnych grup na osobne dyski i osobne kanały, tak, aby proces zapisu do kolejnego pliku nie konkurował z procesem ARCH kopiującym zapisany i użyty plik do lokalizacji archiwalnej.

Pliki bazy danych

Pliki bazy danych pozostają w bezpośrednim związku planowaniem przestrzeni tabel. Alokowany wstępnie rozmiar plików powinien uwzględniać wzrost objętości z tytułu wprowadzania danych do bazy. Często stosowana praktyką jest alokacja odpowiadająca przewidywanemu rocznemu przyrostowi wielkości bazy, zaś z obserwacji na bieżąco przyrostu danych administrator wnioskuje o ewentualnej konieczności powiększenia plików.

Pliki bazy danych mogą być zakładane jako samorozszerzające się (autoextend), co zabezpiecza przed zatrzymaniem bazy z powodu braku miejsca na nowo alokowane obszary w aktywnych segmentach. Należy jednak pamiętać, że każde zwiększanie rozmiaru pliku podczas pracy bazy wiąże się z zatrzymaniem przetwarzania na czas potrzebny na organizację nowo alokowanej przestrzeni i jest tym dłuższy, im większy jest obszar nowo alokowany. Stąd wynika zalecenie by wykonywać prealokację przestrzeni dyskowych. W zależności od systemu operacyjnego może istnieć ograniczenie na maksymalny rozmiar pojedynczego pliku, w takim przypadku duże przestrzenie tabel należy umieszczać w kilku plikach. Przykładowo, jeżeli ograniczenie na rozmiar pliku wynosi 4GB, a potrzebna przestrzeń tabel musi mieć rozmiar 10 GB to rozkład plików powinien być następujący:

- Plik1 – 4 GB
- Plik2 – 4 GB
- Plik3 – 2 GB typu autoextend on

Wielkość pliku dla systemowej przestrzeni tabel zależy od wielkości składowanych w bazie jednostek programowych (funkcji, procedur, pakietów, wyzwalaczy). Przykładowo, Oracle w wersji 8.1.6 wymaga

minimalnego rozmiaru pliku systemowej przestrzeni tabel o wielkości 50 MB dla konfiguracji podstawowej (bez instalacji dodatkowych opcji np. replikacyjnej). Plik ten powinno się zakładać w trybie dopuszczającym automatyczne rozszerzanie.

Niektóre systemy operacyjne (np. UNIX) umożliwiają umieszczenie przestrzeni plików bezpośrednio na surowych partycjach z pominięciem systemu plików. To rozwiązanie jest trudniejsze do administrowania, ale może przynieść korzyści wydajnościowe przede wszystkim podczas zapisów ze względu na uniknięcie podwójnego buforowania zapisu (w buforach SGA i w buforach systemu plików).

Parametry pliku init.ora

Podczas podnoszenia instancji bazy danych konieczne jest istnienie pliku *init.ora* umożliwiającego zidentyfikowanie plików kontrolnych opisujących daną bazę oraz określenie szeregu parametrów dotyczących alokacji pamięci operacyjnej, maksymalnej liczby uruchamianych procesów, segmentów wycofania itp. Uruchomienie na tej samej maszynie wielu baz Oracle jest możliwe dzięki wyróżnikowi SID. Powinien on być również zawarty jako człon nazwy pliku konfiguracyjnego. Domyślna nazwa SID to ORCL; plik konfiguracyjny ma wtedy nazwę *initORCL.ora*.

Dokładne ustalenie wartości parametrów w pliku *init.ora* jest możliwe dopiero na etapie strojenia bazy, niemniej jednak główne parametry należy określić na początku. Należy pamiętać, że istotne parametry np. alokacji pamięci nie mogą być zmieniane dynamicznie, zaś po zmianie ich wartości w pliku *init.ora* konieczne jest przeładowanie bazy, by odniosły skutek.

Poniżej podano istotne parametry zawarte w pliku konfiguracyjnym, które powinny być określone przed utworzeniem bazy danych:

- DB_BLOCK_SIZE – określa rozmiar bloku w bazie jak dyskutowano to w punkcie 1.1.1
- DB_NAME – określa globalną nazwę bazy danych (maksymalnie 8 znaków), która wraz z DB_DOMAIN jednoznacznie identyfikuje bazę w ramach struktury sieciowej. Nazwa bazy jest zapisywana równocześnie w plikach bazy, plikach dzienników powtórzeń i pliku kontrolnym.
- DB_DOMAIN – określa domenę w ramach której jest zlokalizowana baza. Ma istotne znaczenie w rozproszonym systemie bazy.
- CONTROL_FILE – określa położenie wszystkich kopii plików kontrolnych opisujących daną bazę danych. Dla bezpieczeństwa powinno się utworzyć co najmniej dwie kopie plików kontrolnych, najlepiej na różnych wolumenach dyskowych. Podczas tworzenia bazy zostaną utworzone pliki kontrolne we wskazanych przez ten parametr lokalizacjach, zaś podczas kolejnych uruchomień istniejące już pliki zostaną użyte do identyfikacji wszystkich plików wchodzących w skład bazy.
- DB_BLOCK_BUFFERS – określa liczbę buforów w pamięci alokowanej przez SGA przeznaczonych na cache bloków danych. Wysoka wartość tego parametru umożliwia zwiększenie współczynnika trafień do SGA, co poprawia znacznie wydajność przetwarzania, ale wymaga fizycznego istnienia dużej pamięci RAM. Należy ostrożnie zwiększać wartość tego parametru by nie dopuścić jednocześnie do zwiększenia stronicowania i swapowania pamięci na dysk (memory paging and swapping).
- SHARED_POOL_SIZE – określa rozmiar obszaru dzielonego w bajtach. Wysoka wartość podobnie jak DB_BLOCK_BUFFERS polepsza wydajność przetwarzania. Określone aplikacje, szczególnie oparte o struktury programowe składowane w bazie mogą wymagać określonej minimalnej wielkości dla tego obszaru.
- PROCESSES – określa maksymalna liczbę procesów systemu operacyjnego, które jednocześnie, mogą być podłączone do Oracle. Należy zarezerwować 5 procesów dla procesów drugoplanowych i po jednym dla każdego procesu użytkownika. Ten parametr powinien być określany na podstawie przewidywanej liczby połączeń do bazy.
- ROLLBACK_SEGMENTS – zawiera listę wszystkich segmentów wycofania, które zostaną podczas startu udostępnione dla aplikacji (status online). Zamiast tego parametru można określić liczbę udostępnionych segmentów wycofania poprzez parametry: TRANSACTIONS oraz TRANSACTIONS_PER_ROLLBACK_SEGMENT
- OPEN_CURSORS – określa maksymalną liczbę jednocześnie otwartych kursorów przez sesję. Parametr jest zależny od aplikacji używających bazy danych i decyduje o możliwości jej uruchomienia. Wartość domyślna 64 jest zwykle za mała dla większości aplikacji produkcyjnych.
- MAX_ENABLED_ROLES – określa maksymalną liczbę ról, które mogą być przyznane użytkownikowi. Podobnie jak OPEN_CURSORS zależy od aplikacji. Wartość domyślna 20 jest zwykle za mała dla większości aplikacji produkcyjnych.
- JOB_QUEUE_PROCESSES – określa liczbę procesów drugoplanowych typu SN*Pn* dla instancji. Jeżeli w bazie istnieją zmaterializowane perspektywy, które maja być automatycznie odświeżane, lub zdefiniowane zadania (jobs), które mają być automatycznie uruchamiane to wartość tego parametru powinna być większa od 0 (zalecane min. 2).

```
Przykład.
Zawartość pliku init.ora

db_name = ORACLE
db_domain = FIRMA.COM.PL
db_block_size = 2048
control_file="c:\Oracle\oradata\ORACLE\control01.ctl",c:\Oracle\oradata\ORACLE\control02.ctl",
"c:\Oracle\oradata\ORACLE\control03.ctl")
db_block_buffers = 5000
shared_pool_size = 50M
processes = 100
rollback_segments = ( RBS0, RBS1, RBS2, RBS3, RBS4 )
open_cursors = 200
max_enabled_roles = 100
job_queue_processes = 4
```

2.2 Struktura katalogów

Niezależnie od platformy, na której instalowane jest oprogramowanie Oracle struktura katalogów jest podobna. Korzeniem dla niej jest katalog określony przez zmienną środowiska ORACLE_HOME w systemie UNIX, lub analogiczną wartość w kluczu ORACLE rejestrów systemu Windows. Dodatkowo, między innymi dla potrzeb wydzielonej lokalizacji plików logów i śladu używana jest zmienna ORACLE_BASE. Z punktu widzenia administratora bazy szczególne znaczenie mają następujące katalogi:

- $ORACLE_HOME/database – zawiera plik parametrów init%SID%.ora i plik hasła dostępu do bazy PWD%SID%.ora
- $ORACLE_HOME/network/admin – zawiera pliki konfiguracyjne dla SQL*Net.
- $ORACLE_BASE/admin/%SID%/pfile – może zawierać plik parametrów jeśli take wskazanie znajdzie się w $ORACLE_HOME/database/init%SID%.ora
- $ORACLE_BASE/admin/%SID%/bdump – zawiera plik *alrt.log* zapisujący informacje o wszystkich uruchomieniach bazy, oraz istotnych zdarzeniach w trakcie pracy bazy
- $ORACLE_BASE/admin/%SID%/udump – zawiera pliki śladu sesji użytkownika
- $ORACLE_BASE/admin/%SID%/cdump – zawiera pliki śladu procesów bazy

Podana powyżej struktura jest charakterystyczna dla wersji Oracle8i; dla innych wersji może się nieco różnić.

W przypadku podnoszenia instancji wielu baz na jednej maszynie możliwe jest umieszczenie jednego pliku *init.ora* dla wszystkich baz i zawierającego zbiór wspólnych parametrów, zaś parametry charakterystyczne dla poszczególnych baz umieścić w oddzielnych plikach *init%SID%.ora*. W takim przypadku pliki odrębne powinny posiadać specyfikację określoną parametrem *ifile*.

```
Przykład.
Zawartość pliku init.ora

IFILE = 'c:\Oracle\admin\ORACLE\pfile\init.ora'
IFILE = /oracle/admin/ORCL/pfile/init.ora
```

2.3 Zakładanie bazy

Mając zaplanowaną bazę danych można przystąpić do jej utworzenia. Uniwersalnym narzędziem administratora jest program **svrmgrl** (server manager) standardowo dostarczany z bazą na wszystkie platformy. Umożliwia on wykonywanie poleceń DDL podawanych w linii komend, jak również wykonywanie skryptów z poleceniami SQL. Oprócz tego umożliwia wykonywanie poleceń DML oraz bloków PL*SQL.

Server manager posiada dodatkowo grupę komend służących do podnoszenia i kładzenia bazy, określania trybu archiwizacji logów oraz odtwarzania bazy po awarii. Listę dostępnych komend można uzyskać wydając komendę **help**.

Podstawowe komendy serwer managera to:

- Startup – podniesienie instancji, otwarcie bazy danych
- Shutdown – położenie bazy
- Archive log – wystartowanie/zatrzymanie trybu archiwizowania logów, wyświetlenie statusu archiwizacji
- Recover – odtworzenie bazy po awarii
- Connect – podłączenie do bazy

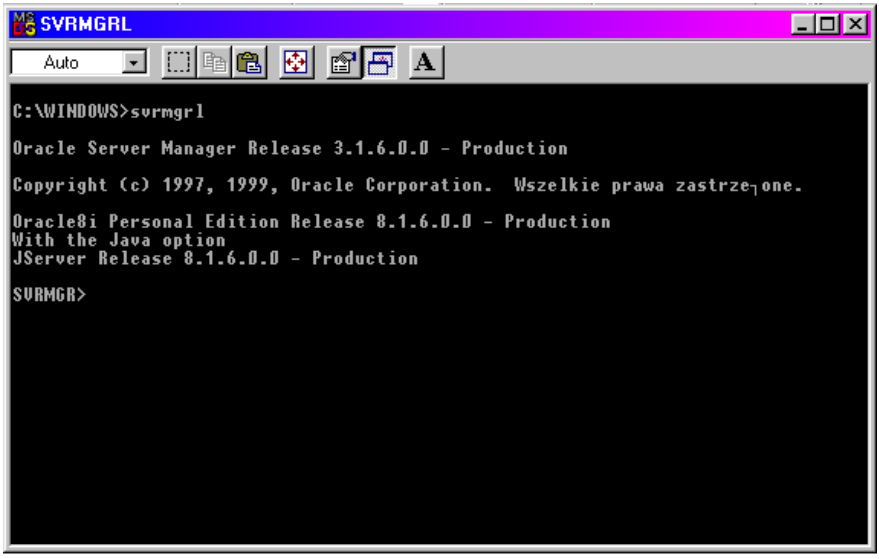
- Disconnect – odłączenie od bazy

Narzędziem graficznym ułatwiającym założenie bazy danych jest **Oracle Database Configuration Assistant,** przeznaczony raczej dla początkującego administratora. Napisany w Javie funkcjonuje na wszystkich platformach systemowych i pozwala w kolejnych krokach ustalić w trybie interakcji z operatorem wszystkie potrzebne parametry bazy, a następnie wygenerować ją samodzielnie.

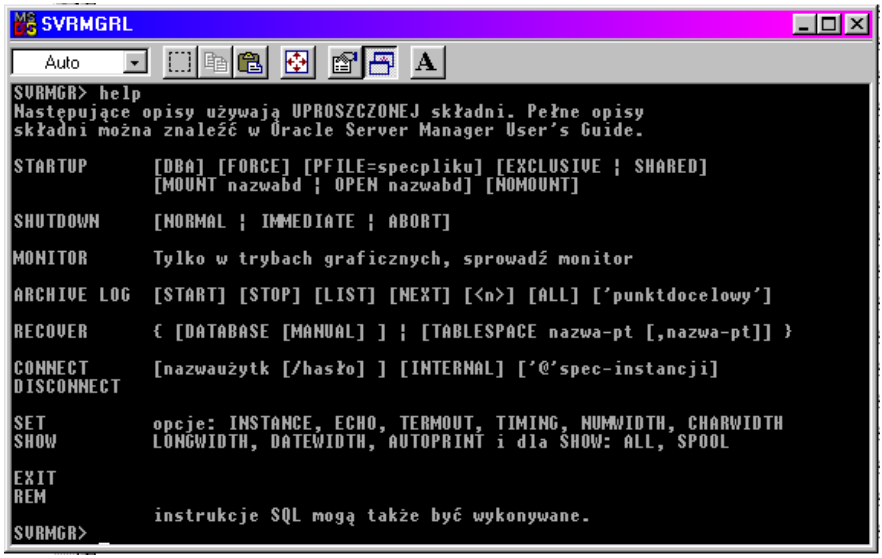
W dalszej części niniejszego opracowania omawiane będą operacje na bazie wykonywane przy pomocy **Server Managera** lub **SQL*Plusa**.

Po utworzeniu bazy komendą CREATE DATABASE zostaną automatycznie utworzone dwa schematy:

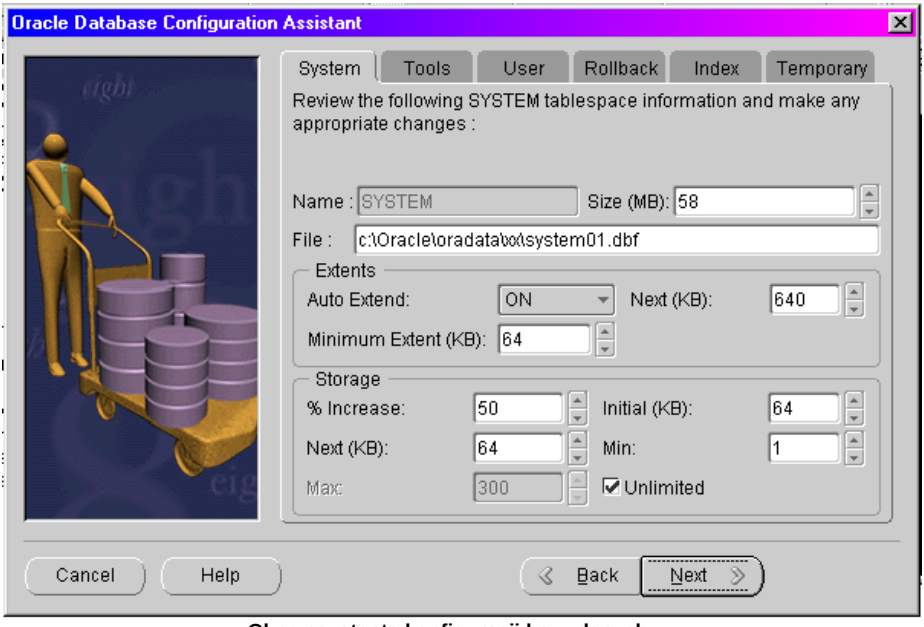
- SYS z hasłem change_on_install
- SYSTEM z hasłem manager



Ekran startowy Server Managera



Dostępne komendy Server Managera



Okno asystenta konfiguracji bazy danych

Bazę danych tworzy się w następujących krokach:

1. Utworzenie pliku parametrów – *init.ora*. Plik parametrów tworzy się najczęściej w oparciu o wzorzec wygenerowany w trakcie instalacji software.
2. Utworzenie pliku hasła – służy do tego program **pwdorcl** posiadający trzy parametry:

- File – nazwa pliku hasła
- Password – hasło dla użytkownika SYS lub INTERNAL
- Entries – maksymalna liczba różnych użytkowników łączących się do bazy na prawach DBA lub OPER (parametr opcjonalny).

```
Przykład.

orapwd file=$ORACLE_HOME/database/PWDORCL.ora password=oracle
orapwd file=c:\oracle\ora81\database\PWDORCL.ora password=oracle
```

W przykładzie pokazano tworzenie pliku hasła na platformie Unix i Windows 2000.

3. Uruchomienie **svrmgrl**. Program svrmgrl uruchamia się z linii komend systemu operacyjnego.
4. Podłączenie do Oracle jako **internal**. Podłączenie wykonuje się komendą **connect internal**. Użytkownik **internal** jest odpowiednikiem użytkownika SYS, ale nie wymaga autoryzacji w bazie danych, co umożliwia podłączenie do bazy przed jej otwarciem. Hasło dla niego jest zapamiętane w pliku utworzonym w kroku drugim.
5. Wystartowanie instancji Oracle. Dla utworzenia nowej bazy danych należy wystartować instancję w trybie nomount.
6. Utworzenie bazy komendą **create database**. Ponieważ komenda **create database** ma dosyć rozbudowaną składnię, wygodnie jest przygotować wcześniej odpowiednie skrypty i wydawać wyłącznie komendy przetworzenia skryptu np. @createdb.sql
7. Wykonanie skryptów tworzących słownik danych – w zależności od potrzeb wykonuje się różne skrypty dostarczone z bazą. Zwykle są to następujące:
   - $ORACLE_HOME/rdbms/admin/catalog.sql – tworzy słownik danych
   - $ORACLE_HOME/rdbms/admin/catproc.sql – generuje standardowe pakiety
   - $ORACLE_HOME/rdbms/admin/catrep.sql – w celu użycia zaawansowanej opcji replikacyjnej
8. Wykonanie na prawach użytkownika SYSTEM skryptu *$ORACLE_HOME/sqlplus/admin/pupbld.sql* – umożliwia ograniczenie wybranym użytkownikom dostępu do wykonywania wskazanych komend poprzez **SQL*Plus** i jednocześnie eliminuje pojawianie się ostrzeżenia przy podłączaniu do bazy z tego narzędzia. Szczegóły dotyczące ograniczania dostępu dla użytkowników opisane są w rozdziale „Security" podręcznika „SQL*Plus User's Guide and Reference"
9. Utworzenie potrzebnych przestrzeni tabel i segmentów wycofania – podobnie jak w przypadku tworzenia bazy najlepiej użyć przygotowanych wcześniej skryptów.

```
Przykład.

C:\>svrmgrl

Oracle Server Manager Release 3.1.6.0.0 - Production

Copyright (c) 1997, 1999, Oracle Corporation.  Wszelkie prawa zastrzeżone.

Oracle8i Personal Edition Release 8.1.6.0.0 - Production
With the Java option
JServer Release 8.1.6.0.0 - Production

SVRMGR> connect internal
Połączony.
SVRMGR> startup nomount
Instancja ORACLE wystartowała.
Całkowity globalny obszar systemu                    64094964 bajty(-ów)
Fixed Size                                              70388 bajty(-ów)
Variable Size                                        53612544 bajty(-ów)
Database Buffers                                     10240000 bajty(-ów)
Redo Buffers                                           172032 bajty(-ów)
SVRMGR>@createdb

Uruchomienie skryptów:
SVRMGR>@%ORACLE_HOME%\rdbms\admin\catalog
SVRMGR>@%ORACLE_HOME%\rdbms\admin\catproc
SVRMGR> connect system/manager
Połączony.
SVRMGR>@%ORACLE_HOME%\sqlplus\admin\pupbld
SVRMGR> connect internal
Połączony.
SVRMGR>@create_tablespace
SVRMGR>@create_rollback_segs
```

Poniżej podany jest przykład skryptów tworzących bazę, przestrzenie tabel i segmenty wycofania.

Przykład.

Skrypt createdb.sql

```
CREATE DATABASE ORACLE
LOGFILE 'E:\ORADB\LOG1A.ORA' SIZE 15M,
        'E:\ORADB\LOG2A.ORA' SIZE 15M,
        'E:\ORADB\LOG3A.ORA' SIZE 15M,
'E:\ORADB\LOG4A.ORA' SIZE 15M,
'E:\ORADB\LOG5A.ORA' SIZE 15M
MAXLOGFILES 32
MAXLOGMEMBERS 2
MAXLOGHISTORY 1
DATAFILE 'E:\ORADB\SYSTEM.ORA' SIZE 50M  REUSE AUTOEXTEND ON NEXT 640K
MAXDATAFILES 254
MAXINSTANCES 1
CHARACTER SET EE8MSWIN1250
NATIONAL CHARACTER SET EE8MSWIN1250;
```

Skrypt create_tablespace.sql

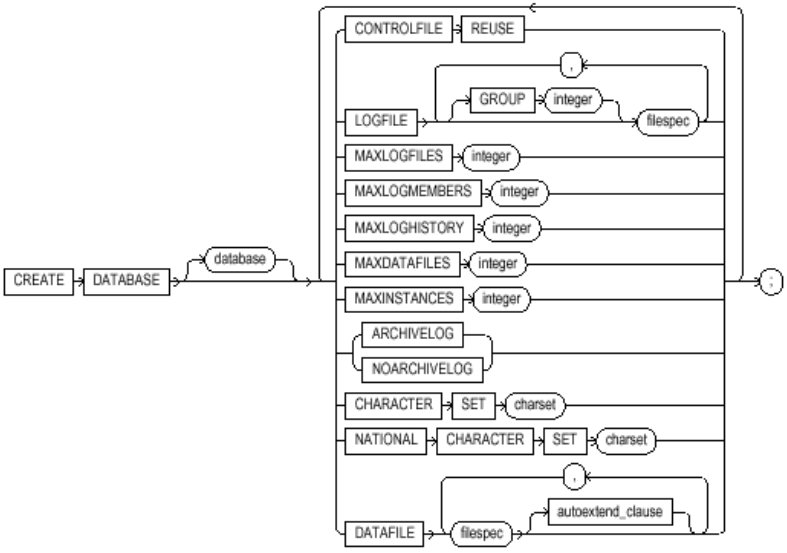
```
CREATE TABLESPACE TEMP DATAFILE 'E:\ORADB\TEMP.ORA' SIZE 10M REUSE
 AUTOEXTEND ON NEXT 2M MAXSIZE 50M TEMPORARY;

CREATE TABLESPACE USER_DATA DATAFILE 'E:\ORADB\USER.ORA' SIZE 10M REUSE
 AUTOEXTEND ON NEXT 2M MAXSIZE 50M;

CREATE TABLESPACE RBS DATAFILE 'E:\ORADB\RBS.ORA' SIZE 10M REUSE
 AUTOEXTEND ON NEXT 2M MAXSIZE 50M;
```

Skrypt create_rollback_segs

```
CREATE PUBLIC ROLLBACK SEGMENT R01 TABLESPACE RBS
  STORAGE (INITIAL 100K NEXT 100K MINEXTENTS 10 OPTIMAL 1000K );
CREATE PUBLIC ROLLBACK SEGMENT R02 TABLESPACE RBS
  STORAGE (INITIAL 100K NEXT 100K MINEXTENTS 10 OPTIMAL 1000K );
CREATE PUBLIC ROLLBACK SEGMENT R03 TABLESPACE RBS
  STORAGE (INITIAL 100K NEXT 100K MINEXTENTS 10 OPTIMAL 1000K );
CREATE PUBLIC ROLLBACK SEGMENT R04 TABLESPACE RBS
  STORAGE (INITIAL 100K NEXT 100K MINEXTENTS 10 OPTIMAL 1000K );
CREATE PUBLIC ROLLBACK SEGMENT R05 TABLESPACE RBS
  STORAGE (INITIAL 100K NEXT 100K MINEXTENTS 10 OPTIMAL 1000K );

alter rollback segment R01 online;
alter rollback segment R02 online;
alter rollback segment R03 online;
alter rollback segment R04 online;
alter rollback segment R05 online;
```

W przykładzie pokazano tworzenie bazy przy pomocy przygotowanych wcześniej skryptów.

Polecenie CREATE DATABASE



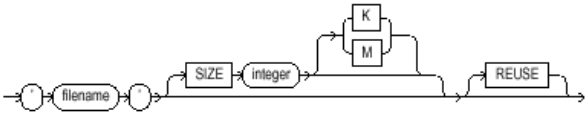
Znaczenie parametrów polecenia:

- DATABASE – nazwa bazy danych (max 8 znaków ASCII); zapisywana jest w pliku kontrolnym i weryfikowana przy wydawaniu komend ALTER DATABASE z użyciem nazwy jako parametru.
- CONTROLFILE REUSE – użycie tego parametru umożliwia w niektórych przypadkach ponowne użycie istniejącego pliku kontrolnego (nadpisanie go nowymi wartościami); stosuje się przy ponownym tworzeniu bazy, gdyż nie wymaga wcześniejszego usuwania już istniejącego pliku kontrolnego
- LOGFILE – specyfikuje pliki dzienników powtórzeń; do pracy bazy wymagane są co najmniej dwie grupy plików. Ze względów bezpieczeństwa można określić więcej niż jednego członka w każdej grupie.
- MAXLOGFILES – określa maksymalną liczbę grup plików dzienników powtórzeń, które można utworzyć dla bazy. Parametr potrzebny jest do rezerwacji odpowiedniej ilości miejsca na nazwy w pliku kontrolnym. Wartość domyślna, minimalna i maksymalna zależy od systemu operacyjnego.
- MAXLOGMEMBERS - określa maksymalną liczbę członków (members) dla grup plików dzienników powtórzeń. Parametr potrzebny jest do rezerwacji odpowiedniej ilości miejsca na nazwy w pliku kontrolnym. Wartość minimalna wynosi 1, domyślna i maksymalna zależą od systemu operacyjnego.
- MAXLOGHISTORY – określa maksymalną liczbę archiwalnych plików dzienników powtórzeń dla automatycznego odtwarzania w instalacji serwera równoległego (Oracle Parallel Server). Parametr potrzebny jest do rezerwacji odpowiedniej ilości miejsca na nazwy w pliku kontrolnym. Wartość minimalna wynosi 0, domyślna zależy od systemu operacyjnego, maksymalna jest limitowana wyłącznie rozmiarem pliku kontrolnego.
- MAXDATAFILES – określa początkowy rozmiar sekcji w pliku kontrolnym przewidzianej na przechowywanie informacji o plikach bazy danych. Liczba plików, których może używać dana instancja bazy jest limitowany również przez parametr DB_FILES w pliku init.ora.
- MAXINSTANCES – określa maksymalną liczbę instancji, które mogą montować i otwierać daną bazę.
- ARCHIVELOG – tworzy bazę w trybie archiwizacji logów
- NOARCHIVELOG – tworzy bazę w trybie bez archiwizacji logów. Parametr domyślny.
- CHARACTER SET – określa stronę kodową, w której będą składowane dane.
- NATIONAL CHARACTER SET – określa stronę kodową, w której będą składowane dane w kolumnach typu: NCHAR, NCLOB i NVARCHAR2. Jeśli parametr nie zostanie określony to jego wartość zostanie przyjęta taka jak podano w CHARACTER SET.
- DATAFILE – określa jeden lub więcej plików, które zostaną użyte do utworzenia przestrzeni tabel SYSTEM.

Specyfikacja pliku w poleceniach DDL

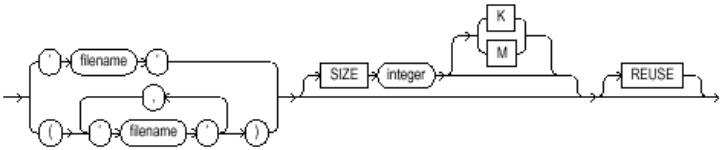
W wielu poleceniach DDL mających związek z plikami używana jest fraza **filespec**. Takimi poleceniami są np. **CREATE DATABASE, ALTER DATATBASE, CREATE TABLESPACE, ALTER TABLESPACE, CREATE CONTROLFILE**. W odniesieniu plików danych ma ona poniższą postać:

filespec_datafiles & filespec_tempfiles::=



W odniesieniu do plików dzienników powtórzeń fraza **filespec** ma postać:

filespec_redo_log_file_groups::=

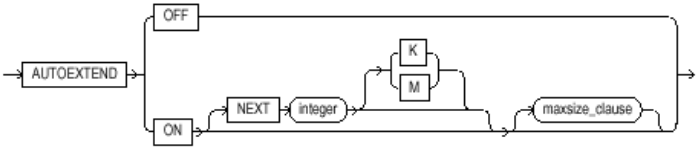


Znaczenie parametrów:

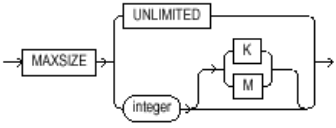
- filename – określa nazwę pliku według konwencji systemu operacyjnego serwera bazy np. 'C:\ORADB\PLIK.ORA' dla Windows, '/oradb/plik.ora' dla UNIX, 'VOL1:ORADB\PLIK.ORA' dla NetWare.
- SIZE – określa rozmiar pliku w bajtach, kilobajtach (K) lub megabajtach (M)
- REUSE – pozwala ponownie użyć istniejący plik (o ile zgadza się jego rozmiar z parametrem SIZE), ale jeżeli zawierał on dane to zostaną one utracone.

Pliki, które mają mieć własność automatycznego rozszerzania się wraz ze wzrostem składowanych danych powinny mieć wyspecyfikowaną własność **autoextend on**. Służy do tego następująca fraza:

autoextend_clause::=



maxsize_clause::=

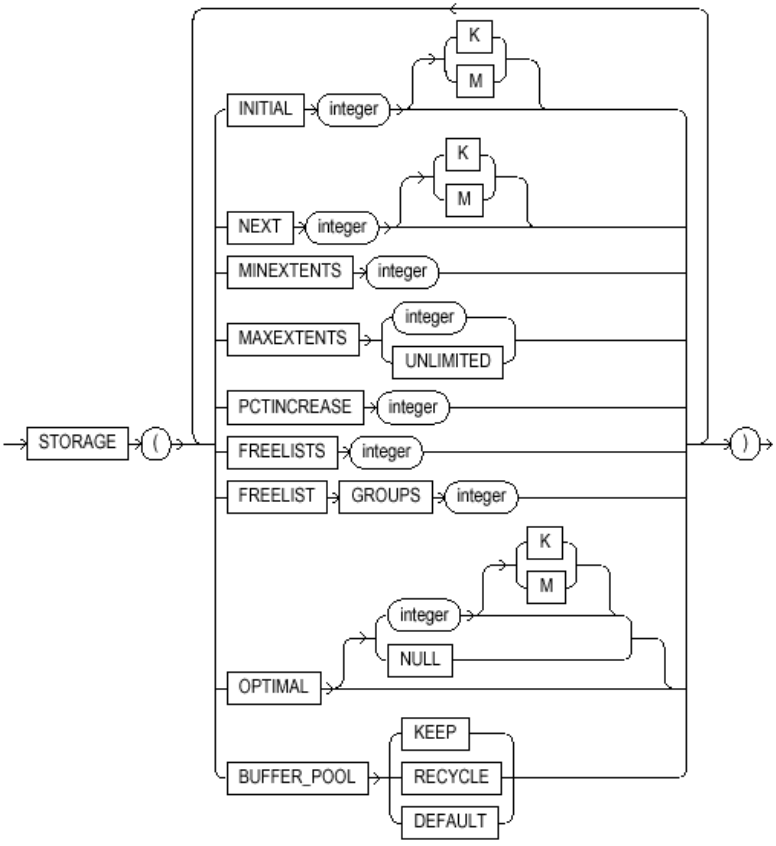


Znaczenie parametrów:

- ON – włączenie trybu automatycznego rozszerzania
- NEXT – określa rozmiar w bajtach, kilobajtach (K) lub megabajtach (M) obszaru, o który zostanie rozszerzony plik w razie potrzeby. Wartością domyślną jest rozmiar jednego bloku.
- MAXSIZE – określa maksymalny rozmiar do jakiego może zostać rozszerzony plik.
- UNLIMITED – określa brak limitu na rozmiar pliku; faktycznie jest on równy maksymalnemu rozmiarowi pliku dostępnego w danym systemie operacyjnym.
- OFF – wyłącza tryb automatycznego rozszerzania. Wartości NEXT i MAXSIZE zostają ustawione na 0 i wymagają ponownego określenia przy ponownym włączeniu trybu.

Specyfikacja parametrów składowania w poleceniach DDL

W poleceniach DDL dotyczących parametrach składowania jest używana fraza storage_clause. Poniżej podane jej składnię.



Znaczenie parametrów:

- INITIAL – rozmiar w bajtach (K – kilobajtach, M – megabajtach) pierwszego obszaru (extend) alokowanego w chwili tworzenia obiektu.
- NEXT – rozmiar w bajtach (K – kilobajtach, M – megabajtach) następnego alokowanego obszaru
- MINEXTENTS – liczba obszarów alokowanych w chwili tworzenia obiektu
- MAXEXTENTS – maksymalna liczba obszarów możliwych do zaalokowania dla obiektu. Do tej liczby wlicza się również pierwszy obszar.
- PCTINCREASE – wyrażona w procentach wartość przyrostu trzeciego i następnych obszarów w stosunku do poprzedzającego. Wartość ta jest używana do obliczenia wielkości kolejnych obszarów alokowanych wraz ze wzrostem wielkości obiektu
- FREELISTS – liczba list wolnych bloków
- FREELIST GROUP – liczba grup list wolnych bloków. Wartość istotna dla pracy serwera równoległego.
- OPTIMAL – optymalny rozmiar segmentu wycofania w bajtach. Parametr stosuje się tylko do segmentów wycofania i służy automatycznemu utrzymaniu przez Oracle założonego rozmiaru segmentu
- BUFFER_POOL – parametr stosuje się do obiektów schematu i określa sposób buforowania w SGA. Przyjmuje wartości:

- KEEP – obiekt jest trzymany w pamięci w celu zmniejszenia liczby operacji IO. Parametr ma wyższy priorytet niż parametr NOCACHE w definicji obiektu
- RECYCLE – bloki obiektu są usuwane z pamięci, gdy nie są już wykorzystywane
- DEFAULT – przypisuje obiekt do domyślnego bufora

3. Podnoszenie i kładzenie bazy

Instancję bazy danych można podnosić oraz zamykać w kilku trybach. Niektóre czynności związane z administracją bazy danych można wykonywać w określonych trybach pracy bazy. Niniejszy rozdział opisuje te zagadnienia.

3.1 Podnoszenie bazy

Bazę można podnosić w trzech trybach:

- Start instancji bez montowania bazy
- Start instancji i zamontowanie bazy bez jej otwarcia
- Start instancji z zamontowaniem i otwarciem bazy w trybie:
  - Nieograniczonym (unrestricted mode) – baza dostępna dla wszystkich użytkowników
  - Ograniczonym (restricted mode) – baza dostępna wyłącznie dla administratorów bazy, użytkowników posiadających jednocześnie dwa przywileje: CREATE SESSION i RESTRICTED SESSION

Bazę można podnosić wyłącznie łącząc się poprzez serwer dedykowany, nigdy wielowątkowy.

Aby podnieść bazę należy podłączyć się przy pomocy **svrmgrl** jako internal a następnie wykonać polecenie:

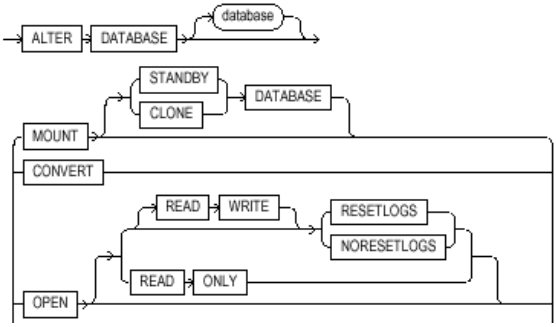
Startup tryb pfile=init%SID%.ora

Znaczenie parametrów:

- tryb – określa tryb podniesienia bazy; wartością domyślna jest start instancji z zamontowaniem i otwarciem bazy
- pfile – określa nazwę pliku inicjacyjnego, na podstawie którego podniesiona baza. Pominięcie tego parametru spowoduje, że zostanie użyty standardowy plik init.ora. Jeżeli zostanie określona zmienna środowiska (lub wartość w rejestrze) ORACLE_SID to zostanie ona użyta do wyznaczenia nazwy pliku inicjacyjnego.

TRYB	Znaczenie	Potrzebny do wykonania operacji
Nomount	Start instancji bez montowania	Tworzenie bazy danych
Mount	Start instancji i zamontowanie bazy	➢ zmiana nazwy pliku bazy danych ➢ dodanie, usunięcie, zmiana nazwy plików dzienników powtórzeń ➢ zmiana trybu archiwizacji logów ➢ wykonanie odtwarzania bazy
		Normalny tryb pracy bazy dostępnej dla wielu użytkowników
Restrict	Start instancji, zamontowanie i otwarcie bazy w trybie ograniczonym	Operacje wymagające otwartej bazy danych, ale przy gwarancji braku podłączenia innych użytkowników, np.: ➢ utrzymanie/konserwacja aplikacji np. przebudowa indeksów ➢ eksport lub import bazy ➢ ładowanie danych za pomocą SQL*Loader
Force	Start instancji, zamontowanie i otwarcie bazy	Start instancji bazy, gdy występują problemy z normalnym startem, lub pracująca baza nie daje się położyć poleceniami SHUTDOWN NORMAL, SHUTDOWN IMMEDIATE, lub SHUTDOWN TRANSACTIONAL.. Jeśli instancja już pracowała przed jej podniesieniem zostanie wykonany SHUTDOWN ABORT.
Open recover	Start instancji, zamontowanie bazy i wykonanie automatycznego odtwarzania	Automatyczne odtwarzanie bazy z logów np. po awarii.

Po podniesieniu instancji bazy możliwe jest wykonanie kolejnych kroków, a więc zamontowanie i otwarcie bazy przy pomocy polecenia **ALTER DATABASE**.



Znaczenie parametrów:

- MOUNT – wykonuje montowanie bazy
- OPEN – wykonuje otwarcie zamontowanej wcześniej bazy danych. Otwarcie możliwe jest w dwóch trybach:
  - READ ONLY – tryb tylko do odczytu, nie powoduje powstawania zapisów do plików dziennika powtórzeń. Jest najczęściej stosowany jako sposób pobierania danych z bazy stand-by.
  - READ WRITE – domyślny tryb pracy bazy, umożliwiający zarówno odczyt jak i zapis do bazy. Dodatkowe parametry RESETLOGS i NORESETLOGS mogą być użyte tylko po niepełnym odtwarzaniu bazy danych, lub odtwarzaniu pełnym, ale z użyciem archiwalnego pliku kontrolnego i wskazują systemowi czy pozostawić bieżący numer sekwencyjny logów i kontynuować proces zapisu do istniejących logów, czy też rozpocząć ten proces od początku, zaczynając numerację logów od 1.

Stan instancji bazy danych można odczytać z perspektywy systemowej V$INSTANCE.

```
Przykład.

SQL> select INSTANCE_NAME,HOST_NAME,STARTUP_TIME,STATUS,ARCHIVER,LOG_SWITCH_WAIT,LOGINS,
  2         SHUTDOWN_PENDING,DATABASE_STATUS "DB STAT"
  3   from v$instance;

INSTANCE_NAME    HOST_NAME  STARTUP_ STATUS  ARCHIVE LOG_SWITCH_ LOGINS     SHU DB STAT
---------------- ---------- -------- ------- ------- ----------- ---------- --- ---------
ORCL             ux0        02/02/01 OPEN    STARTED             ALLOWED    NO  ACTIVE
```

3.2 Kładzenie bazy

Otwartą bazę danych można położyć w kilku trybach. Służy do tego komenda **shutdown** wydawana przy pomocy **serwer managera** przez użytkownika internal.

Parametry komendy shutdown:

- NORMAL – domyślny parametr. Powoduje zamknięcie bazy danych po odłączeniu się wszystkich użytkowników. Po wydaniu komendy żaden nowy użytkownik nie może podłączyć się do bazy, ale baza nie zostanie położona dopóki choć jeden klient jest podłączony do bazy.
- IMMEDIATE – stosuje się, gdy należy położyć bazę danych bez oczekiwania na odłączenie się wszystkich użytkowników, nawet za cenę przerwanych transakcji. Komenda **shutdown immediate** powoduje przerwanie i odwinięcie wszystkich transakcji będących w toku oraz zakończenie wszystkich połączeń użytkowników. Jakkolwiek w większości przypadków ta opcja umożliwia szybkie położenie bazy, to w sytuacji gdy w bazie są obecne długie, niezatwierdzone transakcje czas położenia bazy może okazać się nadspodziewanie długi.
- TRANSACTIONAL – umożliwia położenie bazy bez straty danych przetwarzanych przez aktywnych użytkowników. Po wydaniu komendy z tą opcją niemożliwe jest podłączenie się do bazy nowych użytkowników, wszystkie aktywne transakcje są kończone normalnie, po czym następuje odłączenie użytkowników i położenie bazy.
- ABORT – powoduje natychmiastowe położenie instancji bazy. Procesy użytkowników są przerywane, zaś niezatwierdzone transakcje nie zostają odwinięte przed położeniem bazy. Opcję tę stosuje się w wyjątkowych sytuacjach, gdy okoliczności zmuszają administratora do natychmiastowego i bezwarunkowego położenia bazy.

4. Zabezpieczenie bazy przed awarią

System bezpieczeństwa bazy Oracle oparty jest o mechanizm plików dziennika powtórzeń (redo logs). Gwarantuje on, że wszelkie informacje o zmianach w bazie danych zostaną zapisane na dysku w plikach logów niezależnie od tego czy zostaną zapisane do plików bazy danych. Mechanizm ten umożliwia uzyskanie wysokiej wydajności przetwarzania dzięki silnemu buforowaniu bloków danych w pamięci operacyjnej serwera przy jednoczesnym zagwarantowaniu najwyższego poziomu bezpieczeństwa. Transakcja zostaje uznana za zatwierdzoną dopiero wtedy, gdy system uzyska potwierdzenie, iż informacja o transakcji znajduje się w pliku logu. Należy jednak pamiętać, iż przy stosowaniu macierzy dyskowych może to oznaczać faktycznie, że dane te zostały umieszczone w jej cache, a nie fizycznie na dyskach. Bezpieczeństwo danych wiąże się tu oczywiście z koniecznością zapewnienia odpowiednio bezpiecznego zasilania systemów dyskowych. Większość nowoczesnych, silnie buforowanych macierzy posiada własne, niezależne podtrzymanie pamięci podręcznej.

4.1 Tryby pracy bazy

Z punktu widzenia użycia plików dziennika powtórzeń baza może pracować w dwóch trybach:

- Bez archiwizacji plików dziennika powtórzeń (noarchivelog) – pliki logów są używane naprzemiennie, przed ponownym użyciem dany plik nie jest archiwizowany. Logi zabezpieczają wyłącznie bazę na wypadek awarii instancji. Odtworzenie bazy jest możliwe tylko z informacji zawartych w redo logach (brak archiwalnych logów), a więc wyłącznie z ostatnich operacji bazodanowych. W przypadku uszkodzenia plików bazy nie istnieje możliwość odzyskania danych. Ten tryb zwykle nie jest stosowany w instalacjach produkcyjnych, chyba, że zakłada się możliwość ręcznego uzupełniania danych w bazie w przedziale czasu od ostatniej kopii bezpieczeństwa do chwili wystąpienia awarii.
- Z archiwizacją plików dziennika powtórzeń (archivelog) – zawartość plików danej grupy logów musi być zarchiwizowana przed ponownym użyciem plików tej grupy. Archiwizowanie może być wykonane ręcznie, lub automatycznie poprzez proces ARCH. W oparciu o logi archiwalne i dobrą, pełną kopię bazy można odtworzyć całą bazę danych na stan przed awarią.

Tryb pracy bazy określa się w momencie jej utworzenia poleceniem **CREATE DATABASE** albo zmienia później dla bazy działającej. Zaleca się tworzenie bazy w trybie noarchivelog by uniknąć niepotrzebnego tworzenia archiwalnych logów w trakcie przetwarzania skryptów generujących słownik danych oraz dane aplikacji, zaś włączenie trybu archivelog w chwili przejścia do działania produkcyjnego. Przestawianie trybu pracy bazy odbywa się po jej uprzednim, prawidłowym zamknięciu poleceniem **shutdown normal** lub **shutdown immediate** za pomocą opcji polecenia **ALTER DATABASE**. Przestawienie trybu pracy bazy może być wykonane tylko na bazie zamontowanej (nieotwartej).

```
Przykład.

ALTER DATABASE ARCHIVELOG;

ALTER DATABASE NOARCHIVELOG;
```

Tryb pracy bazy można odczytać z perspektywy V$DATABASE w kolumnie LOG_MODE.

```
Przykład.

select NAME, TO_CHAR( TO_DATE(CREATED,'YY/MM/DD HH24:Mi:SS'),'DD/MM/YYYY HH24:Mi:SS') CREATED,
LOG_MODE,CHECKPOINT_CHANGE#,ARCHIVE_CHANGE#
   from v$database;

NAME      CREATED                   LOG_MODE     CHECKPOINT_CHANGE#  ARCHIVE_CHANGE#
--------- ------------------------- ------------ ------------------ ----------------
BAZA      11/10/2000 00:00:00       ARCHIVELOG               582546           514498
```

5. Zarządzanie plikami dziennika powtórzeń

5.1 Zarządzanie plikami online dziennika powtórzeń (online redo log)

Stany plików dziennika powtórzeń

W każdej grupie plik dziennika może być w jednym z następujących stanów:

- CURRENT – plik jest aktualnie zapisywany przez proces LGWR
- ACTIVE – plik został w całości zapisany przez proces LGWR, ale jest niezbędny do ewentualnego użycia w procesie naprawy instancji (instance recovery). W tym stanie nie można go jeszcze użyć do ponownego zapisu
- INACTIVE - plik został w całości zapisany przez proces LGWR i nie jest już potrzebny do ewentualnego użycia w procesie naprawy instancji (instance recovery). W tym stanie można go już użyć do ponownego zapisu
- UNUSED – plik nie był jeszcze nigdy użyty do zapisu. Stan ten występuje zawsze po dodaniu nowego pliku logu, lub po komendzie RESETLOGS.
- CLEARING – plik logu jest ponownie tworzony jako pusty po komendzie ALTER DATABASE CLEAR LOGFILE. Po zakończeniu procesu otrzymuje status UNUSED
- CLEARING_CURRENT – bieżący log jest w trakcie czyszczenia. Utrzymywanie się w tym stanie świadczy o błędach w trakcie przełączania logów najprawdopodobniej z powodu błędów I/O.

Aktualny stan plików dziennika powtórzeń można odczytać z perspektywy V$LOG w kolumnie STATUS. Dodatkowo, przydatne informacje dla administratora znajdują się w kolumnach:

- GROUP# - numer grupy plików dziennika powtórzeń
- SEQUENCE# - zawiera kolejny numer sekwencyjny pliku
- BYTES – rozmiar pliku
- FIRST_TIME – podaje dokładny czas przełączenia plików dziennika

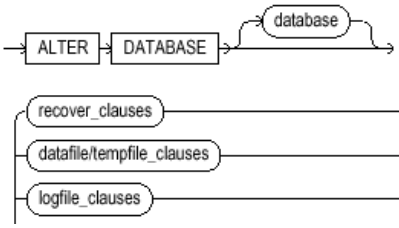
Po każdym przełączeniu plików (log switch), a więc po zakończeniu zapisywania przez LGWR jednego logu i rozpoczęciu zapisywania następnego, numer sekwencyjny jest zwiększany o 1. Określony dla danego logu numer sekwencyjny jest również zachowywany w archiwalnym pliku dziennika powtórzeń, jeśli baza jest w trybie archiwizacji logów. Dzięki mechanizmowi inkrementacji numerów sekwencyjnych możliwa jest unikalna identyfikacja zarówno archiwalnych jak i online redo logów, co jest wykorzystywane w procesie odtwarzania danych po awarii.

Przykład.

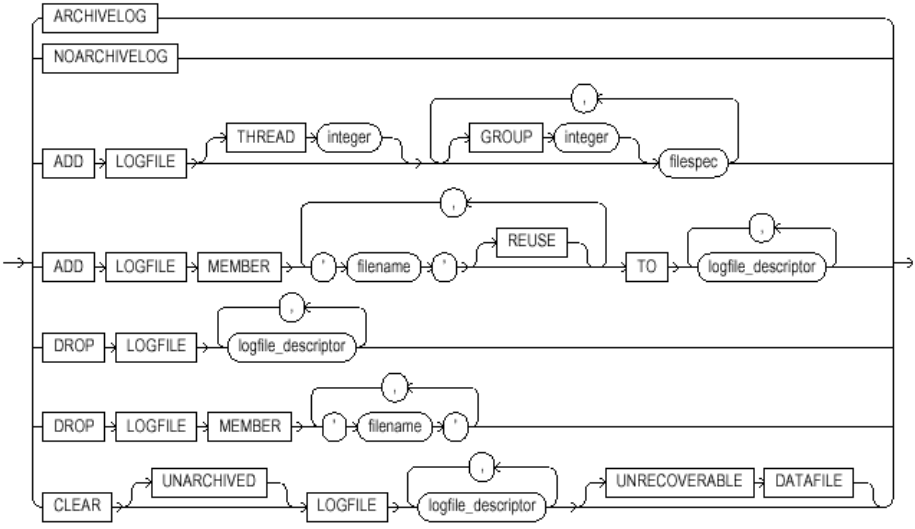
```
SQL>  select GROUP#,THREAD#,SEQUENCE#,BYTES,MEMBERS,ARCHIVED,
  2          STATUS,FIRST_CHANGE#,
  3          TO_CHAR(FIRST_TIME,'DD-MM-YY HH24:MI:SS' ) "FIRST_TIME" from v$log
  4     order by FIRST_CHANGE#;

GROUP#   THREAD# SEQUENCE#     BYTES   MEMBERS ARC STATUS   FIRST_CHANGE# FIRST_TIME
------ --------- --------- --------- --------- --- -------- ------------- -----------------
     5         1        75  10485760         1 YES INACTIVE        461937 03-11-01 10:46:16
     1         1        76  10485760         1 YES INACTIVE        487632 03-11-01 10:48:33
     2         1        77  10485760         1 YES INACTIVE        514498 03-11-01 12:32:03
     3         1        78  10485760         1 YES INACTIVE        540143 03-11-01 12:34:12
     4         1        79  10485760         1 NO  CURRENT         582545 28-11-01 12:16:02
```

Zmiany plików dziennika powtórzeń w ramach istniejącej bazy

Wszystkie parametry takie jak rozmieszczenie, liczba, rozmiar online redo logów itp. są określane w chwili tworzenia bazy danych poleceniem **CREATE DATABASE**. Możliwe jest jednak wykonanie szeregu zmian później, w tym również na pracującej bazie danych. Poniżej podano kilka typowych operacji możliwych do wykonania. Operacje dotyczące logów wykonuje się poleceniem **ALTER DATABASE** w sekcji **logfile_clauses**.



logfile_clauses::=



Informacje o plikach logów można uzyskać z perspektywy V$LOGFILE

```
Przykład.

SQL> select * from v$logfile;

   GROUP# STATUS   MEMBER
--------- -------- ------------------------------
        1          /oracle/oradb/LOG1A.ORA
        2          /oracle/oradb/LOG2A.ORA
        3          /oracle/oradb/LOG3A.ORA
        4          /oracle/oradb/LOG4A.ORA
        5          /oracle/oradb/LOG5A.ORA
```

Tworzenie nowej grupy logów

Wykonuje się poleceniem: **ALTER DATABASE ADD LOGFILE**. W poleceniu tym można również określić dodatkowo numer grupy.

```
Przykład.

SQL>  alter database add logfile group 11 '/oracle/oradb/LOG11A.ORA' size 10M;

Baza danych została zmieniona.
```

W powyższym przykładzie dodano do bazy grupę o numerze 11 logów składającą się z jednego pliku o rozmiarze 10MB.

Dodanie pliku dziennika do istniejącej grupy logów

Wykonuje się poleceniem: **ALTER DATABASE ADD LOGFILE MEMBER**.

```
Przykład.

SQL>  alter database add logfile member '/oracle/oradb/LOG11B.ORA' to
('/oracle/oradb/LOG11A.ORA');

Baza danych została zmieniona.

SQL>  alter database add logfile member '/oracle/oradb/LOG11C.ORA' to
('/oracle/oradb/LOG11A.ORA','/oracle/oradb/LOG11B.ORA');

Baza danych została zmieniona.
```

W powyższym przykładzie dodano do grupy składającej się z pliku */oracle/oradb/LOG11A.ORA* plik */oracle/oradb/LOG11B.ORA*, a następnie do tej samej grupy składającej się już z dwóch plików kolejny element */oracle/oradb/LOG11C.ORA*. Ponieważ wszystkie pliki tej samej grupy muszą mieć jednakowy rozmiar w poleceniu tym nie specyfikuje się rozmiaru pliku.

Usuniecie pliku logu z grupy

Wykonuje się poleceniem **ALTER DATABASE DROP LOGFILE MEMBER**.

```
Przykład.

SQL>  alter database drop logfile member '/oracle/oradb/LOG11C.ORA';

Baza danych została zmieniona.
```

W powyższym przykładzie usunięto z plik */oracle/oradb/LOG11C.ORA*.

Usuniecie grupy logów

Wykonuje się poleceniem **ALTER DATABASE DROP LOGFILE**

```
Przykład.

SQL>  alter database drop logfile group 11;

Baza danych została zmieniona.
```

W powyższym przykładzie usunięto wszystkie pliki logów wchodzących w skład grupy 11.

Należy pamiętać, że usunięcie plików logów poleceniami Oracle nie usuwa ich fizycznie z dysków. Należy tę operację wykonać z poziomu systemu operacyjnego.

Zmiana nazwy plików logów

Zarówno zmianę nazwy plików jak i zmianę ich lokalizacji można wykonać na dwa sposoby:

- Używając polecenia **ALTER DATABASE RENAME FILE** – polecenie to nie ma skutku w systemie operacyjnym, a jedynie zmienia zapisy w pliku kontrolnym. Aby móc go użyć należy po położeniu bazy danych wykonać przy pomocy komend systemu operacyjnego kopiowanie lub zmianę nazwy pliku, a dopiero potem wydać polecenie Oracle. Dla bezpieczeństwa wymagane jest wcześniejsze wykonanie kopii bezpieczeństwa bazy.
- Wykonując kombinację tworzenia nowych i usuwania starych plików dziennika powtórzeń. Operację tę można wykonać nie zatrzymując pracującej bazy.

Należy pamiętać, że po operacjach zmieniających strukturę plików wchodzących w skład bazy danych należy wykonać kopię bezpieczeństwa pliku kontrolnego.

Operacje związane z plikami dzienników powtórzeń

Operacje związane z działaniem plików dziennika powtórzeń wykonywane są za pomocą polecenia **ALTER SYSTEM**

Wymuszenie przełączenia logów

Standardowo przełączenie logów następuje po całkowitym zapisaniu plików jednej grupy i rozpoczęciu przez LGWR zapisu do grupy następnej. Administrator może wymusić przełączenie logów w dowolnej chwili poleceniem **SWITCH LOGFILE**.

```
Przykład.

SQL> alter system switch logfile;

System został zmieniony.
```

Kontrola bloków i czyszczenie logów

W standardowej konfiguracji Oracle nie sprawdza poprawności zapisanych plików dziennika powtórzeń. Istnieje jednak możliwość włączenia opcji kontroli poprawności zapisów poprzez liczenie sumy kontrolnej dla każdego bloku i zapisywanie jej w nagłówku każdego bloku zapisywanego na dysk. O użyciu tej opcji decyduje parametr inicjalizacyjny LOG_BLOCK_CHECKSUM.

```
Przykład.
Zapis w pliku initORCL.ora

LOG_BLOCK_CHECKSUM = TRUE
```

Umieszczenie w pliku inicjalizacyjnym linii jak w przykładzie powoduje, włączenie naliczania i zapisu do logów sum kontrolnych bloków.

Sprawdzenie sumy kontrolnej następuje w dwóch przypadkach:

- Podczas odczytu bloku z pliku dziennika powtórzeń i jego zapisu do logu archiwalnego przez proces ARCH
- Podczas odczytu bloku z archiwalnego pliku dziennika powtórzeń podczas odtwarzania bazy

W pierwszym przypadku przy natrafieniu na błąd Oracle próbuje odczytać blok z innego logu danej grupy. Jeśli nie ma takiego, lub we wszystkich plikach grupy blok jest uszkodzony wtedy proces archiwizacji pliku nie jest możliwy do wykonania i zostaje zatrzymany. Baza jest w stanie pracować jeszcze tak długo jak długo dostępne są kolejne grupy logów. Kiedy jednak kolejne przełączenie plików dziennika powtórzeń spowoduje konieczność zapisu do grupy, na której proces archiwizacji zastał przerwany, praca bazy również zostaje zatrzymana do chwili wznowienia działania procesu ARCH. Jedyną możliwością wznowienia pracy przez proces archiwizacji jest wyczyszczenie uszkodzonego pliku logu komendą **CLEAR**. Po wykonaniu tej komendy uszkodzone pliki zostają wyczyszczone i dostępne do dalszego zapisu, pomimo iż nie zostały zarchiwizowane. Należy pamiętać, że po tej operacji należy wykonać pełną kopię bazy gdyż archiwalne pliki dziennika powtórzeń są niekompletne i nie dają możliwości odtworzenia bazy w przypadku awarii.

```
Przykład.

SQL> alter database clear unarchived logfile group 1;

Baza danych została zmieniona.
```

W przykładzie pliki grupy 1 zostały wyczyszczone.

5.2 Zarządzanie archiwalnymi plikami dziennika powtórzeń (archived redo log)

Wykorzystanie archiwalnych plików dziennika powtórzeń

Archiwalne pliki dziennika powtórzeń są kopią użytych i całkowicie zapisanych plików dziennika powtórzeń (online redo log). W trybie archiwizacji logów bazy danych i przy włączonym automatycznym kopiowaniu procesy ARCn odpowiadają za wykonanie operacji kopiowania. Jednocześnie przed skopiowaniem użytego logu proces LGWR nie ma możliwości ponownego zapisu do niego, co gwarantuje istnienie i spójność wszystkich logów archiwalnych. W przypadku istnienia wieloelementowych grup online redo logów proces ARCn ma możliwość wyboru dobrego pliku o ile jeden z elementów grupy jest uszkodzony. Pliki archiwalne mogą być kopiowane do jednej lub kilku lokalizacji.

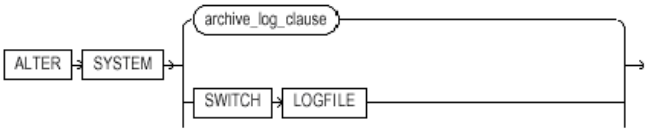
Archiwalne pliki dziennika powtórzeń mogą być użyte do następujących celów:

- Odtwarzania bazy danych po awarii
- Aktualizacji bazy standby

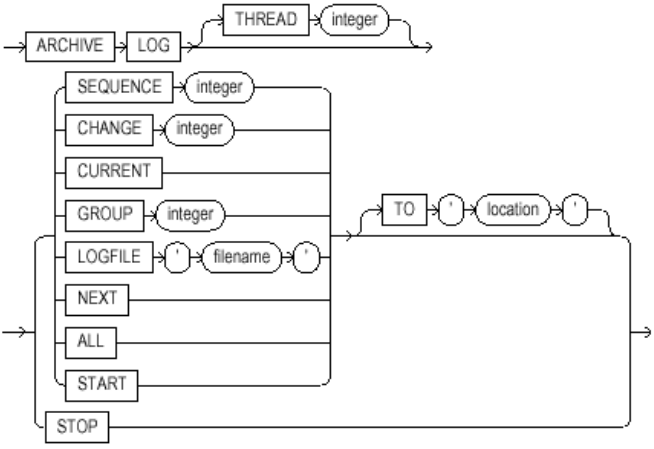
- Pozyskiwania informacji historycznych z bazy danych przy pomocy dodatkowego narzędzia **LogMiner**

Z punktu widzenia bezpieczeństwa bazy danych głównym zastosowaniem archiwalnych plików dziennika powtórzeń jest możliwość ich użycia przy odtwarzaniu bazy po awarii.

Zarządzanie archiwalnymi logami odbywa się przy pomocy komendy **ALTER SYSTEM**



archive_log_clause::=



Włączenie trybu automatycznego archiwizowania

Włączenie trybu automatycznego archiwizowania, a więc uaktywnienie procesów ARCn możliwe jest w momencie startu instancji oraz na polecenie administratora. O tym czy proces archiwizacji wystartuje automatycznie decyduje parametr inicjalizacyjny LOG_ARCHIVE_START.

```
Przykład.
Zapis w pliku initORCL.ora

LOG_ARCHIVE_START = TRUE
```

Umieszczenie w pliku inicjalizacyjnym linii jak w przykładzie powoduje automatyczne włączenie procesu archiwizacji.

Wykonanie tej samej operacji możliwe jest przy użyciu polecenia **ALTER SYSTEM.**

```
Przykład.

SQL> alter system archive log start;

System został zmieniony.

$ ps -fe |grep arc
  oracle   558     1  0 18:30:31 ?        0:00 ora_arc0_ORCL


SQL> alter system archive log stop;

System został zmieniony.
```

```
$ ps -fe |grep arc
```

Pierwsze polecenie przykładu startuje procesy ARCn wykonujące automatyczne archiwizowanie plików dziennika powtórzeń. Drugie polecenie zatrzymuje te procesy. Polecenie systemowe **ps** pokazuje, że w przykładowej konfiguracji uruchomiony jest jeden proces archiwizatora ora_arc0_ORCL dla instancji o identyfikatorze ORCL. Po wykonaniu komendy **stop** proces zostaje zamknięty.

Wymuszanie archiwizacji określonych logów

Administrator może ręcznie sterować archiwizacją logów, wskazując określone *redo logi* do archiwizacji. Służą do tego komendy z frazy **archive_log_clause** polecenia **ALTER SYSTEM**. Znaczenie poszczególnych parametrów:

- THREAD – wskazanie wątka zawierającego grupy logów. Ten parametr jest używany wyłącznie w konfiguracji serwera równoległego (*Parallel Server*)
- SEQUENCE – wskazanie grupy logów do archiwizacji poprzez numer sekwencyjny
- CHANGE – wskazanie logu zawierającego pozycję zmiany (*redo log entry*) ze wskazanym SCN (*system change number*). Jeśli wyspecyfikowany SCN znajduje się w bieżącej grupie logów Oracle wykona przełączenie grup logów
- CURRENT – wskazanie bieżącego logu
- GROUP – wskazanie grupy logów do archiwizacji poprzez jej numer
- LOGFILE – wskazanie pliku logu bezpośrednio przez nazwę
- NEXT – wskazanie następnej grupy logów po ostatnio zarchiwizowanej, która jest już zapisana a nie została jeszcze zarchiwizowana
- ALL – wskazuje wszystkie grupy logów, które są już zapisane a jeszcze nie zostały zarchiwizowane
- TO 'location' – wskazanie miejsca lokalizacji logów archiwalnych. Pominięcie tego parametru powoduje wybranie lokalizacji określonej w pliku inicjalizacyjnym

```
Przykład.

SQL> alter system archive log current;

System został zmieniony.
```

Polecenie wykonuje archiwizację bieżącego logu.

Lokalizacja archiwalnych plików dziennika powtórzeń

Oracle umożliwia kopiowanie plików logów do jednej lub kilku lokalizacji, maksymalnie pięciu. W typowych instalacjach archiwalne pliki kopiuje się do jednej lokalizacji; w określonych przypadkach lokalizacje zwielokrotnia się. Takie przypadki to:

- Potrzeba zwiększenia bezpieczeństwa – zamiast ciągłego kopiowania na nośnik zewnętrzny (zwykle taśmę) wydziela się w lokalnej instalacji odrębne dyski składujące archiwalne logi
- Potrzeba przeniesienia logów poprzez Net8 do zdalnej maszyny utrzymującej bazę standby

Lokalizację, do której kopiuje się pliki dziennika powtórzeń określają parametry pliku inicjalizacyjnego. Możliwe jest zastosowanie dwóch metod wskazania:

- Dla kopiowania lokalnego do maksymalnie dwóch lokalizacji używa się parametrów: LOG_ARCHIVE_DEST oraz LOG_ARCHIVE_DUPLEX_DEST. Pierwszy parametr wskazuje główne miejsce kopiowania, drugi opcjonalną alternatywną lokalizację. Wartością obydwu parametrów jest wskazanie docelowego katalogu na dysku.
- Do kopiowania lokalnego lub zdalnego (poprzez SQL*Net) używa się parametrów LOG_ARCHIVE_DEST_n, gdzie n może przyjmować wartości od 1 do 5, a zatem możliwe jest wskazanie pięciu różnych lokalizacji. Dodatkowo wartość tego parametru jest specyfikowana kluczem LOCATION w celu wskazania miejsca w lokalnym systemie plików, lub kluczem SERVICE w celu zdalnego archiwizowania poprzez Net8. Specyfikując klucz SERVICE używa się nazwy usługi sieciowej tłumaczonej następnie przez Oracle na opis połączenia na podstawie wpisów w pliku *tnsnames.ora*.

```
Przykład.

Wpisy w pliku init.ora określające miejsca kopiowania w lokalnym systemie plików.
```

```
LOG_ARCHIVE_DEST = '/oracle/oralog1'
LOG_ARCHIVE_DUPLEX_DEST = '/oracle/oralog1'
```

W podanym przykładzie pliki dziennika powtórzeń będą kopiowane do dwóch katalogów: /oracle/oralog1 i /oracle/oralog2.

```
Przykład.

Wpisy w pliku init.ora, określające miejsca kopiowania w lokalnym systemie plików.

LOG_ARCHIVE_DEST_1 = 'LOCATION=/oracle/oralog1'
LOG_ARCHIVE_DEST_2 = 'LOCATION=/oracle/oralog2'

LOG_ARCHIVE_DEST_3 = 'SERVICE=bazaSB'
```

W podanym przykładzie pliki dziennika powtórzeń będą kopiowane do dwóch katalogów lokalnych: /oracle/oralog1 i /oracle/oralog2 oraz poprzez SQL*Net do bazy standby. Nazwa serwisu bazaSB musi znajdować się w pliku *tnsnames.ora*.

Administrator może dodatkowo sterować dostępem do wybranych lokalizacji przy pomocy parametrów inicjalizacyjnych LOG_ARCHIVE_DEST_STATE_n nadając im jedną z dwóch wartości:

- ENABLE – zezwala na użycie wskazanej lokalizacji do kopiowania logów
- DEFER – wstrzymuje do odwołania użycie wskazanej lokalizacji

Parametr ten jest dynamiczny i może być zmieniony również podczas pracy bazy poleceniem ALTER SYSTEM.

```
Przykład.

SQL> alter system set log_archive_dest_state_1 = DEFER;

System został zmieniony.
```

W podanym przykładzie po wydaniu komendy pliki logów nie są kopiowane do katalogu wskazanego parametrem LOG_ARCHIVE_DEST_1.

```
Przykład.

SQL> alter system set log_archive_dest_state_1 = ENABLE;

System został zmieniony.
```

W podanym przykładzie po wydaniu komendy pliki logów ponownie są kopiowane do katalogu wskazanego parametrem LOG_ARCHIVE_DEST_1.

Informacje o archiwalnych plikach dziennika powtórzeń można uzyskać z perspektywy V$_ARCHIVE_DEST. Ważnymi kolumnami w tej perspektywie są:

- DEST_ID – określa numer lokalizacji plików logów archiwalnych
- DESTINATION – zawiera miejsce, do którego są kopiowane pliki
- STATUS – określa status dla danej lokalizacji. Może przyjmować wartości:
  - VALID/INVALID – określa czy została wyspecyfikowana lokalizacja plików lub nazwa serwisu dla lokalizacji zdalnej
  - ENABLED/DISABLED – określa, czy Oracle powinien użyć wyspecyfikowanej lokalizacji
  - ACTIVE/INACTIVE – określa, czy wystąpił problem z dostępem do wskazanej lokalizacji

```
Przykład.

SQL> select DEST_ID, STATUS, DESTINATION from V$ARCHIVE_DEST;

  DEST_ID STATUS    DESTINATION
--------- --------- --------------------
        1 VALID     /oracle/oralog1
        2 VALID     /oracle/oralog2
        3 INACTIVE
        4 INACTIVE
        5 INACTIVE
```

Powyższy przykład pokazuje, że w systemie określono dwie lokalizacje plików logów archiwalnych (/oracle/oralog1 i /oracle/oralog2) i nie ma problemów z ich użyciem.

Zasady nazewnictwa archiwalnych plików dziennika powtórzeń

Ze względu na ciągłe kopiowanie tych samych plików dziennika powtórzeń do jednej lokalizacji konieczne jest wyróżnienie kolejnych kopii plików. Oracle stosuje zasadę numerowania plików używając dwóch wartości: numeru sekwencyjnego logu, oraz numeru wątka. Administrator może kształtować nazwy plików używając parametru inicjalizacyjnego LOG_ARCHIVE_FORMAT i zmiennych %s, %S, %t, %T budując z nich wzorzec nazwy. Znaczenie zmiennych:

- %s – numer sekwencyjny logu
- %S – numer sekwencyjny logu dopełniony zerami z lewej strony do stałej długości
- %t – numer wątku
- %T – numer wątku dopełniony zerami z lewej strony do stałej długości

Domyślna wartość parametru LOG_ARCHIVE_FORMAT jest zależna od systemu operacyjnego.

```
Przykład wpisu w pliku init.ora

LOG_ARCHIVE_FORMAT = log%t_%s.arc
```

Określony w przykładzie wzorzec skutkuje tworzeniem kolejnych plików o następujących nazwach (przy założeniu, że aktualny numer sekwencyjny wynosi 1000 i jest to pierwszy wątek logów):

log1_1001.arc, log1_1002.arc, log1_1003.arc ....

Informacje o archiwalnych logach

Administrator może uzyskać informacje dotyczące archiwalnych logów z szeregu perspektyw systemowych. Poniżej podano niektóre z nich z opisem ważniejszych kolumn:

- V$PARAMETER – informacje o parametrach inicjalizacyjnych
- V$ARCHIVED_LOG – szczegółowe informacje o logach archiwalnych pobrane z pliku kontrolnego. Po każdej udanej archiwizacji logu, lub jego wyczyszczeniu dostaje dopisany jeden wiersz widoczny w tej perspektywie. Wiersz jest również dopisany po odtwarzaniu logu z backupu. Ważniejsze kolumny:
  - RECID – identyfikator wiersza logu archiwalnego
  - NAME – nazwa pliku logu archiwalnego. Wartość null wskazuje, że log został wyczyszczony przed archiwizacją
  - SEQUENCE# - numer sekwencyjny logu
  - FIRST_CHANGE# - numer pierwszej zmiany w logu
  - FIRST_TIME – dokładny czas pierwszej zmiany w logu
  - NEXT_CHANGE# - numer pierwszej zmiany w następnym logu
  - NEXT_TIME – dokładny czas pierwszej zmiany w następnym logu
  - BLOCKS – rozmiar archiwizowanego logu w blokach
  - BLOCK_SIZE – rozmiar bloku logu
  - COMPLETION_TIME – dokładny czas zakończenia archiwizacji logu
- V$ARCHIVE_DEST – informacje o lokalizacjach logów archiwalnych. Perspektywa opisana w rozdziale „Lokalizacja archiwalnych plików dziennika powtórzeń”
- V$ARCHIVE_PROCESSES – informacje o stanie procesów ARCn instancji. Ważniejsze kolumny:
  - PROCESS – numer procesu ARC (0-9)
  - STATUS – status procesu. Możliwe są następujące wartości kolumny: STOPPED, SCHEDULED, STARTING, ACTIVE, STOPPING, TERMINATED.
  - STATE – bieżący stan procesu ARCn. Możliwe są następujące wartości kolumny: IDLE, BUSY.
  - LOG_SEQUENCE – numer sekwencyjny logu archiwizowanego w danym momencie. Wartość kolumny wynosi 0 jeśli STATE=IDLE, lub odpowiada numerowi sekwencyjnemu gdy STATE=BUSY
- V$LOG_HISTORY – informacje historyczne o każdym logu pobrane z pliku kontrolnego. Ważniejsze kolumny:
  - RECID – identyfikator wiersza w pliku kontrolnym
  - THREAD# - numer wątka logu
  - SEQUENCE# - numer sekwencyjny logu archiwalnego

- FIRST_TIME – dokładny czas pierwszej pozycji zmiany w logu – najmniejszej wartość SCN w tym logu.
- FIRST_CHANGE# - najmniejsza wartość SCN w tym logu.
- NEXT_CHANGE# - największa wartość SCN w tym logu.

```
Przykład.

SQL> select name,value from V$PARAMETER
  2    where name like '%archive%';

NAME                                 VALUE
------------------------------------ -------------------------
log_archive_start                    TRUE
log_archive_dest
log_archive_duplex_dest
log_archive_dest_1                   location=/oracle/oralog
log_archive_dest_2                   LOCATION=/oracle/oralog1
log_archive_dest_3
log_archive_dest_4
log_archive_dest_5
log_archive_dest_state_1             enable
log_archive_dest_state_2             enable
log_archive_dest_state_3             enable
log_archive_dest_state_4             enable
log_archive_dest_state_5             enable
log_archive_max_processes            2
log_archive_min_succeed_dest         2
standby_archive_dest                 ?/dbs/arch
log_archive_format                   log%t_%s.dbf

17 wierszy zostało wybranych.
```

Przykład pokazuje sposób uzyskania informacji o parametrach w pliku inicjalizacyjnym dotyczącym archiwalnych plików dziennika powtórzeń.

```
Przykład.

SQL> select NAME, SEQUENCE#, FIRST_CHANGE#,
  2         TO_CHAR(FIRST_TIME,'YY-MM-DD HH24:MI:SS') "FIRST_TIME", BLOCKS,
  3         ARCHIVED, DELETED
  4    from V$ARCHIVED_LOG
  5   where SEQUENCE# between 60 and 65;

NAME                         SEQUENCE# FIRST_CHANGE# FIRST_TIME           BLOCKS ARC DEL
---------------------------- --------- ------------- ----------------- --------- --- ---
/oracle/oralog/log1_60.dbf          60        264946 01-02-07 10:39:16     20478 YES NO
/oracle/oralog/log1_61.dbf          61        265050 01-02-07 10:40:29     20480 YES NO
/oracle/oralog/log1_62.dbf          62        265073 01-02-07 10:41:27     20480 YES NO
/oracle/oralog/log1_63.dbf          63        265458 01-02-07 10:42:37      5468 YES NO
/oracle/oralog/log1_64.dbf          64        285738 01-02-08 13:15:49        45 YES NO
/oracle/oralog/log1_65.dbf          65        305780 01-02-08 13:20:06     20480 YES NO

6 wierszy zostało wybranych.
```

W przykładzie pokazano informacje o archiwalnych plikach dziennika powtórzeń o numerach sekwencyjnych 60..65. Z faktu, iż pliki 63 i 64 zawierają mniejszą liczbę bloków można wywnioskować, że wymuszono przełączenie tych logów przed ich całkowitym wypełnieniem.

```
Przykład.

SQL> select PROCESS, STATUS, LOG_SEQUENCE, STATE from V$ARCHIVE_PROCESSES;

  PROCESS STATUS     LOG_SEQUENCE STAT
--------- ---------- ------------ ----
        0 ACTIVE                0 IDLE
        1 ACTIVE                0 IDLE
        2 ACTIVE                0 IDLE
        3 STOPPED               0 IDLE
        4 STOPPED               0 IDLE
        5 STOPPED               0 IDLE
        6 STOPPED               0 IDLE
        7 STOPPED               0 IDLE
        8 STOPPED               0 IDLE
        9 STOPPED               0 IDLE
```

```
10 wierszy zostało wybranych.

$ ps -fe |grep ora_arc
  oracle   729     1  0   gru 13 ?        0:00 ora_arc2_ORCL
  oracle   725     1  0   gru 13 ?        0:00 ora_arc0_ORCL
  oracle   727     1  0   gru 13 ?        0:00 ora_arc1_ORCL
```

Z wydanego w przykładzie polecenia widać, że uruchomiono trzy procesy archiwizacyjne, bezczynne w chwili wydania komendy. Dodatkowo zweryfikowano to przez identyfikację tych procesów poleceniem systemowym **ps**.

```
Przykład.

SQL> select RECID, STAMP, THREAD#, SEQUENCE#, FIRST_CHANGE#,
  2         TO_CHAR(FIRST_TIME,'YY-MM-DD HH24:MI:SS') "FIRST_TIME", NEXT_CHANGE#
  3     from V$LOG_HISTORY
  4   where SEQUENCE# between 60 and 65;

    RECID     STAMP   THREAD# SEQUENCE# FIRST_CHANGE# FIRST_TIME        NEXT_CHANGE#
--------- --------- --------- --------- ------------- ----------------- ------------
       60 421065629         1        60        264946 01-02-07 10:39:16       265050
       61 421065687         1        61        265050 01-02-07 10:40:29       265073
       62 421065758         1        62        265073 01-02-07 10:41:27       265458
       63 421161349         1        63        265458 01-02-07 10:42:37       285738
       64 421161606         1        64        285738 01-02-08 13:15:49       305780
       65 426254545         1        65        305780 01-02-08 13:20:06       312044

6 wierszy zostało wybranych.
```

W przykładzie użyto perspektywy V$LOG_HISTORY do uzyskania informacji o logach.

Przydatnym dla administratora poleceniem umożliwiającym szybkie uzyskanie informacji o stanie archiwizacji logów w systemie jest ARCHIVE LOG LIST. Polecenie można wydać zarówno przy pomocy **server managera** jak i **SQL*Plus**. Wymagane jest posiadanie prawa SYSDBA.

```
Przykład.

SQL> archive log list
Tryb dziennika bazy danych              Tryb archiwizacji
Automatyczne archiwizowanie             Włączone
Miejsce archiwizowania            /oracle/oralog1
Najstarsza sekwencja dziennika online     108
Następna sekwencja dziennika do archiwizowania   112
Bieżąca sekwencja dziennika           112
```

W przykładzie uzyskano informacje o stanie archiwizacji bazy komendą SQL*Plusa.

6. Kopia bezpieczeństwa bazy (backup)

6.1 Wprowadzenie do problematyki kopii bezpieczeństwa

Konieczność zabezpieczenia bazy

Podstawowym obowiązkiem administratora bazy danych jest zabezpieczenie jej na wypadek awarii. W rozdziale 3 i 4 opisano tryby pracy bazy gwarantujące osiągnięcie odpowiedniego poziomu bezpieczeństwa oraz zarządzanie plikami dziennika powtórzeń mającymi krytyczny wpływ na możliwość odtworzenia bazy po awarii. Niniejszy rozdział opisuje zagadnienia związane z zabezpieczaniem bazy na wypadek poważnej awarii skutkującej zniszczeniem części lub całości bazy. Pomimo stosowania różnych zabezpieczeń systemowych i sprzętowych np. dyski lustrzane, macierze RAID0, RAID5 itp. ryzyko uszkodzenia plików bazy istnieje zawsze, co oznacza konieczność okresowego kopiowania bazy na inne nośniki. Najczęstszymi powodami uszkodzenia lub usunięcia ważnych danych z bazy są:

- Awaria sprzętu: dysków, kontrolerów dyskowych itp.
- Awaria logiczna bazy widziana często jako uszkodzenie bloków danych przy prawidłowo działających dyskach
- Błąd ludzki np. przypadkowe usunięcie tabeli, lub schematu, usunięcie lub zniszczenie pliku bazy.

Niezwykle istotne jest posiadanie przez administratora świadomości możliwości popełnienia błędu przy wykonywaniu kopii bezpieczeństwa. Opracowane procedury powinny zapewniać możliwość korekcji takich błędów np. poprzez istnienie redundantnych taśm oraz alternatywnych metod odzyskania bazy.

Kopie bezpieczeństwa można podzielić na dwie grupy:

- Fizyczne – oznacza skopiowanie fizyczne plików bazy danych
- Logiczne – oznacza zabezpieczenie wybranych, lub wszystkich danych z bazy w pliku utworzonym narzędziem **export**. Ta metoda nie jest stosowana jako podstawowy sposób backupu, można jej używać jako metody dodatkowej lub uzupełniającej. W przypadku bardzo dużych baz danych odtwarzanie nawet części bazy tą metodą jest problematyczne ze względu na bardzo długi czas importu.

Elementy składowe kopii bezpieczeństwa

Dla skutecznego odtworzenia bazy po awarii konieczne jest zabezpieczenie odpowiednich elementów składowych bazy. Należą do nich:

- Pliki bazy danych (datafiles)
- Pliki kontrolne (controlfiles)
- Segmenty wycofania (rollback segments)
- Pliki dziennika powtórzeń (online redo log files)
- Archiwalne pliki dziennika powtórzeń (archived redo log files)

Dodatkowo należy również pamiętać o zabezpieczeniu pliku konfiguracyjnego, zawierającego niejednokrotnie istotne parametry inicjalizacyjne ustalone w procesie strojenia instancji bazy. Szybkość otworzenia wszystkich plików w tym również plików konfiguracyjnych bazy, listenera SQL*Net, profilu właściciela bazy (oracle) itp. decyduje ostatecznie o długości przerwy w pracy bazy spowodowanej awarią.

Informacje o elementach składowych bazy danych uzyskuje się z następujących perspektyw:

- V$DATAFILE – informacje o plikach danych
- V$LOGFILE – informacje o plikach dziennika powtórzeń
- V$CONTROLFILE – informacje o pliku kontrolnym

Rodzaje fizycznej kopii bezpieczeństwa

Z punktu widzenia zakresu zabezpieczanych części bazy, kopię bezpieczeństwa można podzielić na:

- Pełną kopię bazy danych – obejmuje wszystkie elementy składowe bazy wyszczególnione w punkcie 5.2. Jest standardowo przyjętą metodą zabezpieczania bazy.
- Kopię przestrzeni tabel – obejmuje backup wybranej przestrzeni tabel, czyli wszystkich plików tworzących tę przestrzeń. Kopię taką można wykonać tylko, gdy baza pracuje w trybie archiwizacji logów, gdyż są one niezbędne do uzgodnienia spójności, po odtworzeniu, z resztą przestrzeni tabel w bazie. Wyjątkiem są przestrzenie tabel w trybie read-only lub offline, których stan jest niezmienny.

- Kopię pliku bazy danych – obejmuje kopię wybranego pliku wchodzącego w skład określonej przestrzeni tabel. Metoda ta jest raczej rzadko wykonywana i stosują się do niej zasady analogiczne jak przy kopii przestrzeni tabel.
- Kopię pliku kontrolnego – obejmuje wykonanie wyłącznie kopii pliku kontrolnego.

Spójność kopii (consistent backup)

Przez spójność kopii rozumie się taki stan, w którym wszystkie pliki bazy danych pracujące w trybie read-write oraz plik kontrolny posiadają zapisany taki sam numer SCN (system change number). Przy próbie odzyskania bazy z kopii, Oracle sprawdza spójność kopii przez porównanie SCN odczytanego z nagłówków wszystkich plików bazy z wartością zapisaną w pliku kontrolnym. W trakcie pracy bazy uzgodnienie spójności odbywa się podczas wykonywania punktu kontrolnego (checkpoint). Spójna kopia bezpieczeństwa umożliwia start bazy bez potrzeby odtwarzania z logów. Kopię taką uzyskuje się z bazy położonej w trybie *normal* lub *immediate*. Kopie utworzone z baz zamkniętych w trybie *abort* lub po upadku instancji są zawsze niespójne i przy starcie wymagają odtworzenia (recovery).

Jeżeli baza pracuje w trybie *noarchivelog* wtedy jedyna nadająca się do uruchomienia kopia bezpieczeństwa musi być wykonana jako kopia spójna, dla bazy pracującej w trybie *archivelog* możliwe jest odtworzenie stanu spójnego na podstawie plików logów i informacji zawartej w pliku kontrolnym.

Metody wykonania kopii bezpieczeństwa

Oracle8i umożliwia wykonanie kopii bezpieczeństwa następującymi metodami:

- Oprogramowaniem Oracle **rman** (Recovery Manager) – umożliwia wykonanie kopii fizycznej, zarządza procesem kopiowania i odtwarzania. Implementuje backup przyrostowy (incremental backup).
- Komendami systemu operacyjnego – umożliwia wykonanie kopii fizycznej poprzez kopiowanie plików np. komendą **cp**, **dd** itp. Istnieje specjalistyczne oprogramowanie firm trzecich automatyzujące ten proces; możliwe jest również użycie skryptów napisanych przez administratora.
- Oprogramowaniem Oracle **export** – umożliwia wykonanie kopii logicznej. Operację odwrotną – odtworzenie danych wykonuje program **import**

6.2 Realizacja kopii bezpieczeństwa

Zarządzanie plikiem kontrolnym

Jak podano powyżej plik kontrolny ma szczególne znaczenie dla problemu bezpieczeństwa bazy. Zawiera on zapisaną informację o plikach wchodzących w skład bazy danych, używaną w przypadku:

- Identyfikacji plików otwieranych przy starcie bazy
- Identyfikacji plików wymagających odtworzenia przy podnoszeniu upadłej instancji

Zapisy te ulegają aktualizacji w przypadku:

- Dodania nowego pliku do bazy
- Zmiany nazwy pliku bazy
- Usunięcia pliku używanego przez bazę

Dotyczy to zarówno plików wchodzących w skład przestrzeni tabel jak i plików dzienników powtórzeń (online redo logs). Jeżeli zostaną wykonane na bazie operacje opisane powyżej, należy natychmiast wykonać kopię bezpieczeństwa pliku kontrolnego.

Przykład.

```
ALTER DATABASE [ADD | DROP] LOGFILE
ALTER DATABASE [ADD | DROP] LOGFILE MEMBER
ALTER DATABASE [ADD | DROP] LOGFILE GROUP
ALTER DATABASE [ARCHIVELOG | NOARCHIVELOG]
ALTER DATABASE RENAME FILE
CREATE TABLESPACE
ALTER TABLESPACE [ADD | RENAME] DATAFILE
ALTER TABLESPACE [READ WRITE | READ ONLY]
DROP TABLESPACE
```

W przykładzie podano komendy, po wykonaniu których należy wykonać kopię bezpieczeństwa pliku kontrolnego.

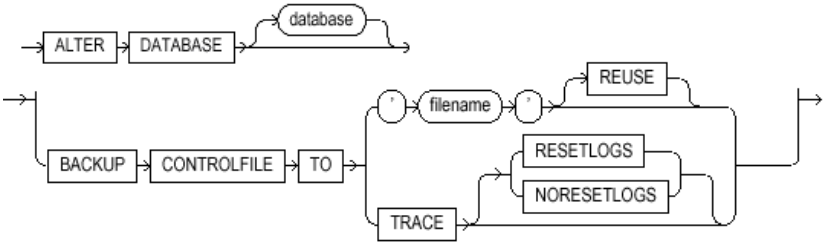
Duplikacja pliku kontrolnego

Oracle potrafi utrzymywać więcej niż jeden plik kontrolny. Zasadą powinno być istnienie co najmniej dwóch plików kontrolnych w bazie położonych fizycznie na dwóch różnych dyskach. Liczbę i położenie plików kontrolnych określa parametr inicjalizacyjny CONTROL_FILES

```
Przykład.
Zapis w pliku init.ora.

control_files = ("/oracle/oradb/ctrl01.ctl","/oracle1/oradb/ctrl02.ctl")
```

Kopia pliku kontrolnego

Kopię pliku kontrolnego można wykonać w trybie on-line komendą **ALTER DATABASE**



Komenda ta umożliwia wykonanie kopii pliku kontrolnego albo w postaci pliku binarnego – opcja *filename* lub bezpośrednio w postaci skryptu generującego plik kontrolny zapisanego w pliku *trace*. Opcja RESETLOGS/NORESETLOGS wskazuje jaka składnia komendy do otwarcia bazy zostanie umieszczona w pliku. W zależności od użytej opcji zostanie wygenerowana komenda **ALTER DATABASE OPEN RESETLOGS** lub **ALTER DATABASE OPEN NORESETLOGS**.

Zabezpieczona opisaną metodą kopia pliku kontrolnego może być użyta w sytuacji utraty wszystkich plików kontrolnych bazy. Dysponując skryptem generującym plik kontrolny można w takiej sytuacji utworzyć go ponownie.

```
Przykład.

SQL> alter database backup controlfile to '/oracle/oradb/CTRL.bak' reuse;

Baza danych została zmieniona.
```

W przykładzie wykonano kopię pliku kontrolnego w postaci binarnej. Została utworzona kopia w postaci pliku *CTRL.bak* w katalogu */oracle/oradb*. Użyta opcja reuse umożliwia nadpisanie pliku kopii, jeśli taki sam był już utworzony wcześniej.

```
Przykład.

SQL> alter database backup controlfile to trace noresetlogs;

Baza danych została zmieniona.
```

W przykładzie wykonano kopię pliku kontronego w postaci skryptu zapisanego w pliku śladu.

```
Przykład.

$ more ora_10818.trc
Dump file /oracle/admin/udump/ora_10818.trc
Oracle8i Enterprise Edition Release 8.1.5.0.0 - Production
With the Partitioning and Java options
PL/SQL Release 8.1.5.0.0 - Production
ORACLE_HOME = /oracle/product/8.1.5
System name:    SunOS
Node name:      ux1
Release:        5.8
Version:        Generic
Machine:        i86pc
```

```
Instance name: ORCL
Redo thread mounted by this instance: 1
Oracle process number: 13
Unix process pid: 10818, image: oracle@ux1 (TNS V1-V3)

*** SESSION ID:(15.4) 2002.01.30.14.16.39.000
*** 2002.01.30.14.16.39.000
# The following commands will create a new control file and use it
# to open the database.
# Data used by the recovery manager will be lost. Additional logs may
# be required for media recovery of offline data files. Use this
# only if the current version of all online logs are available.
STARTUP NOMOUNT
CREATE CONTROLFILE REUSE DATABASE "BAZACOM0" NORESETLOGS ARCHIVELOG
    MAXLOGFILES 32
    MAXLOGMEMBERS 4
    MAXDATAFILES 64
    MAXINSTANCES 4
    MAXLOGHISTORY 1630
LOGFILE
  GROUP 1 '/oracle/oradb/LOG1A.ORA'  SIZE 10M,
  GROUP 2 '/oracle/oradb/LOG2A.ORA'  SIZE 10M,
  GROUP 3 '/oracle/oradb/LOG3A.ORA'  SIZE 10M,
  GROUP 4 '/oracle/oradb/LOG4A.ORA'  SIZE 10M,
  GROUP 5 '/oracle/oradb/LOG5A.ORA'  SIZE 10M
DATAFILE
  '/oracle/oradb/SYSTEM.ORA',
  '/oracle/oradb/USER.ORA',
  '/oracle/oradb/TEMP.ORA',
  '/oracle/oradb/ROLL.ORA',
  '/oracle/oradb/DATA.ORA',
  '/oracle/oradb/IND.ORA',
CHARACTER SET EE8ISO8859P2
;
# Recovery is required if any of the datafiles are restored backups,
# or if the last shutdown was not normal or immediate.
RECOVER DATABASE
# All logs need archiving and a log switch is needed.
ALTER SYSTEM ARCHIVE LOG ALL;
# Database can now be opened normally.
ALTER DATABASE OPEN;
# No tempfile entries found to add.
#
```

Przykład zawiera treść pliku *trace* z wygenerowaną komendą utworzenia pliku kontrolnego. Należy zwrócić uwagę, że jest to kompletny skrypt służący do podniesienia bazy po awarii z utratą pliku kontrolnego.

Planowanie strategii backupu

Strategia kopii bezpieczeństwa obejmuje ustaloną dla każdej instalacji metodę utrzymania i wykonywania kopii bezpieczeństwa. Nie istnieje jedna, najlepsza strategia odpowiednia dla dowolnej bazy danych. Czynnikami wpływającymi na określenie strategii są:

- Środki techniczne posiadane przez administratora
- Tryb pracy bazy – wymagana dostępność systemu w ciągu dnia/tygodnia
- Maksymalny zakładany czas odtworzenia bazy po awarii
- Dopuszczalność odtworzenia części zapisów z dokumentów papierowych.

Opisane czynniki są wzajemnie zależne, np. zakładany tryb pracy bazy 24x7 (24 godziny dziennie przez 7 dni w tygodniu) jest możliwy do uzyskania przy zastosowaniu określonych środków technicznych umożliwiających nieprzerwaną pracę systemu i redundancję wszystkich komponentów sprzętowych. Wymaga również zapewnienia możliwości wymiany uszkodzonych elementów bez przerywania pracy bazy. Bardzo krótki czas podniesienia bazy po awarii wymaga z kolei stosowania rozwiązań typu baza standby, kopie dyskowe itp.

Planując strategię backupu należy jednak bezwzględnie przyjąć jako priorytet bezpieczeństwo, rozumiane jako bezwzględnie pewną możliwość odzyskania bazy z kopii. Należy przewidzieć sytuację, w której utracie ulega również kopia bezpieczeństwa. Zabezpieczeniem przed tym jest jedynie posiadanie alternatywnych ścieżek odzyskiwania danych z redundantnych kopii. Oznacza to konieczność przechowywania w dłuższym okresie czasu starych kopii bezpieczeństwa i wielokrotnych kopii archiwalnych plików dziennika powtórzeń oraz eksploatację bazy w trybie *archivelog*.

Dobrą praktyką administratora jest okresowe weryfikowanie zastosowanych procedur, polegające na odzyskaniu bazy z kopii na innej maszynie i testowe uruchomienie tak utworzonego systemu.

6.3 Kopia bezpieczeństwa wykonywana poprzez system operacyjny

Niniejszy rozdział opisuje wykonywanie kopii bezpieczeństwa przy pomocy poleceń systemu operacyjnego. Polega ona na kopiowaniu wskazanych plików do innej lokalizacji, którą może być: inny dysk, odległy system plików, biblioteka taśmowa lub streamer itp.

Pełna kopia bezpieczeństwa off-line (zimna)

Celem jest uzyskanie pełnej i spójnej kopii bezpieczeństwa (consistent database). Baza odtworzona z takiej kopii może zostać bezproblemowo uruchomiona. Pliki danych mogą być również podstawą do odtworzenia bazy do punktu w czasie, lub wystąpienia awarii na podstawie informacji zapisanych w logach. Procedura składa się z trzech kroków:

1. Położenie bazy w trybie spójnym poleceniem **SHUTDOWN NORMAL, SHUTDOWN IMMEDIATE** lub **SHUTDOWN TRANSACTIONAL**.
2. Wykonanie kopii plików poleceniem systemu operacyjnego np. **cp**, **copy** itp.
3. Uruchomienie bazy poleceniem **STARTUP**

Kopia bezpieczeństwa przestrzeni plików w trybie on-line (gorąca)

Celem jest uzyskanie kopii plików wybranej przestrzeni plików bez zatrzymywania pracy bazy. Operację można wykonać tylko w trybie *archivelog* pracy bazy. Procedura składa się z następujących kroków:

1. Identyfikacja wszystkich plików wchodzących w skład przestrzeni tabel, której backup jest planowany. Informacja ta jest zawarta w perspektywie DBA_DATA_FILES
2. Przestawienie przestrzeni tabel w tryb backupu on-line komendą **ALTER TABLESPACE BEGIN BACKUP**
3. Skopiowanie plików wchodzących w skład przestrzeni tabel zgodnie z zawartością perspektywy DBA_DATA_FILES
4. Zakończenie procedury backupu komendą **ALTER TABLESPACE END BACKUP**

```
Przykład.

SQL> select tablespace_name, file_name from dba_data_files
  2    order by tablespace_name;

TABLESPACE_NAME         FILE_NAME
----------------------- ------------------------------
DANE                    /oracle/oradb/DANE.ORA
IND                     /oracle/oradb/IND.ORA
ROLL                    /oracle/oradb/ROLL.ORA
TEMP                    /oracle/oradb/TEMP.ORA
DATA                    /oracle/oradb/DATA.ORA
SYSTEM                  /oracle/oradb/SYSTEM.ORA
USER_DATA               /oracle/oradb/USER.ORA

7 wierszy zostało wybranych.

SQL> alter tablespace USER_DATA begin backup;

Przestrzeń tabel została zmieniona.

$cp /oracle/oradb/USER.ORA /backup

SQL> alter tablespace USER_DATA end backup;

Przestrzeń tabel została zmieniona.
```

W przykładzie wykonano kopię przestrzeni tabel USER_DATA w skład której wchodził jeden plik. Plik został skopiowany do katalogu */backup*.

Uwaga

W przypadku nieoczekiwanej awarii instancji bazy danych w trakcie wykonywania gorącego backupu, a więc gdy administrator nie zdążył wydać komendy **ALTER TABLESPACE END BACKUP**, po ponownym starcie bazy Oracle rozpozna kopiowane pliki jako wymagające odtworzenia. W takiej sytuacji należy wykonać komendę **ALTER DATABASE DATAFILE 'nazwa pliku' END BACKUP**, gdzie jako

nazwę pliku należy użyć pełnej ścieżki dostępu do pliku, który Został uznany przez Oracle za wymagający odtworzenia.

Informacje o bieżącym stanie plików bazy z tytułu wykonywania kopii bezpieczeństwa są zawarte w perspektywie V$BACKUP. Kolumny perspektywy:

- FILE# - identyfikator pliku
- STATUS – status pliku. Przyjmuje wartości:
  - NOT ACTIVE
  - ACTIVE – w trakcie backupu
  - OFFLINE NORMAL
  - Ewentualny status błędu
- CHANGE# - SCN na moment startu backupu
- TIME – czas startu backupu

```
Przykład.

SQL> select * from v$backup;

    FILE# STATUS               CHANGE# TIME
--------- ------------------ --------- --------
        1 NOT ACTIVE                 0
        2 ACTIVE             1,100E+12 02/02/01
        3 NOT ACTIVE                 0
        4 NOT ACTIVE                 0
        5 NOT ACTIVE                 0
        6 NOT ACTIVE                 0
        7 NOT ACTIVE                 0

7 wierszy zostało wybranych.
```

Z przykładu widać, że plik o numerze 2 jest w trybie gorącego backupu.

Kopia bezpieczeństwa przestrzeni tabel read-only

Celem jest uzyskanie kopii plików wybranej przestrzeni plików pracujące w trybie read-only bez zatrzymywania pracy bazy. Ponieważ przestrzeń tabel w trybie read-only nie podlega zmianom podczas pracy procedura składa się tylko z dwóch kroków:

1. Identyfikacja wszystkich plików wchodzących w skład przestrzeni tabel, której backup jest planowany. Informacja ta jest zawarta w perspektywie DBA_DATA_FILES
2. Skopiowanie plików wchodzących w skład przestrzeni tabel zgodnie z zawartością perspektywy DBA_DATA_FILES

Kopia bezpieczeństwa bazy w trybie wstrzymania (Suspend Mode)

Celem jest uzyskanie kopii bezpieczeństwa plików bazy w najkrótszym możliwym czasie zatrzymania operacji bazodanowych. Możliwe jest to jeśli baza danych jest położona na dyskach macierzy trzymającej synchroniczną kopię plików (mirror) i umożliwiającej wykonanie szybkiego rozpięcia kopii (split). Całą operację przeprowadza się zatrzymując na chwilę wszystkie operacje I/O na plikach bazy i wykonując w tym momencie split kopii. Oracle wspiera tę operację poprzez ustanowienie trybu wstrzymania na bazie danych. W trybie tym zatrzymane są wszystkie operacje I/O dotyczące:

- Nagłówków plików danych
- Bloków danych
- Pliku kontrolnego.

Tryb wstrzymania wykonuje się komendą **ALTER SYSTEM SUSPEND**. Po wydaniu komendy zostają dokończone już rozpoczęte operacje I/O, ale wszystkie następujące po niej próby dostępu do bazy są kolejkowane i wykonywane po odwołaniu trybu wstrzymania komendą **ALTER SYSTEM RESUME**. Ze względu na możliwe opóźnienie wynikające z przetwarzanych bieżąco operacji I/O Oracle rekomenduje połączenie trybu suspend z przestawieniem plików bazy danych w tryb gorącego backupu. Daje to gwarancję pomyślnego wykonania spójnego rozdzielenia luster macierzy.

```
Przykład.

SQL> alter system suspend;

System został zmieniony.
```

```
SQL> select INSTANCE_NAME,HOST_NAME,STARTUP_TIME,STATUS,DATABASE_STATUS "DB STAT"
  2   from v$instance;

INSTANCE_NAME    HOST_NAME  STARTUP_ STATUS  DB STAT
---------------- ---------- -------- ------- -----------------
ORCL             com0       02/02/01 OPEN    SUSPENDED

SQL> alter system resume;

System został zmieniony.

SQL> select INSTANCE_NAME,HOST_NAME,STARTUP_TIME,STATUS,DATABASE_STATUS "DB STAT"
  2   from v$instance;

INSTANCE_NAME    HOST_NAME  STARTUP_ STATUS  DB STAT
---------------- ---------- -------- ------- -----------------
ORCL             com0       02/02/01 OPEN    ACTIVE
```

W przykładzie stan bazy jest widoczny w kolumnie DATABASE_STATUS perspektywy systemowej V$INSTANCE.

Kopię w trybie wstrzymania wykonuje się w następujących krokach:

1. Identyfikacja wszystkich plików wchodzących w skład bazy, której backup jest planowany. Informacja ta jest zawarta w perspektywie DBA_DATA_FILES
2. Przestawienie wszystkich przestrzeni tabel w tryb backupu on-line komendą **ALTER TABLESPACE BEGIN BACKUP**
3. Przestawienie bazy w tryb wstrzymania komendą **ALTER SYSTEM SUSPEND** i weryfikacja stanu w perspektywie V$INSTANCE.
4. Wykonanie podziału luster macierzy zgodnie ze specyfikacją sprzętu
5. Odwołanie trybu wstrzymania komendą **ALTER SYSTEM RESUME** i weryfikacja stanu w perspektywie V$INSTANCE.
6. Zakończenie procedury backupu komendą **ALTER TABLESPACE END BACKUP**

Kopia plików przestrzeni tabel w trybie off-line

Celem jest wykonanie kopii wybranej przestrzeni tabel po przestawieniu jej w tryb off-line, bez zatrzymywania pracy bazy na pozostałych przestrzeniach tabel. Przestrzeń w trybie off-line jest niedostępna dla użytkowników. Kopię wykonuje się w następujących krokach:

1. Identyfikacja wszystkich plików wchodzących w skład przestrzeni tabel, której backup jest planowany. Informacja ta jest zawarta w perspektywie DBA_DATA_FILES
2. Przestawienie przestrzeni tabel w tryb off-line komendą **ALTER TABLESPACE OFFLINE NORMAL**
3. Skopiowanie plików wchodzących w skład przestrzeni tabel
4. Przestawienie przestrzeni tabel w tryb online komendą **ALTER TABLESPACE ONLINE**

7. Odtwarzanie bazy po awarii

7.1 Wprowadzenie do problematyki odtwarzania bazy

Definicja zagadnienia

W przypadku awarii instancji bazy Oracle, lub innego nienormalnego zakończenia pracy, stan plików bazy jest niespójny. Występują wtedy następujące zjawiska:

- Brak zgodności zapisanego w nagłówkach plików i pliku kontrolnym SCN
- Brak zapisanych w plikach zmodyfikowanych bloków danych
- Zapisane, częściowo wykonane ale nie zatwierdzone transakcje

Przy kolejnym podniesieniu bazy Oracle automatycznie wykonuje operację naprawy instancji (instance recovery). Naprawa odbywa się w dwóch krokach:

1. Odwinięcie w przód (rolling forward) – odtworzenie na podstawie dzienników powtórzeń (on-line redo logs) wszystkich zmian w bazie, a więc zarówno zatwierdzonych jak i niezatwierdzonych transakcji wykonanych po ostatnim punkcie kontrolnym
2. Odwinięcie w tył (rolling back) – wycofanie z bazy wszystkich niezatwierdzonych w momencie awarii transakcji.

W sytuacjach poważniejszych awarii procedura automatycznej naprawy instancji jest niewystarczająca i wymagane jest wykonanie przez administratora manualnej naprawy bazy najczęściej z użyciem kopii zapasowej plików bazy i archiwalnych plików dziennika powtórzeń. Proces nosi nazwę *media recovery* i jest opisany w niniejszym rozdziale.

Postępowanie w przypadku awarii

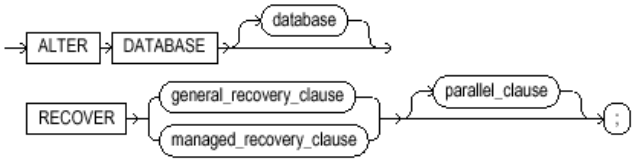
W sytuacji wystąpienia błędu uniemożliwiającego podniesienie bazy do obowiązków administratora należy podjęcie kroków umożliwiających możliwie najszybsze uruchomienie bazy. Przed podjęciem procedur odtwarzania bazy należy znaleźć przyczynę awarii. Jeżeli jest nią element sprzętowy należy rozpocząć od naprawy sprzętu, jeżeli systemowy – usunąć jego przyczynę. Następnie przystępuje się do odtwarzania bazy. Dobrą zasadą jest stosowanie się do poniższych wskazówek:

- Zidentyfikować problem – określić jaki element bazy uległ uszkodzeniu.
- Określić metodę naprawy – wybrać najlepszy sposób naprawy bazy. W sytuacji istnienia alternatywnych metod naprawy np. posiadanie kopii bezpieczeństwa na różnych nośnikach wybrać metodę najszybszą i najpewniejszą
- O ile jest to możliwe zabezpieczyć bazę uszkodzoną – jeżeli wielkość bazy na to pozwala dobrą praktyką jest skopiowanie bazy uszkodzonej na wypadek ewentualnych problemów z kopią bezpieczeństwa. W sytuacji, gdy operację odtwarzania trzeba powtórzyć, gdyż za pierwszym razem nie powiodła się, konieczne będą niektóre pliki z bazy uszkodzonej. Dobrą praktyką jest skopiowanie z bazy uszkodzonej pliku kontrolnego i plików dziennika powtórzeń.
- Podczas procedury odtwarzania prowadzić zapis wykonywanych czynności. W razie niepowodzenia łatwiej jest znaleźć błąd w stosowanej metodzie.

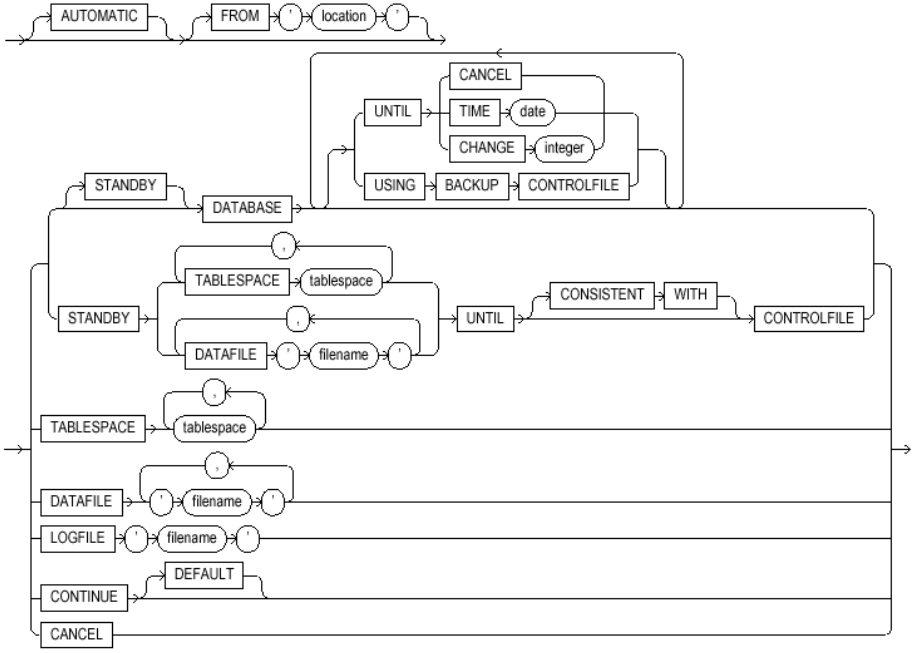
Należy pamiętać, że przy odtwarzaniu bazy wykonywane są czynności nieodwracalne np. nadpisywanie plików zbiorami archiwalnymi; dlatego też opracowane procedury muszą gwarantować eliminację błędów ludzkich, lub możliwość ich korekty. O ile awaria bazy jest zdarzeniem przewidywalnym z technicznego punktu widzenia, o tyle popełniony przez administratora na etapie naprawy błąd może skutkować całkowitą utratą bazy.

Komenda recover database

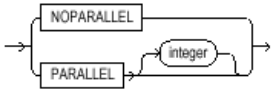
Odtworzenie bazy danych wykonuje się komendą **ALTER DATABASE RECOVER**, lub przyjmowaną przez **SQL*PLUS** i **server manager** komendą **RECOVER DATABASE**. Fraza **managed_recovery_clause** dotyczy bazy standby i nie będzie omawiana w tym rozdziale.



general_recovery_clause::=



parallel_clause::=



Warunki wykonania komendy

Odtworzenie całej bazy wymaga aby była on zamknięta, zaś instancja bazy podniesiona i zamontowana w trybie wyłącznym. Możliwe jest odtwarzanie poszczególnych przestrzeni tabel lub plików również gdy baza jest otwarta, ale wymaga to ustawienia na nich trybu offline. Proces odtwarzania musi być wykonywany poprzez serwer dedykowany, podłączenie do bazy poprzez serwer wielowątkowy wyklucza wykonanie tej operacji.

Znaczenie opcji

Poszczególne opcje komendy **ALTER DATABASE RECOVER** mają następujące znaczenie:

- AUTOMATIC – Oracle wykona automatyczne generowanie nazwy archiwalnego pliku dziennika powtórzeń potrzebnego do odtwarzania na podstawie zdefiniowanych w pliku inicjalizacyjnym parametrów LOG_ARCHIVE_DEST_n lub gdy nie jest on określony LOG_ARCHIVE_DEST oraz LOG_ARCHIVE_FORMAT. Proces będzie powtarzany, aż wszystkie potrzebne pliki dzienników powtórzeń zostaną przetworzone. W przypadku nie

znalezienia potrzebnego pliku, Oracle zgłosi zapytanie o wskazanie nazwy pliku przez operatora.

- FROM 'location' – wskazuje inną lokalizację archiwalnych plików dziennika powtórzeń niż zapisana w pliku inicjalizacyjnym parametrami LOG_ARCHIVE_DEST_n lub LOG_ARCHIVE_DEST.
- TABLESPACE – określa przestrzeń tabel dla której ma być wykonane odtwarzanie
- DATAFILE – określa plik bazy danych dla którego ma być wykonane odtwarzanie
- LOGFILE – wskazuje określony plik dziennika powtórzeń, który ma być użyty do odtwarzania
- CONTINUE DEFAULT – jest ekwiwalentem opcji AUTOMATIC, ale w odróżnieniu od niej nie wymaga potwierdzenia przez administratora nazwy archiwalnego pliku dziennika powtórzeń wygenerowanej przez Oracle.
- CANCEL – zatrzymuje proces odtwarzania prowadzony w trybie UNTIL CANCEL
- UNTIL – określa do jakiego momentu ma być prowadzone odtwarzanie. Możliwe są następujące warianty:
  - CANCEL – odtwarzanie jest prowadzone do chwili wydania komendy **ALTER DATABASE RECOVER CANCEL**
  - TIME date – odtwarzanie jest wykonywane do osiągnięcia punktu w czasie. Czas musi być podany w sztywnym formacie 'YYYY-MM-DD:HH24:MI:SS'
  - CHANGE integer – odtwarzanie jest prowadzone do punktu określonego przez SCN o podanym numerze. Baza zostanie odtworzona do pierwszej spójnej transakcja po podanym numerze SCN.
- USING BACKUP CONTROLFILE – wymusza użycie innego pliku kontrolnego niż bieżący. Stosuje się w przypadku utraty bieżącego pliku kontrolnego i konieczności odtworzenia go z kopii.
- NOPARALLEL – odtwarzanie szeregowe
- PARALLEL integer – przetwarzanie równoległe o podanym stopniu zrównoleglenia. Parametr może być stosowany na maszynach wieloprocesorowych w celu przyspieszenie procesu odtwarzania.

7.2 Odtwarzanie bazy

Zgodnie z zasadą minimalizacji czasu potrzebnego do usunięcia awarii, odtwarzanie z kopii zapasowej powinno dotyczyć wyłącznie plików wymagających naprawy. Perspektywa V$RECOVER_FILE zawiera informacje o plikach wymagających naprawy. Znaczenie kolumn perspektywy:

- FILE# - numer identyfikacyjny pliku
- ONLINE – status pliku (ONLINE, OFFLINE)
- ERROR – powód konieczności odtworzenia. Możliwe wartości:
  - NULL – nieznana przyczyna
  - OFFLINE NORMAL – odtwarzanie niepotrzebne
- CHANGE# - numer SCN od którego należy rozpocząć odtwarzanie
- TIME – czas SCN od którego należy rozpocząć odtwarzanie

Pełne odtwarzanie bazy do chwili awarii

Celem jest odtworzenie bazy danych, lub jej części (przestrzeń tabel, plik) do ostatniej zatwierdzonej transakcji przed awarią. Jest to najczęściej wykonywana operacja po awariach sprzętowych lub systemowych, kiedy uszkodzeniu ulegają pliki bazy danych. Odtwarzanie bazy należy wykonać w następujących krokach:

1. Odtworzyć uszkodzone pliki z kopii bezpieczeństwa
2. Zamontować instancję bazy komendą **startup mount**
3. Ustawić wszystkie pliki bazy w trybie on-line. Do odtwarzania wszystkie pliki muszą być w tym trybie, jeżeli określony plik jest w trybie off-line należy zmienić tryb komendą **ALTER DATABASE DATAFILE 'nazwa' online;**
4. Wykonać komendę **RECOVER DATABASE** lub **ALTER DATABASE RECOVER DATABASE**
5. W zależności od użytej opcji odtwarzania administrator będzie zmuszony akceptować lub podawać nazwy plików archiwalnych redo logów, lub dla opcji AUTOMATIC proces przebiegnie bez jego ingerencji. Odtwarzanie przebiegać będzie do momentu wykorzystania wszystkich niezbędnych archiwalnych plików dziennika powtórzeń.

6. Otworzyć bazę komendą **ALTER DATABASE OPEN;**

W kroku trzecim użyto komendy odtworzenia wszystkich uszkodzonych plików jedną operacją. Możliwe jest odtwarzanie poszczególnych plików lub przestrzeni tabel, w tym przypadku krok 3 może zostać zmodyfikowany i stosuje się wtedy odpowiednio komendy: **RECOVER DATAFILE**, lub **RECOVER TABLESPACE**.

Opisana powyżej procedura dotyczy odtwarzania zamkniętej bazy danych. Na czas wykonania tej operacji dostępu użytkowników do bazy jest niemożliwy. W określonych przypadkach gdy uszkodzeniu uległa część bazy np. jeden plik (z wyjątkiem plików wchodzących w skład systemowej przestrzeni tabel) możliwe jest odtwarzanie na bazie otwartej przy aktywnej pracy użytkowników w nieuszkodzonej części bazy. Operację przeprowadza się w następujących krokach:

1. Przestawić przestrzenie tabel zawierające uszkodzone pliki bazy w tryb off-line komendą **ALTER TABLESPACE 'nazwa' OFFLINE TEMPORARY**
2. Odtworzyć z kopii bezpieczeństwa uszkodzone pliki
3. Wykonać odtworzenie komendą **RECOVER TABLESPACE 'nazwa'**
4. W zależności od użytej opcji odtwarzania administrator będzie zmuszony akceptować lub podawać nazwy plików archiwalnych redo logów, lub dla opcji AUTOMATIC proces przebiegnie bez jego ingerencji. Odtwarzanie przebiegać będzie do momentu wykorzystania wszystkich niezbędnych archiwalnych plików dziennika powtórzeń.
5. Udostępnić naprawione przestrzenie tabel komendą: **ALTER TABLESPACE 'nazwa' ONLINE**

Częściowe odtwarzanie bazy

Celem jest odtworzenie częściowe bazy danych. Stosuje się je w określonych przypadkach np. gdy zostanie przypadkowo usunięta z bazy informacja. Jako przykład można podać przypadkowe usunięcie tabeli z bazy. Jest to operacja poprawna z punktu widzenia bazy, ale niszcząca dla aplikacji. W takim przypadku możliwa jest naprawa pomyłki w następujący sposób:

1. Odtworzenie bazy do chwili poprzedzającej usunięcie tabeli
2. Wyeksportowanie tabeli
3. Odtworzenie aktualnej bazy
4. Zaimportowanie tabeli

Wynikiem przeprowadzonych kroków jest posiadanie aktualnej bazy danych wraz z pomyłkowo usuniętą tabelą.

Częściowe odtwarzanie wykonuje się również w przypadku, gdy baza danych ulegnie uszkodzeniu wraz z częścią archiwalnych plików dziennika powtórzeń. W tym przypadku odzyskać można bazę do stany zapisanego w ostatnim prawidłowym pliku dziennika powtórzeń.

Częściowe odtwarzanie jest operacją wymagającą na wstępie przeanalizowanie stanu bazy oczekiwanego na moment dla którego baza jest odtwarzana. Głównym problemem jest tu stan fizycznej struktury plików bazy na ten moment. Należy doprowadzić do zgodności zapisu informacji o plikach bazy w pliku kontrolnym ze stanem jaki wystąpi w momencie zakończenia procesu odtwarzania. Niezgodność może wystąpić w sytuacji gdy do bazy został dodany nowy plik, którego nie było w chwili na którą ma być odtworzona baza. W takim wypadku należy posłużyć się zachowaną kopią pliku kontrolnego z tego momentu, a jeśli jest ona niedostępna utworzyć nowy plik kontrolny.

Powyższe okoliczności są jedną z przesłanek do wykonywania kopii pliku kontrolnego po każdej zmianie struktury plików bazy, o czym wspomniano wcześniej.

Odtwarzanie do odwołania

Odtwarzanie bazy należy wykonać w następujących krokach:

1. Odtworzyć uszkodzone pliki z kopii bezpieczeństwa. Należy odtworzyć wszystkie pliki, które wchodziły w skład bazy w momencie na który chcemy odtworzyć bazę. Jeżeli któryś z plików jest niedostępny, np. został utworzony po wykonaniu ostatniej dostępnej kopii bezpieczeństwa, należy wygenerować go jako plik pusty. Pliki, które zostały utworzone po momencie do którego odtwarzamy bazę nie muszą być odtwarzane z kopii bezpieczeństwa.
2. Zamontować instancję bazy komendą **startup mount**
3. Wykonać komendę **RECOVER DATABASE UNTIL CANCEL**, lub jeśli użyto kopii pliku kontrolnego komendę **RECOVER DATABASE UNTIL CANCEL USING BACKUP CONTROLFILE**

4. W zależności od użytej opcji odtwarzania administrator będzie zmuszony akceptować lub podawać nazwy plików archiwalnych redo logów, lub dla opcji AUTOMATIC proces przebiegnie bez jego ingerencji.
5. Zakończyć proces odtwarzania komendą CANCEL
6. Otworzyć bazę komendą **ALTER DATABASE OPEN RESETLOGS**

Należy zwrócić uwagę, że po częściowym odtwarzaniu bazy danych otwarcie jej jest możliwe wyłącznie w trybie RESETLOGS. Oznacza to przerwanie ciągłości dotychczas tworzonych archiwalnych plików dziennika powtórzeń i rozpoczęcie nowego ciągu kontynuowanego od momentu do którego baza została odtworzona. Nowa baza danych powstała w wyniku procesu częściowego odtwarzania nosi nazwę nowej *inkarnacji* (incarnation). W konsekwencji archiwalne pliki dziennika powtórzeń, wygenerowane dla starej inkarnacji bazy danych po punkcie częściowego odtworzenia, są niezgodne z nową inkarnacją bazy i nie mogą być już nigdy użyte.

Nowa inkarnacja bazy powinna być zatem traktowana jako punkt wyjścia dla procesu zabezpieczenia bazy i bezpośrednio po zakończeniu odtwarzania należy wykonać z niej pełną kopię bezpieczeństwa.

Odtwarzanie do punktu w czasie

Proces przeprowadza się analogicznie jak przy odtwarzaniu do odwołania, z tą różnicą, że zakończenie odtwarzania nastąpi automatycznie w momencie osiągnięcia przez bazę stanu na podany punkt czasu. Odtwarzanie bazy należy wykonać w następujących krokach:

1. Odtworzyć uszkodzone pliki z kopii bezpieczeństwa. Należy odtworzyć wszystkie pliki, które wchodziły w skład bazy w punkcie czasu na który chcemy odtworzyć bazę. Jeżeli któryś z plików jest niedostępny, np. został utworzony po wykonaniu ostatniej dostępnej kopii bezpieczeństwa, należy wygenerować go jako plik pusty. Pliki, które zostały utworzone po momencie do którego odtwarzamy bazę nie muszą być odtwarzane z kopii bezpieczeństwa.
2. Zamontować instancję bazy komendą **startup mount**
3. Wykonać komendę **RECOVER DATABASE UNTIL TIME 'YYYY-MM-DD:HH24:MI:SS'**, lub jeśli użyto kopii pliku kontrolnego komendę **RECOVER DATABASE UNTIL TIME 'YYYY-MM-DD:HH24:MI:SS' USING BACKUP CONTROLFILE**. Należy pamiętać o użyciu odpowiedniego formatu czasu zgodnie ze wzorcem podanym w tym punkcie.
4. W zależności od użytej opcji odtwarzania administrator będzie zmuszony akceptować lub podawać nazwy plików archiwalnych redo logów, lub dla opcji AUTOMATIC proces przebiegnie bez jego ingerencji.
5. Otworzyć bazę komendą **ALTER DATABASE OPEN RESETLOGS**

Odtwarzanie do wskazanego numeru SCN

Proces przeprowadza się analogicznie jak przy odtwarzaniu do punktu w czasie, z tą różnicą, że zakończenie odtwarzania nastąpi automatycznie w momencie osiągnięcia przez bazę stanu odpowiadającemu podanemu numerowi SCN. Odtwarzanie bazy należy wykonać w następujących krokach:

1. Odtworzyć uszkodzone pliki z kopii bezpieczeństwa. Należy odtworzyć wszystkie pliki, które wchodziły w skład bazy w momencie wykonania SCN o podanym numerze. Jeżeli któryś z plików jest niedostępny, np. został utworzony po wykonaniu ostatniej dostępnej kopii bezpieczeństwa, należy wygenerować go jako plik pusty. Pliki, które zostały utworzone po momencie do którego odtwarzamy bazę nie muszą być odtwarzane z kopii bezpieczeństwa.
2. Zamontować instancję bazy komendą **startup mount**
3. Wykonać komendę **RECOVER DATABASE UNTIL CHANGE nr_zmiany**, lub jeśli użyto kopii pliku kontrolnego komendę **RECOVER DATABASE UNTIL CHANGE nr_zmiany USING BACKUP CONTROLFILE**. Wartość numeru SCN podaje się w formacie liczby całkowitej.
4. W zależności od użytej opcji odtwarzania administrator będzie zmuszony akceptować lub podawać nazwy plików archiwalnych redo logów, lub dla opcji AUTOMATIC proces przebiegnie bez jego ingerencji.
5. Otworzyć bazę komendą **ALTER DATABASE OPEN RESETLOGS**

8. Możliwe uszkodzenia elementów bazy danych

W punkcie tym przedstawione zostaną skrótowo najczęściej występujące uszkodzenia bazy danych oraz sposoby jej naprawy.

W dokumentacji technicznej Oracle używane jest pojęcie uszkodzenia nośnika (media failure) oznaczające zjawisko powodujące przerwanie pracy bazy z powodów urządzeń na których jest ona postawiona. Wyróżnia się dwa rodzaje uszkodzeń:

- Trwałe (permanent) – oznaczające poważne i zwykle nieodwracalne uszkodzenie skutkujące zniszczeniem i w konsekwencji utratą danych. Typowym uszkodzeniem trwałym jest awaria dysku. Po usunięciu awarii np. wymianie dysku baza wymaga odtworzenia s kopii bezpieczeństwa.
- Tymczasowe (temporary) – oznaczające chwilowy brak dostępu do danych, ale nie powodujący ich zniszczenia. Typową awarią tego typu jest awaria zasilacza, lub kontrolera dyskowego. Po usunięciu awarii np. wymianie zasilacza baza może zostać podniesiona, zaś dane są ponownie dostępne. Nie ma potrzeby odtwarzania bazy z kopii zapasowej.

Poniżej opisane zostaną awarie typu trwałego, wymagające ingerencji administratora bazy w celu jej naprawy. Jak zostało wspomniane wcześniej możliwości naprawy bazy pracującej w trybie noarchive log są bardzo ograniczone, dlatego też poniższe rozważania dotyczyć będą bazy pracującej w trybie archive log.

8.1 Utrata plików

W przypadku awarii dysku następuje utrata plików na nim położonych. Zabezpieczeniem sprzętowym przed tego typu awarią są dyski lustrzane (mirror) lub macierze typu RAID 1, lub RAID 5, a więc zawierające dane zapisane redundantnie. Każde zabezpieczenie tego typu oparte jest na statystycznym prawdopodobieństwie awarii pojedynczego elementu opartego o przeciętny czas między awariami (MTTF – mean time to failure). W wyliczeń opartych na MTTF wynika, że prawdopodobieństwo jednoczesnego uszkodzenia dwóch dysków lustrzanych jest wielokrotnie niższe niż dysku pojedynczego, ale niezależnie od stopnia zwielokrotnienia luster prawdopodobieństwo to jest zawsze większe od zera. Oznacza to, że w każdym przypadku należy liczyć się z utratą plików i zabezpieczyć dodatkową kopią bezpieczeństwa.

Utrata pliku danych

Przypadek ten oznacza konieczność odtworzenia pliku z ostatniej kopii bezpieczeństwa i wykonanie pełnego odtwarzania bazy do chwili awarii. Jeżeli uszkodzeniu ulegnie plik wchodzący w skład systemowej przestrzeni tabel, lub plik zawierający aktywne segmenty wycofania jedynym dopuszczalnym wariantem jest wykonanie odtwarzania na zamkniętej bazie danych. Jeżeli uszkodzeniu ulegnie plik nie wchodzący w skład systemowej przestrzeni tabel, lub plik nie zawierający aktywnych segmentów wycofania, odtwarzanie można wykonać zarówno na zamkniętej jak i otwartej bazie danych. W ostatnim przypadku przestrzeń tabel należy przestawić w tryb off-line, tak jak opisano to w rozdziale „Pełne odtwarzanie bazy do chwili awarii"

Utrata pliku dziennika powtórzeń

Dobrym zabezpieczeniem plików dziennika powtórzeń jest użycie zwielokrotnionych plików w ramach każdej grupy, przy czym każdy element grupy musi być położony na osobnym dysku. W przypadku uszkodzenia plików na jednym z dysków pozostają prawidłowe, nieuszkodzone inne pliki tej grupy co umożliwia dalszą, nieprzerwaną pracę bazy. Informacja o uszkodzeniu plików dziennika powtórzeń zostaje przez Oracle zapisana do pliku *alrt.log* oraz do pliku *trace*. Czynnością naprawczą po wymianie uszkodzonego dysku jest odtworzenie pierwotnej konfiguracji logów poprzez usunięcie uszkodzonego logu komendą **ALTER DATABASE DROP LOGFILE MEMBER** i ponowne dodanie go do grupy komendą **ALTER DATABASE ADD LOGFILE MEMBER**. Uszkodzony plik ma status INVALID w perspektywie V$LOGFILE. Operacje te zostały opisane w rozdziale „Zarządzanie plikami dziennika powtórzeń"

Sytuacja komplikuje się zdecydowanie gdy nastąpi utrata całej grupy plików dziennika powtórzeń. Możliwość odtworzenia bazy jest wówczas zależna od stanu bazy (czas jaki upłynął od ostatniego punktu kontrolnego) oraz statusu utraconej grupy. Można wyróżnić trzy sytuacje w jakich znalazła się utracona grupa logów:

1. Grupa nie jest już potrzebna do odtwarzania bazy. Pliki zostały skopiowane do miejsca składowania plików archiwalnych. Grupa posiada status INACTIVE. Administrator powinien w tym przypadku wykonać czyszczenie logów komendą **ALTER DATABASE CLEAR LOGFILE**
2. Grupa jest już nieużywana do zapisu, ale ciągle potrzebna do odtwarzania instancji. Grupa posiada status ACTIVE. W tym przypadku należy podjąć próbę zapisu zmodyfikowanych bloków

danych do plików poprzez wymuszenie punktu kontrolnego, a następnie analogicznie jak w punkcie 1. wyczyścić grupę logów. Jeżeli operacja nie powiedzie się należy wykonać częściowe odtwarzanie bazy na podstawie kopii bezpieczeństwa używając ostatniego dostępnego logu.

3. Grupa jest aktualnie używana do zapisu zmian w bazie. Grupa posiada status CURRENT. Należy podjąć próbę wyczyszczenia logów tej grupy. Jeżeli operacja nie powiedzie się należy wykonać częściowe odtwarzanie bazy na podstawie kopii bezpieczeństwa używając ostatniego dostępnego logu.

Utrata archiwalnego pliku dziennika powtórzeń

Utrata archiwalnego pliku dziennika powtórzeń oznacza brak możliwości odtworzenia baz po awarii na podstawie wyjściowej kopii bezpieczeństwa i ciągu wygenerowanych plików logów. Jeżeli utrata pliku nastąpiła po jego wcześniejszym skopiowaniu na inny nośnik np. taśmę, administrator w dalszym ciągu dysponuje pełnym kompletem wymaganych logów. W przypadku potrzeby odtwarzania bazy należy jedynie przenieść na dysk potrzebne pliki archiwalnych logów. W przypadku całkowitej utraty archiwalnego logu należy natychmiast wykonać nową, pełną kopię bezpieczeństwa bazy, która od tego momentu będzie stanem wyjściowym do ewentualnego odtwarzania po awarii.

Utrata pliku kontrolnego

Podobnie jak pliki dzienników powtórzeń plik kontrolny powinien być zwieloktrotniony na różnych dyskach. W przypadku utraty jednego z nich należy, po położeniu bazy, po prostu skopiować poleceniem systemu operacyjnego np. cp plik zachowany (nieuszkodzony) do wszystkich wymaganych lokalizacji, a następnie podnieść bazę. W przypadku utraty wszystkich plików kontrolnych należy utworzyć nowy plik kontrolny poleceniem **CREATE CONTROLFILE** na podstawie ostatniej kopii bezpieczeństwa pliku kontrolnego, która powinna być wykonana po ostatniej zmianie strukturalnej bazy. W krańcowej sytuacji administrator musi podjąć próbę utworzenia nowego pliku ręcznie. Operacje te zostały opisane w rozdziale „Zarządzanie plikiem kontrolnym".

Jeżeli kopia bezpieczeństwa pliku kontrolnego była wykonywana poleceniem **ALTER DATABASE BACKUP CONTROLFILE TO file** tworzącym kopię binarną pliku należy najpierw przekształcić ten plik w skrypt tworzący nowy plik kontrolny. Wykonuje się to w następujących krokach:

- Przeniesienie kopii bezpieczeństwa pliku kontrolnego w miejsce oryginalnych plików kontrolnych
- Podniesienie instancji bazy (tryb nomount)
- Wygenerowanie skryptu w pliku trace poleceniem **ALTER DATABASE BACKUP CONTROLFILE TO TRACE**
- Uruchomienie skryptu z pliku *trace*

8.2 Uszkodzenia ukryte bazy

Poważnym zagrożeniem bezpieczeństwa bazy są uszkodzenia ukryte, ujawniające się nie w chwili jej uruchomienia, ale w momencie dostępu do niepoprawnych danych. Uszkodzenia te mogą nie być wykryte przez dłuższy czas, w konsekwencji czego nawet szereg kolejnych kopii bezpieczeństwa może zawierać uszkodzone pliki. Baza w takim stanie może nie nadawać się już do pełnej naprawy, jeśli uszkodzenie znajduje się w blokach zajmowanych przez wiersze tabel, zaś administrator nie dysponuje dobrą kopią bezpieczeństwa i kompletem wygenerowanych od chwili jej utworzenia logów archiwalnych. Nawet w tak złej sytuacji istnieje jednak możliwość podjęcia kroków naprawczych, dających szansę odzyskania przynajmniej części danych.

Uszkodzenia bloków danych

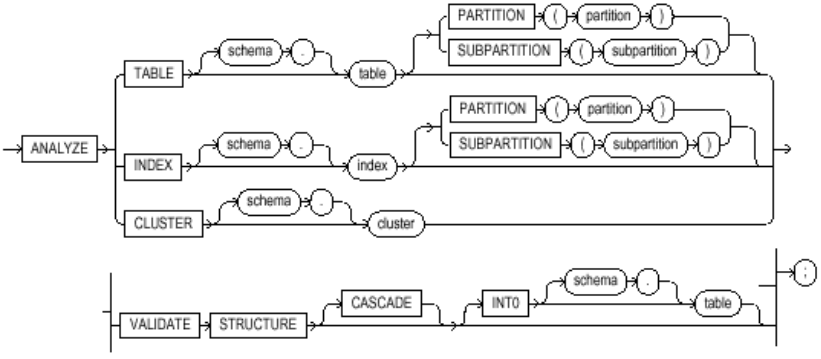
Typowymi błędami które mogą wystąpić w bazie są uszkodzenia bloków. Trudno jednoznacznie określić przyczynę pojawiania się uszkodzeń bloków; w części przypadków winę za to ponosi sprzęt, w części są to błędy software'owe. Co charakterystyczne, testy skanowania powierzchni dyskowej mogą dać wynik pozytywny, zaś uszkodzenia plików posadowionych na takim dysku mogą występować, a nawet mieć charakter powtarzalny. Baza zachowuje się poprawnie tak długo, jak długo nie występuje odwołanie do uszkodzonych bloków. Zapytanie w wyniku którego czytane są uszkodzone bloki zgłasza błąd.

```
Przykład.
W bazie znajduje się uszkodzona tabela TEST. Pierwsze bloki są poprawne.
```

```
SQL> select * from test where rownum<2;

COL1
----------
ABCDEFGHIJ

SQL> select count(*) from test;
select count(*) from test
                     *
BŁĄD w linii 1:
ORA-01578: blok danych ORACLE jest uszkodzony (plik nr. 8, blok nr. 5)
ORA-01110: plik danych 8: 'D:\ORACLE\ORADATA\ORACLE\TEST.ORA'

Zapis w pliku ALRT.LOG
***
Corrupt block relative dba: 0x02000005 (file 8, block 5)
Fractured block found during buffer read
Data in bad block -
 type: 6 format: 2 rdba: 0x02000005
 last change scn: 0x0000.00064957 seq: 0x1 flg: 0x02
 consistency value in tail: 0x58585858
 check value in block header: 0x0, block checksum disabled
 spare1: 0x0, spare2: 0x0, spare3: 0x0
***
Reread of rdba: 0x02000005 (file 8, block 5) found same corrupted data
```

Pierwsze zapytanie zwraca jeden wiersz, znajdujący się w poprawnym bloku. Drugie zapytanie wymusza czytanie całej tabeli, co jest niemożliwe ze względu na uszkodzenie w bloku nr 5. Poza zgłoszeniem błędu uszkodzenie bloku jest raportowane w pliku alertu.

Sprawdzenie poprawności tabeli można wykonać poleceniem **ANALYZE**.



Opcja VALIDATE STRUCTURE powoduje sprawdzenie wszystkich bloków i wierszy wskazanej tabeli, opcja CASCADE dodatkowo sprawdza poprawność wszystkich indeksów opartych o wskazaną tabele.

```
Przykład.

SQL> analyze table test validate structure;
analyze table test validate structure
*
BŁĄD w linii 1:
ORA-01578: blok danych ORACLE jest uszkodzony (plik nr. 8, blok nr. 5)
ORA-01110: plik danych 8: 'D:\ORACLE\ORADATA\ORACLE\TEST.ORA'
```

Polecenie **ANALYZE** jest poleceniem SQL i wymaga otwarcia bazy. Poza nim Oracle dostarcza szybszego narzędzia **DBVERIFY** do kontroli poprawności wewnętrznej wskazanego pliku. Jest to polecenie systemu operacyjnego.

Przykład.

```
D:\oracle\ora81\bin>dbv file=D:\ORACLE\ORADATA\ORACLE\TEST.ORA blocksize=8192

DBVERIFY: Release 8.1.7.0.0 - Production on śro Mar 6 22:14:43 2002

(c) Copyright 2000 Oracle Corporation.  All rights reserved.

DBVERIFY - Początek weryfikacji : FILE = D:\ORACLE\ORADATA\ORACLE\TEST.ORA
Strona 5 jest influx - prawdopodobnie uszkodzony nośnik
***
Corrupt block relative dba: 0x02000005 (file 0, block 5)
Fractured block found during dbv:
Data in bad block -
 type: 6 format: 2 rdba: 0x02000005
 last change scn: 0x0000.00064957 seq: 0x1 flg: 0x02
 consistency value in tail: 0x58585858
 check value in block header: 0x0, block checksum disabled
 spare1: 0x0, spare2: 0x0, spare3: 0x0
***

Strona 6 jest zaznaczona jako uszkodzona
***
Corrupt block relative dba: 0x02000006 (file 0, block 6)
Bad header found during dbv:
Data in bad block -
 type: 88 format: 88 rdba: 0x58585858
 last change scn: 0x5858.58585858 seq: 0x58 flg: 0x58
 consistency value in tail: 0x47230603
 check value in block header: 0x5858, block checksum disabled
 spare1: 0x58, spare2: 0x58, spare3: 0x5858
***



DBVERIFY - Weryfikacja zakończona

Liczba stron sprawdzonych                : 8
Liczba stron przetworzonych (Dane)       : 4
Liczba stron błędnych       (Dane)       : 0
Liczba stron przetworzonych (Indeks)     : 0
Liczba stron błędnych       (Indeks)     : 0
Całkowita liczba przetworzonych stron (Inne): 2
Liczba stron pustych                     : 0
Liczba stron zaznaczonych jako uszkodzone: 2
Liczba stron napływowych (Influx)        : 1

D:\oracle\ora81\bin>
```

Polecenie **DBV** podaje dodatkowo statystykę wykonanej analizy pliku.

O ile warunki na to pozwalają wskazane jest sprawdzenie poleceniem **DBV** wszystkich plików zawsze po wykonaniu kopii bazy na taśmę. Procedura ta upewnia administratora, że zabezpieczona kopia bazy zawiera poprawną fizyczną strukturę plików.

Identyfikacja uszkodzonej tabeli

W przypadku zapytań do bazy będących złączeniem kilku tabel nie można bezpośrednio określić, której tabeli dotyczy błąd. Jak pokazano w przykładzie Oracle raportuje błąd podając numer pliku i numer bloku. Na podstawie tych informacji można znaleźć tabelę przy pomocy perspektywy systemowej DBA_EXTENTS.

Kolumny perspektywy DBA_EXTENTS zawierające potrzebną informację:

- SEGMENT_NAME – nazwa segmentu zawierającego poszukiwany obszar
- SEGMENT_TYPE – typ segmentu
- TABLESPACE_NAME – przestrzeń tabel zawierająca poszukiwany obszar
- FILE_ID – numer pliku zawierający poszukiwany obszar
- BLOCK_ID – numer pierwszego bloku obszaru
- BLOCKS – liczba bloków wchodzących w skład obszaru

Przykład.

```
SQL> select OWNER, SEGMENT_NAME, SEGMENT_TYPE, TABLESPACE_NAME,
  2         BLOCK_ID, BLOCKS
  3   from dba_extents
  4  where FILE_ID=8 and
  5        5 between BLOCK_ID and BLOCK_ID + BLOCKS-1;

OWNER           SEGMENT_NAME         SEGMENT_TY TABLESPACE_NAME        BLOCK_ID     BLOCKS
--------------- -------------------- ---------- -------------------- ---------- ----------
SYSTEM          TEST                 TABLE      TEST                          4          5

SQL>
```

Zapytanie w przykładzie zidentyfikowało segment typu tabela o nazwie TEST, który zawiera uszkodzony blok nr 5 w pliku nr 8.

Użycie pakietu DBMS_REPAIR

W przypadku uszkodzenia bloków danych pomocnym narzędziem jest dostarczony przez Oracle pakiet DBMS_REPAIR. Umożliwia on znalezienie uszkodzeń, oraz pomaga w naprawie tabeli. Nie gwarantuje odzyskania danych z uszkodzonych bloków, co jest ewidentnie niemożliwe, ale umożliwiając dostęp do nieuszkodzonych bloków daje możliwość dostępu do wszystkich poprawnych wierszy w tabeli. W sytuacji gdy brak innej możliwości odtworzenia bazy, częściowe odzyskanie danych z tabeli może okazać się jedynym sposobem na ocalenie przynajmniej części danych.

Pakiet zawiera następujące procedury:

- ADMIN_TABLES – zarządza techniczną tabelą używaną do zapisu danych o uszkodzonych blokach lub wskazaniach indeksów na uszkodzone wiersze

```
DBMS_REPAIR.ADMIN_TABLES (
     table_name IN VARCHAR2,
     table_type IN BINARY_INTEGER,
     action IN BINARY_INTEGER,
     tablespace IN VARCHAR2 DEFAULT NULL);
```

Parametry procedury:

  - TABLE_NAME – nazwa tabeli technicznej. W zależności od typu tabeli nazwa musi zawierać prefix ORPHAN_, lub REPAIR_
  - TABLE_TYPE – typ tabeli, dopuszczalne wartości: ORPHAN_TABLE, REPAIR_TABLE
  - ACTION – określenie czynności na tabeli. Dopuszczalne wartości: CREATE_ACTION, PURGE_ACTION, DROP_ACTION
  - TABLESPACE – przestrzeń tabel w której wygenerowana będzie tabela. Domyślnie używana jest systemowa przestrzeń tabel

- CHECK_OBJECT – sprawdza określony obiekt i zapisuje dane o uszkodzeniach w tabeli technicznej

```
DBMS_REPAIR.CHECK_OBJECT (
     schema_name IN VARCHAR2,
     object_name IN VARCHAR2,
     partition_name IN VARCHAR2 DEFAULT NULL,
     object_type IN BINARY_INTEGER DEFAULT TABLE_OBJECT,
     repair_table_name IN VARCHAR2 DEFAULT 'REPAIR_TABLE',
     flags IN BINARY_INTEGER DEFAULT NULL,
     relative_fno IN BINARY_INTEGER DEFAULT NULL,
     block_start IN BINARY_INTEGER DEFAULT NULL,
     block_end IN BINARY_INTEGER DEFAULT NULL,
     corrupt_count OUT BINARY_INTEGER);
```

Parametry procedury:

  - SCHEMA_NAME – nazwa schematu zawierającego sprawdzany obiekt
  - OBJECT_NAME – nazwa sprawdzanego obiektu
  - PARTITION_NAME – nazwa sprawdzanej partycji, NULL – oznacza sprawdzenie całego obiektu
  - OBJECT_TYPE – typ sprawdzanego obiektu. Dopuszczalne wartości: TABLE_OBJECT, INDEX_OBJECT
  - REPAIR_TABLE_NAME – nazwa tabeli technicznej
  - FLAGS – nieużywany

- RELATIVE_FNO – względny numer pliku, wymagany gdy określa się zakres sprawdzanych bloków
- BLOCK_START – pierwszy blok do sprawdzenia, wymagany gdy określa się zakres sprawdzanych bloków
- BLOCK_END – ostatni blok do sprawdzenia, wymagany gdy określa się zakres sprawdzanych bloków
- CORRUPT_COUNT – liczba wykrytych uszkodzeń

- DUMP_ORPHAN_KEYS – sprawdza indeksy i zapisuje w tabeli technicznej pozycje indeksów, które wskazują na wiersze w uszkodzonych blokach

```
DBMS_REPAIR.DUMP_ORPHAN_KEYS (
     schema_name IN VARCHAR2,
     object_name IN VARCHAR2,
     partition_name IN VARCHAR2 DEFAULT NULL,
     object_type IN BINARY_INTEGER DEFAULT INDEX_OBJECT,
     repair_table_name IN VARCHAR2 DEFAULT 'REPAIR_TABLE',
     orphan_table_name IN VARCHAR2 DEFAULT 'ORPHAN_KEYS_TABLE',
     flags IN BINARY_INTEGER DEFAULT NULL,
     key_count OUT BINARY_INTEGER);
```

Parametry procedury:

- SCHEMA_NAME – nazwa schematu zawierającego sprawdzany obiekt
- OBJECT_NAME – nazwa sprawdzanego obiektu
- PARTITION_NAME – nazwa sprawdzanej partycji, NULL – oznacza sprawdzenie całego obiektu
- OBJECT_TYPE – typ sprawdzanego obiektu. Dopuszczalne wartości: TABLE_OBJECT, INDEX_OBJECT
- REPAIR_TABLE_NAME – nazwa tabeli technicznej zawierającej informacje o uszkodzonych blokach
- ORPHAN_TABLE_NAME - nazwa tabeli technicznej zawierającej informacje o pozycjach indeksów wskazujących na wiersze w uszkodzonych blokach
- FLAGS – nieużywany
- KEY_COUNT – liczba przetworzonych pozycji indeksów

- FIX_CORRUPT_BLOCKS – naprawia uszkodzone bloki poprzez zaznaczenie ich jako uszkodzone software'owo. Przed uruchomieniem tej procedury należy wykonać procedurę CHECK_OBJECT.

```
DBMS_REPAIR.FIX_CORRUPT_BLOCKS (
     schema_name IN VARCHAR2,
     object_name IN VARCHAR2,
     partition_name IN VARCHAR2 DEFAULT NULL,
     object_type IN BINARY_INTEGER DEFAULT TABLE_OBJECT,
     repair_table_name IN VARCHAR2 DEFAULT 'REPAIR_TABLE',
     flags IN BINARY_INTEGER DEFAULT NULL,
     fix_count OUT BINARY_INTEGER);
```

Parametry procedury:

- SCHEMA_NAME – nazwa schematu zawierającego naprawiany obiekt
- OBJECT_NAME – nazwa naprawianego obiektu
- PARTITION_NAME – nazwa naprawianej partycji, NULL – oznacza naprawienie całego obiektu
- OBJECT_TYPE – typ sprawdzanego obiektu. Dopuszczalne wartości: TABLE_OBJECT, INDEX_OBJECT
- REPAIR_TABLE_NAME – nazwa tabeli technicznej zawierającej informacje o uszkodzonych blokach
- FLAGS – nieużywany
- FIX_COUNT – liczba naprawionych bloków

- REBUILD_FREELISTS – przebudowuje listy wolnych bloków, umieszczając wszystkie wolne bloki na głównej liście i zerując wszystkie inne listy.

```
DBMS_REPAIR.REBUILD_FREELISTS (
     schema_name IN VARCHAR2,
     object_name IN VARCHAR2,
     partition_name IN VARCHAR2 DEFAULT NULL,
```

```
      object_type IN BINARY_INTEGER DEFAULT TABLE_OBJECT);
```

Parametry procedury:

- SCHEMA_NAME – nazwa schematu zawierającego naprawiany obiekt
- OBJECT_NAME – nazwa naprawianego obiektu
- PARTITION_NAME – nazwa naprawianej partycji, NULL – oznacza naprawienie całego obiektu
- OBJECT_TYPE – typ sprawdzanego obiektu. Dopuszczalne wartości: TABLE_OBJECT, INDEX_OBJECT

- SKIP_CORRUPT_BLOCKS – włącza lub wyłącza przeskakiwanie uszkodzonych bloków podczas przeszukiwania tabeli lub indeksu. Jeżeli przeskakiwanie uszkodzonych bloków jest wyłączone Oracle zgłasza przy dostępie do uszkodzonych bloków błąd ORA-1578

```
DBMS_REPAIR.SKIP_CORRUPT_BLOCKS (
     schema_name IN VARCHAR2,
     object_name IN VARCHAR2,
     object_type IN BINARY_INTEGER DEFAULT TABLE_OBJECT,
     flags IN BINARY_INTEGER DEFAULT SKIP_FLAG);
```

Parametry procedury:

- SCHEMA_NAME – nazwa schematu zawierającego naprawiany obiekt
- OBJECT_NAME – nazwa naprawianego obiektu
- OBJECT_TYPE – typ sprawdzanego obiektu. Dopuszczalne wartości: TABLE_OBJECT, CLUSTER_OBJECT
- FLAGS – specyfikacja działania procedury. Dopuszczalne wartości: SKIP_FLAG, NOSKIP_FLAG

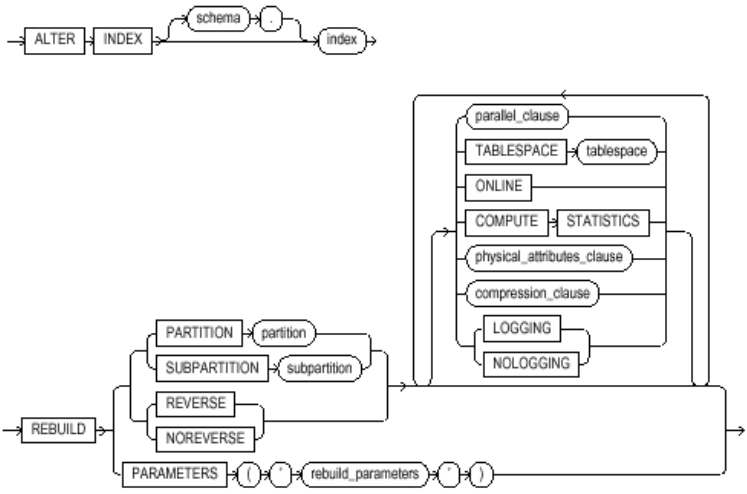
Użycie pakietu umożliwia szczegółową analizę uszkodzeń w bazie. Umożliwia również odczytanie wszystkich nieuszkodzonych wierszy po zaznaczeniu uszkodzonych bloków i wykonaniu procedury SKIP_CORRUP_BLOCKS z flagą SKIP_FLAG. Pakiet może być uruchamiany wyłącznie przez użytkownika SYS. Informacje o uszkodzeniach w poszczególnych blokach są zapisywane w kolejnych wierszach tabeli technicznej o nazwie domyślnej REPAIR_TABLE. Informacje o pozycjach indeksów, które wskazują na wiersze w uszkodzonych blokach są zapisywane w wierszach tabeli technicznej o nazwie domyślnej ORPHAN_KEY_TABLE.

```
Przykład.

SQL> desc repair_table
 Nazwa                                     Wartość NULL? Typ
 ----------------------------------------- -------- --------------
 OBJECT_ID                                 NOT NULL NUMBER
 TABLESPACE_ID                             NOT NULL NUMBER
 RELATIVE_FILE_ID                          NOT NULL NUMBER
 BLOCK_ID                                  NOT NULL NUMBER
 CORRUPT_TYPE                              NOT NULL NUMBER
 SCHEMA_NAME                               NOT NULL VARCHAR2(30)
 OBJECT_NAME                               NOT NULL VARCHAR2(30)
 BASEOBJECT_NAME                                    VARCHAR2(30)
 PARTITION_NAME                                     VARCHAR2(30)
 CORRUPT_DESCRIPTION                                VARCHAR2(2000)
 REPAIR_DESCRIPTION                                 VARCHAR2(200)
 MARKED_CORRUPT                            NOT NULL VARCHAR2(10)
 CHECK_TIMESTAMP                           NOT NULL DATE
 FIX_TIMESTAMP                                      DATE
 REFORMAT_TIMESTAMP                                 DATE

SQL> desc orphan_key_table
 Nazwa                                              Wartość null? Typ
 -------------------------------------------------- -------- ------------------
 SCHEMA_NAME                                        NOT NULL VARCHAR2(30)
 INDEX_NAME                                         NOT NULL VARCHAR2(30)
 IPART_NAME                                                  VARCHAR2(30)
 INDEX_ID                                           NOT NULL NUMBER
 TABLE_NAME                                         NOT NULL VARCHAR2(30)
 PART_NAME                                                   VARCHAR2(30)
 TABLE_ID                                           NOT NULL NUMBER
 KEYROWID                                           NOT NULL ROWID
 KEY                                                NOT NULL ROWID
 DUMP_TIMESTAMP                                     NOT NULL DATE
```

Uszkodzenia indeksów

Jeżeli uszkodzony blok zawiera indeks możliwe jest jego powtórne utworzenie na podstawie danych źródłowych zawartych w tabeli. Sprawdzenie poprawności indeksu można wykonać poleceniem **ANALYZE INDEX VALIDATE STRUCTURE**, albo przy okazji kontroli tabeli poleceniem **ANALYZE TABLE VALIDATE STRUCTURE CASCADE**.

Odtworzenie indeksu wykonuje się komendą **ALTER INDEX REBUILD**.



Jak widać składnia komendy jest bardzo rozbudowana i umożliwia wykonanie dodatkowych, innych operacji podczas odtwarzania indeksu. Przydatną operacją jest odtworzenie indeksu w innej przestrzeni tabel, stosowana często do przeniesienia indeksu pomyłkowo utworzonego w przestrzeni tabel dla niego nie przeznaczonej. Służy do tego opcja TABLESPACE.

```
Przykład.

SQL> select index_name, tablespace_name from dba_indexes
  2    where index_name = 'TEST_IDX';

INDEX_NAME                     TABLESPACE_NAME
------------------------------ ----------------
TEST_IDX                       TEST

SQL> alter index test_idx rebuild tablespace INDX;

Indeks został zmieniony.

SQL> select index_name, tablespace_name from dba_indexes
  2    where index_name = 'TEST_IDX';

INDEX_NAME                     TABLESPACE_NAME
------------------------------ ----------------
TEST_IDX                       INDX
```

W przykładzie przeniesiono indeks znajdujący się pierwotnie w przestrzeni tabel TEST do przewidzianej dla indeksów przestrzeni tabel INDX.

9. Zarządzanie przestrzeniami tabel i plikami danych

W rozdziale „Tworzenie bazy danych" wspomniano o zakładaniu przestrzeni tabel w chwili tworzenia nowej bazy danych. W bieżącym rozdziale omówione zostaną główne czynności administratorskie wykonywane na przestrzeniach tabel istniejącej bazy danych. Najczęściej wykonywanymi czynnościami w tym obszarze są:

- Kontrola wolnego miejsca w pliku
- Zakładanie nowej przestrzeni tabel
- Zmiana rozmiaru alokowanego pliku na dysku
- Zmiana parametrów przestrzeni tabel
- Dodawanie nowego pliku do przestrzeni tabel
- Przenoszenie pliku do innej lokalizacji
- Zmiana statusu przestrzeni – online/offline
- Usuwanie przestrzeni tabel

9.1 Obserwacja przestrzeni tabel

Przestrzenie tabel są strukturami obsługującymi dynamicznie zmieniające się dane, w szczególności krytyczne znaczenie ma przyrost używanych bloków w segmentach danych i indeksów. Dlatego ważna jest bieżąca obserwacja wypełnienia plików wchodzących w skład przestrzeni tabel oraz wielkość fragmentacji danych. Informacje dostępne są w perspektywach systemowych. Do najważniejszych z nich należą:

- DBA_TABLESPACES – zawiera informacje o parametrach wszystkich przestrzeni tabel wchodzących w skład bazy. Znaczenie kolumn
  - TABLESPACE_NAME – nazwa przestrzeni tabel
  - INITIAL_EXTENT – domyślna wartość rozmiaru pierwszego (inicjacyjnego) obszaru
  - NEXT_EXTENT – domyślna wartość rozmiaru drugiego obszaru
  - MIN_EXTENTS – domyślna wartość minimalnej liczby obszarów
  - MAX_EXTENTS – domyślna wartość minimalnej liczby obszarów
  - PCT_INCREASE – domyślna wartość procentowego przyrostu następnych alokowanych obszarów
  - MIN_EXTLEN – minimalna wielkość obszaru przestrzeni tabel
  - STATUS – status przestrzeni tabel, przyjmuje wartości: ONLINE, OFFLINE, READ ONLY
  - CONTENTS – typ przechowywanych danych, przyjmuje wartości: PERMANENT, TEMPORARY
  - LOGGING – domyślna wartość parametru logging dla tabel ładowanych ścieżką bezpośrednią (direct-load insert)
  - EXTENT_MANAGEMENT – sposób zarządzania obszarami: DICTIONARY, LOCAL
  - ALLOCATION_TYPE – typ alokacji obszarów. Przyjmuje wartości zależne od sposobu zarządzania obszarami np. USER, SYSTEM, UNIFORM
  - PLUGGED_IN – wskaźnik włączenia transportowalnych przestrzeni tabel. Przyjmuje wartości: YES, NO
- V$TABLESPACE - zawiera informacje o numerze i nazwie przestrzeni tabel zapisane w pliku kontrolnym. Znaczenie kolumn:
  - TS# - numer przestrzeni tabel
  - NAME – nazwa przestrzeni tabel
- DBA_DATA_FILES – zawiera informacje o parametrach plików wchodzących w skład bazy. Znaczenie kolumn
  - FILE_NAME – nazwa pliku
  - FILE_ID – identyfikator pliku
  - TABLESPACE_NAME – nazwa przestrzeni tabel do której należy plik
  - BYTES – rozmiar pliku w bajtach
  - BLOCKS – rozmiar pliku w blokach (bloki bazy danych Oracle)
  - STATUS – status pliku. Przyjmuje wartości: AVAILABLE, INVALID (plik o tym numerze nie jest używany, np. został usunięty)
  - RELATIVE_FNO – względny numer pliku
  - AUTOEXTENSIBLE – flaga wskazująca czy plik jest samorozszerzający się. Przyjmuje wartości: YES, NO

  - MAXBYTES – maksymalny rozmiar pliku w bajtach
  - MAXBLOCKS – maksymalny rozmiar pliku w blokach
  - INCREMENT_BY – wartość kroku przyrostu pliku w czasie automatycznego rozszerzania
  - USER_BYTES – liczba bajtów, które mogą być użyte na dane
  - USER_BLOCKS – liczba bloków, które mogą być użyte na dane
- V$DATAFILE – zawiera informacje o bieżącym stanie plików odczytane z pliku kontrolnego. Znaczenie kolumn:
  - FILE# - numer pliku
  - CREATION_CHANGE# - numer zmiany przy której plik został utworzony
  - CREATION_TIME – czas utworzenia pliku
  - TS# - numer przestrzeni tabel
  - RFILE# - względny numer pliku w przestrzeni tabel
  - STATUS – status pliku. Przyjmuje wartości: ONLINE, OFFLINE, SYSTEM, RECOVER, SYSOFF (oznacza plik w stanie offline należący do systemowej przestrzeni tabel)
  - ENABLED – określa dostęp z poziomu języka SQL. Interpretacja wartości kolumny:
    - DISABLED – brak dostępu
    - READ ONLY – dopuszczalne wyłącznie odczyty
    - READ WRITE – pełen dostęp
    - UNKNOWN – wartość wyświetlana w przypadku uszkodzenia pliku kontrolnego
  - CHECKPOINT_CHANGE# - numer SCN dla ostatnio wykonanego punktu kontrolnego
  - CHECKPOINT_TIME – czas ostatnio wykonanego punktu kontrolnego
  - UNRECOVERABLE_CHANGE# - numer ostatniej nieodtwarzalnej zmiany w pliku
  - UNRECOVERABLE_TIME – czas ostatniej nieodtwarzalnej zmiany w pliku
  - LAST_CHANGE# - numer ostatniej zmiany wykonanej w pliku. Wartość ustawiana na NULL gdy plik jest zmieniany
  - LAST_TIME – czas ostatniej zmiany wykonanej w pliku
  - OFFLINE_CHANGE# - numer zmiany ostatniego przełączenia w tryb offline
  - ONLINE_CHANGE# - numer zmiany ostatniego przełączenia w tryb online
  - ONLINE_TIME – czas ostatniego przełączenia w tryb online
  - BYTES – rozmiar pliku w bajtach
  - BLOCKS – rozmiar pliku w blokach
  - CREATE_BYTES – rozmiar pliku w chwili jego utworzenia
  - BLOCK_SIZE – rozmiar bloku w pliku
  - NAME – nazwa pliku
  - PLUGGED_IN – wskaźnik włączenia. Przyjmuje wartości: 1 - gdy przestrzeń tabel jest włączona, 0 – gdy nie jest włączona.
- DBA_FREE_SPACE – zawiera informacje o wolnych obszarach w przestrzeniach tabel. Znaczenie kolumn:
  - TABLESPACE_NAME – nazwa przestrzeni tabel zawierającej obszar
  - FILE_ID – numer identyfikacyjny pliku zawierającego obszar
  - BLOCK_ID – początkowy numer bloku wchodzącego w skład obszaru
  - BYTES – rozmiar obszaru w bajtach
  - BLOCKS – rozmiar obszaru w blokach
  - RELATIVE_FNO – względny numer pliku zawierającego pierwszy blok obszaru

Przykład.

```
SQL> select TABLESPACE_NAME, STATUS, CONTENTS, LOGGING,
  2         EXTENT_MANAGEMENT, ALLOCATION_TYPE, PLUGGED_IN
  3   from dba_tablespaces
  4   order by TABLESPACE_NAME;

TABLESPACE_NAME STATUS    CONTENTS  LOGGING   EXTENT_MAN ALLOCATIO PLU
--------------- --------- --------- --------- ---------- --------- ---
DRSYS           ONLINE    PERMANENT LOGGING   DICTIONARY USER      NO
INDX            ONLINE    PERMANENT LOGGING   DICTIONARY USER      NO
RBS             ONLINE    PERMANENT LOGGING   DICTIONARY USER      NO
SYSTEM          ONLINE    PERMANENT LOGGING   DICTIONARY USER      NO
TEMP            ONLINE    TEMPORARY LOGGING   DICTIONARY USER      NO
TEST            ONLINE    PERMANENT LOGGING   LOCAL      UNIFORM   NO
```

```
TOOLS           ONLINE    PERMANENT LOGGING   DICTIONARY USER      NO
USERS           ONLINE    PERMANENT LOGGING   DICTIONARY USER      NO

SQL> select TABLESPACE_NAME, INITIAL_EXTENT "INIT_E",
  2         NEXT_EXTENT "NEXT_E", MIN_EXTENTS "MIN_E",
  3         MAX_EXTENTS, PCT_INCREASE, MIN_EXTLEN
  4   from dba_tablespaces
  5   order by TABLESPACE_NAME;

TABLESPACE_NAME     INIT_E     NEXT_E      MIN_E MAX_EXTENTS PCT_INCREASE MIN_EXTLEN
--------------- ---------- ---------- ---------- ----------- ------------ ----------
DRSYS                65536      65536          1  2147483645           50      65536
INDX                131072     131072          1        4096            0     131072
RBS                 524288     524288          8        4096           50     524288
SYSTEM               65536      65536          1  2147483645           50      65536
TEMP                 65536      65536          1                       0      65536
TEST                 16384      16384          1  2147483645            0      16384
TOOLS                32768      32768          1        4096            0      32768
USERS               131072     131072          1        4096            0     131072
```

W przykładzie pobrano z bazy informacje o istniejących przestrzeniach tabel. Należy pamiętać, że wartości dotyczące obszarów np. INITIAL_EXTENT są używane do alokacji obszarów tworzonych w przestrzeni tabel obiektów dla których wartości te nie zostały jawnie określone.

```
Przykład.

SQL> select FILE_NAME, FILE_ID, TABLESPACE_NAME "TBS", BYTES, STATUS
  2   from dba_data_files
  3   order by TABLESPACE_NAME;

FILE_NAME                                FILE_ID TBS             BYTES STATUS
---------------------------------------- ---------- ---------- ---------- ----------
D:\ORACLE\ORADATA\ORACLE\DR01.DBF              7 DRSYS        20971520 AVAILABLE
D:\ORACLE\ORADATA\ORACLE\INDX01.DBF            6 INDX         20971520 AVAILABLE
D:\ORACLE\ORADATA\ORACLE\RBS01.DBF             2 RBS          52428800 AVAILABLE
D:\ORACLE\ORADATA\ORACLE\SYSTEM01.DBF          1 SYSTEM      287309824 AVAILABLE
D:\ORACLE\ORADATA\ORACLE\TEMP01.DBF            4 TEMP         20971520 AVAILABLE
D:\ORACLE\ORADATA\ORACLE\TEST.ORA              8 TEST           131072 AVAILABLE
D:\ORACLE\ORADATA\ORACLE\TOOLS01.DBF           5 TOOLS        10485760 AVAILABLE
D:\ORACLE\ORADATA\ORACLE\USERS01.DBF           3 USERS        20971520 AVAILABLE


SQL> select FILE_ID, TABLESPACE_NAME "TBS", BYTES,
  2         AUTOEXTENSIBLE, MAXBYTES, MAXBLOCKS, INCREMENT_BY,
  3         USER_BYTES
  4   from dba_data_files
  5   order by TABLESPACE_NAME;

   FILE_ID TBS             BYTES AUT   MAXBYTES  MAXBLOCKS INCREMENT_BY USER_BYTES
---------- ---------- ---------- --- ---------- ---------- ------------ ----------
         7 DRSYS        20971520 YES 3,4360E+10    4194302           80   20963328
         6 INDX         20971520 YES 3,4360E+10    4194302          160   20963328
         2 RBS          52428800 YES 3,4360E+10    4194302          640   52420608
         1 SYSTEM      287309824 YES 3,4360E+10    4194302         1280  287301632
         4 TEMP         20971520 YES 3,4360E+10    4194302           80   20963328
         8 TEST           131072 NO           0          0            0      65536
         5 TOOLS        10485760 YES 3,4360E+10    4194302           40   10477568
         3 USERS        20971520 YES 3,4360E+10    4194302          160   20963328
```

W przykładzie pobrano z bazy informacje o plikach tworzących istniejące przestrzenie tabel. Należy zwrócić uwagę na parametry plików typu autoincrement, dla których określono maksymalnie możliwy rozmiar w systemie operacyjnym (w przykładzie pliki były posadowione na Windows 2000). Ilość miejsca przeznaczonego na dane nie jest równa rozmiarowi pliku i pomniejszona jest o przestrzeń wykorzystywaną przez system, co widać szczególnie w przestrzeniach tabel zarządzanych lokalnie (tu TEST).

```
Przykład.

SQL> select TABLESPACE_NAME, sum(BYTES), sum(BYTES)/1024/1024 "MB",
  2    sum(BLOCKS) from dba_free_space
  3  group by TABLESPACE_NAME;

TABLESPACE_NAME  SUM(BYTES)      MB SUM(BLOCKS)
---------------- ---------- ------- -----------
```

```
DRSYS                16637952    15.9        2031
INDX                 20963328    20.0        2559
RBS                  23060480    22.0        2815
SYSTEM                7979008     7.6         974
TEMP                 20963328    20.0        2559
TEST                    65536      .1           8
TOOLS                10477568    10.0        1279
USERS                19783680    18.9        2415
```

W przykładzie pobrano z bazy informacje o sumarycznej wolnej przestrzeni w poszczególnych przestrzeniach tabel. Wartości wyprowadzono w bajtach, megabajtach i blokach.

9.2 Operacje na przestrzeniach tabel

Kontrola wolnego miejsca w pliku i ograniczenie fragmentacji

Dla uzyskania maksymalnej wydajności bazy korzystne jest wstępne alokowanie miejsca dla tabel i indeksów. Każde tworzenie nowego obszaru jest kosztowne czasowo, szczególnie jeśli wiąże się dodatkowo z alokacja miejsca na dysku, co występuje gdy w pliku brak wolnej przestrzeni. W takim przypadku dla plików o stałym rozmiarze występuje błąd i transakcja która potrzebowała dodatkowego miejsca zostaje przerwana. Dla plików typu autoincrement następuje automatyczne rozszerzenie pliku, ale operacja ta jest stosunkowo długotrwała i może odbić się na wydajności aplikacji. W przypadku bardzo dużych baz danych, gdzie z konieczności przestrzenie tabel są podzielone na pliki, zwykle maksymalnych rozmiarów kontrola wolnego miejsca ma fundamentalne znaczenie, gdyż brak miejsca oznacza konieczność utworzenia nowego pliku. Jest to wyłączne zadanie administratora, gdyż Oracle nie wspiera mechanizmu automatycznego zakładania pliku. Operacja ta powinna być wykonywana odpowiednio wcześniej, tak by w bazie istniał zawsze bezpieczny zapas wolnego miejsca.

Odczyt wolnego miejsca w bazie odbywa się poprzez perspektywę DBA_FREE_SPACE tak jak pokazano to w przykładzie. Należy pamiętać, że zapytanie o SUM(BLOCKS) podaje sumę wszystkich wolnych obszarów i nie jest wystarczająco miarodajne. Ze względu na fragmentację bazy i brak wystarczająco dużego ciągłego obszaru może zabraknąć miejsca na utworzenie nowego obszaru dla rozszerzającej się tabeli. Dlatego należy dodatkowo sprawdzać fragmentację przestrzeni tabel.

```
Przykład.

SQL> select TABLESPACE_NAME, FILE_ID, BYTES, BLOCK_ID, BLOCKS
  2    from dba_free_space
  3   where TABLESPACE_NAME='USERS'
  4   order by TABLESPACE_NAME, FILE_ID;

TABLESPACE_NAME     FILE_ID      BYTES   BLOCK_ID     BLOCKS
---------------- ---------- ---------- ---------- ----------
USERS                     3   19128320        226       2335
USERS                     3     131072        114         16
USERS                     3     131072         82         16
USERS                     3     131072         98         16
USERS                     3     131072        146         16
USERS                     3     131072        130         16
```

W przykładzie pokazano, że na wolne miejsce w przestrzeni USERS składa się 6 ciągłych obszarów o różnych rozmiarach.

Jeżeli przestrzeń tabel (dotyczy przestrzeni zarządzanych przez słownik – Dictionary-Managed Tablespace) jest sfragmentowana można dokonać łączenia obszarów poleceniem **ALTER TABLESPACE COALESCE**.

```
Przykład.

SQL> alter tablespace USERS coalesce;

Przestrzeń tabel została zmieniona.

SQL> select TABLESPACE_NAME, FILE_ID, BYTES, BLOCK_ID, BLOCKS
  2    from dba_free_space
  3   where TABLESPACE_NAME='USERS'
  4   order by TABLESPACE_NAME, FILE_ID;
```

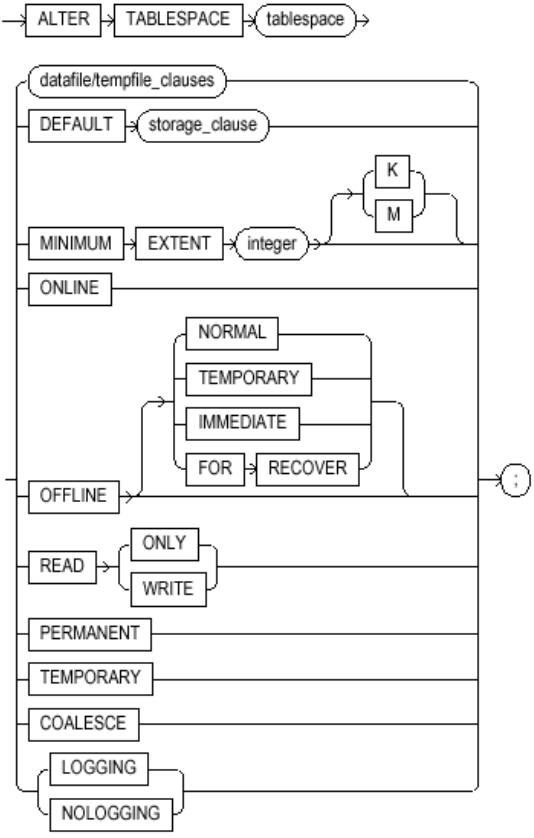
```
TABLESPACE_NAME      FILE_ID      BYTES    BLOCK_ID      BLOCKS
---------------- ---------- ---------- ---------- ----------
USERS                     3   19128320        226        2335
USERS                     3     655360         82          80
```

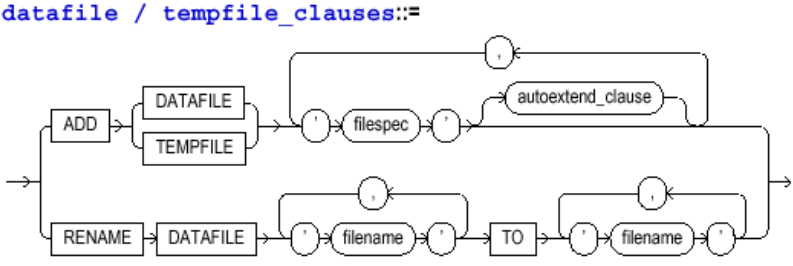
W przykładzie dokonano scalenia 5 obszarów w jeden.

Operacja łączenia jest również wykonywana automatycznie przez Oracle np. podczas alokacji nowego obszaru. Powyższy przykład pokazał, że operacja łączenia dotyczy tylko obszarów przylegających i niemożliwe jest całkowite usunięcie w ten sposób fragmentacji bazy.

Zmiana parametrów przestrzeni

Operacje dotyczące zmiany parametrów przestrzeni tabel wykonywane są poleceniem **ALTER TABLESPACE**. Niektóre operacje dotyczące poszczególnych plików są dostępne z polecenia **ALTER DATABASE**.





Opcja MINIMUM EXTENT dotyczy tylko przestrzeni tabel zarządzanych przez słownik i specyfikuje rozmiar najmniejszego dopuszczalnego obszaru (wolnego lub zajętego) w przestrzeni tabel. Parametr ten specyfikuje najmniejszą dopuszczalną fragmentację w przestrzeni.

```
Przykład.

SQL> alter tablespace DANE2
  2   default storage (next 128K);

Przestrzeń tabel została zmieniona.

SQL> alter database
  2   datafile '/oracle/oradb/dane2.ora'
  3   autoextend on next 1M maxsize unlimited;

Baza danych została zmieniona.
```

W przykładzie zmieniono parametr NEXT w domyślnych parametrach składowania przestrzeni tabel DANE2 oraz parametry automatycznego rozszerzania pliku */oracle/oradb/dane2.ora*.

Dodawanie przestrzeni tabel i plików

Nowe przestrzenie tabel dodaje się do bazy zwykle na skutek potrzeb aplikacyjnych, lub ze względu na potrzeby wydajnościowe (strojenie). Odpowiedni rozkład przestrzeni tabel pomiędzy dyski w istotny sposób może poprawić wydajność systemu przy operacjach prowadzonych równolegle. Nowe pliki dodaje się do istniejących przestrzeni tabel zwykle w następujących przypadkach:

- Istniejąca przestrzeń tabel zawarta jest w pliku o maksymalnym rozmiarze dopuszczalnym dla systemu operacyjnego i brakuje miejsca na nowe dane
- Na dysku zawierającym przestrzeń tabel nie ma już miejsca na rozszerzenie rozmiaru istniejących plików

```
Przykład.

SQL> create tablespace DANE1
  2   datafile '/oracle/oradb/dane1.ora' size 128k reuse
  3   autoextend on next 500K maxsize 10M
  4   extent management local uniform size 4k;

Przestrzeń tabel została utworzona.

SQL> create tablespace DANE2
  2   datafile '/oracle/oradb/dane2.ora' size 128k reuse
  3   autoextend on next 500K maxsize 10M
  4   default storage ( initial 16K next 64K pctincrease 50);

Przestrzeń tabel została utworzona.
```

W przykładzie utworzono dwie przestrzenie tabel jedną zarządzaną lokalnie i drugą zarządzaną przez słownik. Każdą z przestrzeni tworzy jeden plik samorozszerzający się o maksymalnym rozmiarze 10MB.

```
Przykład.

SQL> alter tablespace DANE2
  2   add datafile '/oracle/oradb/dane2_1.ora' size 128k reuse
  3   autoextend on next 500K maxsize unlimited;

Przestrzeń tabel została zmieniona.
```

```
SQL> select TABLESPACE_NAME "TBS", FILE_NAME, BYTES, BLOCKS, STATUS,
  2         AUTOEXTENSIBLE, MAXBYTES, MAXBLOCKS, INCREMENT_BY "INC"
  3         from dba_data_files
  4   where TABLESPACE_NAME = 'DANE2';

TBS   FILE_NAME                      BYTES  BLOCKS STATUS      AUT  MAXBYTES MAXBLOCKS   INC
----- ------------------------- --------- ------- ----------- --- --------- --------- -----
DANE2 /oracle/oradb/dane2.ora      131072      64 AVAILABLE   YES  10485760      5120   250
DANE2 /oracle/oradb/dane2_1.ora    131072      64 AVAILABLE   YES 8,590E+09   4194302   250
```

W przykładzie dodano jeden plik do przestrzeni tabel DANE2 i zweryfikowano parametry plików tworzących tę przestrzeń tabel odczytując dane z perspektywy DBA_DATA_FILES.

Zmiana rozmiaru alokowanego pliku na dysku

Zmianę można wykonać zarówno zwiększając jak i zmniejszając rozmiar pliku. W drugim przypadku operacja powiedzie się o ile w pliku znajduje się wolne, nie używane miejsce. Operację tę wykonuje się poleceniem **ALTER DATABASE DATAFILE RESIZE**.

```
Przykład.

SQL> alter database
  2   datafile '/oracle/oradb/dane2.ora' resize 256k;

Baza danych została zmieniona.

TBS   FILE_NAME                      BYTES  BLOCKS STATUS      AUT  MAXBYTES MAXBLOCKS   INC
----- ------------------------- --------- ------- ----------- --- --------- --------- -----
DANE2 /oracle/oradb/dane2.ora      262144     128 AVAILABLE   YES  10485760      5120   250
DANE2 /oracle/oradb/dane2_1.ora    131072      64 AVAILABLE   YES 8,590E+09   4194302   250
```

W przykładzie powiększono plik */oracle/oradb/dane2.ora* do wielkości 256 i zweryfikowano operację odczytując dane z perspektywy DBA_DATA_FILES.

Przenoszenie pliku do innej lokalizacji

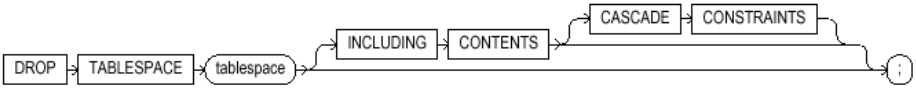
Przenoszenie pliku do innej lokalizacji jest operacją tożsamą ze zmianą nazwy pliku i wykonywane jest operacją **ALTER TABLESPACE RENAME DATAFILE**. Wykonanie tej komendy nie powoduje fizycznie zmiany nazwy pliku, a jedynie zapisuje zmiany w pliku kontrolnym. Dlatego też operację tę należy przeprowadzić z uwzględnieniem konieczności wykonania zmiany nazwy pliku w systemie operacyjnym. Jeżeli zmiana nazwy wiąże się ze zmianą lokalizacji należy plik przekopiować w odpowiednie miejsce. Na czas wykonania operacji należy przestrzeń tabel przestawić w tryb offline, lub wykonać czynności w systemie operacyjnym na położonej bazie, zaś komendę **ALTER TABLESPACE RENAME DATAFILE** wydać po zamontowaniu, ale przed otwarciem bazy.

```
Przykład.

SQL> alter tablespace DANE2 offline;

Przestrzeń tabel została zmieniona.

$ mv dane2.ora dane2_0.ora

SQL> alter tablespace DANE2
  2   rename datafile '/oracle/oradb/dane2.ora' to '/oracle/oradb/dane2_0.ora';

Przestrzeń tabel została zmieniona.

SQL> alter tablespace DANE2 online;

Przestrzeń tabel została zmieniona.
```

W przykładzie pokazano ciąg czynności wykonanych w bazie i systemie operacyjnym skutkujących ostatecznie zmianą nazwy pliku (przeniesieniem do innej lokalizacji komendą **mv**).

Usuwanie przestrzeni tabel

Usuwanie przestrzeni tabel wykonuje się poleceniem **DROP TABLESPACE**.



Jeżeli w przestrzeni tabel znajdują się wygenerowane obiekty np. tabele należy użyć opcji **INCLUDING CONTENTS**. Usunięcie przestrzeni tabel nie skutkuje fizycznym usunięciem plików które ją tworzą, należy usunąć je później ręcznie z poziomu systemu operacyjnego.

```
Przykład.

SQL> drop tablespace DANE2;

Przestrzeń tabel została usunięta.

$ ls -l dane2*
-rw-r-----   1 oracle   dba       133120 mar 14 13:19 dane2_0.ora
-rw-r-----   1 oracle   dba       133120 mar 14 13:19 dane2_1.ora
$ rm dane2*
```

W przykładzie usunięto przestrzeń tabel DANE2, a następnie poleceniem systemu operacyjnego pliki tworzące tę przestrzeń. W momencie usuwania przestrzeń nie zawierała żadnych danych.

10. Zarządzanie użytkownikami

Zarządzanie użytkownikami jest jedną ze standardowych i często wykonywanych czynności przez administratora. Zagadnienie to jest podobne do zarządzania kontami użytkowników w systemach operacyjnych. Oracle umożliwia autoryzację użytkowników poprzez system operacyjny i jednolite zarządzanie użytkownikami w niektórych systemach operacyjnych. W bieżącym rozdziale omówione zostanie wyłącznie zarządzanie użytkownikami poprzez bazę danych. Typowymi czynnościami związanymi z zarządzaniem użytkownikami są:

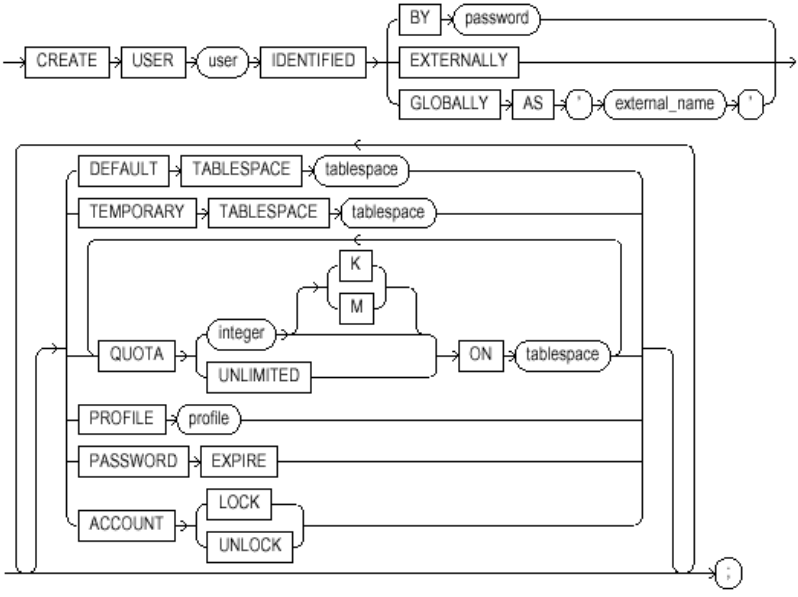
- Zakładanie i usuwanie użytkowników
- Nadawanie przywilejów i ról bazodanowych
- Przydział i zarządzanie profilami
- Zarządzanie hasłami
- Zarządzanie przydziałem zasobów dla użytkowników
- Blokowanie i odblokowanie dostępu do bazy
- Przydział miejsca w przestrzeniach tabel

10.1 Administracja kontami użytkowników

Większość czynności administratorskich wykonuje się poleceniem **CREATE USER** i **ALTER USER**

Zakładanie użytkownika

Polecenie **CREATE USER** służy do zakładania nowego użytkownika w bazie i przypisania mu szeregu parametrów.

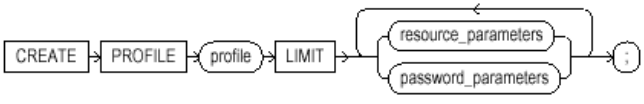


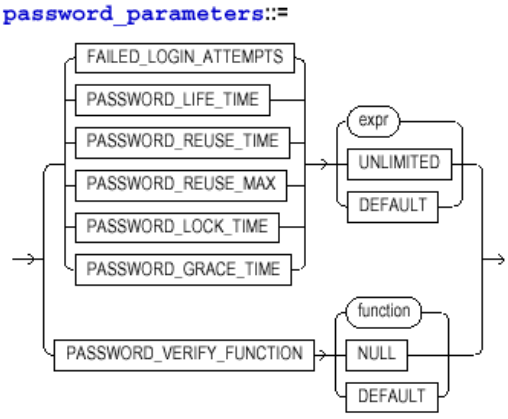
Hasło użytkownika

Użytkownik w bazie jest identyfikowany poprzez hasło zapisane w bazie danych – opcja **IDENTIFIED BY password**, lub poprzez usługę zewnętrzną np. system operacyjny, lub usługę katalogową. Hasło użytkownika jest obligatoryjne. Możliwa jest następująca kontrola haseł:

- Trywialna – wymagane jest jedynie istnienie hasła użytkownika, hasło nigdy nie traci ważności i brak jest kontroli jego komplikacji
- Zaawansowana standardowa – system kontroluje parametry hasła poprzez profil przypisany do użytkownika i standardową funkcję kontroli komplikacji hasła
- Zaawansowana niestandardowa – system kontroluje parametry hasła poprzez profil przypisany do użytkownika, i niestandardową funkcję napisaną przez administratora do kontroli komplikacji hasła

Stopień kontroli hasła jest wymuszany profilem użytkownika. Standardowo użytkownicy mają przypisany profil *default*, który nie zawiera żadnych ograniczeń na hasło. Ograniczenia takie nakłada się tworząc nowy profil, lub modyfikując dla wszystkich profil domyślny. Profil zakłada się poleceniem **CREATE PROFILE**.





Znaczenie parametrów:

- FAILED_LOGIN_ATTEMPTS – liczba nieudanych prób logowania (autoryzacji) przed blokadą konta
- PASSWORD_LIFE_TIME – liczba dni obowiązywania hasła. W przypadku, gdy użytkownik nie zmieni hasła przed upływem terminu, hasło staje się przestarzałe i bez jego zmiany połączenie do bazy jest niemożliwe
- PASSWORD_REUSE_TIME – liczba dni przed upływem których nie można ponownie użyć hasła. Parametr zabezpiecza przed wielokrotnym używaniem przez użytkownika tych samych haseł
- PASSWORD_REUSE_MAX – liczba zmian hasła przed upływem których nie można ponownie użyć hasła. Parametr zabezpiecza przed wielokrotnym używaniem przez użytkownika tych samych haseł
- PASSWORD_LOCK_TIME – liczba dni na które konto zostanie zablokowane po określonej liczbie kolejnych nieudanych prób logowania
- PASSWORD_GRACE_TIME – liczba dni w czasie których logowanie jest jeszcze możliwe, po upływie terminu określonego przez PASSWORD_LIFE_TIME. W okresie tym użytkownik otrzymuje ostrzeżenia o przeterminowaniu hasła, następnie gdy nie zmieni hasła staje się ono przestarzałe i bez jego zmiany połączenie do bazy jest niemożliwe
- PASSWORD_VERIFY_FUNCTION – funkcja kontroli komplikacji hasła. Administrator może wskazać funkcję standardową, dostarczoną przez Oracle, lub napisaną przez siebie.

Oracle dostarcza przykładową funkcję weryfikacji komplikacji hasła w skrypcie *utlpwdmg.sql*. Zawarte tam są następujące ograniczenia:

- Długość hasła co najmniej 4
- Hasło musi różnić się od identyfikatora użytkownika
- Hasło musi zawierać co najmniej jedną literę, jedną cyfrę i jeden znak interpunkcyjny
- Nowe hasło musi różnić się od starego co najmniej trzema znakami
- Hasło nie może być słowem trywialnym, w funkcji zawarto sprawdzenie kilku prostych słów

Przykład.

```
CREATE OR REPLACE FUNCTION verify_function
(username varchar2,
  password varchar2,
  old_password varchar2)
  RETURN boolean IS

           ...

--Check for the minimum length of the password
IF length(password) < 4 THEN
raise_application_error(-20002, 'Password length less than 4');
END IF;
--Check if the password is too simple. A dictionary of words may be
--maintained and a check may be made so as not to allow the words
--that are too simple for the password.
IF NLS_LOWER(password) IN ('welcome', 'database', 'account', 'user',
'password', 'oracle', 'computer', 'abcd')
THEN raise_application_error(-20002, 'Password too simple');
END IF;
```

W przykładzie wylistowano fragment standardowej funkcji o nazwie VERIFY_FUNCTION ze skryptu *utlpwdmg.sql*. Sprawdzeniu podlega długość i trywialność hasła.

Skrypt *utlpwdmg.sql* tworzy funkcję VERIFY_FUNCTION a następnie modyfikuje domyślny profil operacją **ALTER PROFILE**. Ustawieniu podlegają również inne parametry hasła.

```
Przykład.

ALTER PROFILE DEFAULT LIMIT
      PASSWORD_LIFE_TIME 60
      PASSWORD_GRACE_TIME 10
      PASSWORD_REUSE_TIME 1800
      PASSWORD_REUSE_MAX UNLIMITED
      FAILED_LOGIN_ATTEMPTS 3
      PASSWORD_LOCK_TIME 1/1440
      PASSWORD_VERIFY_FUNCTION verify_function;
```

W przykładzie pokazano parametry ustawiane w domyślnym profilu przez skrypt *utlpwdmg.sql*

Blokada konta

Blokada konta wykonywana jest poprzez opcję **ACCOUNT LOCK**, uniemożliwia ona dostęp użytkownika do bazy. Można jej użyć w momencie tworzenia użytkownika np. przygotowując wcześniej konta użytkowników w bazie i udostępniając je w okresie późniejszym. Opcji można również użyć blokując dostęp do schematów przechowujących dane, a których konta nie są przywidziane do bezpośredniej pracy. W tym przypadku konto odblokowuje się tylko na czas administracji obiektami tego schematu.

Odblokowanie konta wykonuje się opcją **ACCOUNT UNLOCK**.

Przeterminowanie hasła

Przeterminowanie hasła wykonywane jest poprzez opcję **PASSWORD EXPIRE**. Opcja jest użyteczna gdy administrator chce wymusić zmianę hasła przy następnym logowaniu. Używając jest w poleceniu **CREATE USER** wymusza się zmianę hasła przy pierwszym logowaniu do bazy.

Domyślna przestrzeń tabel

W przypadku braku przypisania użytkownikowi domyślnej przestrzeni tabel wszystkie obiekty tworzone przez niego, dla których nie wyspecyfikował jawnie przestrzeni tabel będą tworzone w przestrzeni systemowej. Jest to zjawisko niepożądane, dlatego też tworząc użytkownika powinno się określać jego przestrzeń domyślną. Planując przestrzenie tabel w bazie wskazane jest utworzyć przestrzeń tabel dla użytkowników. Przestrzeń domyślną określa się dla użytkownika opcją DEFAULT TABLESPACE.

Tymczasowa przestrzeń tabel

Tymczasowa przestrzeń tabel używana jest dla utworzenia segmentów tymczasowych głównie dla operacji sortowania, które wymagają więcej miejsca niż przewidziano w obszarze pamięci (SORT_AREA_SIZE). Przestrzeń taką należy koniecznie utworzyć w bazie i przypisywać ją użytkownikom jako przestrzeń tymczasową opcją TEMPORARY TABLESPACE.

Limity miejsca w przestrzeniach tabel

Standardowo utworzony użytkownik w bazie danych pomimo prawa do tworzenia obiektów w bazie nie ma możliwości ich utworzyć bez przypisanego mu limitu dostępnego miejsca. Przywilej systemowy UNLIMITED TABLESPACE umożliwia zajęcie dowolnie dużego obszaru bazy na obiekty użytkownika, zaś gdy użytkownik go nie posiada należy opcją QUOTA określić dla niego limit miejsca w określonych przestrzeniach tabel.

Przykład.

```
--- polecenia użytkownika system ---
SQL> create user jzazul identified by jz
  2  default tablespace user_data;

Utworzono użytkownika.

SQL> grant connect to jzazul;

Przyznanie uprawnień zakończone powodzeniem.

--- polecenia użytkownika jzazul ---
SQL> create table TEST
  2   ( col1 number );
create table TEST
*
BŁĄD w linii 1:
ORA-01536: space quota exceeded for tablespace 'USER_DATA'

--- polecenia użytkownika system ---
SQL> alter user jzazul quota 10M on USER_DATA;

Użytkownik został zmieniony.

--- polecenia użytkownika jzazul ---
SQL> create table TEST
  2   ( col1 number );

Tabela została utworzona.

SQL> insert into test
  2    values
  3   (1);

1 wiersz został utworzony.

SQL> commit;

Zatwierdzanie zostało ukończone.
```

W przykładzie utworzono użytkownika bez prawa zapisu do żadnej z przestrzeni tabel. Próba utworzenia tabeli w domyślnej przestrzeni tabel generuje błąd. Po określeniu limitu dostępnego miejsca w przestrzeni USER_DATA użytkownik może w niej tworzyć obiekty i zapisywać dane tak długo jak długo nie wyczerpie przydzielonego limitu.

W przypadku użytkowników, których dostęp do bazy ogranicza się wyłącznie do korzystania z gotowych aplikacji, których obiekty umieszczone są w oddzielnych schematach, nie istnieje potrzeba przydzielania tym użytkownikom limitów miejsca w przestrzeniach tabel. Użytkownicy ci nie posiadają żadnych obiektów we własnych schematach, zaś limit miejsca można określić wyłącznie dla użytkowników – właścicieli schematów aplikacji.

Limity założone na przestrzeniach tabel są widoczne w perspektywie DBA_TS_QUOTAS.

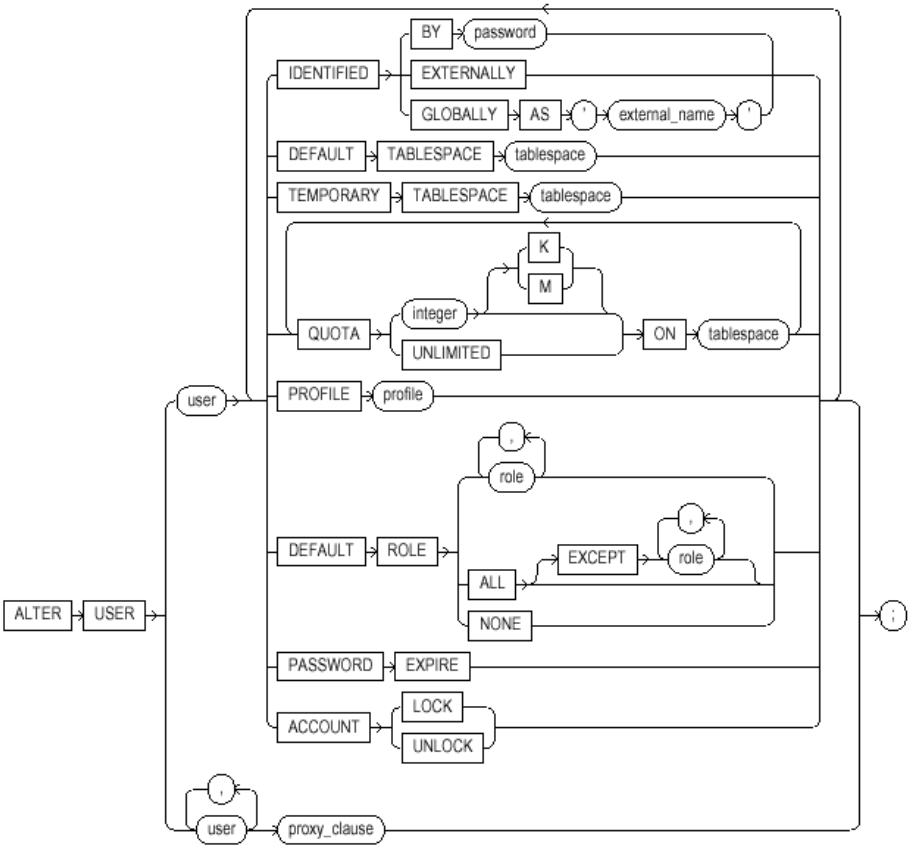
Przykład.

```
SQL> select TABLESPACE_NAME, USERNAME, BYTES, MAX_BYTES
  2   from DBA_TS_QUOTAS
  3  where TABLESPACE_NAME='USER_DATA';

TABLESPACE_NAME USERNAME                         BYTES MAX_BYTES
--------------- ------------------------------ --------- ---------
USER_DATA       JZAZUL                             0  10485760
```

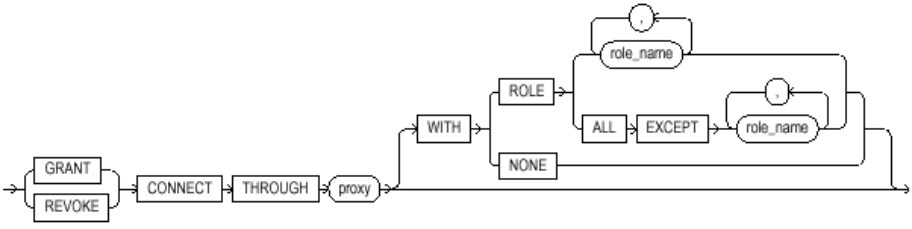
W przykładzie wyświetlono limity miejsca ustawione na przestrzeni tabel USER_DATA.

Zmiana parametrów użytkownika

Parametry użytkownika określone poleceniem CREATE USER mogą być następnie zmienione przy pomocy polecenia ALTER USER.

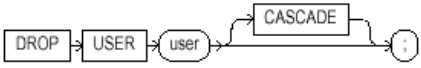


proxy_clause::=



Usuwanie użytkownika z bazy

Usunięcie użytkownika z bazy odbywa się poleceniem **DROP USER**.



Jeżeli użytkownik posiada w swoim schemacie obiekty należy użyć opcji **CASCADE**. W tym przypadku zostanie on usunięty z bazy wraz ze wszystkimi obiektami należącymi do niego. Usuwany użytkownik musi być odłączony od bazy.

```
Przykład.

SQL> drop user jzazul;
drop user jzazul
*
BŁĄD w linii 1:
ORA-01922: CASCADE must be specified to drop 'JZAZUL'


SQL> drop user jzazul cascade;
drop user jzazul cascade
*
BŁĄD w linii 1:
ORA-01940: cannot drop a user that is currently connected

--- po odłączeniu użytkownika od bazy ---
SQL> drop user jzazul cascade;

Usunięto użytkownika.
```

W przykładzie pokazano próbę usunięcia podłączonego do bazy użytkownika posiadającego obiekty w swoim schemacie. Operacja została wykonana po odłączeniu użytkownika i użyciu opcji **CASCADE**.

Informacje o użytkownikach w perspektywach systemowych

Informacje o użytkownikach zawarte są następujących perspektywach systemowych:

- DBA_USERS – statyczne informacje o użytkownikach w bazie. Kolumny:
  - USERNAME – nazwa użytkownika
  - USER_ID - identyfikator użytkownika
  - PASSWORD - zaszyfrowane hasło
  - ACCOUNT_STATUS – status konta użytkownika. Przyjmuje wartości: LOCKED, EXPIRED, UNLOCKED
  - LOCK_DATE – data zablokowania konta użytkownika (o statusie LOCKED)
  - EXPIRY_DATE – data końca ważności konta
  - DEFAULT_TABLESPACE – domyślna przestrzeń tabel użytkownika
  - TEMPORARY_TABLESPACE – tymczasowa przestrzeń tabel użytkownika
  - CREATED – data założenia konta użytkownika w bazie
  - PROFILE – nazwa profilu zasobów użytkownika
  - INITIAL_RSRC_CONSUMER_GROUP – startowa grupa przydziału zasobów
  - EXTERNAL_NAME – zewnętrzna nazwa użytkownika
- DBA_TS_QUOTAS – limity miejsca w przestrzeniach tabel dla użytkowników
  - TABLESPACE_NAME – nazwa przestrzeni tabel
  - USERNAME – nazwa użytkownika
  - BYTES – liczba bajtów wykorzystanych przez użytkownika
  - MAX_BYTES – limit miejsca dla użytkownika w bajtach (-1 brak limitu)
  - BLOCKS – liczba bloków Oracle wykorzystanych przez użytkownika
  - MAX_BLOCKS – limit miejsca dla użytkownika w blokach Oracle (-1 brak limitu)
- DBA_PROFILES – profile w bazie
  - PROFILE – nazwa profilu
  - RESOURCE_NAME – nazwa zasobu
  - RESOURCE_TYPE – typ zasobu. Przyjmuje wartości: KERNEL, PASSWORD
  - LIMIT – wartość limitu
- V$SESSION – bieżące informacje o sesjach pracujących w bazie

10.2 Administracja przywilejami i rolami

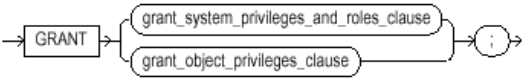
W większości przypadków przywileje i role dla użytkowników bazy danych są projektowane przez twórców aplikacji używanych do pracy z bazą. Rola administratora ogranicza się w tym przypadku do

nadania odpowiednich praw nowo tworzonym użytkownikom bazy oraz kontroli zakresu przyznanych uprawnień. Z punktu widzenia bezpieczeństwa systemu ważne jest sprawdzenie czy standardowi użytkownicy aplikacji nie mają nadmiernych praw, szczególnie w zakresie ról i przywilejów systemowych.

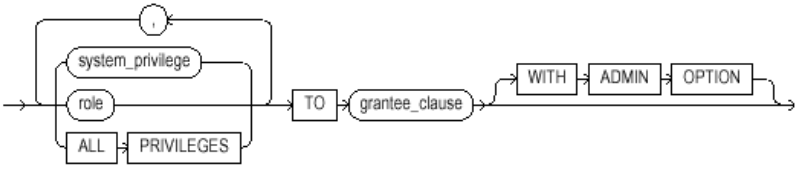
Przyznawanie użytkownikom przywilejów i ról

Prawa użytkownika do działania na obiektach bazy danych są limitowane przez przyznane mu przywileje i role. Przywileje dotyczą obiektów schematu (przywileje obiektowe) lub wykonywania określonych czynności w systemie bazy danych (przywileje systemowe).

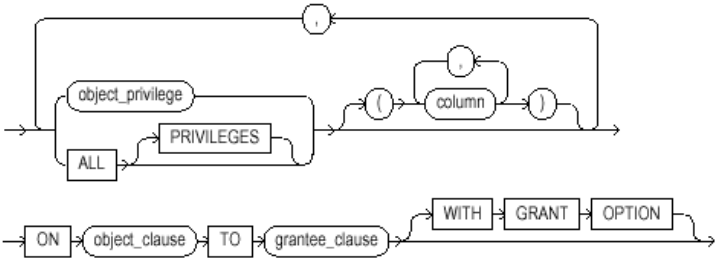
Przyznanie użytkownikowi przywileju lub roli wykonuje się komendą **GRANT**. Ta sama komenda służy do przypisania przywileju lub roli do innej roli. Przyznanie określonego przywileju pseudoużytkownikowi PUBLIC jest równoznaczne z przyznaniem go każdemu użytkownikowi bazy.



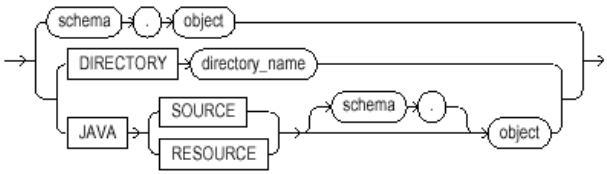
grant_system_privileges_and_roles_clause::=



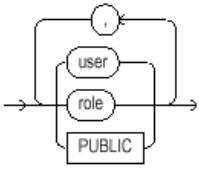
grant_object_privileges_clause::=



object_clause::=



grantee_clause::=



Przyznanie użytkownikowi określonego przywileju umożliwia mu wykonywanie działania dopuszczalnego przez ten przywilej. Przykładowo nadanie przywileju SELECT ANY TABLE umożliwia odczyt danych z tabel dowolnego schematu. Szczególne znaczenie mają jednak opcje ADMIN i GRANT.

Opcja ADMIN używana we frazie WITH ADMIN OPTION dotyczy przywilejów systemowych i przyznaje dodatkowo następujące prawa:

- Nadawanie przywileju (roli) innemu użytkownikowi (roli)
- Odbieranie przywileju (roli) innemu użytkownikowi (roli)
- Zmianę potrzeby autoryzacji potrzebnej do używania roli
- Usunięcie roli

Opcja GRANT używana we frazie WITH GRANT OPTION dotyczy przywilejów obiektowych i przyznaje dodatkowo prawo nadawanie roli innemu użytkownikowi lub roli.

Opcja ALL PRIVILEGES oznaczania nadanie wszystkich praw do obiektu. Właściciel schematu automatycznie uzyskuje wszystkie prawa do obiektów w swoim schemacie wraz z opcją WITH GRANT OPTION. Umożliwia mu to zarządzanie dostępem do swoich obiektów dla innych użytkowników bazy.

Przywileje obiektowe do tabeli lub perspektywy mogą być bardziej szczegółowe i dotyczyć wyspecyfikowanych kolumn, ale wyłącznie dla przyznania praw do wykonywania następujących operacji:

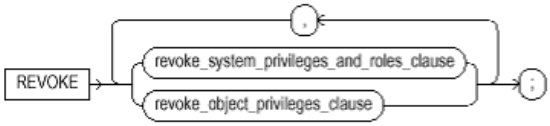
- INSERT
- REFERENCES
- UPDATE

Oprócz klasycznych obiektów schematu można przypisywać przywileje do:

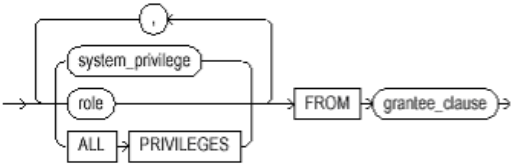
- Katalogów (DIRECTORY) – aliasów w bazie do katalogów systemu plików serwera w których są składowane zewnętrznie pliki binarne (BFILES)
- Obiektów Javy (JAVA SOURCE|RESOURCE) – zawierających źródła, klasy lub zasoby opisane językiem Java.

Odbieranie użytkownikom przywilejów i ról

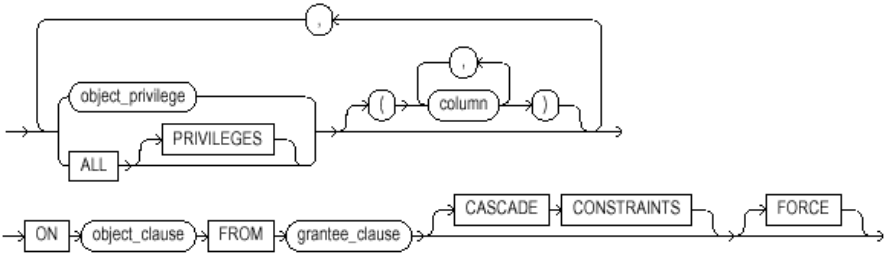
Przyznany użytkownikowi lub roli przywilej (rola) może zostać mu odebrany przy pomocy komendy **REVOKE**. Skutek wykonania komendy jest natychmiastowy tzn. zaraz po jej wykonaniu użytkownik nie może korzystać z praw wynikających z odebranego przywileju. Odebranie przywileju pseudoużytkownikowi PUBLIC oznacza odebranie tych praw wszystkim użytkownikom, którzy nabyli go poprzez PUBLIC. Nie wpływa to na prawa uzyskane poprzez bezpośrednie przyznanie poprzez przywilej lub rolę.



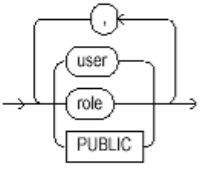
revoke_system_privileges_and_roles_clause::=



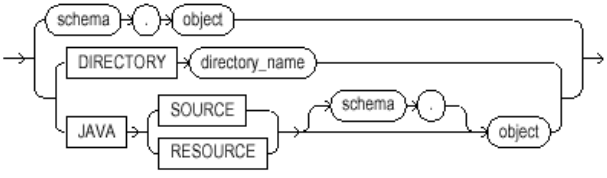
revoke_object_privileges_clause::=



grantee_clause::=



object_clause::=



Znaczenie poszczególnych parametrów i opcji jest analogiczne jak w komendzie **GRANT**.

Przywilej obiektowy REFERENCES oznacza prawo tworzenia więzów referencyjnych (kluczy obcych) do obiektu. Dlatego też odebranie tego przywileju wiąże się automatycznie z koniecznością likwidacji więzów utworzonych przez użytkownika korzystającego z tego prawa. Składnia komendy uwzględnia to we frazie **CASCADE CONSTARAINTS**, która jest w tym przypadku obowiązkowa.

Opcja **FORCE** wiąże się z odebraniem prawa EXECUTE na typie obiektowym zdefiniowanym przez użytkownika zawierającym zależne tabele lub typy obiektowe. Efektem odebrania prawa jest oznaczenie

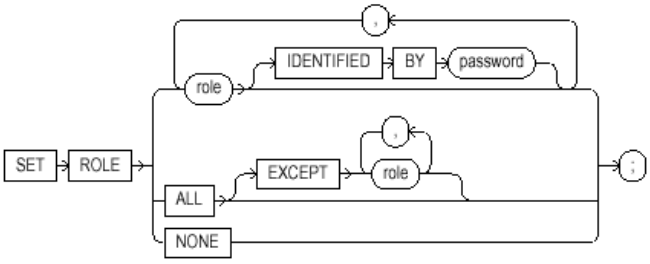
wszystkich zależnych obiektów flagą INVALID oraz brak dostępu do danych w zależnych tabelach. Również wszystkie zależne indeksy oparte na funkcji są oznaczane flagą UNUSABLE.

Przydzielanie domyślnych ról

Użytkownik ma przydzielaną w chwili logowania do systemu domyślną rolę określaną w poleceniu **ALTER USER DEFAULT ROLE**. Dzięki temu w momencie logowania nie jest wymagane podawanie hasła do roli. Tak określona rola musi być wcześniej bezpośrednio nadana użytkownikowi poleceniem GRANT. Nie można określić roli domyślnej jeżeli:

- Nie przydzielono roli poleceniem **GRANT**
- Rola jest przydzielona poprzez inna rolę (przydział pośredni)
- Rola jest zarządzana poprzez usługę zewnętrzną np. system operacyjny lub Oracle Internet Directory.

Uaktywnienie lub dezaktywacja ról w bieżącej sesji wykonywana jest przez użytkownika poleceniem **SET ROLE**. Operacja ta może dotyczyć również roli domyślnej.



Perspektywy systemowe związane z przywilejami i rolami

Poniżej podano ważniejsze perspektywy zawierające informacje dotyczące perspektyw i ról

- DBA_ROLES – opisuje zdefiniowane role. Kolumny:
  - ROLE – nazwa roli
  - PASSWORD_REQUIRED – wartość wskazująca czy rola wymaga hasła
- ROLE_ROLE_PRIVS – opisuje role przypisane innym rolom. Kolumny:
  - ROLE - nazwa roli
  - GRANTED_ROLE – nazwa przypisanej roli
  - ADMIN_OPTION - wartość wskazująca czy nadano prawo ADMIN
- ROLE_SYS_PRIVS – opisuje przywileje systemowe nadane rolom. Kolumny:
  - ROLE - nazwa roli
  - PRIVILEGE – nazwa przyznanego przywileju
  - ADMIN_OPTION - wartość wskazująca czy nadano prawo ADMIN
- ROLE_TAB_PRIVS – opisuje przywileje obiektowe nadane rolom. Kolumny:
  - ROLE - nazwa roli
  - OWNER – właściciel obiektu
  - TABLE_NAME – nazwa obiektu
  - COLUMN_NAME – nazwa kolumny do której wyspecyfikowany szczegółowe prawo dostępu
  - PRIVILEGE – nazwa przyznanego przywileju
  - GRANTABLE – wskazuje czy przyznano ADMIN OPTION
- DBA_ROLE_PRIVS – opisuje role przypisane użytkownikom lub rolom. Kolumny
  - GRANTEE – nazwa użytkownika lub roli otrzymującego przywilej
  - GRANTED_ROLE – przyznana rola
  - ADMIN_OPTION - wartość wskazująca czy nadano prawo ADMIN
  - DEFAULT_ROLE – wskazuje czy jest to rola domyślna
- DBA_SYS_PRIVS - opisuje przywileje systemowe nadane użytkownikom lub rolom. Kolumny:
  - GRANTEE – nazwa użytkownika lub roli otrzymującego przywilej
  - PRIVILEGE – nazwa przyznanego przywileju
  - ADMIN_OPTION - wartość wskazująca czy nadano prawo ADMIN

- DBA_TAB_PRIVS – opisuje przywileje obiektowe nadane użytkownikom. Kolumny:
  - GRANTEE – nazwa użytkownika otrzymującego przywilej
  - OWNER – właściciel obiektu
  - TABLE_NAME – nazwa obiektu
  - GRANTOR – nazwa użytkownika nadającego przywilej
  - PRIVILEGE – nazwa przyznanego przywileju
  - GRANTABLE – wskazuje czy przyznano ADMIN OPTION

```
Przykład.

SQL> select * from dba_roles;

ROLE                           PASSWORD
------------------------------ --------
CONNECT                        NO
RESOURCE                       NO
DBA                            NO
AQ_USER_ROLE                   NO
SELECT_CATALOG_ROLE            NO
EXECUTE_CATALOG_ROLE           NO
DELETE_CATALOG_ROLE            NO
AQ_ADMINISTRATOR_ROLE          NO
EXP_FULL_DATABASE              NO
IMP_FULL_DATABASE              NO
RECOVERY_CATALOG_OWNER         NO
SNMPAGENT                      NO
XX                             YES
```

W przykładzie wyświetlono wszystkie role zdefiniowane w systemie. Z wyjątkiem roli XX pozostałe tworzone są podczas instalacji systemu. Rola DBA zarezerwowana jest standardowo dla administratora systemu.

```
Przykład.

SQL> select ROLE, GRANTED_ROLE, ADMIN_OPTION
  2    from role_role_privs
  3   where ROLE='EXP_FULL_DATABASE';

ROLE                           GRANTED_ROLE                   ADM
------------------------------ ------------------------------ ---
EXP_FULL_DATABASE              EXECUTE_CATALOG_ROLE           NO
EXP_FULL_DATABASE              SELECT_CATALOG_ROLE            NO


SQL> select ROLE, PRIVILEGE, ADMIN_OPTION
  2    from role_sys_privs
  3   where ROLE='EXP_FULL_DATABASE';

ROLE                           PRIVILEGE                                ADM
------------------------------ ---------------------------------------- ---
EXP_FULL_DATABASE              ADMINISTER RESOURCE MANAGER              NO
EXP_FULL_DATABASE              BACKUP ANY TABLE                         NO
EXP_FULL_DATABASE              EXECUTE ANY PROCEDURE                    NO
EXP_FULL_DATABASE              EXECUTE ANY TYPE                         NO
EXP_FULL_DATABASE              SELECT ANY TABLE                         NO
```

W przykładzie wyświetlono role a następnie przywileje systemowe przypisane roli EXP_FULL_DATABASE.

```
Przykład.

SQL> select ROLE, OWNER, TABLE_NAME, COLUMN_NAME, PRIVILEGE, GRANTABLE
  2    from ROLE_TAB_PRIVS
  3  where ROLE='XX';

ROLE  OWNER      TABLE_NAME COLUMN_NAM PRIVILEGE  GRA
----- ---------- ---------- ---------- ---------- ---
XX    SYSTEM     TEST                  SELECT     NO
```

W przykładzie pokazano, że rola XX ma prawo odczytu danych ze wszystkich kolumn tabeli SYSTEM.TEST bez prawa **GRANT OPTION**.

11. Zarządzanie kolejkami zadań

Kolejki zadań obsługiwane są przez procesy tła SNPx. Wymagany jest co najmniej jeden proces SNP w systemie. W przypadku konieczności obsługi wielu zadań np. odświeżanie wielu zmaterializowanych widoków względy wydajnościowe wymagają uruchomienia więcej niż jednego procesu SNP. Oracle umożliwia uruchomienie do 36 takich procesów dla pojedynczej instancji. W przypadku istnienia wielu procesów SNP uruchamianie zadań jest dzielone pomiędzy nie, przy czym każde zadanie może być uruchomione w danej chwili tylko przez jeden proces. O liczbie uruchomionych procesów decyduje parametr inicjalizacyjny JOB_QUEUE_PROCESSES przyjmujący wartość w zakresie 0-36.

Procesy SNP sprawdzają konieczność uruchomienia zadań okresowo ze stałym interwałem czasowym. Wartość interwału czasowego jest sterowana parametrem inicjalizacyjnym JOB_QUEUE_INTERVAL przyjmującym wartości w zakresie 1-3600. Wartość ta określa w sekundach co jaki czas proces SNP sprawdza czy są zadania do uruchomienia.

```
Przykład.

Parametry pliku initORCL.ora

JOB_QUEUE_PROCESSES = 2
JOB_QUEUE_INTERVAL = 60
```

W przykładzie podano wartości z pliku inicjalizacyjnego oznaczające, że dla tej instancji zostaną uruchomione dwa procesy tła SNP1 i SNP2, które co 60 sekund będą sprawdzały i ewentualnie uruchamiały zaplanowane zadania.

Zarządzanie kolejkami zadań odbywa się poprzez pakiet bazodanowy DBMS_JOB. Informacje o zadaniach zawarte są w perspektywach systemowych.

Właścicielem zadania staje się użytkownik, który je uruchomił. Istnieje istotne ograniczenie w administracji kolejkami zadań polegające na tym, iż wyłącznie właściciel zadania ma prawo je zmienić, wymusić uruchomienie lub usunąć z kolejki.

Zadaniem administratora bazy jest nadzór nad prawidłowym funkcjonowaniem kolejek zadań utworzonych przez aplikacje, w szczególności sprawdzenie czy wszystkie planowane zadania wykonują się codziennie, oraz diagnozowanie przyczyny zatrzymania zadań (dla których flaga BROKEN ma wartość Y). Często administrator uruchamia własne zadania np. obliczanie statystyk, planując ich uruchomienie w czasie przerw w pracy użytkowników np. w nocy.

11.1 Pakiet DBMS_JOB

Pakiet DBMS_JOB zawiera następujące procedury:

- SUBMIT – definiuje nowe zadanie

```
DBMS_JOB.SUBMIT (
        job OUT BINARY_INTEGER,
        what IN VARCHAR2,
        next_date IN DATE DEFAULT sysdate,
        interval IN VARCHAR2 DEFAULT 'null',
        no_parse IN BOOLEAN DEFAULT FALSE,
        instance IN BINARY_INTEGER DEFAULT any_instance,
        force IN BOOLEAN DEFAULT FALSE );
```

Parametry procedury:

- JOB – wartość numeru zadania zwracana przez procedurę. Jest ustalana w oparciu o sekwencję SYS.JOBSEQ.
- WHAT – zapisana w języku PL/SQL procedura opisująca zadanie
- NEXT_DATE – czas w którym zostanie uruchomione zadanie
- INTERVAL – wyliczenie następnego punktu w czasie w którym zostanie uruchomione zadanie
- NO_PARSE – flaga logiczna. Przyjmuje wartości: FALSE – oznacza, że procedura będzie parsowana zawsze przy uruchamianiu zadania, TRUE – oznacza, że procedura będzie parsowana tylko przy pierwszym uruchomieniu zadania.
- INSTANCE – określa instancję bazy która może uruchamiać zadanie
- FORCE – flaga logiczna. Przyjmuje wartości: TRUE – dowolna wartość dodatnia jest dopuszczalna jako numer instancji, FALSE - określona instancja musi być czynna aby zadanie mogło być uruchomione. Parametry INSTANCE i FORCE są używane w celu sztywnego przypisania zadania do instancji bazy, lub zezwolenia na uruchamianie zadania przez dowolną instancję.

- REMOVE – usuwa zadanie z kolejki zadań. Nie zatrzymuje uruchomionego zadania

```
DBMS_JOB.REMOVE (
        job IN BINARY_INTEGER );
```

Parametry procedury:

- JOB – numer zadania

- CHANGE – zmienia definicję zadania

```
DBMS_JOB.CHANGE (
      job IN BINARY_INTEGER,
      what IN VARCHAR2,
      next_date IN DATE,
      interval IN VARCHAR2,
      instance IN BINARY_INTEGER DEFAULT NULL,
      force IN BOOLEAN DEFAULT FALSE);
```

Parametry procedury:

- JOB – numer zadania
- WHAT – zapisana w języku PL/SQL procedura opisująca zadanie
- NEXT_DATE – czas w którym zostanie uruchomione zadanie
- INTERVAL – wyrażenie wyliczające następny punktu w czasie w którym zostanie uruchomione zadanie
- INSTANCE – określa instancję bazy która może uruchamiać zadanie. Wartość NULL oznacza brak zmiany przypisania zadania do instancji.
- FORCE – flaga logiczna. Przyjmuje wartości: TRUE – dowolna wartość dodatnia jest dopuszczalna jako numer instancji, FALSE - określona instancja musi być czynna aby zadanie mogło być uruchomione. Parametry INSTANCE i FORCE są używane w celu sztywnego przypisania zadania do instancji bazy, lub zezwolenia na uruchamianie zadania przez dowolną instancję

- WHAT – zmienia treść procedury tworzącej zadanie

```
DBMS_JOB.WHAT (
      job IN BINARY_INTEGER,
      what IN VARCHAR2,
```

Parametry procedury:

- JOB – numer zadania
- WHAT – zapisana w języku PL/SQL procedura opisująca zadanie

- NEXT_DATE – zmienia czas uruchomienia zadania

```
DBMS_JOB.NEXT_DATE (
      job IN BINARY_INTEGER,
      next_date IN DATE );
```

Parametry procedury:

- JOB – numer zadania
- NEXT_DATE – czas w którym zostanie uruchomione zadanie

- INSTANCE – zmienia przypisanie zadania do instancji

```
DBMS_JOB.INSTANCE (
      job IN BINARY_INTEGER,
      instance IN BINARY_INTEGER DEFAULT NULL,
      force IN BOOLEAN DEFAULT FALSE );
```

Parametry procedury:

- JOB – numer zadania
- INSTANCE – określa instancję bazy która może uruchamiać zadanie. Wartość NULL oznacza brak zmiany przypisania zadania do instancji.
- FORCE – flaga logiczna. Przyjmuje wartości: TRUE – dowolna wartość dodatnia jest dopuszczalna jako numer instancji, FALSE - określona instancja musi być czynna aby zadanie mogło być uruchomione. Parametry INSTANCE i FORCE są używane w celu sztywnego przypisania zadania do instancji bazy, lub zezwolenia na uruchamianie zadania przez dowolną instancję

- INTERVAL – zmienia częstotliwość uruchamiania zadania

```
DBMS_JOB.INTERVAL (
       job OUT BINARY_INTEGER,
       interval IN VARCHAR2 DEFAULT 'null';
```

Parametry procedury:

- JOB – numer zadania
- INTERVAL – wyliczenie następnego punktu w czasie w którym zostanie uruchomione zadanie

- BROKEN – ustawia flagę oznaczającą, że zadanie nie zostanie nigdy uruchomione

```
DBMS_JOB.BROKEN (
        job IN BINARY_INTEGER,
        broken IN BOOLEAN,
        next_date IN DATE DEFAULT SYSDATE);
```

Parametry procedury:

- JOB – numer zadania
- BROKEN – flaga. Ustawienie flagi skutkuje tylko gdy zadanie nie jest aktualnie wykonywane. W przeciwnym przypadku Oracle automatycznie przestawia tę flagę na NORMAL po zakończeniu uruchomionego zadania.
- NEXT_DATE – czas w którym zostanie uruchomione zadanie

- RUN – wymusza natychmiastowe uruchomienie zadania, nawet jeśli jest oznaczone flagą BROKEN

```
DBMS_JOB.RUN (
      job IN BINARY_INTEGER,
      force IN BOOLEAN DEFAULT FALSE );
```

Parametry procedury:

- JOB – numer zadania
- IFORCE – flaga logiczna. Przyjmuje wartości: TRUE – dowolna wartość dodatnia jest dopuszczalna jako numer instancji, FALSE - określona instancja musi być czynna aby zadanie mogło być uruchomione. Parametry INSTANCE i FORCE są używane w celu sztywnego przypisania zadania do instancji bazy, lub zezwolenia na uruchamianie zadania przez dowolną instancję

- USER_EXPORT – zwraca tekst umożliwiający ponowne utworzenie wskazanego zadania

```
DBMS_JOB.USER_EXPORT (
        job IN BINARY_INTEGER,
        mycall IN OUT VARCHAR2,
        myinst IN OUT VARCHAR2);
```

Parametry procedury:

- JOB – numer zadania
- MYCALL – tekst tworzący zadanie
- MYINST – tekst umożliwiający zmianę przypisania zadania do instancji

11.2 Informacje o zadaniach w słowniku systemowym

Słownik systemowy zawiera informacje o zadaniach w następujących perspektywach:

- DBA_JOBS – zawiera informacje o zadaniach zdefiniowanych w bazie. Znaczenie kolumn:
  - JOB – numer zadania
  - LOG_USER – identyfikator użytkownika tworzącego zadanie
  - PRIV_USER – identyfikator użytkownika na prawach którego wykonywane jest zadanie
  - SCHEMA_USER – domyślny schemat używany do parsowania poleceń używanych w zadaniu. Obiekty używane w zadaniu, dla których nie wyspecyfikowano jawnie schematu będą interpretowane przez parser jako należące do schematu SCHEMA_USER
  - LAST_DATE – data ostatniego udanego uruchomienia zadania
  - LAST_SEC - czas ostatniego udanego uruchomienia zadania
  - THIS_DATE – data uruchomienia wystartowanego zadania
  - THIS_SEC – czas uruchomienia wystartowanego zadania
  - NEXT_DATE – data następnego uruchomienia zadania
  - NEXT_SEC – czas następnego uruchomienia zadania
  - TOTAL_TIME- czas zużyty przez system na obsługę zadania w sekundach.
  - BROKEN – przyjmuje wartości: Y – zadanie nie będzie przez system uruchamiane, N – zadanie będzie przez system uruchamiane
  - INTERVAL – wyrażenie wyliczające następny punktu w czasie w którym zostanie uruchomione zadanie
  - FAILURES – liczba nieudanych prób wystartowania zadania liczona od ostatniego udanego uruchomienia
  - WHAT – ciało anonimowego bloku PL/SQL które uruchamia zadanie. Odpowiada wartości parametru WHAT w procedurze DBMS_JOB.SUBMIT
  - NLS_ENV – parametry sesji związanej z zadaniem dotyczące ustawień narodowych

  - MISC_ENV – inne parametry sesji używane przez zadanie
  - INSTANCE – identyfikator instancji która może uruchomić zadanie
- DBA_JOBS_RUNNING – zawiera informacje o zadaniach aktualnie uruchomionych w bazie. Znaczenie kolumn:
  - SID – identyfikator procesu wykonującego zadanie
  - JOB – numer zadania aktualnie wykonywanego
  - FAILURES - liczba nieudanych prób wystartowania zadania liczona od ostatniego udanego uruchomienia
  - LAST_DATE - data ostatniego udanego uruchomienia zadania
  - LAST_SEC - czas ostatniego udanego uruchomienia zadania
  - THIS_DATE – data uruchomienia wystartowanego zadania
  - THIS_SEC – czas uruchomienia wystartowanego zadania
  - INSTANCE – identyfikator instancji która może uruchomić zadanie

```
Przykład.

SQL> VAR job number;
SQL> begin
  2   DBMS_JOB.SUBMIT(:job, 'begin DBMS_UTILITY.ANALYZE_SCHEMA(''BS'',''ESTIMATE''); end;',
  3     TO_DATE( TO_CHAR( sysdate, 'YYYY-MM-DD' )||'' 00:05:00','YYYY-MM-DD HH24:MI:SS'),
  4    'TO_DATE( TO_CHAR( sysdate, ''YYYY-MM-DD'' )||'' 00:05:00'',''YYYY-MM-DD
HH24:MI:SS'') + 1' );
  5  end;
  6  /

Procedura PL/SQL została zakończona pomyślnie.

SQL> print job

      JOB
---------
       25

SQL> select JOB, LOG_USER, WHAT, BROKEN, FAILURES, INTERVAL
  2   from dba_jobs
  3  where job=25;

 JOB LOG_USER WHAT                                                          BROKEN  FAILURES
---- -------- ------------------------------------------------------------ ------ ---------
INTERVAL
--------------------------------------------------------------------------------
  25 SYSTEM   begin DBMS_UTILITY.ANALYZE_SCHEMA('BS','ESTIMATE'); end;     N
TO_DATE( TO_CHAR( sysdate, 'YYYY-MM-DD' )||'' 00:05:00','YYYY-MM-DD HH24:MI:SS') + 1
```

Utworzono zadanie polegające na wykonaniu obliczenia statystyk w bazie dla wszystkich obiektów schematu BS metodą przybliżoną. Zadanie będzie uruchamiane raz dziennie pięć minut po północy. Utworzone zadanie otrzymało numer 25. Informacja o zadaniu jest opisana w perspektywie DBA_JOBS.

```
Przykład.

SQL> select NLS_ENV from dba_jobs
  2    where job=1;

NLS_ENV
--------------------------------------------------------------------------------
NLS_LANGUAGE='POLISH' NLS_TERRITORY='POLAND' NLS_CURRENCY='zł' NLS_ISO_CURRENCY='POLAND'
NLS_NUMERIC TE_LANGUAGE='POLISH' NLS_SORT='POLISH'
```

W przykładzie pokazano ustawienia narodowe sesji używane przez zadanie numer 1.

```
Przykład.

SQL> select SID, JOB, FAILURES, LAST_DATE, LAST_SEC, THIS_DATE, THIS_SEC
  2    from dba_jobs_running
  3   where job=2;

nie wybrano żadnych wierszy

SQL> select JOB, LAST_DATE, LAST_SEC, THIS_DATE, THIS_SEC,
  2         NEXT_DATE, NEXT_SEC, FAILURES
```

```
  3    from DBA_JOBS
  4   where job=2;

       JOB LAST_DAT LAST_SEC THIS_DAT THIS_SEC NEXT_DAT NEXT_SEC   FAILURES
---------- -------- -------- -------- -------- -------- -------- ----------
         2 02/05/25 13:14:14                  02/05/26 13:14:14          0
```

Zadanie numer 2 jest zdefiniowane w bazie, ale w chwili wykonywania zapytania do bazy nie jest uruchomione. Świadczy o tym pusta odpowiedź z perspektywy DBA_JOBS_RUNNING, oraz wartości NULL w kolumnach THIS_DATE, THIS_SEC w perspektywie DBA_JOBS.

```
Przykład.

SQL> select SID, JOB, FAILURES, LAST_DATE, LAST_SEC, THIS_DATE, THIS_SEC
  2    from dba_jobs_running
  3   where job=2;

       SID        JOB   FAILURES LAST_DAT LAST_SEC THIS_DAT THIS_SEC
---------- ---------- ---------- -------- -------- -------- --------
         8          2          0 02/05/25 13:14:14 02/05/25 13:17:54

SQL>
SQL> select JOB, LAST_DATE, LAST_SEC, THIS_DATE, THIS_SEC,
  2         NEXT_DATE, NEXT_SEC, FAILURES
  3    from DBA_JOBS
  4   where job=2;

       JOB LAST_DAT LAST_SEC THIS_DAT THIS_SEC NEXT_DAT NEXT_SEC   FAILURES
---------- -------- -------- -------- -------- -------- -------- ----------
         2 02/05/25 13:14:14 02/05/25 13:17:54 02/05/26 13:14:14          0
```

Zadanie numer 2 jest uruchomione w chwili odczytu danych z perspektyw systemowych. Perspektywa DBA_JOBS_RUNNING zawiera wiersz opisujący to zadanie. Kolumny THIS_DATE, THIS_SEC w perspektywach DBA_JOBS i DBA_JOBS_RUNNING zawierają te same wartości – dokładną datę i czas uruchomienia zadania.

```
Przykład.

SQL> var c1 varchar2(4000)
SQL> var c2 varchar2(4000)
SQL> begin
  2   DBMS_JOB.USER_EXPORT ( 26,:c1,:c2 );
  3  end;
  4  /

Procedura PL/SQL została zakończona pomyślnie.

SQL> print c1

C1
--------------------------------------------------------------------------------
------
dbms_job.isubmit(job=>26,what=>'begin DBMS_UTILITY.ANALYZE_SCHEMA(''BS'',''ESTIMATE'');
end;',next_d
ate=>to_date('2002-05-23:00:05:00','YYYY-MM-DD:HH24:MI:SS'),interval=>'TO_DATE( TO_CHAR(
sysdate, ''
YYYY-MM-DD'' )||'' 00:05:00'',''YYYY-MM-DD HH24:MI:SS'') + 1',no_parse=>TRUE);

SQL> print c2

C2
--------------------------------------------------------------------------------
null;
```

W przykładzie wyeksportowane dane o zadaniu w postaci gotowego do uruchomienia wywołania procedury. Wartość drugiej zmiennej jest NULL gdyż nie określono w definicji zadania przywiązania zadania do instancji bazy danych.

```
Przykład.

SQL> begin
  2   DBMS_JOB.NEXT_DATE( 25, TO_DATE( TO_CHAR( sysdate, 'YYYY-MM-DD' )||' 01:05:00','YYYY-MM-
DD HH24:MI:SS') );
  3  end;
```

```
  4  /

Procedura PL/SQL została zakończona pomyślnie.

SQL> select JOB, LOG_USER, SCHEMA_USER,
  2         LAST_DATE, LAST_SEC, THIS_DATE, THIS_SEC,
  3         TO_CHAR(NEXT_DATE,'YYYY/MM/DD HH24:MI:SS')
  4    from dba_jobs
  5   where job=25;

 JOB LOG_USER     SCHEMA_USER  LAST_DAT LAST_SEC THIS_DAT THIS_SEC TO_CHAR(NEXT_DATE,'
---- ------------ ------------ -------- -------- -------- -------- -------------------
  25 SYSTEM       SYSTEM                                           2002/05/23 01:05:00
```

W przykładzie zmieniono godzinę uruchomienia zadania na 1:05. Zmianę zweryfikowano odczytując perspektywę DBA_JOBS.

```
Przykład.

SQL> exec DBMS_JOB.Remove(25);

Procedura PL/SQL została zakończona pomyślnie.

SQL> select JOB, LOG_USER, SCHEMA_USER,
  2         LAST_DATE, LAST_SEC, THIS_DATE, THIS_SEC,
  3         TO_CHAR(NEXT_DATE,'YYYY/MM/DD HH24:MI:SS')
  4    from dba_jobs
  5   where job=25;

nie wybrano żadnych wierszy
```

W przykładzie usunięto zadanie o numerze 25. W perspektywie DBA_JOBS brak wierszy opisujących zadanie o tym numerze.

12. Zarządzanie segmentami wycofania

Segmenty wycofania (rollback segments) niezbędne są do przeprowadzania transakcji w bazie danych. W bazie istnieje zawsze co najmniej jeden segment wycofania o nazwie SYSTEM umiejscowiony w systemowej przestrzeni tabel. Jest on używany do wewnętrznych potrzeb motoru bazy danych, przede wszystkim do zabezpieczenia transakcji wykonywanych na danych słownika bazy danych. Systemowy segment wycofania nie powinien być używany do innych celów, dlatego też konieczne jest utworzenie w bazie dodatkowych segmentów przeznaczonych dla transakcji użytkowników. Również niezbędnym warunkiem do wykonywania operacji na wierszach tabel składowanych w niesystemowych przestrzeniach tabel jest istnienie dodatkowych segmentów wycofania.

Oracle przypisuje bieżące transakcje do kolejnych segmentów wycofania, tak aby zapewnić równomierne obciążenie wszystkich segmentów. Mechanizm ten jest automatyczny i jedynym zadaniem administratora jest zapewnienie istnienia w bazie odpowiedniej liczby segmentów wycofania oraz wystarczającej przestrzeni do zapisywania w nich informacji o transakcjach. Ważne jest również aby wielkość segmentów była dostosowana do potrzeb aplikacji używających długich zapytań w celu wyeliminowania możliwości pojawienia się błędu „SNAPSHOT TOO OLD". Administrator powinien obserwować zachowanie się segmentów wycofania używanych przez aplikacje pracujące na bazie, analizować ich rozrost i planować ewentualne ich zwijanie do wielkości optymalnej. Jeżeli aplikacja używa specjalnie zaprojektowanych segmentów wycofania np. dla potrzeb realizacji bardzo dużych transakcji i stosuje przywiązanie transakcji do określonego segmentu wycofania zadaniem administratora jest udostępnienie takiego segmentu dla tej operacji. Konieczne jest wtedy odpowiednie planowanie miejsca w przestrzeni tabel zawierającej segmenty wycofania, oczywiście jeśli obszar jest limitowany przez zasoby dyskowe.

12.1 Aktywacja segmentów wycofania

Segmenty wycofania mogą być przypisane do określonej instancji (segmenty prywatne), lub dostępne dla wszystkich instancji (segmenty publiczne). Udostępnienie segmentu wycofania określonej instancji wymaga umieszczenia jego nazwy w pliku inicjalizacyjnym jako wartość parametru ROLLBACK_SEGMENTS lub wykonania komendy **ALTER ROLLBACK SEGMENT ONLINE** dla wszystkich potrzebnych segmentów wycofania.

```
Przykład.

Parametr pliku init.ora:

ROLLBACK_SEGMENTS = ( R01,R02 )


SQL> alter rollback segment R01 online;

Segment wycofania został zmieniony.

SQL> alter rollback segment R02 online;

Segment wycofania został zmieniony.
```

W przykładzie udostępniono instancji bazy danych dwa segmenty wycofania R01 i R02; w pierwszym przypadku umieszczając nazwy segmentów w pliku inicjalizacyjnym, w drugim wydając komendy SQL.

Publiczne segmenty wycofania mogą być automatycznie udostępniane instancji poprzez odpowiednie ustawienie następujących parametrów inicjalizacyjnych:

- TRANSACTIONS – określa maksymalną liczbę jednocześnie wykonywanych transakcji
- TRANSACTIONS_PER_ROLLBACK_SEGMENT – określa maksymalną liczbę współbieżnych transakcji, które mogą być obsłużone przez jeden segment wycofania.

Liczba udostępnionych segmentów wycofania jest w tym przypadku obliczana ze wzoru:

Liczba segmentów = TRANSACTIONS / TRANSACTIONS_PER_ROLLBACK_SEGMENT

Wyliczona wartość jest zaokrąglana w górę do najbliższej liczby całkowitej.

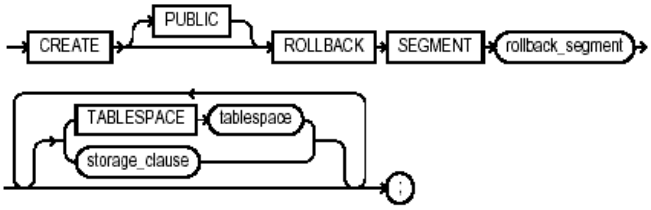
```
Przykład.
Parametry pliku init.ora:

TRANSACTIONS = 205
TRANSACTIONS_PER_ROLLBACK_SEGMENT = 10
```

Dla wartości podanych w przykładzie podczas podnoszenia bazy danych system udostępni 21 segmentów wycofania. Segmenty te muszą być wcześniej utworzone w bazie.

12.2 Operacje na segmentach wycofania

Tworzenie segmentów wycofania

Segmenty wycofania tworzone są komendą **CREATE ROLLBACK SEGMENT**.



Znaczenie parametrów:

- PUBLIC – określa dostępność segmentu wycofania dla wszystkich instancji
- TABLESPACE – określa przestrzeń tabel w której jest tworzony segment wycofania. Parametr powinien być specyfikowany, gdyż w przeciwnym przypadku Oracle utworzy segment w przestrzeni systemowej. Segmenty wycofania mogą być tworzone w przestrzeniach tabel zarządzanych poprzez słownik, lub zarządzanych lokalnie typu UNIFORM. Nie mogą być tworzone w przestrzeniach tabel zarządzanych lokalnie przez system.

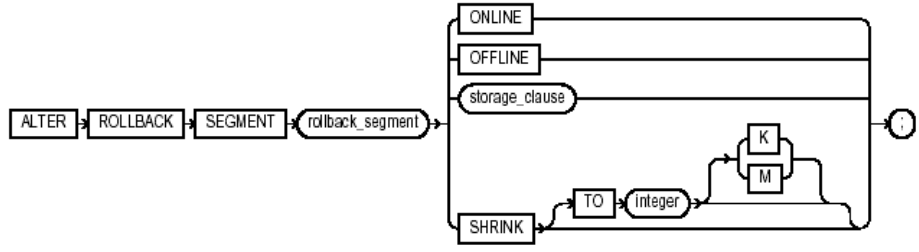
Cechą specyficzną wyłącznie dla segmentów wycofania jest istnienie opcji OPTIMAL we frazie STORAGE_CLAUSE. Określa ona optymalną wielkość segmentu wycofania w bajtach. Oracle dynamicznie zmniejsza wielkość segmentu do wartości optymalnej gdy transakcja nie potrzebuje dłużej danych zapisanych w tym obszarze. Dzięki temu mechanizmowi wiele segmentów wycofania jest w stanie wspólnie korzystać z tej samej powierzchni dyskowej w ramach jednej przestrzeni tabel.

Wszystkie obszary segmentu wycofania mają ten sam rozmiar, z tego powodu parametr PCTINCREASE jest niedopuszczalny.

Po utworzeniu segment wycofania jest nieaktywny – offline.

Zmiana parametrów segmentów wycofania

Istniejące segmenty wycofania zmienia się komendą **ALTER ROLLBACK SEGMENT**



Znaczenie parametrów:

- ONLINE – uaktywnienie segmentu wycofania
- OFFLINE – dezaktywacja segmentu wycofania. Operacja jest wykonywana natychmiast jeśli żadna aktywna transakcja nie używa tego segmentu. W przeciwnym przypadku segment nie może być używany przez kolejne transakcje, ale status offline zostaje ustawiony po zakończeniu transakcji aktualnie korzystających z tego segmentu
- SHRINK – powoduje zmniejszenie rozmiaru segmentu wycofania do podanej wartości, nie mniejszej jednak niż 2 obszary (extents). Jeżeli nie zostanie wyspecyfikowana wartość a w definicji segmentu określono wartość OPTIMAL, to do tej wartości zostanie on zmniejszony. W przeciwnym przypadku Oracle zmniejszy rozmiar segmentu do wartości MINEXTENTS.

Usuwanie segmentów wycofania

Usunięcie segmentu wycofania wykonywane jest poleceniem **DROP ROLLBACK SEGMENT**.

Przed usunięciem segment wycofania musi być przestawiony w tryb offline.

Przypisanie transakcji do określonego segmentu

Zarządzanie przydziałem transakcji do określonego segmentu wycofania odbywa się automatycznie. Mechanizm ten może okazać się niewystarczający w przypadku wykonywania szczególnie długiej transakcji. W typowych aplikacjach OLTP transakcji jest wiele, lecz każda z nich jest krótkotrwała i dokonuje zmiany niewielkiej liczby wierszy – typowo poniżej 10% wszystkich wierszy w tabeli. Dlatego też dla takich nietypowych transakcji powinno się planować oddzielne segmenty wycofania uaktywniane tylko dla nich. Przypisanie transakcji do określonego segmentu wycofania wykonywane jest komendą:

SET TRANSACTION USE ROLLBACK SEGMENT nazwa_segmentu;

Komenda jest skuteczna tylko dla najbliższej transakcji. Po jej zatwierdzeniu, dla kolejnej funkcjonuje standardowy mechanizm przydziału.

12.3 Informacja o segmentach wycofania

Informacje o istniejących w bazie segmentach wycofania można odczytać z perspektywy DBA_ROLLBACK_SEGS. Znaczenie kolumn:

- SEGMENT_NAME – nazwa segmentu wycofania
- OWNER – właściciel segmentu
- TABLESPACE_NAME – przestrzeń tabel w której utworzono segment
- SEGMENT_ID – numer identyfikacyjny segmentu
- FILE_ID – numer identyfikacyjny pliku zawierającego segment
- BLOCK_ID – numer bloku zawierającego nagłówek segmentu
- INITIAL_EXTENT – rozmiar pierwszego obszaru segmentu wycofania
- NEXT_EXTENT – rozmiar kolejnych obszarów segmentu wycofania
- MIN_EXTENTS – minimalna liczba obszarów segmentu wycofania
- MAX_EXTENTS – maksymalna liczba obszarów segmentu wycofania
- PCT_INCREASE – procent przyrostu kolejnych obszarów, wartość 0 dla segmentów wycofania
- STATUS – status segmentu (online, offline)
- INSTANCE_NUM – numer instancji będącej właścicielem segmentu wycofania.
- RELATIVE_FNO – względny numer pliku zawierającego nagłówek segmentu wycofania

Informacje o aktualnym stanie segmentów wycofania, w tym również o liczbie transakcji korzystających z segmentów znajdują się w dynamicznej perspektywie V$ROLLSTAT, zawierającej szczegółowe statystyki pracy segmentu. W perspektywie brak kolumny zawierającej nazwę segmentu, można ją uzyskać z perspektywy dynamicznej V$ROLLNAME.

Kolumny perspektywy V$ROLLNAME:

- USN – numer segmentu wycofania
- NAME – nazwa segmentu wycofania

Kolumny perspektywy V$ROLLSTAT:

- USN – numer segmentu wycofania
- EXTENTS – liczba obszarów segmentu wycofania
- RSSIZE – rozmiar segmentu w bajtach
- WRITES – liczba bajtów zapisanych do segmentu
- XACTS – liczba aktywnych transakcji w segmencie
- GETS – liczba żądań nagłówka segmentu wycofania
- WAITS - liczba żądań nagłówka segmentu wycofania, które zakończyły się oczekiwaniem.
- OPTSIZE – wartość parametru OPTIMAL segmentu

- HWMSIZE – najwyższa wartość rozmiaru segmentu w bajtach osiągnięta przez niego w trakcie obsługi transakcji (tzw. znacznik wysokiej wody – high water mark)
- SHRINKS – liczba wykonanych redukcji rozmiaru segmentu wykonana w celu utrzymania rozmiaru optymalnego
- WRAPS – liczba przesunięć pozycji zapisu w segmencie z jednego obszaru do drugiego
- EXTENDS – liczba alokowań nowych obszarów
- AVESHRINK – średnia liczba bajtów zwalnianych podczas redukcji rozmiaru segmentu, wykonana w celu utrzymania rozmiaru optymalnego
- AVEACTIVE – średnia liczba bajtów w aktywnych obszarach segmentu wycofania
- STATUS – status segmentu
- CUREXT – aktualny obszar
- CURBLK – aktualny blok

Perspektywy V$ROLLSTAT używa się do strojenia segmentów wycofania. Umożliwia ona dobranie odpowiedniej liczby i rozmiarów segmentów wycofania, oraz prawidłowego doboru parametru OPTIMAL.

```
Przykład.

SQL> create public rollback segment R03 storage
  2   ( initial 40K next 40K minextents 5 maxextents 121) tablespace RBS;

Segment wycofania został utworzony.

SQL> select SEGMENT_NAME, TABLESPACE_NAME, INITIAL_EXTENT, NEXT_EXTENT,
  2         MIN_EXTENTS, MAX_EXTENTS, STATUS
  3   from dba_rollback_segs;

SEGMENT_NA TABLESPACE_NAM INITIAL_EXTENT NEXT_EXTENT MIN_EXTENTS MAX_EXTENTS STATUS
---------- -------------- -------------- ----------- ----------- ----------- -------
SYSTEM     SYSTEM                  53248       53248           2         249 ONLINE
R01        RBS                     40960       40960           5         121 ONLINE
R02        RBS                     40960       40960           5         121 OFFLINE
R03        RBS                     40960       40960           5         121 OFFLINE

SQL> alter rollback segment R03 online;

Segment wycofania został zmieniony.

SQL> select SEGMENT_NAME, TABLESPACE_NAME, INITIAL_EXTENT, NEXT_EXTENT,
  2         MIN_EXTENTS, MAX_EXTENTS, STATUS
  3   from dba_rollback_segs;

SEGMENT_NA TABLESPACE_NAM INITIAL_EXTENT NEXT_EXTENT MIN_EXTENTS MAX_EXTENTS STATUS
---------- -------------- -------------- ----------- ----------- ----------- -------
SYSTEM     SYSTEM                  53248       53248           2         249 ONLINE
R01        RBS                     40960       40960           5         121 ONLINE
R02        RBS                     40960       40960           5         121 OFFLINE
R03        RBS                     40960       40960           5         121 ONLINE
```

W przykładzie utworzono segment wycofania R03. Informacje o nim odczytano z perspektywy DBA_ROLLBACK_SEGS. Po utworzeniu segment ma status offline. Segment został udostępniony do użycia komendą **alter rollback segment**. Przestawiony status jest widoczny w perspektywie DBA_ROLLBACK_SEGS.

```
Przykład.

SQL> select NAME, EXTENTS, RSSIZE, WRITES, XACTS,
  2         HWMSIZE, SHRINKS, WRAPS, EXTENDS
  3   from v$rollname, v$rollstat
  4  where v$rollname.usn = v$rollstat.usn and NAME='R03';

NAME        EXTENTS     RSSIZE     WRITES  XACTS    HWMSIZE  SHRINKS  WRAPS    EXTENDS
---------- -------- ---------- ---------- ------ ---------- -------- ------ ----------
R03               5     200704       4214      0     200704        0      0          0

SQL> select NAME, EXTENTS, RSSIZE, WRITES, XACTS,
  2         HWMSIZE, SHRINKS, WRAPS, EXTENDS
  3   from v$rollname, v$rollstat
  4  where v$rollname.usn = v$rollstat.usn and NAME='R03';

NAME        EXTENTS     RSSIZE     WRITES  XACTS    HWMSIZE  SHRINKS  WRAPS    EXTENDS
```

```
---------- -------- ---------- ---------- ------ ---------- -------- ------ ----------
R03              22     897024     838448      1     897024        0     21         17

SQL> select NAME, EXTENTS, RSSIZE, WRITES, XACTS,
  2         HWMSIZE, SHRINKS, WRAPS, EXTENDS
  3   from v$rollname, v$rollstat
  4  where v$rollname.usn = v$rollstat.usn and NAME='R03';

NAME        EXTENTS     RSSIZE     WRITES  XACTS    HWMSIZE  SHRINKS  WRAPS    EXTENDS
---------- -------- ---------- ---------- ------ ---------- -------- ------ ----------
R03              22     897024     838448      0     897024        0     21         17

SQL> alter rollback segment R03 shrink to 20K;

SQL> select NAME, EXTENTS, RSSIZE, WRITES, XACTS,
  2         HWMSIZE, SHRINKS, WRAPS, EXTENDS
  3   from v$rollname, v$rollstat
  4  where v$rollname.usn = v$rollstat.usn and NAME='R03';


NAME        EXTENTS     RSSIZE     WRITES  XACTS    HWMSIZE  SHRINKS  WRAPS    EXTENDS
---------- -------- ---------- ---------- ------ ---------- -------- ------ ----------
R03               2      77824     848386      0     897024        2     21         17
```

W przykładzie obserwowany jest segment wycofania R03. Na początku segment jest nie używany, zaś jego rozmiar jest taki jak w chwili utworzenia. Po następnym odczytaniu stanu z perspektyw systemowych widać, że w segmencie jest prowadzona jedna transakcja (XACTS=1). Widać dodatkową alokację obszarów i wzrost rozmiaru segmentu. Po komendzie powodującej zmniejszenie segmentu (shrink) rozmiar ustala się na żądanym poziomie, zaś kolumna SHRINKS rejestruje wykonanie tej operacji.

```
Przykład.

SQL> alter rollback segment R03 offline;

Segment wycofania został zmieniony.

SQL> select NAME, EXTENTS, RSSIZE, WRITES, XACTS,
  2         HWMSIZE, SHRINKS, WRAPS, EXTENDS, STATUS
  3   from v$rollname, v$rollstat
  4  where v$rollname.usn = v$rollstat.usn and NAME='R03';

NAME  EXTENTS RSSIZE    WRITES  XACTS    HWMSIZE  SHRINKS  WRAPS    EXTENDS STATUS
---- -------- ------ --------- ------ ---------- -------- ------ ---------- ---------------
R03         22 897024   2526978      1     897024        0     21         17 PENDING OFFLINE

SQL> set transaction use rollback segment R03;
set transaction use rollback segment R03
*
BŁĄD w linii 1:
ORA-01634: segment wycofania o numerze 'R03' ma zmienić tryb na offline

SQL> alter rollback segment R03 offline;

Segment wycofania został zmieniony.

SQL> drop rollback segment R03;

Segment wycofania został usunięty.

SQL> select SEGMENT_NAME, TABLESPACE_NAME, INITIAL_EXTENT, NEXT_EXTENT,
  2         MIN_EXTENTS, MAX_EXTENTS, STATUS
  3   from dba_rollback_segs;

SEGMENT_NA TABLESPACE_NAM INITIAL_EXTENT NEXT_EXTENT MIN_EXTENTS MAX_EXTENTS STATUS
---------- -------------- -------------- ----------- ----------- ----------- -------
SYSTEM     SYSTEM                  53248       53248           2         249 ONLINE
R01        RBS                     40960       40960           5         121 ONLINE
R02        RBS                     40960       40960           5         121 OFFLINE
```

W przykładzie wykonano komendę przestawienia segmentu wycofania w tryb offline w trakcie obsługiwania przez niego transakcji. W rezultacie segment otrzymał status PENDING OFFLINE i niemożliwe już było obsługiwanie przez niego kolejnych transakcji co pokazano przy próbie jawnego przydziału transakcji. Po zakończeniu transakcji segment można było przestawić w tryb offline i

ostatecznie usunąć z bazy. Perspektywa DBA_ROLLBACK_SEGS nie zawiera już informacji o tym segmencie.

13. Śledzenie zmian w bazie danych

13.1 Obserwacja zmian w logach

Oracle dostarcza narzędzie **LogMiner** umożliwiające bezpośredni odczyt informacji zapisanych w plikach dziennika powtórzeń i archiwalnych plikach dziennika powtórzeń. Narzędzie to udostępniając relacyjne perspektywy zawierające szczegółowe informacje o zmianach w bazie danych może służyć między innymi do następujących celów:

- Śledzenie zmian wykonanych przez określonego użytkownika
- Zbieranie statystyk dostępu do określonych tabel w zadanym przedziale czasu
- Odczyt historycznych zapisów zmian w bazie
- Identyfikcja i naprawa logicznych uszkodzeń bazy

LogMiner opiera się o dwa pakiety:

- DBMS_LOGMNR_D – służy do tworzenia pliku pomocniczego słownika metadanych
- DBMS_LOGMNR – służy do określenia analizowanych logów oraz przedziału czasu lub SCN w ramach których odczytywane są informacje z logów

Pakiet DBMS_LOGMNR_D zawiera jedną procedurę:

- BUILD – tworzy plik słownika o nazwie *dictionary_filename* w lokalizacji *dictionary_location*

```
DBMS_LOGMNR_D.BUILD (
    dictionary_filename IN VARCHAR2,
    dictionary_location IN VARCHAR2);
```

Pakiet DBMS_LOGMNR zawiera procedury:

- ADD_LOGFILE – dodaje plik logu do listy logów podlegających odczytowi

```
DBMS_LOGMNR.ADD_LOGFILE(
     LogFileName IN VARCHAR2,
     Options IN BINARY_INTEGER default ADDFILE );
```

Parametry procedury:

  - LogFileName – nazwa pliku logu
  - Options – rodzaj akcji. Przyjmuje wartości:
    - DBMS_LOGMNR.NEW – utworzenie nowej listy
    - DBMS_LOGMNR.ADDFILE – dodanie nowego pliku do listy
    - DBMS_LOGMNR.REMOVEFILE – usunięcie pliku z listy

- START_LOGMNR – uruchomienie sesji LogMiner

```
DBMS_LOGMNR.START_LOGMNR(
     startScn IN NUMBER default 0,
     endScn IN NUMBER default 0,
     startTime IN DATE default '01-jan-1988',
     endTime IN DATE default '01-jan-2988',
     DictFileName IN VARCHAR2 default '',
     Options IN BINARY_INTEGER default 0 );
```

Parametry procedury:

  - startSCN – wartość początkowa SCN od której pozycje logu są analizowane
  - endSCN – wartość końcowa SCN od której pozycje logu są analizowane
  - startTime – wartość początkowa czasu od którego pozycje logu są analizowane
  - endTime – wartość końcowa czasu od którego pozycje logu są analizowane
  - DictFileName – pełna nazwa (ze ścieżką dostępu) pliku słownika
  - Options – opcje. Przyjmuje wartość:
    - DBMS_LOGMNR.USE_COLMAP – LogMiner używa mapowania kolumn opisanych w pliku logmnr.opt. Plik musi być w tej samej lokalizacji co plik słownika
    - DBMS_LOGMNR.SKIP_CORRUPTION – pomijanie uszkodzonych bloków

- END_LOGMNR – zatrzymanie sesji LogMiner

```
DBMS_LOGMNR.END_LOGMNR;
```

Informacje związane z LogMiner dostępne są poprzez perspektywy:

- V$LOGMNR_CONTENTS

- V$LOGMNR_DICTIONARY
- V$LOGMNR_LOGS
- V$LOGMNR_PARAMETERS

Uruchomienie LogMiner

Zapisy w logach zawierają odwołania do obiektów w bazie danych np. tabel poprzez wewnętrzne identyfikatory. Chcąc uzyskać nazwy tych obiektów należy wygenerować plik pomocniczy zawierający słownik systemowy. Należy pamiętać, że plik jest generowany z aktualnej zawartości słownika danych i jeżeli w chwili generacji nie zawierał informacji o jakimś historycznym obiekcie to zapisy w logach odnoszące się do niego będą dostępne tylko poprzez identyfikator wewnętrzny.

Aby uruchomić LogMiner należy wykonać następujące kroki:

1. Wygenerować plik meta danych procedurą DBMS_LOGMNR_D
2. Utworzyć nową listę logów przetwarzanych przez LogMiner procedurą DBMS_LOGMNR.ADD_LOGFILE z opcją NEW i dodać do niej pierwszy log
3. W razie potrzeby dodać do listy kolejne logi procedurą DBMS_LOGMNR.ADD_LOGFILE z opcją ADDFILE
4. Wystartować proces LogMinera procedurą DBMS_LOGMNR.START_LOGMNR. Wartości dostępnego przedziału czasu lub numerów SCN muszą zawierać się w logach umieszczonych na liście.
5. Po zakończeniu prac zatrzymać LogMiner procedurą DBMS_LOGMNR.END_LOGMNR

Dla prawidłowego działania **LogMiner** należy w pliku inicjalizacyjnym ustawić wartość parametru UTL_FILE_DIR na wartość katalogu zawierającego plik słownika.

Odczyt danych z logów możliwy jest za pośrednictwem instancji innej bazy niż ta, która wygenerowała logi. Muszą być jednak spełnione następujące warunki:

- Plik słownika musi być wygenerowany na podstawie bazy oryginalnej i z taką samą stroną kodową
- Rozmiary bloku bazy danych muszą być identyczne
- Obie bazy muszą pracować na jednakowej platformie sprzętowej

Parametry ustawione przez opisane procedury mogą być wyświetlone z perspektywy V$LOGMNR_PARAMETERS.

```
SQL> desc V$LOGMNR_PARAMETERS
 Nazwa                                     Wartość null? Typ
 ----------------------------------------- -------- ------------
 START_DATE                                         DATE
 END_DATE                                           DATE
 START_SCN                                          NUMBER
 END_SCN                                            NUMBER
 INFO                                               VARCHAR2(32)
 STATUS                                             NUMBER
```

Odczyt danych z logów

W trakcie działania procesu LogMiner dane z logu są dostępne poprzez perspektywę V$LOGMNR_CONTENTS. Przykładowe kolumny perspektywy:

- SCN – numer SCN
- TIMESTAMP – dokładny czas
- THREAD# - numer wątka generującego log
- LOG_ID – identyfikator logu
- ABS_FILE# - bezwzględny numer pliku zawierającego blok danych
- REL_FILE# - względny numer pliku zawierającego blok danych
- DATA_BLK# - numer bloku
- DATA_OBJ# - numer obiektu zwierającego blok
- SEG_OWNER – nazwa schematu zawierającego segment
- SEG_NAME – nazwa segmentu
- SEG_TYPE – typ segmentu
- SEG_TYPE_NAME – nazwa typu segmentu
- TABLE_SPACE – nazwa przestrzeni tabel zawierającej segment
- ROW_ID – identyfikator wiersza
- SESSION# - numer sesji (wchodzi w skład identyfikatora sesji)
- SERIAL# - numer seryjny (wchodzi w skład identyfikatora sesji)

- USERNAME – nazwa użytkownika
- SESSION_INFO – informacje o sesji
- ROLLBACK – segment wycofania
- OPERATION - operacja wykonana w bazie
- SQL_REDO – instrukcja wykonana w bazie
- SQL_UNDO – instrukcja odwracająca czynność wykonaną w bazie

Przykład.

Wykonanie operacji w bazie:

```
SQL> select to_char(sysdate,'DD-MM-YY HH24:MI:SS') from dual;

TO_CHAR(SYSDATE,'
-----------------
03-04-02 18:58:51

SQL> insert into test
  2    values
  3   ('ABCD');

1 wiersz został utworzony.

SQL> commit;

Zatwierdzanie zostało ukończone.

SQL> select to_char(sysdate,'DD-MM-YY HH24:MI:SS') from dual;

TO_CHAR(SYSDATE,'
-----------------
03-04-02 18:59:07
```

Identyfikacja logu:

```
SQL> select l.STATUS,f.MEMBER,TO_CHAR(l.FIRST_TIME,'DD-MM-YY HH24:MI:SS' ) "FIRST_TIME"
  2    from v$log l, v$logfile f
  3   where f.GROUP#=l.GROUP#;

STATUS   MEMBER                         FIRST_TIME
-------- ------------------------------ -----------------
INACTIVE /oracle/oradb/LOG1A.ORA        19-03-02 14:24:59
INACTIVE /oracle/oradb/LOG2A.ORA        21-03-02 09:27:55
INACTIVE /oracle/oradb/LOG3A.ORA        03-04-02 18:18:55
INACTIVE /oracle/oradb/LOG4A.ORA        03-04-02 18:51:57
CURRENT  /oracle/oradb/LOG5A.ORA        03-04-02 18:59:34
```

Uruchomienie LogMiner dla operacji w zadanym przedziale czasu:

```
SQL> execute sys.DBMS_LOGMNR.ADD_LOGFILE( LOGFILENAME => '/oracle/oradb/LOG4A.ORA', OPTIONS =>
sys.DBMS_LOGMNR.NEW );

Procedura PL/SQL została zakończona pomyślnie.

SQL> execute sys.DBMS_LOGMNR.START_LOGMNR( STARTTIME => TO_DATE('03-04-2002 18:58:51','DD-MM-
YYYY HH24:MI:SS'), ENDTIME   => TO_DATE('03-04-2002 18:59:07','DD-MM-YYYY HH24:MI:SS'),
DICTFILENAME => '/oracle/lgmnr/dictionary.ora', OPTIONS => sys.DBMS_LOGMNR.USE_COLMAP );

Procedura PL/SQL została zakończona pomyślnie.
```

Odczyt operacji wykonanych na tabeli TEST:

```
SQL> select OPERATION, SEG_NAME, SQL_REDO, SQL_UNDO, TO_CHAR(TIMESTAMP,'DD-MM-YY HH24:MI:SS')
  2    from V$LOGMNR_CONTENTS
  3   where SEG_NAME='TEST';

OPERATION                      SEG_NAME
------------------------------ --------------------------------
SQL_REDO
--------------------------------------------------------------------------------------------------
SQL_UNDO
--------------------------------------------------------------------------------------------------
TO_CHAR(TIMESTAMP
-----------------
INSERT                         TEST
insert into SYSTEM.TEST(COL1) values ('ABCD      ');
delete from SYSTEM.TEST where COL1 = 'ABCD      ' and ROWID = 'AAAA8DAAOAAAAAiAAA';
03-04-02 18:59:02
```

```
Zatrzymanie LogMiner:
SQL> execute sys.DBMS_LOGMNR.END_LOGMNR;

Procedura PL/SQL została zakończona pomyślnie.
```

W przykładzie wykonano wstawienie jednego wiersza do tabeli TEST. Dla celów poglądowych odczytano dokładny czas z serwera przed i po wykonaniu tej operacji. Następnie odszukano log zawierający zapisy zmian w bazie z tego przedziału czasu i uruchomiono **LogMiner** dla tego dziennika powtórzeń w interesującym nas przedziale czasu. W kolejnym kroku odczytano z perspektywy V$LOGMNR_CONTENTS wszystkie informacje dotyczące operacji wykonanych na tabeli TEST (SEG_NAME='TEST'). Ostatnim krokiem było zatrzymanie LogMiner.

13.2 Auditing

Auditing jest wbudowanym w bazę Oracle niskopoziomowym mechanizmem śledzenia i rejestrowania różnych operacji wykonywanych na bazie. Użytkownik posiadający odpowiedni przywilej AUDIT może określić zdarzenia w bazie, dla których ma zostać podjęta akcja rejestracji. Ponieważ mechanizm auditingu jest wbudowany w system obsługi bazy nie istnieje możliwość zatajenia wykonywanych operacji i uniknięcia ich rejestracji.

Zwykle auditingu używa się do:

- tropienia podejrzanej aktywności
- monitorowania aktywności bazy
- zbierania statystyk o aktywności bazy

Dane zebrane w procesie auditingu mogą być zapisywane w bazie danych Oracle lub w plikach systemu operacyjnego, przy użyciu wbudowanych w system operacyjny mechanizmów śledzenia. Druga opcja jest specyficzna dla każdego systemu operacyjnego serwera i nie będzie tu omawiana. O aktywności auditingu decyduje wartość parametru AUDIT_TRAIL w pliku konfiguracyjnym init.ora. Parametr ten jest statyczny, a zatem jego zmiana skutkuje dopiero po przeładowaniu bazy. Parametr ten może przyjąć następujące wartości:

- FALSE – auditing jest nieaktywny
- NONE – auditing jest nieaktywny, znaczenie parametru analogiczne jak FALSE
- DB – auditing jest aktywny, zapisy są wykonywane do bazy ORACLE
- TRUE – auditing jest aktywny, zapisy są wykonywane do bazy ORACLE, znaczenie parametru analogiczne jak DB
- OS – auditing jest aktywny, zapisy są wykonywane do plików śladu systemu operacyjnego

```
Przykład.
Parametr pliku init.ora

AUDIT_TRAIL = DB
```

Baza danych umożliwia aktywację auditingu, zapisy odbywają się do bazy danych.

Włączenie zapisu danych o zdarzeniach

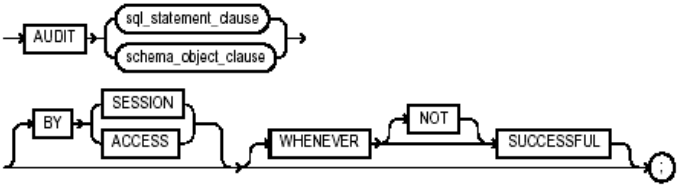
W bazie danych istnieje tabela w schemacie SYS o nazwie AUD$ przeznaczona do rejestracji zdarzeń na których założono auditing. Jeżeli wystąpi oczekiwane zdarzenie do tabeli dopisywany jest wiersz. Rejestracji podlegają zdarzenia niezależnie od efektu zakończenia transakcji użytkownika, a więc zarówno transakcje zatwierdzone jak i wycofane zostają zarejestrowane**.**

Auditingowi mogą podlegać zdarzenia systemowe lub operacje na obiektach użytkownika. Ogólnie obserwacje dzieli się na:

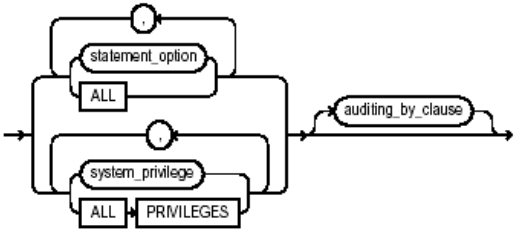
- obserwacje poleceń
- obserwacje przywilejów
- obserwacje obiektów

Obserwacja może dotyczyć wszystkich lub tylko wybranych użytkowników, nie jest wykonywana dla połączeń INTERNAL i SYS. Auditing zapisuje fakt wystąpienia obserwowanego zdarzenia, chcąc zapamiętać wartości zmienione w tablicach należy użyć specjalnie napisanych wyzwalaczy.

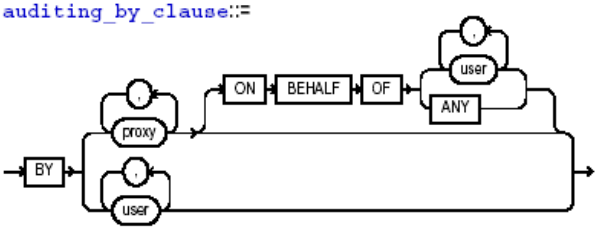
Ustawienie śledzenia operacji wykonuje się komendą **AUDIT**.



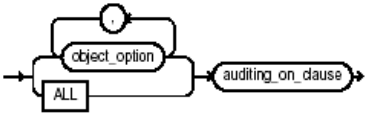
sql_statement_clause::=



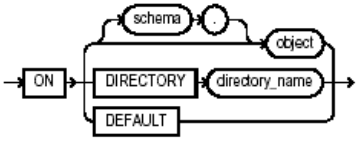
auditing_by_clause::=



schema_object_clause::=



auditing_on_clause::=



Znaczenie parametrów:

- BY SESSION - generacja jednego zapisu z każdego typu polecenia SQL w sesji (zakres czasu od połączenia do rozłączenia z bazą)
- BY ACCESS - generacja zapisu za każdym razem gdy wykonywana jest obserwowana akcja
- WHENEVER SUCCESSFUL – zapis wykonany tylko z udanych operacji. Pominięcie parametru oznacza zapis wszystkich operacji (udanych i nieudanych)
- WHENEVER NOT SUCCESSFUL – zapis wykonany tylko z operacji zakończonych błędem. Pominięcie parametru oznacza zapis wszystkich operacji (udanych i nieudanych)
- STATEMENT_OPTION – określa polecenie, które podlega śledzeniu
- ALL – włączenie śledzenia dla wszystkich poleceń

- SYSTEM_PRIVILEGE – określa śledzone polecenia SQL, które są wykonywane na mocy podanego przywileju systemowego lub roli.
- AUDITING_BY_CLAUSE – parametry określające wykonawcę śledzonych poleceń:
  - USER – śledzenie dotyczy określonego użytkownika
  - PROXY - śledzenie dotyczy wyłącznie poleceń wydanych przez określone proxy. Możliwe jest dalsze uszczegółowienie specyfikując:
    - ON BEHALF OF user - śledzenie dotyczy wyłącznie poleceń wykonanych na rzecz określonego użytkownika
    - ON BEHALF OF ANY - śledzenie dotyczy poleceń wykonanych na rzecz dowolnego użytkownika
- SCHEMA_OBJECT_CLAUSE – dotyczy śledzenia obiektów w schemacie:
  - OBJECT_OPTION – określa szczegółowo śledzone operacje w kontekście obiektów których dotyczą.
  - ALL – dotyczy wszystkich dopuszczalnych operacji na obiektach
- AUDITING_ON_CLAUSE – dotyczy specyfikacji obiektów w schemacie:
  - ON OBJECT – nazwa obiektu podlegającego obserwacji (tabela, perspektywa, sekwencja, składowana jednostka programu, zmaterializowana perspektywa, biblioteka). Parametr SCHEMA określa schemat w którym śledzone są obiekty. Brak specyfikacji oznacza określenie schematu bieżącego
  - ON DEFAULT – określa wyspecyfikowaną opcję jako domyślną dla tworzonych później obiektów. Umożliwia to śledzenie zmian w obiektach nie istniejących w bazie w chwili wydania komendy AUDIT.
  - ON DIRECTORY nazwa – określa katalog podlegający śledzeniu.

Poniższa tabela zawiera listę opcji określających obserwację zdarzeń systemowych oraz listę poleceń których dotyczą.

Opcja polecenia AUDIT	Obserwowana operacja lub polecenie
CLUSTER	CREATE CLUSTER AUDIT CLUSTER DROP CLUSTER TRUNCATE CLUSTER
CONTEXT	CREATE CONTEXT DROP CONTEXT
DATABASE LINK	CREATE DATABASE LINK DROP DATABASE LINK
DIMENSION	CREATE DIMENSION ALTER DIMENSION DROP DIMENSION
DIRECTORY	CREATE DIRECTORY DROP DIRECTORY
INDEX	CREATE INDEX ALTER INDEX DROP INDEX
NOT EXISTS	Wszystkie instrukcje kończące się błędem z powodu braku obiektu
PROCEDURE	CREATE FUNCTION CREATE LIBRARY CREATE PACKAGE CREATE PACKAGE BODY CREATE PROCEDURE DROP FUNCTION DROP LIBRARY DROP PACKAGE DROP PROCEDURE
PROFILE	CREATE PROFILE ALTER PROFILE DROP PROFILE
PUBLIC DATABASE LINK	CREATE PUBLIC DATABASE LINK DROP PUBLIC DATABASE LINK
PUBLIC SYNONYM	CREATE PUBLIC SYNONYM DROP PUBLIC SYNONYM
ROLE	CREATE ROLE ALTER ROLE DROP ROLE

	SET ROLE
ROLLBACK STATEMENT	CREATE ROLLBACK SEGMENT ALTER ROLLBACK SEGMENT DROP ROLLBACK SEGMENT
SEQUENCE	CREATE SEQUENCE DROP SEQUENCE
SESSION	Logowanie do bazy
SYNONYM	CREATE SYNONYM DROP SYNONYM
SYSTEM AUDIT	AUDIT sql_statements NOAUDIT sql_statements
SYSTEM GRANT	GRANT system_privileges_and_roles REVOKE system_privileges_and_roles
TABLE	CREATE TABLE DROP TABLE TRUNCATE TABLE
TABLESPACE	CREATE TABLESPACE ALTER TABLESPACE DROP TABLESPACE
TRIGGER	CREATE TRIGGER ALTER TRIGGER with ENABLE and DISABLE clauses DROP TRIGGER ALTER TABLE with ENABLE ALL TRIGGERS clause and DISABLE ALL TRIGGERS clause
TYPE	CREATE TYPE CREATE TYPE BODY ALTER TYPE DROP TYPE DROP TYPE BODY
USER	CREATE USER ALTER USER DROP USER
VIEW	CREATE VIEW DROP VIEW
ALTER SEQUENCE	ALTER SEQUENCE
ALTER TABLE	ALTER TABLE
COMMENT TABLE	COMMENT ON TABLE table, view, materialized view COMMENT ON COLUMN table.column, view.column, materialized view.column
DELETE TABLE	DELETE FROM table DELETE FROM view
EXECUTE PROCEDURE	CALL Wywołanie dowolnej procedury lub funkcji. Dostęp do dowolnej zmiennej, biblioteki lub kursora w pakiecie.
GRANT DIRECTORY	GRANT privilege ON directory REVOKE privilege ON directory
GRANT PROCEDURE	GRANT privilege ON procedure, function, package REVOKE privilege ON procedure, function, Package
GRANT SEQUENCE	GRANT privilege ON sequence REVOKE privilege ON sequence
GRANT TABLE	GRANT privilege ON table, view, materialized view. REVOKE privilege ON table, view, materialized view
GRANT TYPE	GRANT privilege ON TYPE REVOKE privilege ON TYPE
INSERT TABLE	INSERT INTO table INSERT INTO view
LOCK TABLE	LOCK TABLE table LOCK TABLE view
SELECT SEQUENCE	Dowolna instrukcja zawierająca funkcje: sequence.CURRVAL lub sequence.NEXTVAL

SELECT TABLE	SELECT FROM table SELECT FROM view SELECT FROM materialized view
UPDATE TABLE	UPDATE table UPDATE view

Poniższe opcje dotyczą ustawienia obserwacji na obiektach:

- ALTER
- AUDIT
- COMMENT
- DELETE
- EXECUTE
- GRANT
- INDEX
- INSERT
- LOCK
- READ
- RENAME
- SELECT
- UPDATE

Mają one zastosowanie odpowiednio do zakresu działania polecenia np. **SELECT** odnosi się do tabel, perspektyw, zmaterializowanych perspektyw i sekwencji, READ do katalogów (DIRECTORY) itp.

Informacje o ustawionych opcjach śledzenia dostępne są poprzez perspektywy: DBA_STMT_AUDIT_OPTS, DBA_PRIV_AUDIT_OPTS, DBA_OBJ_AUDIT_OPTS.

Perspektywa DBA_STMT_AUDIT_OPTS – zawiera informacje o ustawionych obserwacjach zdarzeń systemowych. Kolumny:

- USER_NAME – nazwa obserwowanego użytkownika. Kolumna ma wartość NULL jeśli obserwacja dotyczy wszystkich użytkowników lub ANY CLIENT jeśli dotyczy dostępu przez proxy
- PROXY_NAME – nazwa użytkownika proxy wykonującego operację dla innych klientów. Kolumna ma wartość NULL gdy klient wykonuje operację bezpośrednio.
- AUDIT_OPTION – nazwa obserwowanego zdarzenia
- SUCCESS – tryb obserwacji (by access/by session) dla ustawionego warunku WHENEVER SUCCESSFUL
- FAILURE - tryb obserwacji (by access/by session) dla ustawionego warunku WHENEVER NOT SUCCESSFUL

Perspektywa DBA_PRIV_AUDIT_OPTS – zawiera informacje o ustawionych obserwacjach przywilejów systemowych. Kolumny:

- USER_NAME – nazwa obserwowanego użytkownika. Kolumna ma wartość NULL jeśli obserwacja dotyczy wszystkich użytkowników lub ANY CLIENT jeśli dotyczy dostępu przez proxy
- PROXY_NAME – nazwa użytkownika proxy wykonującego operację dla innych klientów. Kolumna ma wartość NULL gdy klient wykonuje operację bezpośrednio.
- PRIVILEGE – nazwa obserwowanego przywileju systemowego
- SUCCESS – tryb obserwacji (by access/by session) dla ustawionego warunku WHENEVER SUCCESSFUL
- FAILURE - tryb obserwacji (by access/by session) dla ustawionego warunku WHENEVER NOT SUCCESSFUL

Perspektywa DBA_OBJ_AUDIT_OPTS – zawiera informacje o obserwacjach ustawionych na obiektach bazy danych. Kolumny:

- OWNER – właściciel obserwowanego obiektu
- OBJECT_NAME – nazwa obiektu
- OBJECT_TYPE – typ obiektu
- ALT – obserwacja polecenia alter wykonanego na obiekcie
- AUD – obserwacja polecenia audit wykonanego na obiekcie

- COM – obserwacja polecenia comment wykonanego na obiekcie
- DEL – obserwacja polecenia delete wykonanego na obiekcie
- GRA – obserwacja polecenia grant wykonanego na obiekcie
- IND – obserwacja polecenia index wykonanego na obiekcie
- INS – obserwacja polecenia insert wykonanego na obiekcie
- LOC – obserwacja polecenia lock wykonanego na obiekcie
- REN – obserwacja polecenia rename wykonanego na obiekcie
- SEL – obserwacja polecenia select wykonanego na obiekcie
- UPD – obserwacja polecenia update wykonanego na obiekcie
- REF – obserwacja polecenia reference wykonanego na obiekcie
- EXE – obserwacja polecenia execute wykonanego na obiekcie
- CRE – obserwacja polecenia create wykonanego na obiekcie
- REA – obserwacja polecenia read wykonanego na obiekcie
- WRI – obserwacja polecenia write wykonanego na obiekcie

Kolumny szczegółów ALT .. WRI zawierają wartość -/- gdy obserwacja nie jest włączona lub odpowiedni atrybut:

- A – gdy auditing dotyczy pojedynczego dostępu (by access)
- S - gdy auditing dotyczy dostępu w sesji (by session)

Pierwsza wartość oznacza ustawienie obserwacji dla operacji udanych (whenever successful), druga dla nieudanych (whenever not successful). Przykładowo wartość kolumny UPD='A/S' dla określonej tabeli oznacza, że rejestrowane są wszystkie udane próby modyfikacji tabeli, oraz rejestrowana nieudana próba modyfikacji tabeli w sesji.

Perspektywa ALL_DEF_AUDIT_OPTS – zawiera informacje o domyślnych obserwacjach ustawionych na obiektach bazy danych. Wartości domyślne zostaną ustawione na obiektach w chwili ich utworzenia. Kolumny:

- ALT – obserwacja polecenia alter wykonanego na obiekcie
- AUD – obserwacja polecenia audit wykonanego na obiekcie
- COM – obserwacja polecenia comment wykonanego na obiekcie
- DEL – obserwacja polecenia delete wykonanego na obiekcie
- GRA – obserwacja polecenia grant wykonanego na obiekcie
- IND – obserwacja polecenia index wykonanego na obiekcie
- INS – obserwacja polecenia insert wykonanego na obiekcie
- LOC – obserwacja polecenia lock wykonanego na obiekcie
- REN – obserwacja polecenia rename wykonanego na obiekcie
- SEL – obserwacja polecenia select wykonanego na obiekcie
- UPD – obserwacja polecenia update wykonanego na obiekcie
- REF – obserwacja polecenia reference wykonanego na obiekcie
- EXE – obserwacja polecenia execute wykonanego na obiekcie

Kolumny szczegółów ALT...WRI zawierają wartość -/- gdy obserwacja nie jest włączona lub odpowiedni atrybut:

- A – gdy auditing dotyczy pojedynczego dostępu (by access)
- S - gdy auditing dotyczy dostępu w sesji (by session)

Pierwsza wartość oznacza ustawienie obserwacji dla operacji udanych (whenever successful), druga dla nieudanych (whenever not successful). Przykładowo wartość kolumny UPD='A/S' oznacza, że po utworzeniu nowej tabeli rejestrowane będą wszystkie udane próby modyfikacji tabeli, oraz rejestrowana nieudana próba modyfikacji tabeli w sesji.

```
Przykład.

SQL>  audit connect whenever not successful;

Obserwowanie zostało włączone.

SQL> audit system grant by bs by session;

Obserwowanie zostało włączone.

SQL> audit delete table by bs by session whenever successful;
```

```
Obserwowanie zostało włączone.

SQL> select USER_NAME, PROXY_NAME, AUDIT_OPTION, SUCCESS, FAILURE
  2   from dba_stmt_audit_opts;

USER_NAME       PROXY_NAME      AUDIT_OPTION                    SUCCESS    FAILURE
--------------- --------------- ------------------------------- ---------- ----------
                                CREATE SESSION                  NOT SET    BY ACCESS
BS                              SYSTEM GRANT                    BY ACCESS  BY ACCESS
BS                              DELETE TABLE                    BY SESSION NOT SET

SQL> select USER_NAME, PROXY_NAME, PRIVILEGE, SUCCESS, FAILURE
  2   from dba_priv_audit_opts;

USER_NAME       PROXY_NAME      PRIVILEGE                       SUCCESS    FAILURE
--------------- --------------- ------------------------------- ---------- ---------
                                CREATE SESSION                  NOT SET    BY ACCESS
```

W przykładzie włączono auditing dla nieudanych połączeń do bazy (każde zdarzenie rejestrowane), dla zmian przywilejów systemowych wykonanych przez użytkownika BS (każde zdarzenie rejestrowane) oraz dla udanego usuwania wierszy przez użytkownika BS z dowolnej tabeli (jedna rejestracja dla sesji). Ustawione obserwacje są widoczne w perspektywie DBA_STMT_AUDIT_OPTS. Perspektywa DBA_PRIV_AUDIT_OPTS pokazuje również auditing ustawiony na przywileju

```
Przykład.

SQL> audit delete on BS.TEST by access whenever successful;

Obserwowanie zostało włączone.

SQL> select OWNER,OBJECT_NAME,OBJECT_TYPE,
  2         ALT,AUD,COM,DEL,GRA,IND,INS,LOC,REN,SEL,UPD,REF,EXE,CRE,REA,WRI
  3   from dba_obj_audit_opts
  4   where OBJECT_NAME='TEST' and OWNER='BS';

OWNER OBJECT_NAM OBJECT_TYP ALT AUD COM DEL GRA IND INS LOC REN SEL UPD REF EXE CRE REA WRI
----- ---------- ---------- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
BS    TEST       TABLE      -/- -/- -/- A/- -/- -/- -/- -/- -/- -/- -/- -/- -/- -/- -/- -/-
```

W przykładzie ustawiono obserwację dla wszystkich udanych operacji usuwania wierszy z tabeli TEST. Rejestracji podlega każda taka operacja. Ustawienie auditingu zostało potwierdzone zapytaniem skierowanym do perspektywy systemowej DBA_OBJ_AUDIT_OPTS.

```
Przykład.

SQL> audit execute on BS.WORK30 by session;

Obserwowanie zostało włączone.

SQL> select OWNER,OBJECT_NAME,OBJECT_TYPE,
  2         ALT,AUD,COM,DEL,GRA,IND,INS,LOC,REN,SEL,UPD,REF,EXE,CRE,REA,WRI
  3   from dba_obj_audit_opts
  4   where OBJECT_NAME='WORK30' and OWNER='BS';

OWNER OBJECT_NAM OBJECT_TYP ALT AUD COM DEL GRA IND INS LOC REN SEL UPD REF EXE CRE REA WRI
----- ---------- ---------- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
BS    WORK30     PROCEDURE  -/- -/- -/- -/- -/- -/- -/- -/- -/- -/- -/- -/- -/- S/S -/- -/- -/-
```

W przykładzie ustawiono obserwację dla udanych i nieudanych wywołań procedury WORK30. Rejestracji podlega wystąpienie zdarzenia w sesji. Ustawienie auditingu zostało potwierdzone zapytaniem skierowanym do perspektywy systemowej DBA_OBJ_AUDIT_OPTS.

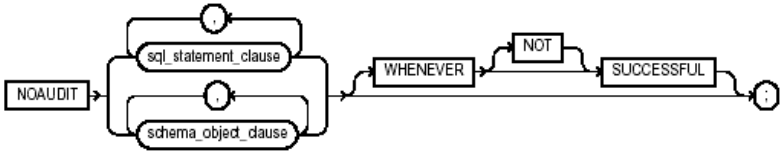
```
Przykład.

SQL> audit update on default;

Obserwowanie zostało włączone.

SQL> select ALT,AUD,COM,DEL,GRA,IND,INS,LOC,REN,SEL,UPD,REF,EXE
  2     from all_def_audit_opts;

ALT AUD COM DEL GRA IND INS LOC REN SEL UPD REF EXE
--- --- --- --- --- --- --- --- --- --- --- --- ---
```

```
-/- -/- -/- -/- -/- -/- -/- -/- -/- -/- S/S -/- -/-

SQL> create table NEW_TABLE
  2    ( col1 number );

Tabela została utworzona.

SQL> select OWNER,OBJECT_NAME,OBJECT_TYPE,
  2         ALT,AUD,COM,DEL,GRA,IND,INS,LOC,REN,SEL,UPD,REF,EXE,CRE,REA,WRI
  3    from dba_obj_audit_opts
  4   where OBJECT_NAME = 'NEW_TABLE';

OWNER OBJECT_NAM OBJECT_TYP ALT AUD COM DEL GRA IND INS LOC REN SEL UPD REF EXE CRE REA WRI
----- ---------- ---------- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
BS    NEW_TABLE  TABLE      -/- -/- -/- -/- -/- -/- -/- -/- -/- -/- S/S -/- -/- -/- -/- -/-
```

W przykładzie włączono auditing dla wszystkich nowo utworzonych tabel. Będą dla nich rejestrowane operacje update. Potwierdza to odczyt z perspektywy DBA_OBJ_AUDIT_OPTS po utworzeniu nowej tabeli.

Wyłączenie zapisu danych o zdarzeniach

Odwołanie obserwacji odbywa się komenda **NOAUDIT**. Składnia opcji jest analogiczna jak dla komendy **AUDIT**. Należy zauważyć, że w komendzie **NOAUDIT** nie występuje opcja BY ACCESS / BY SESSION, co oznacza, że auditing zostaje wyłączony niezależnie od wariantu włączenia (dla sesji lub dla dostępu)




```
Przykład.

SQL> select USER_NAME, PROXY_NAME, AUDIT_OPTION, SUCCESS, FAILURE
  2   from dba_stmt_audit_opts;

USER_NAME       PROXY_NAME      AUDIT_OPTION                   SUCCESS    FAILURE
--------------- --------------- ------------------------------ ---------- ----------
                                CREATE SESSION                 NOT SET    BY ACCESS
BS                              SYSTEM GRANT                   BY ACCESS  BY ACCESS
BS                              DELETE TABLE                   BY SESSION NOT SET

SQL> noaudit connect whenever not successful;

Obserwowanie zostało wyłączone.

SQL> noaudit system grant by bs;

Obserwowanie zostało wyłączone.

SQL> noaudit delete table by bs whenever successful;

Obserwowanie zostało wyłączone.

SQL> select USER_NAME, PROXY_NAME, AUDIT_OPTION, SUCCESS, FAILURE
  2   from dba_stmt_audit_opts;

nie wybrano żadnych wierszy
```

W przykładzie wyłączono auditing na wszystkich zdarzeniach, na których była ustawiona obserwacja w poprzednim przykładzie. Operacje zweryfikowano odczytem z perspektywy DBA_STMT_AUDIT_OPTS.

```
Przykład.

SQL> select OWNER,OBJECT_NAME,OBJECT_TYPE,
  2         ALT,AUD,COM,DEL,GRA,IND,INS,LOC,REN,SEL,UPD,REF,EXE,CRE,REA,WRI
```

```
  3     from dba_obj_audit_opts
  4   where OWNER='BS';

OWNER OBJECT_NAM OBJECT_TYP ALT AUD COM DEL GRA IND INS LOC REN SEL UPD REF EXE CRE REA WRI
----- ---------- ---------- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
BS    TEST       TABLE      -/- -/- -/- A/- -/- -/- -/- -/- -/- -/- -/- -/- -/- -/- -/- -/-
BS    WORK30     PROCEDURE  -/- -/- -/- -/- -/- -/- -/- -/- -/- -/- -/- -/- -/- S/S -/- -/- -/-

SQL> noaudit all on BS.TEST;

Obserwowanie zostało wyłączone.

SQL> noaudit execute on BS.WORK30;

Obserwowanie zostało wyłączone.

SQL> select OWNER,OBJECT_NAME,OBJECT_TYPE,
  2         ALT,AUD,COM,DEL,GRA,IND,INS,LOC,REN,SEL,UPD,REF,EXE,CRE,REA,WRI
  3     from dba_obj_audit_opts
  4   where OWNER='BS';

OWNER OBJECT_NAM OBJECT_TYP ALT AUD COM DEL GRA IND INS LOC REN SEL UPD REF EXE CRE REA WRI
----- ---------- ---------- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
BS    TEST       TABLE      -/- -/- -/- -/- -/- -/- -/- -/- -/- -/- -/- -/- -/- -/- -/- -/-
BS    WORK30     PROCEDURE  -/- -/- -/- -/- -/- -/- -/- -/- -/- -/- -/- -/- -/- -/- -/- -/-
```

W przykładzie wyłączono auditing na obiektach, na których była ustawiona obserwacja w poprzednim przykładzie. Operacje zweryfikowano odczytem z perspektywy DBA_OBJ_AUDIT_OPTS.

Możliwe jest szybkie wyłączenie wszystkich ustawionych obserwacji na poleceniach lub przywilejach systemowych, a także na określonym obiekcie tak jak w poniższym przykładzie.

```
Przykład.

SQL> noaudit all;

Obserwowanie zostało wyłączone.

SQL> noaudit all privileges;

Obserwowanie zostało wyłączone.

SQL> noaudit all on BS.TEST;

Obserwowanie zostało wyłączone.
```

W przykładzie pokazano warianty szybkiego wyłączania obserwacji.

Odczyt informacji z tabeli auditingu

Zapisy dotyczące wykonanych operacji, na których zostały ustawione obserwacje są wykonywane do tabeli SYS.AUD$. W praktycznym użyciu wygodniejsze jest korzystanie z następujących perspektyw opartych na tabeli AUD$: DBA_AUDIT_TRAIL, DBA_AUDIT_OBJECT, DBA_AUDIT_SESSION, DBA_AUDIT_STATEMENT, DBA_AUDIT_EXISTS.

Perspektywa DBA_AUDIT_TRAIL – zawiera informacje o wszystkich zapisach w tabeli AUD$. Kolumny:

- OS_USERNAME – identyfikator użytkownika (w systemie operacyjnym), którego akcje są obserwowane
- USERNAME - identyfikator użytkownika (w bazie danych), którego akcje są obserwowane
- USERHOST – identyfikator instancji dla instancji Oracle, z której użytkownik miał dostęp do bazy
- TERMINAL – identyfikator terminala użytkownika
- TIMESTAMP – dokładny zapis czasu wykonania operacji
- OWNER – właściciel obiektu którego dotyczyła operacja
- OBJ_NAME - nazwa obiektu którego dotyczyła operacja
- ACTION – kod wykonanej akcji
- ACTION_NAME – nazwa wykonanej akcji
- NEW_OWNER – nowy właściciel obiektu (dotyczy komendy REANAME)
- NEW_NAME – nowa nazwa obiektu (dotyczy komendy REANAME)

- OBJ_PRIVILEGE – przywilej obiektowy nadany komendą GRANT lub odebrany komendą REVOKE
- SYS_PRIVILEGE - przywilej systemowy nadany komendą GRANT lub odebrany komendą REVOKE
- ADMIN_OPTION – wskazuje, że rola lub przywilej systemowy został nadany z opcją ADMIN
- GRANTEE – nazwa użytkownika lub roli, której nadano lub odebrano przywilej
- AUDIT_OPTION – opcja auditingu ustawiona poleceniem audit
- SES_ACTIONS – podsumowanie sesji dla obserwacji ustawionych z opcją BY SESSION. Kolumna jest 16-znakowym łańcuchem którego kolejne znaki (- brak akcji, S – udane akcje, F – nieudane akcje, B – udane i nieudane akcje) oznaczają następujące operacje:
  - ALTER
  - AUDIT
  - COMMENT
  - DELETE
  - GRANT
  - INDEX
  - INSERT
  - LOCK
  - RENAME
  - SELECT
  - UPDATE
  - REFERENCES
  - EXECUTE
  - Zarezerwowane dla rozwoju systemu
  - Zarezerwowane dla rozwoju systemu
  - Zarezerwowane dla rozwoju systemu
- LOGOFF_TIME – czas odłączenia użytkownika od bazy
- LOGOFF_LREAD – logiczne odczyty w sesji
- LOGOFF_PREAD – fizyczne odczyty w sesji
- LOGOFF_LWRITE – logiczne zapisy w sesji
- LOGOFF_DLOCK – wykryte zakleszczenia (deadlock) w sesji
- COMMENT_TEXT – komentarz zawierający dodatkowe informacje o obserwowanej sesji. Kolumna zawiera również informacje o sposobie autentykacji połączenia do bazy – poprzez hasło w bazie, poprzez SQL*Net (opcję Advanced Security), lub poprzez proxy.
- SESSIONID – identyfikator sesji Oracle
- ENTRYID – identyfikator każdej pozycji zapisu auditingu w sesji
- STATEMENTID – identyfikator wydanej komendy
- RETURNCODE – kod komunikatu serwera Oracle po wykonanej akcji (0 – akcja zakończona powodzeniem)
- PRIV_USED – przywilej systemowy użyty do wykonania akcji

Perspektywa DBA_AUDIT_OBJECT – zawiera zapisy auditingu dotyczące wszystkich obiektów w systemie. Kolumny:

- OS_USERNAME – identyfikator użytkownika (w systemie operacyjnym), którego akcje są obserwowane
- USERNAME - identyfikator użytkownika (w bazie danych), którego akcje są obserwowane
- USERHOST – identyfikator instancji dla instancji Oracle, z której użytkownik miał dostęp do bazy
- TERMINAL – identyfikator terminala użytkownika
- TIMESTAMP – dokładny zapis czasu wykonania operacji
- OWNER – właściciel obiektu którego dotyczyła operacja
- OBJ_NAME - nazwa obiektu którego dotyczyła operacja
- ACTION_NAME – nazwa wykonanej akcji
- NEW_OWNER – nowy właściciel obiektu (dotyczy komendy REANAME)
- NEW_NAME – nowa nazwa obiektu (dotyczy komendy REANAME)
- SES_ACTIONS – podsumowanie sesji dla obserwacji ustawionych z opcją BY SESSION. Kolumna jest 16-znakowym łańcuchem którego kolejne znaki (- brak akcji, S – udane akcje, F – nieudane akcje, B – udane i nieudane akcje) oznaczają następujące operacje:
  - ALTER

    - AUDIT
    - COMMENT
    - DELETE
    - GRANT
    - INDEX
    - INSERT
    - LOCK
    - RENAME
    - SELECT
    - UPDATE
    - REFERENCES
    - EXECUTE
    - Zarezerwowane dla rozwoju systemu
    - Zarezerwowane dla rozwoju systemu
    - Zarezerwowane dla rozwoju systemu
- COMMENT_TEXT – komentarz zawierający dodatkowe informacje o obserwowanej sesji. Kolumna zawiera również informacje o sposobie autentykacji połączenia do bazy – poprzez hasło w bazie, poprzez SQL*Net (opcję Advanced Security), lub poprzez proxy.
- SESSIONID – identyfikator sesji Oracle
- ENTRYID – identyfikator każdej pozycji zapisu auditingu w sesji
- STATEMENTID – identyfikator wydanej komendy
- RETURNCODE – kod komunikatu serwera Oracle po wykonanej akcji (0 – akcja zakończona powodzeniem)
- PRIV_USED – przywilej systemowy użyty do wykonania akcji

Perspektywa DBA_AUDIT_SESSION – zawiera informacje o wszystkich zapisach w tabeli AUD$ związanych z połączeniami i odłączeniami od bazy (polecenia connect i disconnect). Kolumny:

- OS_USERNAME – identyfikator użytkownika (w systemie operacyjnym), którego akcje są obserwowane
- USERNAME - identyfikator użytkownika (w bazie danych), którego akcje są obserwowane
- USERHOST – identyfikator instancji dla instancji Oracle, z której użytkownik miał dostęp do bazy
- TERMINAL – identyfikator terminala użytkownika
- TIMESTAMP – dokładny zapis czasu wykonania operacji
- ACTION_NAME – nazwa wykonanej akcji
- LOGOFF_TIME – czas odłączenia użytkownika od bazy
- LOGOFF_LREAD – logiczne odczyty w sesji
- LOGOFF_PREAD – fizyczne odczyty w sesji
- LOGOFF_LWRITE – logiczne zapisy w sesji
- LOGOFF_DLOCK – wykryte zakleszczenia (deadlock) w sesji
- SESSIONID – identyfikator sesji Oracle
- RETURNCODE – kod komunikatu serwera Oracle po wykonanej akcji (0 – akcja zakończona powodzeniem)

Perspektywa DBA_AUDIT_STATEMENT – zawiera informacje o wszystkich zapisach w tabeli AUD$ dotyczących wykonanych operacji: GRANT, REVOKE, AUDIT, NOAUDIT, ALTER SYSTEM. Kolumny:

- OS_USERNAME – identyfikator użytkownika (w systemie operacyjnym), którego akcje są obserwowane
- USERNAME - identyfikator użytkownika (w bazie danych), którego akcje są obserwowane
- USERHOST – identyfikator instancji dla instancji Oracle, z której użytkownik miał dostęp do bazy
- TERMINAL – identyfikator terminala użytkownika
- TIMESTAMP – dokładny zapis czasu wykonania operacji
- OWNER – właściciel obiektu którego dotyczyła operacja
- OBJ_NAME - nazwa obiektu którego dotyczyła operacja
- ACTION_NAME – nazwa wykonanej akcji
- NEW_NAME – nowa nazwa obiektu (dotyczy komendy REANAME)
- OBJ_PRIVILEGE – przywilej obiektowy nadany komendą GRANT lub odebrany komendą REVOKE

- SYS_PRIVILEGE - przywilej systemowy nadany komendą GRANT lub odebrany komendą REVOKE
- ADMIN_OPTION – wskazuje, że rola lub przywilej systemowy został nadany z opcją ADMIN
- GRANTEE – nazwa użytkownika lub roli, której nadano lub odebrano przywilej
- AUDIT_OPTION – opcja auditingu ustawiona poleceniem **audit**
- SES_ACTIONS – podsumowanie sesji dla obserwacji ustawionych z opcją BY SESSION. Kolumna jest 16-znakowym łańcuchem którego kolejne znaki (- brak akcji, S – udane akcje, F – nieudane akcje, B – udane i nieudane akcje) oznaczają następujące operacje:
  - ALTER
  - AUDIT
  - COMMENT
  - DELETE
  - GRANT
  - INDEX
  - INSERT
  - LOCK
  - RENAME
  - SELECT
  - UPDATE
  - REFERENCES
  - EXECUTE
  - Zarezerwowane dla rozwoju systemu
  - Zarezerwowane dla rozwoju systemu
  - Zarezerwowane dla rozwoju systemu
- COMMENT_TEXT – komentarz zawierający dodatkowe informacje o obserwowanej sesji. Kolumna zawiera również informacje o sposobie autentykacji połączenia do bazy – poprzez hasło w bazie, poprzez NET8 (opcję Advanced Security), lub poprzez proxy.
- SESSIONID – identyfikator sesji Oracle
- ENTRYID – identyfikator każdej pozycji zapisu auditingu w sesji
- STATEMENTID – identyfikator wydanej komendy
- RETURNCODE – kod komunikatu serwera Oracle po wykonanej akcji (0 – akcja zakończona powodzeniem)
- PRIV_USED – przywilej systemowy użyty do wykonania akcji

Perspektywa DBA_AUDIT_EXISTS – zawiera informacje o wszystkich zapisach w tabeli AUD$ wynikłych z faktu nie istnienia obiektu, którego dotyczyło zapytanie. Perspektywa zawiera wiersze, dla których kod komunikatu serwera miał jedną z następujących wartości: 942, 943, 959, 1418, 1432, 1434, 1435, 1534, 1917, 1918, 1919, 2019,2024, 2289,4042, 4043, 4080, 1, 951, 955, 957, 1430, 1433, 1452, 1471, 1535, 1543,1758, 1920, 1921, 1922, 2239, 2264, 2266, 2273, 2292, 2297, 2378, 2379,2382, 4081, 12006, 12325 (np. 942 – table or view does not exist). Kolumny:

- OS_USERNAME – identyfikator użytkownika (w systemie operacyjnym), którego akcje są obserwowane
- USERNAME - identyfikator użytkownika (w bazie danych), którego akcje są obserwowane
- USERHOST – identyfikator instancji dla instancji Oracle, z której użytkownik miał dostęp do bazy
- TERMINAL – identyfikator terminala użytkownika
- TIMESTAMP – dokładny zapis czasu wykonania operacji
- OWNER – właściciel obiektu którego dotyczyła operacja
- OBJ_NAME - nazwa obiektu którego dotyczyła operacja
- ACTION_NAME – nazwa wykonanej akcji
- NEW_OWNER – nowy właściciel obiektu (dotyczy komendy REANAME)
- NEW_NAME – nowa nazwa obiektu (dotyczy komendy REANAME)
- OBJ_PRIVILEGE – przywilej obiektowy nadany komendą GRANT lub odebrany komendą REVOKE
- SYS_PRIVILEGE - przywilej systemowy nadany komendą GRANT lub odebrany komendą REVOKE
- GRANTEE – nazwa użytkownika lub roli, której nadano lub odebrano przywilej
- SESSIONID – identyfikator sesji Oracle
- ENTRYID – identyfikator każdej pozycji zapisu auditingu w sesji

- STATEMENTID – identyfikator wydanej komendy
- RETURNCODE – kod komunikatu serwera Oracle po wykonanej akcji (0 – akcja zakończona powodzeniem)

```
Przykład.

SQL> audit connect whenever not successful;

Obserwowanie zostało włączone.

SQL> connect bs/xxxxx
ERROR:
ORA-01017: invalid username/password; logon denied

SQL> select USERNAME, ACTION_NAME,
  2         To_Char(TIMESTAMP,'DD/MM/YY HH24:MI:SS' ) "CZAS", SES_ACTIONS, RETURNCODE
  3   from dba_audit_trail
  4   order by CZAS;

USERNAME   ACTION_NAME          CZAS              SES_ACTIONS      RETURNCODE
---------- -------------------- ----------------- ---------------- ----------
BS         LOGON                18/08/02 13:38:05                       1017
```

W przykładzie ustawiono obserwację na nieudane logowanie np. w celu rejestrowania prób nieuprawnionego łączenia do bazy. Następnie podjęto próbę dołączenia do bazy z niewłaściwym hasłem. Zdarzenie zostało zarejestrowane w tabeli sys.aud$ I odczytane poprzez perspektywę DBA_AUDIT_TRAIL. Kod błędu ORA-01017 został również zapisany do tabeli auditingu.

```
Przykład.

SQL> audit system grant by bs by session;

Obserwowanie zostało włączone.

Komendy wykonane przez użytkownika BS:
SQL> grant select any table to test;

Przyznanie uprawnień zakończone powodzeniem.

SQL> grant unlimited tablespace to test;

Przyznanie uprawnień zakończone powodzeniem.

SQL> select USERNAME, ACTION_NAME,
  2         To_Char(TIMESTAMP,'DD/MM/YY HH24:MI:SS' ) "CZAS",
  3         GRANTEE, SYS_PRIVILEGE
  4    from dba_audit_trail
  5   order by CZAS;

USERNAME   ACTION_NAME          CZAS              GRANTEE  SYS_PRIVILEGE
---------- -------------------- ----------------- -------- --------------------
BS         SYSTEM GRANT         18/08/02 13:54:20 TEST     SELECT ANY TABLE
BS         SYSTEM GRANT         18/08/02 14:00:39 TEST     UNLIMITED TABLESPACE
```

W przykładzie ustawiono obserwację na nadawanie uprawnień systemowych przez użytkownika BS. Wykonane przez niego komendy **grant** zostały zarejestrowane w tabeli auditingu. Przy pomocy perspektywy DBA_AUDIT_TRAIL odczytano czas wykonania komend oraz rodzaj przywilejów nadanych użytkownikowi TEST.

```
Przykład.

SQL> audit update on BS.TEST1 by access whenever successful;

Obserwowanie zostało włączone.

Komendy wykonane przez użytkownika BS:
SQL> update TEST1
  2    set col1=10
  3   where col1=1;

1 wiersz został zmodyfikowany.

SQL> update TEST1
  2    set col1=0
  3   where col1=1;
```

```
0 wierszy zostało zmodyfikowanych.

SQL> update TEST1
  2    set col1=0
  3   where col1=2;
update TEST1
*
BŁĄD w linii 1:
ORA-02290: naruszono więzy kontrolne (BS.ZERO_CKS)

SQL> select USERNAME, OWNER, OBJ_NAME, ACTION_NAME,
  2         To_Char(TIMESTAMP,'DD/MM/YY HH24:MI:SS' ) "CZAS"
  3    from dba_audit_trail
  4   order by CZAS;

USERNAME   OWNER           OBJ_NAME             ACTION_NAME          CZAS
---------- --------------- -------------------- -------------------- -----------------
BS         BS              TEST1                UPDATE               19/08/02 09:17:56
BS         BS              TEST1                UPDATE               19/08/02 09:18:44
```

W przykładzie ustawiono obserwację na udanych operacjach modyfikacji (update) wartości w tabeli TEST1. Do tabeli sys.aud$ trafiają informacje o każdej udanej operacji update niezależnie od tego czy wiersz został faktycznie zmodyfikowany (patrz druga komenda). Nie są natomiast zapisywane informacje o poleceniach update, które zakończyły się niepowodzeniem, w tym przypadku został zgłoszony błąd ORA-02290.

```
Przykład.

SQL> audit update on BS.TEST1 by session whenever successful;

Obserwowanie zostało włączone.

Komendy wykonane przez użytkownika BS:
SQL> update TEST1
  2    set col1=10
  3   where col1=1;

1 wiersz został zmodyfikowany.

SQL> update TEST1
  2    set col1=0
  3   where col1=1;

0 wierszy zostało zmodyfikowanych.

SQL> update TEST1
  2    set col1=0
  3   where col1=2;
update TEST1
*
BŁĄD w linii 1:
ORA-02290: naruszono więzy kontrolne (BS.ZERO_CKS)

SQL> select USERNAME, OWNER, OBJ_NAME, ACTION_NAME, SES_ACTIONS,
  2         To_Char(TIMESTAMP,'DD/MM/YY HH24:MI:SS' ) "CZAS"
  3    from dba_audit_trail
  4   order by CZAS;

USERNAME   OWNER  OBJ_NAME  ACTION_NAME    SES_ACTIONS      CZAS
---------- ------ --------- -------------- ---------------- -----------------
BS         BS     TEST1     SESSION REC    ----------S----- 19/08/02 09:58:36
```

Przykład jest analogiczny do poprzedniego, z tą różnicą, że ustawiono obserwację na całą sesję, w rezultacie w tabeli auditingu zostaje zapisany tylko jeden wiersz, zaś kolumna SES_ACTIONS posiada zaznaczony wskaźnik S (success) w pozycji odpowiadającej poleceniu update.

14. Wprowadzenie do problematyki strojenia bazy

Strojenie (tuning) obejmuje grupę działań służących zwiększeniu wydajności pracy bazy danych. Podstawowe strojenie jest wykonywane już na etapie planowania bazy i aplikacji, i polega na odpowiednim zaprojektowaniu struktury bazy (np. podział na przestrzenie tabel, rozkład plików, dobór bloku itp.) oraz właściwemu zaprojektowaniu aplikacji. W praktyce działania te są niewystarczające i już na etapie uruchomienia produkcyjnego aplikacji pojawiają się problemy wydajnościowe. Problemy dotyczące długiego wykonywania poleceń SQL wynikające ze złych planów wykonania, braku indeksów itp. są pochodne konstrukcji aplikacji i powinny być rozwiązane przez jej twórców. Zadaniem administratora jest natomiast optymalne dopasowanie parametrów bazy danych do posiadanych zasobów sprzętowych, oraz wykrycie wąskich gardeł w systemie w celu określenia potrzebnych rozszerzeń.

Strojenie bazy danych uważane jest za najbardziej zaawansowaną część administracji i wymaga od administratora oprócz wiedzy teoretycznej również doświadczenia. Niniejszy rozdział ma na celu zapoznać z problematyką tuningu, przedstawić sposoby pozyskiwania danych statystycznych o aktywności bazy oraz wskazać standardowe narzędzia używane do tego celu. Szczegółowe metody strojenia bazy wykraczają poza zakres opracowania.

Strojenie systemu obejmuje następujące kroki:

- Strojenie poleceń SQL
- Strojenie użycia procesorów
- Strojenie przydziału pamięci
- Strojenie kanałów WE/WY
- Rozstrzyganie sporów o zasoby
- Strojenie sieci
- Strojenie systemu operacyjnego

Problemy wydajnościowe bazy są zawsze związane ze środowiskiem w jakim baza pracuje, a więc z serwerem bazy (hardware) oraz systemem operacyjnym. Dlatego też administrator powinien dobrze znać to środowisko, oraz przynajmniej podstawowe narzędzia monitoringu systemu operacyjnego np. polecenie **sar** (UNIX), Management Console (Windows 2000) itp.

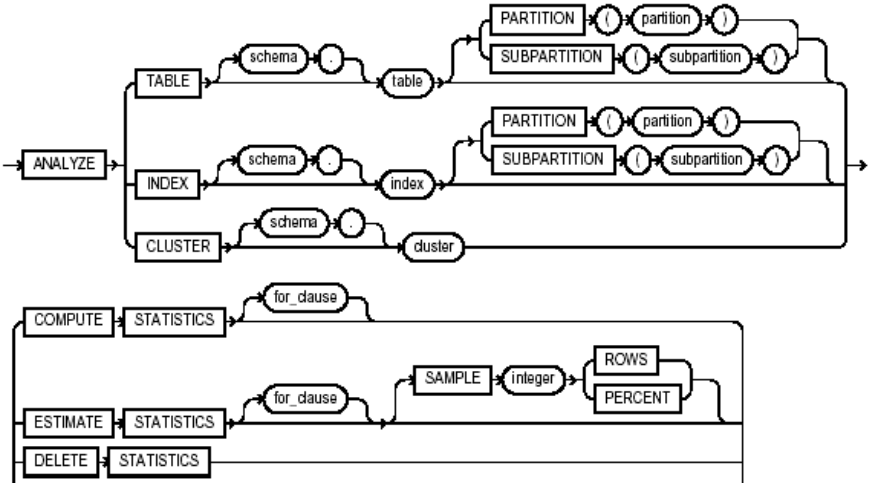
14.1 Warunki wymagane do strojenia

Aktualność statystyk

Przed stwierdzeniem, że pewne zapytania wykonują się nieakceptowalnie długo należy zagwarantować aktualny stan statystyk dla optymalizatora kosztowego, aby wyeliminować sytuację tworzenia planu wykonania na podstawie błędnych danych statystycznych. Dopiero po stwierdzeniu poprawności statystyk należy szukać innych powodów nieoptymalnego działania serwera. Użycie przez serwer Oracle określonego typu optymalizatora determinuje parametr inicjalizacyjny OPTIMIZER_MODE. Przyjmuje wartości:

- RULE – optymalizacja oparta na regułach. Obecnie parametr rzadko stosowany
- CHOOSE – optymalizacja oparta na koszcie
- FIRST_ROWS - optymalizacja oparta na koszcie ukierunkowana na plan wykonania gwarantujący najszybsze uzyskanie pierwszych wierszy odpowiedzi (minimalizacja czasu odpowiedzi)
- ALL_ROWS - optymalizacja oparta na koszcie ukierunkowana na plan wykonania gwarantujący najszybsze uzyskanie wszystkich wierszy odpowiedzi (minimalizacja czasu wykonania)

Obliczenie statystyk wykonywane jest poleceniem **ANALYZE**. Dla dużych baz danych obliczenie statystyk na podstawie wszystkich wierszy (opcja COMPUTE) jest nieopłacalne czasowo i w tym przypadku należy stosować wyliczenie na podstawie próbki statystycznej (opcja ESTIMATE). Częstotliwość obliczania statystyk, oraz wielkość próbki musi być dobrana indywidualnie dla każdej tabeli, uwzględniając wielkość dziennych zmian.



Zbieranie czasu procesora

Oracle umożliwia obliczanie i zapamiętywanie dokładnego czasu wykonywania operacji, oraz czasu spędzonego na oczekiwaniach na pozyskanie zasobu, zwolnienie blokady itp. Czasy te są wyświetlane w plikach śladu oraz perspektywach dynamicznych opisujących statystyki systemowe. O możliwości naliczania czasu decyduje parametr inicjalizacyjny TIMED_STATISTICS. Wartość TRUE parametru umożliwia zapamiętywanie czasu. Parametr ten ma wpływ na ogólne obciążenie systemu, dlatego w normalnej pracy powinien być ustawiony na wartość FALSE.

14.2 Perspektywy zawierające statystyki

Statystyki dostępne są poprzez dynamiczne perspektywy systemowe. Najważniejsze z nich to:

- V$INSTANCE – opis bieżącego stanu instancji bazy
- V$LATCH – statystyki dotyczące rezerwacji (latches).
- V$LIBRARYCACHE – statystyki dotyczące aktywności i wydajności bufora biblioteki SGA
- V$ROLLSTAT – statystyki opisujące aktywność segmentów wycofania
- V$ROWCACHE – statystyki opisujące aktywność danych w buforze słownika danych.
- V$SGA – rozmiar głównych obszarów SGA
- V$SGASTAT - rozmiar poszczególnych obszarów alokowanych w SGA w tym również pozostałej wolnej pamięci
- V$SORT_USAGE – użycie segmentów tymczasowych przez sesje wykonujące sortowanie na dysku.
- V$SQLAREA – statystyki opisujące wspólny obszar SQL, podające szczegóły dla każdego polecenia SQL
- V$SQLTEXT – pełny tekst polecenia SQL należącego do dzielonego kursora w SGA.
- V$SYSSTAT – wartości statystyk systemowych. Nazwy statystyk dostępne są poprzez perspektywę V$STATNAME. Statystyki zbierane są w następujących klasach:
  - 1 – User
  - 2 – Redo
  - 4 – Enqueue
  - 8 – Cache
  - 16 – OS
  - 32 – Parallel Server
  - 64 - SQL
  - 128 - Debug
- V$SYSTEM_EVENT – statystyki oczekiwań na określone zdarzenie.
- V$WAITSTAT – statystyki dotyczące sporów o bloki.

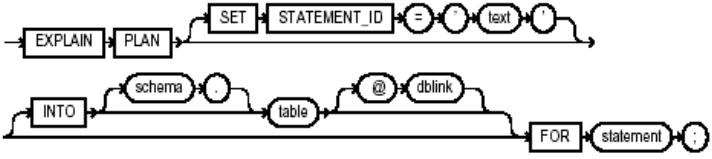
```
Przykład.

SQL>  select s.STATISTIC#, s.VALUE, n.NAME, n.CLASS
  2     from V$SYSSTAT s, V$STATNAME n
  3    where s.STATISTIC#=n.STATISTIC# and
  4          n.CLASS=1;

STATISTIC#      VALUE NAME                                               CLASS
---------- ---------- ---------------------------------------- ----------
         0         13 logons cumulative                                   1
         1          8 logons current                                      1
         2        591 opened cursors cumulative                           1
         3         12 opened cursors current                              1
         4          5 user commits                                        1
         5          0 user rollbacks                                      1
         6        103 user calls                                          1
         7      20395 recursive calls                                     1
         8       2947 recursive cpu usage                                 1
         9      12791 session logical reads                               1
        10          0 session stored procedure space                       1

STATISTIC#      VALUE NAME                                               CLASS
---------- ---------- ---------------------------------------- ----------
        12       2964 CPU used by this session                           1
        13 3094262519 session connect time                               1
        15     408232 session uga memory                                 1
        16     711608 session uga memory max                             1
        20    6465564 session pga memory                                 1
        21    6465564 session pga memory max                             1
       154          0 serializable aborts                                1
       223      22062 bytes sent via SQL*Net to client                   1
       224       7866 bytes received via SQL*Net from client             1
       225         58 SQL*Net roundtrips to/from client                  1
       226          0 bytes sent via SQL*Net to dblink                   1

STATISTIC#      VALUE NAME                                               CLASS
---------- ---------- ---------------------------------------- ----------
       227          0 bytes received via SQL*Net from dblink              1
       228          0 SQL*Net roundtrips to/from dblink                   1
```

W przykładzie wyświetlono aktualne wartości statystyk dla klasy User. Wartości statystyk są zbierane od momentu podniesienia bazy. Do szczegółowej analizy, wykonywanej zwykle dla pewnego przedziału czasu np. gdy występują wąskie gardła przetwarzania, potrzebne są statystyki obejmujące taki przedział. Tak przetworzone dane dostarczane są przez opisane dalej narzędzia.

14.3 Narzędzia używane do strojenia bazy

Explain plan

Polecenie umożliwiające wyświetlenie planu wykonania podanego zdania SQL. Stosowane jest głównie przez programistów aplikacji do analizy planów wykonania.



Przed użyciem polecenia wymagane jest utworzenie pomocniczej tabeli o domyślnej nazwie PLAN_TABLE. Skrypt do tworzenia tej tabeli o nazwie *utlxplan.sql* jest dostarczony przez Oracle i znajduje się w katalogu *$ORACLE_HOME/rdbms/admin*. W określonych przypadkach można użyć również ustawień zmiennej *autotrace* środowiska programu **SQL*Plus**, co umożliwia bezpośrednie wyświetlenie planu wykonania polecenia SQL zaraz po jego realizacji. Aby użyć wspomnianego ustawienia w schemacie użytkownika wykonującego komendy SQL wymagana jest obecność tabeli PLAN_TABLE.

```
Przykład.
```

```
SQL> set autotrace on
SQL> select m.col1, d.col
  2    from master m, detail d
  3   where m.col1=d.m_col1 and m.col2=d.m_col2 and
  4         d.col='Dwa';
      COL1 COL
---------- ----------
         1 Dwa


Plan wykonywania
----------------------------------------------------------
   0      SELECT STATEMENT Optimizer=CHOOSE (Cost=1 Card=1 Bytes=12)
   1    0   NESTED LOOPS (Cost=1 Card=1 Bytes=12)
   2    1     TABLE ACCESS (FULL) OF 'DETAIL' (Cost=1 Card=1 Bytes=8)
   3    1     INDEX (UNIQUE SCAN) OF 'MASTER_PK' (UNIQUE)



Statystyki
----------------------------------------------------------
          0  recursive calls
          2  db block gets
          3  consistent gets
          0  physical reads
          0  redo size
        454  bytes sent via SQL*Net to client
        503  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed
```

W przykładzie pokazano sposób działania automatycznego wyświetlania planu wykonania w programie **SQL*Plus**. Dodatkowo poza planem wykonania wyświetlane są statystyki dla polecenia.

TKPROF

TKPROF jest zawansowanym narzędziem służącym do tłumaczenia plików śladu na postać wymaganą do analizy zapytań. W szczególności w przetworzonym pliku znajdują się plany wykonania i czasy realizacji poszczególnych operacji takich jak: parsowanie zapytań, wykonanie, sprowadzanie (fetch). Aby wymusić zrzucenie zapytań generowanych przez sesję do pliku śladu należy wykonać komendę **ALTER SESSION SET SQL_TRACE = TRUE**. Uzyskany plik śladu może być następnie przetworzony programem narzędziowym **TKPROF**.

```
Przykład.

SQL> alter session set sql_trace=true;

Sesja została zmieniona.

SQL> select m.col1, d.col
  2    from master m, detail d
  3   where m.col1=d.m_col1 and m.col2=d.m_col2 and
  4       d.col='Dwa';

      COL1 COL
---------- ----------
         1 Dwa

SQL> alter session set sql_trace=false;

Sesja została zmieniona.

Zawartość katalogu udump:
$ ls -l
total 16
-rw-r-----   1 oracle   dba         1799 wrz 11 09:41 orcl_ora_18467.trc

$ tkprof orcl_ora_18467.trc trace.txt explain=bs/passwbs
```

```
TKPROF: Release 8.1.7.0.0 - Production on Żro Wrz 11 10:05:24 2002

(c) Copyright 2000 Oracle Corporation.  All rights reserved.


$ ls -l
total 16
-rw-r-----   1 oracle   dba         1799 wrz 11 09:41 orcl_ora_18467.trc
-rw-r--r--   1 oracle   dba         5423 wrz 11 10:05 trace.txt
```

W przykładzie zrzucono do pliku śladu sesję użytkownika i następnie przetworzono plik śladu programem **tkprof**.

Przykład.

Plik orcl_ora_18467.trc:

```
Dump file /oracle/oradb/ORCL/udump/orcl_ora_18467.trc
Oracle8i Enterprise Edition Release 8.1.7.0.0 - Production
With the Partitioning option
JServer Release 8.1.7.0.0 - Production
ORACLE_HOME = /oracle/product/817
System name:       SunOS
Node name:  ux1
Release:    5.8
Version:    Generic_108528-03
Machine:    sun4u
Instance name: ORCL
Redo thread mounted by this instance: 1
Oracle process number: 17
Unix process pid: 18467, image: oracle@ORCL (TNS V1-V3)

*** 2002-09-11 09:41:04.737
*** SESSION ID:(29.3101) 2002-09-11 09:41:04.722
APPNAME mod='SQL*Plus' mh=3669949024 act='' ah=4029777240
=====================
PARSING IN CURSOR #1 len=32 dep=0 uid=395 oct=42 lid=395 tim=93791369 hv=1197935484
ad='83505394'
alter session set sql_trace=true
END OF STMT
EXEC #1:c=0,e=1,p=0,cr=0,cu=0,mis=1,r=0,dep=0,og=4,tim=93791369
=====================
PARSING IN CURSOR #1 len=112 dep=0 uid=395 oct=3 lid=395 tim=93792045 hv=1284252257
ad='837e130c'
select m.col1, d.col
  from master m, detail d
 where m.col1=d.m_col1 and m.col2=d.m_col2 and
       d.col='Dwa'
END OF STMT
PARSE #1:c=0,e=0,p=0,cr=0,cu=0,mis=1,r=0,dep=0,og=4,tim=93792045
EXEC #1:c=0,e=0,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=4,tim=93792045
FETCH #1:c=0,e=0,p=0,cr=2,cu=4,mis=0,r=1,dep=0,og=4,tim=93792045
FETCH #1:c=0,e=0,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=4,tim=93792046
STAT #1 id=1 cnt=1 pid=0 pos=0 obj=0 op='NESTED LOOPS '
STAT #1 id=2 cnt=2 pid=1 pos=1 obj=23642 op='TABLE ACCESS FULL DETAIL '
STAT #1 id=3 cnt=1 pid=1 pos=2 obj=23641 op='INDEX UNIQUE SCAN '
=====================
PARSING IN CURSOR #1 len=33 dep=0 uid=395 oct=42 lid=395 tim=93792955 hv=1615536819
ad='83956658'
alter session set sql_trace=false
END OF STMT
PARSE #1:c=0,e=0,p=0,cr=0,cu=0,mis=1,r=0,dep=0,og=4,tim=93792955
EXEC #1:c=0,e=0,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=4,tim=93792955
```

W przykładzie pokazano zawartość pliku śladu. Jak widać informacje dotyczące wykonania polecenia są SQL są podane w postaci nieczytelnej dla użytkownika.

Przykład.

Fragment pliku trace.txt

```
TKPROF: Release 8.1.7.0.0 - Production on śro Wrz 11 09:41:58 2002

(c) Copyright 2000 Oracle Corporation.  All rights reserved.

Trace file: orcl_ora_18467.trc
Sort options: default
```

```
********************************************************************************
count    = number of times OCI procedure was executed
cpu      = cpu time in seconds executing
elapsed  = elapsed time in seconds executing
disk     = number of physical reads of buffers from disk
query    = number of buffers gotten for consistent read
current  = number of buffers gotten in current mode (usually for update)
rows     = number of rows processed by the fetch or execute call
********************************************************************************

alter session set sql_trace=true


call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        0      0.00       0.00          0          0          0           0
Execute      1      0.00       0.01          0          0          0           0
Fetch        0      0.00       0.00          0          0          0           0
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        1      0.00       0.01          0          0          0           0

Misses in library cache during parse: 0
Optimizer goal: CHOOSE
Parsing user id: 395  ()
********************************************************************************

select m.col1, d.col
  from master m, detail d
 where m.col1=d.m_col1 and m.col2=d.m_col2 and
       d.col='Dwa'

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        2      0.00       0.00          0          2          4           1
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        4      0.00       0.00          0          2          4           1

Misses in library cache during parse: 0
Optimizer goal: CHOOSE
Parsing user id: 395  ()

Rows     Row Source Operation
-------  ---------------------------------------------------
      1  NESTED LOOPS
      2   TABLE ACCESS FULL DETAIL
      1   INDEX UNIQUE SCAN (object id 23641)
```

W przykładzie pokazano fragment przetworzonego przez program **tkprof** pliku śladu. Jak widać postać przetworzona zawiera wszystkie szczegółowe informacje dotyczące wykonań poleceń SQL.

Skrypty narzędziowe

Oracle dostarcza pakiet skryptów przydatnych w strojeniu bazy. Ponieważ perspektywy dynamiczne zawierają dane aktualne na daną chwilę czasu, przydatniejsze dla administratora jest uzyskanie danych z pewnego przedziału czasu. Taką funkcję spełniają skrypty korzystające z pomocniczych tabel. Wymagają one co najmniej dwukrotnego uruchomienia, po czym udostępniają wykonanie raportu tekstowego. Do ich użycia wymagany jest wyłącznie **SQL*Plus**.

Utlbstat/Utlestat

Klasyczne skrypty znane od wcześniejszych wersji Oracle. Użycie odbywa się w dwóch krokach:

1. wykonanie *utlbstat* – skrypt tworzy tabele pomocnicze i zapisuje w nich bieżące stany statystyk na początek analizy
2. wykonanie *utlestat* – skrypt zapisuje bieżące stany statystyk na koniec analizy, a następnie tworzy tekstowy raport różnicowy w pliku *report.txt*

```
Przykład.

SQL> column library  format a12 trunc;
SQL> column pinhitratio   heading 'PINHITRATI';
SQL> column gethitratio   heading 'GETHITRATI';
SQL> column invalidations heading 'INVALIDATI';
```

```
SQL> set numwidth 10;
SQL> Rem Select Library cache statistics.  The pin hit rate should be high.
SQL> select namespace library,
  2             gets,
  3             round(decode(gethits,0,1,gethits)/decode(gets,0,1,gets),3)
  4                gethitratio,
  5             pins,
  6             round(decode(pinhits,0,1,pinhits)/decode(pins,0,1,pins),3)
  7                pinhitratio,
  8             reloads, invalidations
  9    from stats$lib;


LIBRARY            GETS GETHITRATI       PINS PINHITRATI    RELOADS INVALIDATI
------------ ---------- ---------- ---------- ---------- ---------- ----------
BODY                  0          1          0          1          0          0
CLUSTER               0          1          0          1          0          0
INDEX                 0          1          0          1          0          0
JAVA DATA             0          1          0          1          0          0
JAVA RESOURC          0          1          0          1          0          0
JAVA SOURCE           0          1          0          1          0          0
OBJECT                0          1          0          1          0          0
PIPE                  0          1          0          1          0          0
SQL AREA             31       ,871        131       ,931          0          0
TABLE/PROCED         22          1         30          1          0          0
TRIGGER               1          1          1          1          0          0

11 wierszy zostało wybranych.

SQL>
SQL> column "Statistic"       format a27 trunc;
SQL> column "Per Transaction" heading "Per Transact";
SQL> column ((start_users+end_users)/2) heading "((START_USER"
SQL> set numwidth 12;
SQL> Rem The total is the total value of the statistic between the time
SQL> Rem bstat was run and the time estat was run.  Note that the estat
SQL> Rem script logs on to the instance so the per_logon statistics will
SQL> Rem always be based on at least one logon.
SQL> select 'Users connected at ',to_char(start_time, 'dd-mon-yy hh24:mi:ss'),':',start_users
from stats$dates;

'USERSCONNECTEDAT'  TO_CHAR(START_TIME '  START_USERS
------------------- ------------------ - ------------
Users connected at  07-wrz-02 22:18:15 :            8

SQL> select 'Users connected at ',to_char(end_time, 'dd-mon-yy hh24:mi:ss'),':',end_users from
stats$dates;

'USERSCONNECTEDAT'  TO_CHAR(END_TIME,' '    END_USERS
------------------- ------------------ - ------------
Users connected at  07-wrz-02 22:19:51 :            8

SQL> select 'avg # of connections: ',((start_users+end_users)/2) from stats$dates;

'AVG#OFCONNECTIONS:'   ((START_USER
---------------------- ------------
avg # of connections:             8

SQL>
SQL> select n1.name "Statistic",
  2             n1.change "Total",
  3             round(n1.change/trans.change,2) "Per Transaction",
  4             round(n1.change/((start_users + end_users)/2),2)  "Per Logon",
  5             round(n1.change/(to_number(to_char(end_time,   'J'))*60*60*24 -
  6                              to_number(to_char(start_time, 'J'))*60*60*24 +
  7                              to_number(to_char(end_time,   'SSSSS')) -
  8                              to_number(to_char(start_time, 'SSSSS')))
  9                   , 2) "Per Second"
 10         from
 11                      stats$stats n1,
 12                      stats$stats trans,
 13                      stats$dates
 14         where
 15               trans.name='user commits'
 16          and  n1.change != 0
 17         order by n1.name;
```

```
Statistic                      Total Per Transact    Per Logon   Per Second
--------------------------- ------------ ------------ ------------ ------------
background timeouts                111          111        13,88         1,16
buffer is not pinned count          33           33         4,13          ,34
bytes received via SQL*Net        1792         1792          224        18,67
bytes sent via SQL*Net to c       1923         1923       240,38        20,03
calls to get snapshot scn:          79           79         9,88          ,82
calls to kcmgas                     10           10         1,25          ,1
calls to kcmgcs                     41           41         5,13          ,43
cluster key scan block gets         24           24            3          ,25
cluster key scans                   15           15         1,88          ,16
commit cleanouts                    22           22         2,75          ,23
commit cleanouts successful         22           22         2,75          ,23
consistent changes                   6            6          ,75          ,06
consistent gets                    148          148         18,5         1,54
consistent gets - examinati         37           37         4,63          ,39
CPU used by this session            10           10         1,25          ,1
CPU used when call started          10           10         1,25          ,1
CR blocks created                    6            6          ,75          ,06
data blocks consistent read          6            6          ,75          ,06
db block changes                   179          179        22,38         1,86
db block gets                      147          147        18,38         1,53
DBWR checkpoint buffers wri         16           16            2          ,17
DBWR undo block writes               2            2          ,25          ,02
deferred (CURRENT) block cl          3            3          ,38          ,03
enqueue releases                   100          100         12,5         1,04
enqueue requests                    94           94        11,75          ,98
execute count                       67           67         8,38          ,7
free buffer requested               23           23         2,88          ,24
logons cumulative                    2            2          ,25          ,02
messages received                    2            2          ,25          ,02
messages sent                        2            2          ,25          ,02
no work - consistent read g         24           24            3          ,25
opened cursors cumulative           31           31         3,88          ,32
parse count (hard)                   5            5          ,63          ,05
parse count (total)                 31           31         3,88          ,32
parse time cpu                       2            2          ,25          ,02
parse time elapsed                   2            2          ,25          ,02
physical writes                     16           16            2          ,17
physical writes non checkpo         13           13         1,63          ,14
recursive calls                    343          343        42,88         3,57
recursive cpu usage                  1            1          ,13          ,01
redo blocks written                 93           93        11,63          ,97
redo entries                       112          112           14         1,17
redo size                        61616        61616         7702       641,83
redo synch writes                    1            1          ,13          ,01
redo wastage                       152          152           19         1,58
redo write time                      1            1          ,13          ,01
redo writes                          2            2          ,25          ,02
rollbacks only - consistent          6            6          ,75          ,06
session logical reads              295          295        36,88         3,07
session pga memory              -184076      -184076     -23009,5     -1917,46
session pga memory max          -246788      -246788     -30848,5     -2570,71
session uga memory               -2092        -2092       -261,5       -21,79
session uga memory max           62984        62984         7873       656,08
shared hash latch upgrades          57           57         7,13          ,59
sorts (memory)                      22           22         2,75          ,23
sorts (rows)                      3574         3574       446,75        37,23
SQL*Net roundtrips to/from          12           12          1,5          ,13
table fetch by rowid                 3            3          ,38          ,03
user calls                          22           22         2,75          ,23
user commits                         1            1          ,13          ,01

60 wierszy zostało wybranych.

SQL>
```

Przykład jest początkiem wygenerowanego pliku *report.txt* utworzonego przy pomocy pakietu *utlestat*. Informacje dotyczące statystyk systemowych pobierane są z pomocniczej tabeli pakietu STATS$STATS zawierającej wartości statystyk będące różnicą bieżących wartości na chwilę uruchomienia pakietu *utlbstat* i *utlestat*.

```
Przykład.

create table stats$begin_stats as select * from v$sysstat where 0 = 1;
```

```
insert into stats$begin_stats select * from v$sysstat;

create table stats$stats as
select  e.value-b.value change , n.name
   from v$statname n ,  stats$begin_stats b , stats$end_stats e
where n.statistic# = b.statistic# and n.statistic# = e.statistic#;
```

W przykładzie pokazano sposób tworzenia tabel pomocniczych pakietu *utlbstat* i *utlestat*. Jak widać są one oparte o systemową perspektywę dynamiczną V$SYSSTAT. W podobny sposób tworzone są inne tabele pomocnicze wykorzystujące inne perspektywy dynamiczne opisane wcześniej.

Statspack

Narzędzie do bardzo szczegółowego zbierania statystyk, łącznie ze wskazaniem najdłużej wykonywanych poleceń SQL (wraz z ich adresem w SGA). Opiera się o wygenerowanie w bazie osobnego schematu PERFSTAT zawierającego tabele oraz pakiet narzędziowy STATSPACK. Aby użyć pakietu należy:

1. Z poziomu SQL*Plus wykonać pakiet *$ORACLE_HOME/rdbms/admin/spcreate.sql*
2. W wybranych momentach czasowych wykonać migawkę statystyk systemu procedurą PERFSTAT.STATSPACK.SNAP
3. Wykonać różnicowy raport tekstowy z dowolnego przedziału czasu obejmującego dwie dowolnie wybrane migawki. Dostarczony skrypt *spreport.sql* umożliwia interakcyjne podanie numerów migawek.

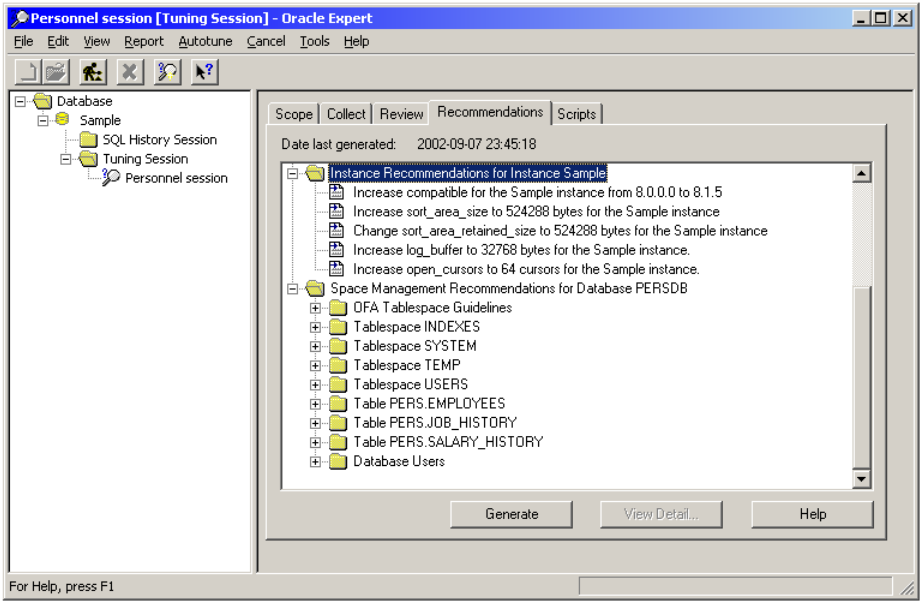
```
Przykład.

STATSPACK report for

DB Name         DB Id    Instance     Inst Num Release     OPS Host
------------ ----------- ------------ -------- ----------- --- ------------
ORACLE        3987557989 ORCL                1 8.1.7.3.0   NO  ux1

                Snap Id     Snap Time      Sessions
                ------- ------------------ --------
 Begin Snap:         11 31-Sie-02 23:16:30       31
   End Snap:         12 31-Sie-02 23:27:20       31
    Elapsed:                 10.83 (mins)

Cache Sizes
~~~~~~~~~~~
           db_block_buffers:    5900000          log_buffer:     262144
              db_block_size:       8192    shared_pool_size:  368435456

Load Profile
~~~~~~~~~~~~                            Per Second       Per Transaction
                                   ---------------       ---------------
                  Redo size:             60,389.55              6,066.96
              Logical reads:              6,121.06                 614.94
              Block changes:                 379.61                  38.14
             Physical reads:                   7.72                   0.78
            Physical writes:                  41.58                   4.18
                 User calls:                 201.23                  20.22
                     Parses:                 114.10                  11.46
                Hard parses:                   0.03                   0.00
                      Sorts:                  26.16                   2.63
                     Logons:                   0.18                   0.02
                   Executes:                 388.49                  39.03
               Transactions:                   9.95

  % Blocks changed per Read:    6.20    Recursive Call %:   87.22
 Rollback per transaction %:    0.00       Rows per Sort: #######

Instance Efficiency Percentages (Target 100%)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
            Buffer Nowait %:  100.00       Redo NoWait %:  100.00
            Buffer  Hit   %:   99.87    In-memory Sort %:   95.44
            Library Hit   %:  100.00        Soft Parse %:   99.97
        Execute to Parse %:   70.63         Latch Hit %:  100.00
Parse CPU to Parse Elapsd %:   63.04     % Non-Parse CPU:   99.98

 Shared Pool Statistics        Begin   End
                               ------  ------
             Memory Usage %:   97.13   97.15
    % SQL with executions>1:   90.20   90.58
  % Memory for SQL w/exec>1:   82.92   83.66
```

```
Top 5 Wait Events
~~~~~~~~~~~~~~~~~
                                                             Wait     % Total
Event                                            Waits  Time (cs)   Wt Time
-------------------------------------------- ---------- ------------ -------
log file parallel write                          8,070        1,038   45.49
log file sync                                    5,752          974   42.68
control file parallel write                        211          210    9.20
control file sequential read                       745           32    1.40
refresh controlfile command                         78           18     .79
        -------------------------------------------------------------
Wait Events for DB: BGZ  Instance: BGZ  Snaps: 11 -12
-> cs - centisecond -  100th of a second
-> ms - millisecond - 1000th of a second
-> ordered by wait time desc, waits desc (idle events last)
```

Przykład jest początkiem pliku raportu wygenerowanego skryptem *spreport* z 10 minutowego przedziału czasu zawartego pomiędzy migawkami o numerach 11 i 12. Jak widać raport zawiera już wyliczone wskaźniki takie jak np. współczynnik trafień w obszar buforów, współczynnik sortowania w pamięci itp. oraz 5 rodzajów zdarzeń o najdłuższym czasie oczekiwania.

Programy specjalistyczne

Poza wymienionymi powyżej narzędziami istnieją specjalizowane programy, pracujące zwykle jako aplikacje okienkowe ułatwiające administratorowi strojenie bazy. Oprócz aplikacji komercyjnych obcych producentów istnieje narzędzie produkcji Oracle o nazwie **Oracle Enterprise Manager** zawierające szereg przydatnych cech w tym również Tuning Pack. Wchodzący w jego skład program **Oracle Expert** posiada wbudowany mechanizm rekomendowania rozwiązań mogących poprawić wydajność bazy.



Okno programu Oracle Expert

14.4 Elementy strojenia systemu

Strojenie pamięci

Strojenie pamięci ma kluczowe znaczenie dla redukcji operacji dyskowych, trwających wielokrotnie dłużej niż dostęp do pamięci operacyjnej. Stroiąc ten obszar należy pamiętać, aby jednocześnie kontrolować zarządzanie pamięci przez system operacyjny, tak aby zwiększenie przydziału pamięci dla

Oracle nie skutkowało jednocześnie wzrostem stronicowania (paging, swaping) w OS co mogłoby całkowicie zniweczyć zysk z powiększenia pamięci alokowanej dla SGA.

Strojenie pamięci w obszarze dzielonym (Shared Pool)

Pamięć operacyjna wykorzystywana jest do buforowania często używanych danych. Jest to szczególnie istotne w typowych zastosowaniach baz danych obsługujących wielu użytkowników pracujących na jednakowym zestawie aplikacji. Wykonywane są wtedy wielokrotnie te same polecenia SQL i bloki PL/SQL. Obserwacji podlegają statystyki:

- PINS – liczba wykonań poleceń
- RELOADS – liczba chybień, w fazie wykonania polecenie było nieobecne w pamięci

Celem strojenia jest dobór wielkości pamięci shared pool tak, aby współczynnik trafień był bliski 100%.

Przykład.

```
SQL> select namespace, pins, reloads
  2   from v$librarycache;

NAMESPACE             PINS    RELOADS
--------------- ---------- ----------
SQL AREA            132904         29
TABLE/PROCEDURE      46283         11
BODY                    47          0
TRIGGER                902          0
INDEX                  351          0
CLUSTER                498          0
OBJECT                   0          0
PIPE                     0          0
JAVA SOURCE              0          0
JAVA RESOURCE            0          0
JAVA DATA                0          0

SQL> select sum(pins)/(sum(pins)+sum(reloads))*100 "Cache Hit Ratio %"
  2   from v$librarycache;

Cache Hit Ratio %
-----------------
       99,9777779
```

W przykładzie wyliczono współczynnik trafień w obszarze bibliotecznym. W przypadku gdyby wyliczony współczynnik był niższy, należy zwiększyć parametr SHARED_POOL_SIZE. Należy również sprawdzić czy aplikacja odpowiada standardom Oracle w zakresie rozpoznawania identycznych poleceń SQL, stosuje zmienne wiązane itp.

Strojenie obszaru dzielonego powinno również uwzględniać sprawdzenie innych odwołań widocznych w perspektywie V$ROWCACHE. Obserwacji podlegają statystyki:

- GETS – liczba żądań pobrania określonej informacji
- GETMISSES – liczba chybionych żądań (brak potrzebnej informacji w słowniku)

Celem strojenia jest dobór wielkości obszaru dzielonego tak, aby współczynnik trafień przekraczał 85%.

Przykład.

```
SQL> select type, parameter, count, gets, getmisses
  2   from v$rowcache;

TYPE        PARAMETER                          COUNT       GETS  GETMISSES
----------- ---------------------------------- ---------- ---------- ----------
PARENT      dc_free_extents                           54        581        109
PARENT      dc_used_extents                           50        102         68
PARENT      dc_segments                              645       5648        663
PARENT      dc_tablespaces                            15       4734          6
PARENT      dc_tablespace_quotas                      23        546          1
PARENT      dc_files                                  22         13          2
PARENT      dc_users                                  27      14749         14
PARENT      dc_rollback_segments                      19       1750         17
PARENT      dc_objects                              1394       7594       1365
PARENT      dc_global_oids                            33        114         31
PARENT      dc_constraints                           498        999        499
PARENT      dc_object_ids                            902       6380        897
PARENT      dc_sequences                              57        260         52
PARENT      dc_usernames                              21       2885         10
PARENT      dc_database_links                          1          0          0
PARENT      dc_histogram_defs                         57        396        279
```

```
PARENT      table scns                          24          28          28
PARENT      dc_outlines                          1           0           0
PARENT      dc_profiles                         14          75           1
PARENT      encrypted object row cache           1           0           0
PARENT      encryption profile row cache         1           0           0
PARENT      ifs_acl_cache_entries                1           0           0
SUBORDINATE dc_users                            12         144           4
SUBORDINATE dc_histogram_data                    1           0           0
SUBORDINATE dc_histogram_data_values             1           0           0
SUBORDINATE partition scns                       1           0           0
SUBORDINATE dc_user_grants                      47       11665          12
SUBORDINATE dc_app_role                          1           0           0

SQL> select sum(gets), sum(getmisses),
  2         100-sum(getmisses)/sum(gets)*100 "Hit Ratio [%]"
  3   from v$rowcache;

 SUM(GETS) SUM(GETMISSES) Hit Ratio [%]
---------- -------------- -------------
     58722           4058    93,0894724
```

W przykładzie wyliczony współczynnik trafień powyżej 90% oznacza prawidłowo dobraną wielkość obszaru dzielonego.

Strojenie buforów danych (Buffer Cache)

Analogicznie do strojenia obszaru wspólnego należy analizować wykorzystanie buforów danych. Właściwe wystrojenie tego obszaru redukuje znacznie operacje dyskowe i pozwala osiągnąć wysoką wydajność przetwarzania. Obserwacji podlegają statystyki zawarte w perspektywie V$SYSSTAT:

- db block gets – ogólna liczba żądań danych
- consistent gets – żądania zrealizowane poprzez odczyt z pamięci
- phisical reads – żądania zrealizowane poprzez odczyt z dysku

Celem strojenia jest uzyskanie współczynnika trafień w obszar pamięci nie mniejszego niż 90%.

```
Przykład.

SQL> select name, value from v$sysstat
  2   where name IN ('db block gets', 'consistent gets',
  3         'physical reads' );

NAME                          VALUE
-------------------- -----------
db block gets             129079
consistent gets           137999
physical reads              2335

SQL> select 100*(1-(v3.value/(v1.value+v2.value))) "Hit Ratio [%]"
  2   from v$sysstat v1, v$sysstat v2, v$sysstat v3
  3   where v1.name='db block gets' and
  4         v2.name='consistent gets' and
  5         v3.name='physical reads';

Hit Ratio [%]
-------------
   99,1258218
```

W przykładzie odczytano wartości potrzebnych statystyk i wyliczono współczynnik trafień. Uzyskana wartość jest bliska 100% co oznacza, że system jest bardzo dobrze buforowany. Należy pamiętać, że tak wysokie wartości są możliwe do uzyskania tylko dla aplikacji OLTP charakteryzujących się zapytaniami czytającymi jednokrotnie niewielką liczbę bloków. Dla aplikacji OLAP, charakteryzującymi się czytaniem często całych tabel, których rozmiar przekracza wielkość buforów w SGA strojenie tego parametru nie ma sensu.

Strojenie obszaru sortowania (Sort Area)

Sortowanie wykonywane jest podczas realizacji poleceń SQL z frazą **order by**, tworzenia indeksów itp. Wykonywane jest w obszarze o rozmiarze określonym parametrem SORT_AREA_SIZE. Gdy obszar ten jest zbyt mały sortowanie odbywa się na dysku, a czas tej operacji może wzrosnąć nawet o rząd wielkości. Obserwacji podlegają statystki liczby sortowań na dysku i w pamięci dostępne w perspektywie V$SYSSTAT. Celem strojenia jest uzyskanie współczynnika sortowania w pamięci bliskiego 100%.

```
Przykład.
```

```
SQL> select name,value from v$sysstat where name like 'sort%';

NAME                      VALUE
-------------------- -----------
sorts (memory)              7000
sorts (disk)                   2
sorts (rows)               41218

SQL> select v1.value/(v1.value+v2.value)*100 "Memory Sort Ratio [%]"
  2    from v$sysstat v1,v$sysstat v2
  3   where v1.name='sorts (memory)' and v2.name='sorts (disk)';

Memory Sort Ratio [%]
---------------------
           99,9717713
```

W przykładzie wyświetlono statystyki sortowania w pamięci i na dysku oraz obliczono współczynnik sortowania w pamięci.

Strojenie WE/WY (I/O)

Strojenie I/O wiąże się bezpośrednio z dostępnymi zasobami sprzętowymi. Obserwacji podlega aktywność plików bazy Oracle w zakresie odczytów i zapisów. Celem strojenia jest uzyskanie jak najmniejszej liczby sporów o zasoby dyskowe poprzez odpowiedni rozkład plików na dyski, stosowanie macierzy, stripe'ów itp. a także dobór wielkości i liczby plików logów oraz częstotliwość wykonywania punktów kontrolnych.

Obserwacji podlegają dane zawarte w perspektywie V$FILESTAT:

- phyrds – liczba odczytów z dysku
- phyblkrd – liczba odczytów bloków z dysku
- phywrts – liczba zapisów na dysk
- phyblkwrt – liczba zapisów bloków na dysk

```
Przykład.

SQL> select name, phyrds, phywrts, phyblkrd, phyblkwrt
  2    from v$datafile df, v$filestat fs
  3   where df.file# = fs.file#
  4   order by phywrts desc;

NAME                    PHYRDS    PHYWRTS   PHYBLKRD  PHYBLKWRT
-------------------- ---------- ---------- ---------- ----------
/oradb/SYSTEM01.DBF        1388       3242       2175       3246
/oradb/RBS01.DBF              9       2277          9       2277
/oradb/USERS01.DBF          104       1882        104       1882
/oradb/INDX01.DBF             4          2          4          2
/oradb/TOOLS01.DBF            4          2          4          2
```

W przykładzie wyświetlono informacje o liczbie zapisów i odczytów do poszczególnych plików bazy w kolejności malejących zapisów.

Strojenie sporów o zasoby (Resource Contention)

Ta część strojenia dotyczy znajdywania i rozwiązywania problemów związanych z próbami jednoczesnego dostępu do współdzielonych zasobów. Spory o zasoby występują przede wszystkim na maszynach wieloprocesorowych umożliwiających faktyczną równoległą pracę wielu sesji Oracle, bez konieczności podziału czasu jednego procesora. Informacje o występowaniu sporów są dostępne w perspektywach V$SYSSTAT, V$WAITSTAT, V$LATCH. Strojenie sporów zostanie pokazane na przykładzie segmentów wycofania i buforów dziennika powtórzeń.

Segmenty wycofania (Rollback Segments)

Spory polegają na próbach jednoczesnego dostępu do segmentów wycofania przez wiele sesji użytkownika.

Obserwacji podlegają statystyki zawarte w perspektywie V$WAITSTAT:

- system undo header – liczba oczekiwań na bufory zawierające bloki nagłówka systemowego segmentu wycofania
- system undo block - liczba oczekiwań na bufory zawierające bloki systemowego segmentu wycofania inne niż bloki nagłówka

- undo header - liczba oczekiwań na bufory zawierające bloki nagłówka segmentów wycofania inne niż bloki nagłówka systemowego segmentu wycofania
- undo block - liczba oczekiwań na bufory zawierające bloki segmentów wycofania inne niż bloki systemowego segmentu wycofania

Celem strojenia jest uzyskanie wartości oczekiwań na poziomie 1% wszystkich oczekiwań drogą ewentualnego zwiększenia liczby segmentów wycofania.

```
Przykład.

SQL> select class, count
  2   from v$waitstat
  3   where class in ('system undo header', 'system undo block',
  4    'undo header', 'undo block' );

CLASS                   COUNT
------------------ ----------
system undo header           0
system undo block            0
undo header                 15
undo block                  10

SQL> select sum(value) from v$sysstat
  2   where name IN ('db block gets', 'consistent gets' );

SUM(VALUE)
----------
    271938
```

W przykładzie wyświetlono statystyki dotyczące segmentów wycofania oraz ogólną liczbę żądań jako wartość odniesienia.

Pliki dziennika powtórzeń (Redo Logs)

Spory polegają na próbach jednoczesnego pozyskania następujących rezerwacji (latch):

- redo allocation latch – rezerwacja alokacji miejsca dla pozycji zmiany w buforze dziennika powtórzeń
- redo copy latches – rezerwacja uzyskiwana na czas kopiowania pozycji zmian do bufora dziennika powtórzeń (spór występuje tylko na maszynach wieloprocesorowych)

Obserwacji podlegają statystyki zawarte w perspektywie V$LATCH:

- GETS – liczba żądań pozyskania rezerwacji w trybie willing-to-wait (uzyskanie rezerwacji jest warunkiem kontynuowania procesu)
- MISSES – liczba nieudanego inicjalnego pozyskania rezerwacji w trybie willing-to-wait
- SLEEPS – liczba kolejnych nieudanych pozyskań rezerwacji w trybie willing-to-wait
- IMMEDIATE GETS – liczba udanych pozyskań rezerwacji w trybie immediate (uzyskanie rezerwacji nie jest konieczne do kontynuacji procesu)
- IMMEDIATE MISSES – liczba nieudanych pozyskań rezerwacji w trybie immediate

Celem strojenia jest uzyskanie wartości misses/gets oraz immediate_misse/(immediate_misses+immediate_gets) na poziomie nie przekraczającym 1%.

```
Przykład.

SQL> select ln.name, gets, misses, immediate_gets, immediate_misses, sleeps
  2   from v$latch l, v$latchname ln
  3   where ln.latch#=l.latch#
  4         and ln.name in ('redo allocation', 'redo copy');

NAME                      GETS     MISSES IMMEDIATE_GETS IMMEDIATE_MISSES     SLEEPS
-------------------- ---------- ---------- -------------- ---------------- ----------
redo allocation           94821         34              0                0         77
redo copy                     0          0          82591               61          0
```

W przykładzie wyświetlono wartości statystyk dla rezerwacji związanych z buforami dziennika powtórzeń. Obliczone wartości mieszczą się w przedziale optymalnym.

Kontrola oczekiwań

Istotnym elementem pomagającym w ocenie wydajności bazy jest kontrola czasu zużytego na oczekiwanie na wykonanie operacji przez system lub spędzonego na oczekiwaniu na pozyskanie zasobu

lub rezerwacji. W tym celu korzystne jest sprawdzanie wartości w perspektywie V$SYSTEM_EVENT w kolumnach:

- EVENT – nazwa zdarzenia
- TOTAL_WAITS – ogólna liczba oczekiwań dla danego zdarzenia
- TOTAL_TIMEOUTS – ogólna liczba timeoutów dla danego zdarzenia
- TIME_WAITED – czas spędzony na oczekiwaniu dla danego zdarzenia (w setnych częściach sekundy)
- AVERAGE_WAIT – średni czas spędzony na oczekiwaniu dla danego zdarzenia (w setnych częściach sekundy)

Przykład.

```
SQL> select event,total_waits,total_timeouts,time_waited,average_wait
  2    from v$system_event
  3   order by time_waited desc;

EVENT                          TOTAL_WAITS TOTAL_TIMEOUTS TIME_WAITED AVERAGE_WAIT
------------------------------ ----------- -------------- ----------- ------------
rdbms ipc message                    33205          30502    12618485          380
pmon timer                            8134           8118     2637680          324
smon timer                              84             78     2612494        31101
SQL*Net message from client          94635              0     2487695           26
jobq slave wait                         84             82       25728          306
control file parallel write           7737              0        6663            1
control file sequential read          2994              0        3455            1
db file sequential read               1416              0        2550            2
db file parallel write                 546            546        1749            3
log file parallel write               2793           2566        1595            1
log file sync                         1545              0         721            0
latch free                             377            371         678            2
rdbms ipc reply                         88              0         493            6
control file heartbeat                   1              1         410          410
db file scattered read                 146              0         307            2
log file sequential read                11              0         105           10
sort segment request                     1              1         104          104
async disk IO                          105              0          78            1
direct path read                        40              0          35            1
library cache load lock                  4              0          30            8
refresh controlfile command              1              0          29           29
SQL*Net message to client            94637              0          26            0
process startup                          1              0          21           21
local write wait                        47              0          11            0
db file single write                    27              0           9            0
LGWR wait for redo copy                 60              1           8            0
SQL*Net more data to client            935              0           7            0
buffer busy waits                        7              0           5            1
SQL*Net more data from client          455              0           5            0
direct path write                       29              0           5            0
log file single write                    7              0           4            1
SQL*Net break/reset to client           90              0           4            0
reliable message                         1              0           2            2
checkpoint completed                     1              0           0            0
instance state change                    1              0           0            0
```

W przykładzie wyświetlono zawartość perspektywy opisującej oczekiwania na zdarzenia systemowe w porządku malejącego czasu. Niektóre wysokie czasy są charakterystyczne dla systemu i nie oznaczają problemów wydajnościowych np. związane z czasem reakcji klienta bazy.

15. Zestawienie statycznych perspektyw systemowych

Poniżej przedstawiono zestawienie ważniejszych perspektyw systemowych przydatnych w administracji bazy danych. Zestawienia wykonano dla perspektyw typu DBA_.

Nazwa	Opis
DBA_ALL_TABLES	Obiektowe i relacyjne tabele w bazie
DBA_ASSOCIATIONS	Zdefiniowane przez użytkownika statystyki
DBA_AUDIT_EXISTS	Zapisy w auditingu dotyczące nieistnienia obiektu wyspecyfikowanego w zapytaniu
DBA_AUDIT_OBJECT	Zapisy auditingu dotyczące wszystkich obiektów w systemie
DBA_AUDIT_SESSION	Zapisy auditingu związane z połączeniami i odłączeniami od bazy
DBA_AUDIT_STATEMENT	Zapisy auditingu dotyczące wykonanych operacji: GRANT, REVOKE, AUDIT, NOAUDIT, ALTER SYSTEM
DBA_AUDIT_TRAIL	Wszystkie zapisy auditingu
DBA_BLOCKERS	Sesje trzymające blokady obiektów, które chcą pozyskać inne sesje, a nie oczekujące na zwolnienie innych blokad
DBA_CATALOG	Tabele, perspektywy, synonimy i sekwencje w bazie
DBA_CLUSTERS	Grona w bazie
DBA_CLUSTER_HASH_EXPRESSIONS	Funkcje mieszające w bazie użyte do budowy gron
DBA_CLU_COLUMNS	Kolumny wchodzące w skład gron.
DBA_COLL_TYPES	Typy zbiorowe (kolekcje) w bazie np. macierze, zagnieżdżone tabele, tabele obiektowe
DBA_COL_COMMENTS	Opisy kolumn (komentarze) tabel I perspektyw
DBA_COL_PRIVS	Przywileje nadane na kolumny tabel
DBA_CONSTRAINTS	Definicje więzów na tabelach w bazie
DBA_CONS_COLUMNS	Kolumny w bazie wyspecyfikowane w definicji więzów
DBA_DATA_FILES	Pliki tworzące bazę danych
DBA_DB_LINKS	Linki bazodanowe
DBA_DEPENDENCIES	Zależności pomiędzy składowanymi jednostkami programowymi np. procedurami, pakietami, wyzwalaczami. Dotyczy również perspektyw zdefinowanych bez użycia linków bazodanowych
DBA_DIMENSIONS	Wymiary zdefiniowane w bazie
DBA_DIM_ATTRIBUTES	Zależności pomiędzy poziomami wymiarów i funkcjonalnie zależnymi kolumnami tabel
DBA_DIM_CHILD_OF DBA_DIM_HIERARCHIES DBA_DIM_JOIN_KEY DBA_DIM_LEVELS DBA_DIM_LEVEL_KEY	Szczegóły dotyczące definicji wymiarów w bazie
DBA_DIRECTORIES	Katalogi zdefiniowane w bazie
DBA_ERRORS	Opis błędów, które wystąpiły podczas kompilacji składowanych jednostek programowych
DBA_EXP_FILES	Opis plików eksportu
DBA_EXP_OBJECTS	Obiekty wyeksportowane przyrostowo
DBA_EXP_VERSION	Numer wersji ostatniej sesji eksportu
DBA_EXTENTS	Opis obszarów tworzących wszystkie segmenty w bazie
DBA_FREE_SPACE	Lista wolnych obszarów w przestrzeniach tabel
DBA_FREE_SPACE_COALESCED	Statystyki spójnej przestrzeni w bazie
DBA_INDEXES	Indeksy w bazie
DBA_INDEXTYPES	Opis typów indeksowych w bazie
DBA_INDEXTYPE_OPERATORS	Lista operatorów działających na typach obiektowych

DBA_IND_COLUMNS	Zaindeksowane kolumny w tabelach i gronach
DBA_IND_EXPRESSIONS	Wyrażenia użyte do zdefiniowania indeksów opartych o funkcje
DBA_IND_PARTITIONS	Opis partycji indeksów
DBA_IND_SUBPARTITIONS	Opis subpartycji indeksów
DBA_INTERNAL_TRIGGERS	Wewnętrzne wyzwalacze w bazie
DBA_JOBS	Zadania w bazie
DBA_JOBS_RUNNING	Uruchomione zadania w bazie
DBA_LIBRARIES	Biblioteki w bazie
DBA_LOBS	Obiekty typu BLOB i CLOB w bazie
DBA_LOB_PARTITIONS	Partycje LOB w bazie
DBA_LOB_SUBPARTITIONS	subpartycje LOB w bazie
DBA_LOCKS	Lista blokad i rezerwacji trzymanych w bazie
DBA_METHOD_PARAMS DBA_METHOD_RESULTS	Szczegóły (parametry, wyniki) metod określonych na typach.
DBA_MVIEW_AGGREGATES	Funkcje grupowe użyte w agregatach zmaterializowanych widokach
DBA_MVIEW_ANALYSIS	Zmaterializowane widoki możliwe do użycia podczas przepisywania zapytań (query rewrite)
DBA_MVIEW_DETAIL_RELATIONS DBA_MVIEW_JOINS DBA_MVIEW_KEYS	Szczegóły definicji zmaterializowanych widoków
DBA_NESTED_TABLES	Zagnieżdżone tabele
DBA_OBJECTS	Wszystkie obiekty w bazie
DBA_OBJECT_SIZE	Rozmiar w bajtach obiektów PL/SQL
DBA_OBJECT_TABLES	Tabele obiektowe w bazie
DBA_OBJ_AUDIT_OPTS	Ustawiona obserwacja na obiektach
DBA_OPERATORS	Operatory w bazie
DBA_OPANCILLARY DBA_OPARGUMENTS DBA_OPBINDINGS	Szczegóły związane z operatorami
DBA_OUTLINES	Wymuszone plany wykonania dla optymalizatora, dla określonych zapytań (outline)
DBA_OUTLINE_HINTS	Zbiór podpowiedzi (hint) dla składowanych outline'ów
DBA_PARTIAL_DROP_TABS	Tabele na których nie zakończyła się całkowicie operacja DROP COLUMN
DBA_PART_COL_STATISTICS	Statystyki i histogramy dla kolumn tabel
DBA_PART_HISTOGRAMS	Szczegóły danych dla histogramów
DBA_PART_INDEXES	Informacje o partycjach indeksów
DBA_PART_KEY_COLUMNS	Kolumny tworzące klucze tabel i indeksów spartycjonowanych
DBA_PART_LOBS	Spartycjonowane LOBy
DBA_PART_TABLES	Spartycjonowane tabele
DBA_PENDING_TRANSACTIONS	Nie rozstrzygnięte transakcje
DBA_POLICIES	Polityki bezpieczeństwa drobnoziarnistego dostępu do danych
DBA_PRIV_AUDIT_OPTS	Przywileje systemowe obserwowane przez auditing
DBA_PROFILES	Profile i określone dla nich limity
DBA_QUEUES DBA_QUEUE_SCHEDULES DBA_QUEUE_TABLES	Perspektywy związane z implementacją mechanizmu kolejek (Advanced Queuing)
DBA_RCHILD DBA_REFRESH DBA_REFRESH_CHILDREN DBA_RGROUP	Perspektywy związane z replikacją danych
DBA_REFS	Kolumny i atrybuty REF w kolumnach typu obiektowego
DBA_REGISTERED_SNAPSHOTS	Zarejestrowane migawki w bazie
DBA_REGISTERED_SNAPSHOT_GROUPS	Grupy replikowanych migawek w danej lokalizacji

	(dotyczy Advanced Replication Option)
DBA_ROLES	Role w bazie
DBA_ROLE_PRIVS	Role przypisane użytkownikom lub innym rolom
DBA_ROLLBACK_SEGS	Segmenty wycofania
DBA_RSRC_CONSUMER_GROUPS DBA_RSRC_CONSUMER_GROUP_PRIVS DBA_RSRC_MANAGER_SYSTEM_PRIVS DBA_RSRC_PLANS DBA_RSRC_PLAN_DIRECTIVES	Perspektywy związane z zarządzaniem zasobami poprzez pakiet DBMS_RESOURCE_MANAGER
DBA_SEGMENTS	Alokacje przestrzeni dla segmentów w bazie
DBA_SEQUENCES	Sekwencje w bazie
DBA_SNAPSHOTS	Migawki w bazie
DBA_SNAPSHOT_LOGS	Logi migawek w bazie
DBA_SNAPSHOT_LOG_FILTER_COLS	Kolumny podlegające logowaniu w logach migawek
DBA_SNAPSHOT_REFRESH_TIMES	Informacje o odświeżaniu migawek
DBA_SOURCE	Kod źródłowy jednostek programowych składowanych w bazie
DBA_STMT_AUDIT_OPTS	Ustawione obserwacje zdarzeń systemowych
DBA_SUBPART_COL_STATISTICS	Statystyki I histogramy dla subpartycji
DBA_SUBPART_HISTOGRAMS	Dane dla histogramów subpartycji
DBA_SUBPART_KEY_COLUMNS	Kolumny tworzące klucze tabel i indeksów spartycjonowanych dla subpartycji
DBA_SYNONYMS	Synonimy w bazie
DBA_SYS_PRIVS	Przywileje systemowe przypisane użytkownikom lub rolom
DBA_TABLES	Tabele w bazie
DBA_TABLESPACES	Przestrzenie tabel tworzące bazę
DBA_TAB_COLUMNS	Kolumny tabel, perspektyw i gron
DBA_TAB_COL_STATISTICS	Statystyki i histogramy kolumn
DBA_TAB_COMMENTS	Komentarze do tabel i perspektyw
DBA_TAB_HISTOGRAMS	Histogramy na kolumnach tabel
DBA_TAB_MODIFICATIONS	Informacje o modyfikacjach tabel (insert, update, delete, truncate) wykonanych od czasu ostatniego zebrania statystyk. Dotyczy tabel z ustawionym atrybutem MONITORING
DBA_TAB_PARTITIONS	Partycje tabel
DBA_TAB_PRIVS	Przywileje obiektowe przyznane użytkownikom
DBA_TAB_SUBPARTITIONS	Subpartycje tabel
DBA_TEMP_FILES	Pliki wchodzące w skład tymczasowych przestrzeni tabel
DBA_TRIGGERS	Wyzwalacze w bazie
DBA_TRIGGER_COLS	Użycie kolumn w wyzwalaczach
DBA_TS_QUOTAS	Limity miejsca przyznane użytkownikom w przestrzeniach tabel
DBA_TYPES	Typy obiektowe w bazie
DBA_TYPE_ATTRS	Atrybuty typów obiektowych
DBA_TYPE_METHODS	Metody zaimplementowane na typach obiektowych
DBA_UNUSED_COL_TABS	Tabele zawierające nieużywane kolumny
DBA_UPDATABLE_COLUMNS	Kolumny w modyfikowalnych perspektywach (updatable join view), które mogą być modyfikowane
DBA_USERS	Użytkownicy bazy
DBA_USTATS	Statystyki definiowane przez użytkowników
DBA_VARRAYS	Obiekty typu VARRAY w bazie
DBA_VIEWS	Perspektywy w bazie
DBA_WAITERS	Sesje oczekujące na blokadę (lock) nie trzymające jednocześnie blokad, na które czekają inne sesje
ROLE_ROLE_PRIVS	Role przyznane innym rolom
ROLE_SYS_PRIVS	Przywileje systemowe przyznane rolom

ROLE_TAB_PRIVS	Przywileje systemowe przyznane rolom

16. Bibliografia

1. Oracle8i Concepts, Part No. A76965-01
   Release 2
   Primary Author: Lefty Leverenz
2. Oracle8i Administrator's Guide, Part No. A76956-01
   Release 2
   Primary Authors: Ruth Baylis, Joyce Fee
3. Oracle8i Backup and Recovery Guide, Part No. A76993-01
   Release 2
   Primary Authors: Connie Dialeris, Joyce Fee
4. Oracle8i Designing and Tuning for Performance, Part No. A76992-01
   Release 2
   Primary Author: Michele Cyran
5. Oracle8i Reference, Part No. A76961-01
   Release 2
   Primary Author: Diana Lorentz
6. Oraclei SQL Reference, Part No. A85397-01
   Release 3
   Primary Author: Diana Lorentz
7. SQL*Plus, User's Guide and Reference, Part No. A82950-01
   Release 8.1.7
   Primary Author: Simon Watt
8. Wybrane zagadnienia z baz danych
   Wanda Gryglewicz-Kacerka
   Rafał Grzybowski
   Bohdan Szymczak
   Wydawnictwo Tempus, 2000

Część II

Magic 8i

Administracja bazą danych

17. Wstęp

Magic jest wydajnym, łatwym i wygodnym systemem służącym do tworzenia aplikacji opartych o nowoczesne relacyjno obiektowe bazy danych. Magic jest narzędziem nowej generacji określanym mianem RAD (Rapid Application Development). Programowanie w Magic polega na wybieraniu predefiniowanych opcji z rozwijalnego menu, ikon i okien dialogowych. Magic zawiera około 300 wbudowanych funkcji oraz zapewnia możliwość dopisywania własnych funkcji użytkownika. Środowiskiem RAD zarządzają tabele.

Nowoczesne narzędzie programistyczne typu RAD powinno posiadać następujące cechy:

- heterogeniczne środowisko i przetwarzanie transakcyjne heterogenicznych baz danych
- dynamiczna adaptacja do zmiennych warunków pracy
- szeroki zbiór rodzajów aplikacji (łącznie z aplikacjami wielowątkowymi)
- zdolność szybkiego przejścia od prototypu do gotowego produktu
- możliwość programowania zespołowego
- możliwość tworzenia aplikacji w czasie rzeczywistym
- przenośność aplikacji (inny system operacyjny, inna platforma sprzętowa)
- integracja aplikacji z innymi zewnętrznymi narzędziami i usługami

Magic jest narzędziem posiadającym takie właśnie cechy. Magic jest jednym z niewielu narzędzi na rynku oferującym zintegrowane, jednorodne środowisko tworzenia aplikacji obejmującym:

- architekturę klient-server
- modele komputerowego przetwarzania sieciowego
- internet

Bardzo ważną cechą tego nowoczesnego narzędzia jest możliwość pracy na wielu platformach sprzętowych (np. PC, IBM RISC, SUN SPARC, SILICON GRAPHICS i innych). Aplikacja Magic może pracować pod kontrolą prawie dowolnego systemu operacyjnego.

18. Magic – narzędzie do tworzenia aplikacji

18.1 Technologia Magic

Tworzenie aplikacji w Magic prowadzi do stworzenia pliku kontrolnego (sterującej bazy danych), który jest następnie analizowany przez maszynę wirtualną czyli motor Magica (binarium) zawierającą wbudowane funkcje przetwarzania bazy danych oraz dane dotyczące platformy sprzętowej i systemu operacyjnego. Takie rozwiązanie pozwala nie tworzyć każdorazowo kodów wykonywalnych. Magic stosuje maszynę wirtualną nazywaną motorem (Magic Application Engine) inną dla każdej platformy sprzętowej. Aplikacja zawiera słownik typów, tabel, formatek ekranowych, raportów oraz zdefiniowanego słownika zadań i powiązań. Projekt aplikacji Magic zawiera struktury danych, operacje obsługujące programy i wiążące struktury danych oraz zabezpieczenia i wygląd (formatki). Przy zastosowaniu maszyny wirtualnej aplikacja (plik kontrolny) jest bazą danych, która może być przez dane środowisko hardwarowo-systemowe akceptowane lub nie. W związku z tym jest możliwa oczywiście przenaszalność aplikacji, ale stworzona aplikacja nie jest tak do końca niezależna od platformy sprzętowej na jakiej ma pracować.

Maszyna wirtualna Magic może pracować w dwóch trybach pracy:

- narzędziowym (toolkit), który pozwala tworzyć i modyfikować aplikację
- wykonawczym (run-time), który pozwala testować i sprawdzać poprawność działania aplikacji (i oczywiście jest to tryb, w którym aplikacja już pracuje u klienta)

Aplikacje Magic mogą współpracować z bazami danych tworzonymi w różnych standardach. Jest to możliwe dzięki specjalnym produktom (gateway) dołączanym do maszyny wirtualnej.

19. Obiekty w bazie Magic

19.1 Typy danych

Typy danych definiują klasy zmiennych lub pól w tabelach, które współdzielą atrybuty, opis wizualnej reprezentacji danych (ang. picture definition) i inne własności.

Typ danych może być:

- alfanumeryczny
- numeryczny
- logiczny
- data/godzina
- memo
- obiekt Ole

Definiując typ danych musimy określić jego pojemność oraz ewentualnie zakres dopuszczalnych wartości. Oznacza to, że typowi danych można przypisać program selekcji wartości.

Typy danych definiuje się w tablicy typów (type dictionary). Typy danych mogą mieć nadane dowolne nazwy, wygodnie jest nadać nazwę taką jaką może mieć odpowiednie pole bazy danych. W Magic tworzenie nowego pola lub zmiennej w oparciu o definicję typu wybranego z tablicy typów powoduje, że domniemana nazwą tworzonego pola jest właśnie nazwa typu. Pole bez specjalnie nadanej nazwy dziedziczy swą nazwę oraz pozostałe atrybuty od macierzystego typu.

Należy pamiętać, że typy pól w różnych tabelach nie zdefiniowane w repozytorium typów nie mogą być automatycznie modyfikowane.

Używanie typów danych nie jest wymagane, ale wprowadza istotne usprawnienia:

- Pozwala oszczędzić czas przy definiowaniu tabel. Jeśli typ danych jest umieszczony w repozytorium typów, można skrócić proces definiowania kolumn w tabeli wskazując wybrany typ jej danych.
- Łatwość zarządzania. Każda modyfikacja atrybutu wcześniej zdefiniowanego w repozytorium typów przenoszona jest automatycznie do wszystkich pól tego typu.
- Zapewnienie zgodności typów pól z różnych tabel przy definiowaniu między nimi relacji, a także przy przekazywaniu parametrów.

19.2 Tabele

Tabele są podstawową strukturą w bazie danych Magic. Tabele zawierają dane opisujące ściśle obiekt lub zdarzenie, który modelują. Wszystkie tabele składają się z pól i rekordów. Tabele definiujemy w repozytorium tabel (file dictionary). Definiujemy nazwę tabeli oraz jej strukturę to znaczy podział na pola. Dokładną specyfikację pól tabeli zawiera tabela pól (fields). Dla pól tabeli należy określić typ danych, atrybut, obraz i zakres. Pole nazywane atrybutem jest najmniejszą strukturą w relacyjnym modelu logicznym. Pola są wykorzystywane do przechowywania jednostkowych danych. Rekord (krotka) jest strukturą składową tabeli, reprezentującą pojedynczą instancję jej tematu.

19.3 Indeksy

Indeksy pozwalają sortować dane wyświetlane w tabeli. Indeks może zawierać wiele segmentów według których wykonywane jest sortowanie. Sortowanie odbywa się w kolejności zadeklarowania segmentów.

19.4 Formatki

Magic zawiera niezbędne narzędzia potrzebne do stworzenia w pełni funkcjonalnego i przyjaznego interfejsu użytkownika (takie jak pole listy, pole combo box i inne elementy standardowego wyposażenia niezbędnego do stworzenia formatek ekranowych. .

19.5 Programy

Magic wyświetla dane z bazy używając programów.

Program (albo **zadanie**) jest to zbiór reguł, który określa kolejne operacje przeprowadzane na wybranej tabeli, aby np. wyświetlić dane na ekranie.

Istnieją dwa typy zadań:

- Wsadowe (ang. batch tasks) - używane do nieinteraktywnego przetwarzania danych, bez interwencji użytkownika, po stronie klienta lub serwera
- Interakcyjne (ang. online tasks) - zawierają interfejs użytkownika do operowania na pojedynczym rekordzie lub liście rekordów z tabeli danych. Służą do wprowadzania danych, sprawdzania poprawności danych, wyświetlania raportów itp.

Programy to obiekty w bazie Magic, które zawarte są w repozytorium programów (program dictionary). Programy mogą zostać wygenerowane automatycznie lub utworzone przez użytkownika. Automatyczną generację programów wykorzystuje się w przypadku interakcyjnego przeglądania pól bazy danych.

W Magic każdy program ma swoją strukturę (program tree). Schemat struktury programu to jego podział na zadania (task), które powinien wykonać. W ogólnym przypadku program może składać się z wielu zadań. Można przyjąć w dużym uproszczeniu, że pojedyncze zadanie zajmuje się przetwarzaniem pojedynczego pliku. Każde zadanie posiada szereg atrybutów – właściwości, wśród których można wybrać tryb redagowania rekordów w chwili wywołania programu (przeglądanie, modyfikacja, wyszukiwanie rekordów, sortowanie i inne właściwości). Zadania mogą się wywoływać inne zadania lub inny program (zgodnie z hierarchią drzewa lub inny program).

W Magic przetwarzanie danych w poszczególnych rekordach odbywa się w taki sam sposób. Realizacja zadania przetwarzania rekordów odbywa się według z góry ustalonego przez Magic schematu. Można tu wyodrębnić następujące fazy:

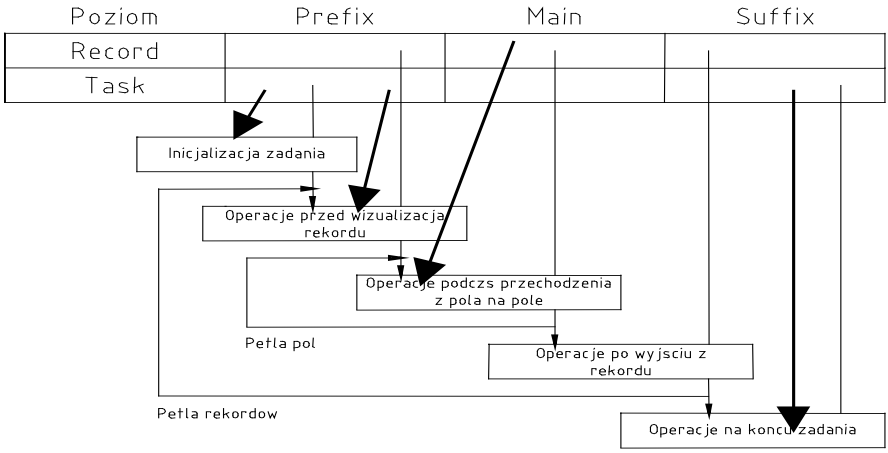
Każda faza zadania ma własną tablicę operacji. Poszczególne wiersze tablicy operacji przedstawiają pojedynczą operację. W Magic różnych operacji jest około 13 (select, update, block, link, call, ...).

Przetwarzanie pojedynczego rekordu podzielone jest na trzy etapy:

- prefix - czynności do wykonania po otwarciu bieżącego rekordu (w operacjach możemy korzystać z danych pobranych z tego rekordu, ale jest to faza nieinterakcyjna. W tej fazie wykonują się operacje przygotowujące rekord do fazy interakcyjnej)
- main - kolejne operacje do wykonania na polach pojedynczego rekordu (Jedyna w zadaniu faza interakcyjna, w której użytkownik panuje nad zadaniem, ale zasada poruszania się po polach odpowiada kolejności umieszczonych poleceń SELECT w tej fazie)
- suffix - czynności do wykonania po opuszczaniu bieżącego rekordu (ale dopiero po wykonaniu tej fazy następuje zapis rekordu i przejście do kolejnego)

Przetwarzanie całego zadania podzielone jest na dwa etapy:

- prefix - wykonywane są czynności przygotowawcze do wykonania zadania np. nadawanie wartości początkowych zmiennym roboczym
- suffix – wykonywane są czynności zamykające zadanie np. kasowanie plików roboczych



Rysunek 1. Schemat działania maszyny wirtualnej Magic

Lista wybranych operacji w Magic

W Magic jest wyróżnionych około 13 możliwych operacji na danych. Oto najczęściej używane operacje:

- **select**

Główne zadanie operacji select to deklarowanie zmiennych. Występują dwie operacje select:

- select real – deklaruje zmienne rzeczywiste (pola z tabeli lub tabel)
- select virtual – deklaruje zmienne robocze, służy do tworzenia i inicjacji zmiennych roboczych (zmiennych lokalnych)

Operacja select może wystąpić tylko w fazie main. W pozostałych fazach przetwarzania rekordu użycie operacji select jest zabronione. Jej zasięg jest jednak szerszy i obejmuje również fazy prefix i suffix. Operacji select virtual można przypisać dowolne wyrażenie do wykonania.

- **update**

Operacja update realizuje przypisanie. Wskazanej zmiennej rzeczywistej lub roboczej (wirtualnej) zostanie przypisana wartość wyrażenia. Operacja update realizuje dwa tryby działania:

- update/N – tryb normalny odpowiada przypisaniu, w którym wskazanej zmiennej zostaje przypisana wartość wyrażenia, natomiast poprzednia wartość tej zmiennej ulegnie zamazaniu
- update/I – tryb inkrementalny, wartość obliczonego wyrażenia nie zastąpi tu pierwotnej wartości wskazanego pola lecz zostanie z nią zsumowana.

- **call**

Operacja call ma cztery warianty, najczęściej używane to:

- call task – wywołuje podzadanie

Wywołanie podzadania związane jest ścisłe z przekazywaniem parametrów wywołania. Przekazywanie parametrów odbywa się za pośrednictwem tablicy parametrów.

Parametry mogą być przekazywane przez wartość (parametr wywołania jest wyrażeniem) lub przez zmienną (parametr wywołania jest zmienną rzeczywistą lub roboczą). (wywołując podzadanie nie jest konieczne jawne przekazanie parametrów, ponieważ będąc w tym podzadaniu mamy wgląd – nawet możliwość modyfikacji zmiennych zadeklarowanych w tym zadaniu)

- call program – wywołuje samodzielny program ze słownika programów aplikacji (przekazanie parametrów poprzez tablicę parametrów jest konieczne)

- **link – end link**

Podstawowym zastosowaniem operacji link jest ustanowienie relacji między rekordami pliku głównego a plikiem dołączonym takiej, że każdemu rekordowi pliku głównego odpowiada jeden rekord pliku dołączonego (tylko jeden rekord z tablicy dołączonej zostanie pobrany)

- **block - end block**

Operacja block oraz end block tworzy parę nawiasów składniowych, które mogą obejmować grupę operacji tworzących blok. Zawartość bloku może być traktowana w programie jak pojedyncza operacja. W bloku wszystkie operacje wykonywane są w porządku zapisania.

- **evaluate exp**

Operacja evaluate exp powoduje obliczenie wyrażenia stanowiącego jego argument (wykonanie konkretnego polecenia zadeklarowanego wyrażeniem)

- **browse on exp**

Operacja browse on exp umożliwia w programie Magic przeglądanie lub edycję dowolnych niesformatowanych plików tekstowych.

- **exit on exp**

Operacja exit on exp jest wyjściem z Magic do systemu operacyjnego. (polecenie to służy do wywołania z poziomu magica programu zewnętrznego systemu operacyjnego, np. uruchomienie kalkulatora z pakietu Windows)

- **verfity exp**

Operacja verfity on exp jest narzędziem diagnostycznym, służącym głównie do zatrzymania – zablokowania użytkownika w wybranym polu – miejscu, do czasu, gdy np. w wybrane pole zostanie wprowadzona oczekiwana wartość).

- **output form**

Operacja output form służy do sformatowanego wyprowadzania danych do pliku zewnętrznego. Operacja ta wymaga zdefiniowania formularza stanowiącego właściwy projekt wydruku. (Jest to uniwersalna operacja służąca do wysłania z Magica na dowolne urządzenie I/O konkretnego formularza. Tym urządzeniem może być: plik tekstowy, ekran monitora, drukarka ... Ogólnie jest to polecenie pozwalające na tworzenie raportów.)

- **input form**

Operacja input form służy do wprowadzania danych z plików zewnętrznych. Niezbędnym parametrem operacji input form jest formularz opisujący format wczytywanych danych oraz odpowiadający na pytanie jakim zmiennym dane te mają zostać przypisane. (plik danych musi być odpowiednio sformatowanym plikiem tekstowym)

19.6 Pomoc

W Magic pomoc jest z założenia wbudowana w każdą opcję menu oraz plansze dialogowe. Dodatkowo każde pole edycyjne planszy dialogowej posiada możliwość wyboru pomocy. Plansza dialogowa jest samodzielnym obiektem, który można przydzielać w dowolny sposób planszom lub polom edycyjnym, Ta sama plansza dialogowa może być przydzielona kilku różnym polom edycyjnym.

20. Architektura klient-server w systemie Magic

Architektura klient-server jest jedną z najważniejszych koncepcji stosowanych w rozproszonych bazach danych. Wykorzystano w niej wiele aspektów dotyczących zasad przetwarzania rozproszonego, obliczeń współbieżnych oraz rozproszonego przetwarzania transakcji. Dzięki zastosowaniu systemu klient-server zmienił się sposób przetwarzania danych. Architektura klient-server jest systemem rozproszonym zapewniającym szybki i ekonomiczny dostęp do danych.

Termin klient-server jest ściśle związany z relacją między dwoma systemami albo procesami. W tej relacji klient jest systemem, który zleca jakieś zadanie systemowi zwanemu serwerem (określenie który z systemów jest klientem, a który serwerem polega na zdefiniowaniu relacji między nimi). Serwery na żądanie klienta wykonują określone usługi. W architekturze klient-server Magic przyjmujemy następujący podział:

- Klienci to stacje, na których działają aplikacje Magic. Te aplikacje wysyłają pewne zlecenia do serwerów. Klientem może być również pojedynczy proces na komputerze realizującym wiele procesów. Proces klienta stanowi najbliższą użytkownikowi warstwę aplikacji Magic.
- Serwery to procesy działające na wieloprocesorowych komputerach. Serwery zajmują się obsługą zleceń przychodzących od strony klienta (dostęp do danych, kolejka wydruku). Proces stanowi wewnętrzną warstwę aplikacji i zapewnia wykonanie usług wykorzystywanych przez wiele podobnych aplikacji.

W architekturze klient-server Magic można wyróżnić następujące warstwy:

dla **Klienta**

- jądro sterujące wykonywaniem programów
- menager plików, zajmujący się lokalnymi bazami danych i systemem operacyjnym
- warstwa bram, odpowiadająca za komunikację z innymi zdalnymi serwerami

dla **Serwera**

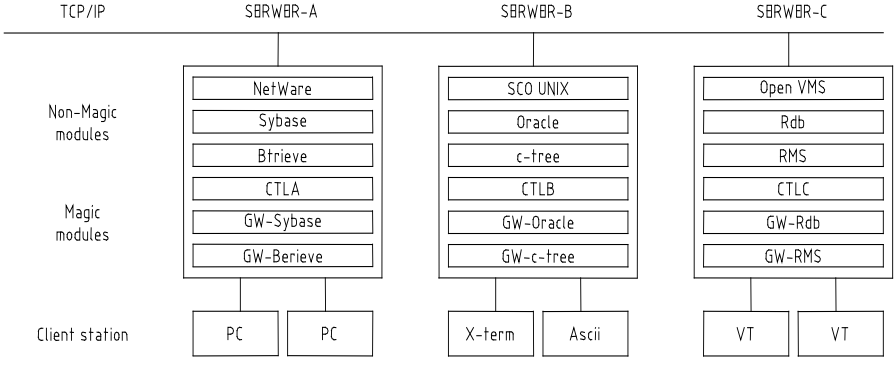
- warstwa logiczna serwera, odpowiedzialna za przyjmowanie zleceń
- manager plików, zajmujący się łącznością z lokalnymi bazami danych
- warstwa bram, zajmująca się komunikacją z innymi zdalnymi serwerami

Serwer Magic nie jest serwerem bazy danych, natomiast świadczy usługi dla klientów będącymi aplikacjami. Do realizacji żądań używane są bramy do systemów zarządzania bazami danych (np. RDBMS). Komunikacja pomiędzy systemami klient-server, działającymi na różnych platformach realizowana jest za pomocą protokołu TCP/IP.

Magic oferuje dwa modele pracy w architekturze klient-server:

- dane znajdują się na serwerze, a zbiór sterujący na stacji klienta
- zbiór sterujący wraz z danymi znajduje się na serwerze

Architektura klient-server w Magic zapewnia interoperacyjność dzięki niezależności od platformy sprzętowej, systemowej i bazy danych. Oznacza to, że aplikacja Magic może korzystać równocześnie z różnych baz danych znajdujących się na różnych platformach sprzętowych. Komunikacja między platformami jest realizowana za pomocą wybranych protokołów sieciowych (w zasadzie używane są tylko 2 protokoły: TCP/IP i NetSoft dla maszyn AS/400).



Rysunek 2. Sieć heterogenicznych modułów klient-serwer

Obecnie klientem może być Runtime pracujący tylko i wyłącznie w środowisku Windows, albo dowolna przeglądarka internetowa, a także dos - owy requester uruchamiający procesy na serverze.

21. System aplikacji internetowej

W Magic 8 istnieje możliwość tworzenia aplikacji internetowych bezpośrednio w środowisku generatora aplikacji. Takie rozwiązanie odznacza się następującymi cechami:

- pełne zintegrowane środowisko tworzenia aplikacji internetowych ze środowiskiem generatora
- wykorzystywanie tych samych narzędzi do tworzenia interfejsu dla aplikacji klient-serwer oraz dla aplikacji internetowych
- kod aplikacji internetowej jest w całości generowany przez Magic (tworzenie aplikacji internetowej nie wymaga znajomości języków Jawa i HTML)
- możliwość użycia standardowych elementów formatek WWW oraz ich rozszerzeń (ramek, apletów Jawy oraz elementów ActiveX)
- możliwość wykorzystania gotowych stron HTML napisanych w narzędziach trzecich i uzupełnianie ich za pośrednictwem Magica danymi pobranymi z bazy danych – taka forma tworzenia aplikacji internetowych jest wydajna i preferowana)
- możliwość wykorzystania techniki WebOnLine, gdzie strona HTML zawiera część logiki aplikacji, takie rozwiązanie obniża liczbę zleceń przeglądarki do servera HTTP)

Pełne funkcjonalne środowisko tworzenia i uruchomienia aplikacji internetowej składa się z:

- serwera aplikacji Magic
- programu koordynującego współpracę wielu serwerów Magic (Magic Requester Broker)
- serwera WWW (Web Server) oraz związanego z nim programu zapewniającego połączenie między serwerem WWW a programem koordynującym pracę wielu serwerów aplikacji Magic (Magic Requester)
- przeglądarki internetowej

Serwer aplikacji Magic zbudowany jest z Runtimu, który przetwarza tylko programy wsadowe oraz zbioru programów internetowych współpracujących z Magic. Programy te mają bezpośredni dostęp do bazy danych, przetwarzają tę bazę zgodnie z zapisanymi formułami logicznymi oraz prezentują wyniki w formie HTML. Serwer aplikacji Magic ściśle współpracuje z programem pośredniczącym Magic Requester Broker, który jest ogniwem łączącym serwer aplikacji i serwer WWW. Magic Requeater Broker pozwala na zwiększenie efektywności działania aplikacji poprzez realizację algorytmu wyrównywania obciążenia poszczególnych serwerów. Poprzez kontrolę pracy aplikacji na wielu serwerach pozwala na tworzenie złożonych wieloplatformowych systemów. (Tak naprawdę to bez Brokera nie ma mowy o tworzeniu czy uruchamianiu aplikacji internetowych bo produkt ten jest niezbędnym ogniwem łączącym cały produkt. To on decyduje o tym, który serwer, a nawet który proces na serwerze będzie przetwarzał nasze zapytanie, a także rzeczywiście rozwiązuje problem optymalnego obciążenia serwerów Magica)

Serwer WWW jest programem zarządzającym zasobami publikowanymi w internecie. Udostępnia on informacje w przeglądarkach (Netscape Web Server, Microsoft Internet Server, Oracle Web Server i inne). Środowisko Magic zawiera trzy ”requestery” (Magic Internet Requester) współpracujące z najbardziej rozpowszechnionymi serwerami WWW:

- requester CGI, pracujący w ogólnie akceptowanym standardzie CGI
- requester NSAPI, wykorzystujący funkcje Netscape Server API obsługiwane przez serwery WWW firmy Netscape
- requester ISAPI, wykorzystujący funkcje Microsoft Internet Server API obsługiwane przez servery WWW firmy Microsoft

Wybór requestera powinien być uzależniony od rodzaju serwera WWW, który będzie współpracował z aplikacją Magic.

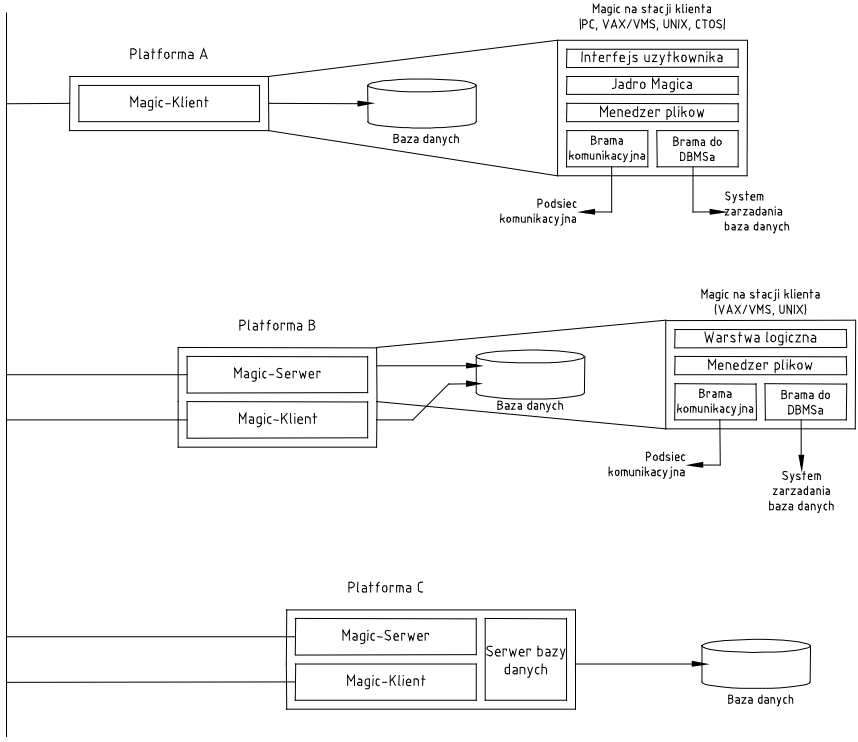
Przeglądarka internetowa (np. Netscape Communicator, Microsoft Internet Explorer) jest programem klienta służącym do wizualizacji danych udostępnionych w internecie.

W przypadku internetowej aplikacji Magic proces przepływu danych (podany na rysunku 3) jest następujący:

- użytkownik internetu, z poziomu przeglądarki łączy się z serwerem WWW, żądając wywołania programu Magic (najczęściej niejawnie poprzez wybór odpowiednio zdefiniowanego łącznika hypertekstowego)
- requester serwera WWW przekazuje zlecenie do programu Magic Requester Broker
- Magic Requester Broker wybiera jeden z dostepnych serwerów Magic, który obsługuje określoną aplikację Magic i wywołuje jej odpowiedni program. Magic Requester Broker zarządza układem zadań, to znaczy zarządza do którego serwera trafi nasze zapytanie. Po jego realizacji przekazuje odpowiedź serwerowi HTTP, który wysyła ją do przeglądarki)

- program Magic analizuje przekazane parametry wejściowe i zgodnie z zaimplementowaną logiką przetwarza dane oraz równocześnie generuje zwrotną stronę HTML, którą następnie przekazuje do serwera WWW
- serwer WWW przesyła wynik działania aplikacji Magic do aktualnie wybranej przez użytkownika internetu przeglądarki

Rysunek 3. Proces przepływu sterowania danymi przy pracy internetowej aplikacji Magic.



Rysunek 3. Proces przepływu danych

22. Aplikacja internetowa

Aplikacja internetowa jest zwykłą aplikacją Magic, która posiada dodatkowe programy służące do obsługi zleceń nadchodzących z sieci. Duże różnice związane są ze sposobem pracy aplikacji Magic oraz internetowej aplikacji Magic. Internetowa aplikacja Magic uruchomiona po stronie klienta nie ma ciągłego połączenia z serwerem aplikacji (w tradycyjnej aplikacji typu klient serwer połączenie z serwerem jest ciągłe). Internetowa aplikacja Magic generuje zlecenia do serwera, tam są przetwarzane a następnie wyświetlane są wyniki. W czasie pomiędzy kolejnymi zleceniami klient pracuje ze stronami HTML, które po przesłaniu do przeglądarki stają się niezależnymi obiektami (mającymi własny wygląd i sposób zachowania). Połączenie z serwerem występuje tylko w chwilach obsługi zleceń. Można przyjąć, że praca serwera aplikacji internetowej ma charakter pracy wsadowej a nie interakcyjnej.

Program internetowy cechuje się następującymi własnościami:

- posiada zdefiniowaną nazwę publiczną (pod którą będzie dostępny w internecie)
- jest programem wsadowym
- opcjonalnie przetwarza przesłane z zewnątrz parametry
- może swobodnie operować na bazie danych
- posiada wyjściową formatkę HTML (lub Javy)

22.1 Wywoływanie programów przez internet

Programy aplikacji Magic są wywoływane z przeglądarki użytkownika internetu pośrednio przez zlecenie kierowane do requestera. Wywołanie requestera można realizować z poziomu:

- przeglądarki – poprzez bezpośrednie podanie adresu URL requestera
- strony HTML
    - poprzez zdefiniowanie łącznika internetowego do odpowiedniego adresu URL
    - poprzez zdefiniowanie formy HTML z parametrem action wskazującym odpowiedni adres URL
- Dos -owego requestera: mgrqcmdl.exe

Parametry przekazywane do requesterów internetowych są następujące:

- appname – nazwa aplikacji
- prgname – nazwa programu w ramach aplikacji
- arguments – parametry przekazywane do programu
- priority – priorytet zlecenia
- user – użytkownik
- pass – hasło

Istnieją dwa sposoby przekazania parametrów do aplikacji:

- poprzez specyfikację parametrów w adresie URL
- poprzez użycie pól ukrytych

22.2 Automatyczna generacja programu internetowego

Generator aplikacji Magic pozwala na wygenerowanie prostego programu internetowego opartego o wybrana tabelę w bazie danych. Wygenerowany program może pełnić jedną z dwóch funkcji:

- Służyć do wprowadzenia kryteriów zapytań kierowanych do bazy
- Wyświetlać zawartość bazy danych (wyniki zapytań)

Wygenerowany program ma nadaną nazwę publiczną i jest dostępny w słowniku programów aplikacji. Poprzez tę nazwę publiczną program jest dostępny w sieci internetowej.

22.3 Formatki aplikacji internetowych

Formatki aplikacji internetowych można zaprojektować w oparciu o:

- Język HTML
- Aplet Javy
- Strony HTML (utworzone przy pomocy innych narzędzi, które następnie są łączone z danymi generowanymi przez Magic)

Każde z narzędzi tworzenia formatek aplikacji internetowych charakteryzuje się nieco innymi właściwościami. W ramach jednego programu istnieje możliwość łączenia formatek wykonanych przy pomocy różnych narzędzi. Narzędzia różnią się możliwością wykorzystania elementów formatek, ustaleniem własności użytych obiektów i sposobu ich rozmieszczenia, szybkości działania oraz możliwości wykorzystania zewnętrznych narzędzi projektowych. W Magic istnieje możliwość użycie ramek, czyli elementów pozwalających wyświetlać w jednym oknie przeglądarki kilku plików HTML.

22.4 Formatki HTML

Edytor formatek HTML pozwala na tworzenie stron HTML:

- statycznych
- dynamicznych

Strony HTML tworzone w edytorze mogą zawierać:

- tekst, tekst sformatowany, przycisk, rysunek, dźwięk
- elementy wprowadzania danych: pole edycyjne, combo box, list box, check box, radio button
- tabele
- ramki
- łączniki (odnośniki) internetowe
- dyrektywy dołączenia zewnętrznych plików HTML do pliku edytowanego
- komponenty Javy i ActiveX

Wiele z tych elementów może być zdefiniowanych w oparciu o zmienne zadeklarowane w programie, których wartości są ustalone dopiero w trakcie działania aplikacji. Jest to sposób na dynamiczne zmiany zawartości publikowanych stron.

Obiektami, które mają szczególne znaczenia są łączniki. Umożliwiają one:

- wywołanie programu Magic
- przejście do innego adresu URL
- przejście do innego miejsca w tym samym dokumencie

22.5 Ramki HTML

Wykorzystanie ramek pozwala na uzyskanie efektu podziału ekranu przeglądarki na kilka niezależnych obszarów, z których każdy może wyświetlać inny dokument HTML. Rozwiązanie to umożliwia między innymi trwałe wyświetlanie na ekranie pewnych informacji (np. adres, logo, spis treści..) bez potrzeby ich ponownego ładowania przy zmianie innych części ekranu. Edytor ramek wbudowany w Magic oferuje bardzo duże możliwości projektowania układu ramek oraz łączników.

22.6 Dołączanie zewnętrznych stron HTML

Magic pozwala na łączenie zaprojektowanych stron (utworzonych przy użyciu dowolnego narzędzia) z aplikacją internetową Magic. Strony takie służą do wyświetlania informacji statycznych a także poprzez wykorzystanie odpowiednich znaczników, mogą udostępniać dane z baz obsługiwanych przez aplikację Magic.

Zewnętrznie utworzone strony WWW są przechowywane poza środowiskiem aplikacji. Takie rozwiązanie ma szereg cech:

- Magic jest środowiskiem bazodanowym i nie pozwala wykorzystać wszystkich możliwości języka HTML
- do opracowywania stron WWW można wykorzystywać inne narzędzia (znane lub lubiane)
- przechowywanie stron WWW poza aplikacją umożliwia korektę wyglądu strony bez przerywania pracy z aplikacją Magic
- daje możliwość integracji z aplikacją już istniejących stron WWW

Dołączanie zewnętrznych stron HTML wykonuje operacja Output. Funkcja Output, jest universalną funkcją. Funkcja Output Merge formatkę wybieramy formatując formatkę. Operacja ta przetwarza plik tekstowy (np. plik HTML) i wyszukuje w nim specjalne znaczniki Magic, które następnie zastępuje wygenerowanymi w programie danymi. Tak przetworzona strona wysyłana jest do requestera. Dla operacji Output Merge definiuje się dwa parametry:

- tabelę parametry łączenia, w której specyfikuje się dane, które powinny zostać zapisane do zewnętrznego pliku HTML w miejsce zawartych w nim znaczników

- plik we/wy typu requester, w którego własnościach deklaruje się nazwę pliku zewnętrznego HTML i format znacznika rozpoznawalnego przez Magic

Wyróżniamy dwa rodzaje znaczników:

- sterujące np.<!$MGREPEAT> .... <!$MGENDREPEAT> - deklaracja dynamicznej tabeli
- znaczniki danych np. <!$MG_ZMIENNA>

22.7 Formatki Javy

Formatki zbudowane na bazie apletu Javy oferują znacznie bogatsze możliwości niż strony oparte na języku HTML. Umożliwiają one:

- Wykorzystanie tablic z suwakiem pozwalającym na przesuwanie zawartości tablicy
- Przeglądanie rekordów tablicy z automatyczną zmianą zawartości kontrolek zawierających dane rekordu, które pozostają poza tablicą
- Wykorzystanie obiektów typu zakładka, grupa i obiekty statyczne (prostokąt, elipsa, linia)
- pełność w pozycjonowaniu elementów
- prostą walidację wprowadzonych informacji
- kontrolę wiadomości wysyłanej w linii statusu
- uruchomienie formatki w oddzielnym oknie

Program Magic posiadający formatkę Javy, w wyniku wywołania generuje dwa pliki:

- stronę HTML
- pliki sterujące dla apletu Javy

22.8 Komponenty Javy i ActiveX

Magic umożliwia wykorzystanie posiadanych apletów Javy i kontrolek ActiveX. Komponenty ActiveX, w przeciwieństwie do apletów Javy muszą być zarejestrowane i zainstalowane na komputerze klienta przed ich uruchomieniem w przeglądarce. Kontrolki ActiveX mogą być ładowane z zasobów lokalnych, natomiast aplety Javy są zawsze ładowane z sieci.

23. Autoryzacja użytkowników

Każdy użytkownik bazy danych Magic musi być zarejestrowany w bazie poprzez unikalny identyfikator oraz hasło dostępu. Magic jest wyposażony w rozbudowany system kontroli uprawnień i praw dostępu to znaczy system autoryzacji. Autoryzacja funkcjonuje zarówno w trybie wykonawczym jak i narzędziowym pracy systemu.

Z każdym użytkownikiem wiążą się następujące elementy autoryzacji:

- Hasło tworzone wraz z użytkownikiem
- Weryfikacja użytkownika przy podłączaniu do bazy
- Możliwość weryfikacji przez system operacyjny

24. Bibliografia

1. The Magic Tutorial and Evaluation Guide, Magic Software Enterprises, 1993
2. The Magic Developers Guide and Reference, Magic Software Enterprises, 1993
3. The Magic Application Deployment and Report Generator Guide, Software Enterprises, 1992
4. Michał Mowiński, Maciej Lasota: Magic Obsługa Aplikacji i Generatora Raportów, tłumaczenie, PPH TELMAX Sp z o.o., 1995
5. Ronald Wacławek: Magic. Wstęp do programowania, Intersoft, 1994
6. Carl L. Hall: Techniczne podstawy systemów klient-serwer, WNT, Warszawa, 1996
7. Marcin Gorawski, Andrzej Konopacki: Magic koncepcja i metodyka, Software 4/95, str. 44-47
8. Marcin Gorawski, Adam Koziatek: Magic Architektura klient-serwer, Software 4/95, str. 48-51
9. Stanisław Plichta, Anna Jasińska-Suwała: Magic jako system do budowy oprogramowania wspomagającego pracę wyższej uczelni, internet magic.htm, 1998