



---

# WSPÓŁCZESNE TECHNOLOGIE INFORMATYCZNE

# EKSPLOATACJA BAZ

# DANYCH



Projekt Absolwent na miarę czasu współfinansowany przez Unię Europejską  
w ramach Europejskiego Funduszu Społecznego  
Nr umowy UDA-POKL.04.01.01-00-421/10-01



Politechnika Lubelska  
Wydział Elektrotechniki i Informatyki  
ul. Nadbystrzycka 38A  
20-618 Lublin

WSPÓŁCZESNE TECHNOLOGIE INFORMATYCZNE

# EKSPLOATACJA BAZ DANYCH

---

Piotr Muryjas

---

Maria Skublewska-Paszkowska

---

Dariusz Gutek

---



**Politechnika Lubelska**  
**Lublin 2011**

Recenzenci:

dr hab. Stanisław Grzegórski, prof. PL

dr inż. Marek Miłoś

Skład komputerowy: Piotr Muryjas

Publikacja finansowana z projektu „Absolwent na miarę czasu”

Projekt „Absolwent na miarę czasu” współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego. Nr umowy UDA-POKL.04.01.01-00-421/10-01

Ta publikacja odzwierciedla jedynie stanowiska jej autorów, a Komisja Europejska nie ponosi odpowiedzialności za informacje w niej zawarte

Publikacja dystrybuowana bezpłatnie

Publikacja wydana za zgodą Rektora Politechniki Lubelskiej

© Copyright by Politechnika Lubelska 2011

ISBN: 978-83-62596-63-8

Wydawca: Politechnika Lubelska

ul. Nadbystrzycka 38D, 20-618 Lublin

Realizacja: Biblioteka Politechniki Lubelskiej

Ośrodek ds. Wydawnictw i Biblioteki Cyfrowej

ul. Nadbystrzycka 36A, 20-618 Lublin

tel. (81) 538-46-59, email: wydawca@pollub.pl

[www.biblioteka.pollub.pl](http://www.biblioteka.pollub.pl)

Druk: ESUS Agencja Reklamowo-Wydawnicza Tomasz Przybylak

[www.esus.pl](http://www.esus.pl)

---

Elektroniczna wersja książki dostępna w Bibliotece Cyfrowej PL [www.bc.pollub.pl](http://www.bc.pollub.pl)

Nakład: 100 egz.

## Spis treści

---

Wstęp .....	7
1. Instalowanie systemu Oracle 11g .....	9
1.1. Ogólna charakterystyka procesu instalacji .....	10
1.2. Instalacja systemu Oracle 11g .....	11
1.3. Pytania kontrolne.....	16
2. Tworzenie bazy danych systemu Oracle 11g .....	17
2.1. Czynności przygotowawcze .....	18
2.2. Tworzenie bazy danych przy użyciu narzędzia DBCA .....	19
2.3. Manualne tworzenie bazy danych .....	29
2.4. Pytania kontrolne.....	38
3. Administrowanie bazą danych Oracle 11g .....	39
3.1. Wprowadzenie.....	40
3.2. Architektura bazy danych Oracle .....	40
3.3. Logiczne struktury przechowywania danych .....	42
3.4. Fizyczne struktury przechowywania danych .....	47
3.5. Struktury pamięci systemu Oracle.....	50
3.6. Strojenie użycia pamięci systemu Oracle.....	53
3.7. Zarządzanie obiektami bazy danych Oracle .....	58
3.8. Mechanizmy bezpieczeństwa bazy danych .....	74
3.9. Pytania kontrolne.....	86

4. Wirtualne bazy danych .....	87
4.1. Mechanizm VPD .....	88
4.2. Wstępna konfiguracja i tworzenie kontekstu .....	91
4.3. Tworzenie strategii bezpieczeństwa .....	95
4.4. Testowanie konfiguracji .....	102
4.5. Pytania kontrolne .....	106
5. Import i eksport danych .....	107
5.1. Oryginalne narzędzia Export i Import .....	108
5.2. Mechanizmy importu i eksportu Data Pump .....	115
5.3. Ładowanie danych przy pomocy SQL*Loader .....	125
5.4. Pytania kontrolne .....	130
6. Optymalizacja dostępu do danych .....	131
6.1. Znaczenie strojenia zapytań SQL .....	132
6.2. Plan wykonania zapytania .....	136
6.3. Optymalizatory .....	138
6.4. Optymalizacja składni zapytań .....	145
6.5. Pytania kontrolne .....	155
7. Kopie zapasowe i archiwizacja danych .....	157
7.1. Rodzaje kopii zapasowych. Kopie online i offline .....	158
7.2. Archiwizacja z wykorzystaniem narzędzia RMAN .....	163
7.3. Pytania kontrolne .....	183
8. Migracja danych .....	185
8.1. Sposoby przeprowadzania migracji danych .....	186
8.2. Asystent aktualizacji bazy danych .....	188
8.3. Bezpośrednia aktualizacja ręczna .....	189
8.4. Wykorzystanie mechanizmów importu i eksportu .....	190
8.5. Metoda oparta o kopiowanie danych .....	192
8.6. Pytania kontrolne .....	193
Literatura .....	195
Indeks .....	199



## Wstęp

---

Współczesne systemy informatyczne wykorzystują bazy danych, w których zaimplementowano zaawansowane rozwiązania technologiczne oraz sprzętowe. W związku z tym eksploatacja takiego środowiska wymaga głębokiej wiedzy, umiejętności i doświadczenia, które zapewnią wysoką efektywność przetwarzania danych oraz wysoki poziom bezpieczeństwa systemu.

Wybór właściwej bazy danych zależy głównie od jej przeznaczenia i powinien uwzględniać charakter procesów, które będą przez nią wspierane. Niezależnie od rodzaju wybranego środowiska, cykl życia każdej bazy rozpoczyna się od procesu instalacji systemu zarządzania, który będzie wspomagał i nadzorował wszystkie czynności, wykonywane podczas jej eksploatacji.

Instalacja i skonfigurowanie bazy danych otwierają drogę do podjęcia działań, których celem będzie utworzenie fizycznych i logicznych struktur przechowywania danych. Jednak administrowanie bazą danych to nie tylko zarządzanie jej obiektami, ale także dbałość o zachowanie bezpieczeństwa danych. Dlatego też tak ważna jest znajomość różnych mechanizmów, jak prawa systemowe, role czy wirtualne bazy danych, które zapewnią dostęp do bazy tylko autoryzowanym jej użytkownikom.

Jednym z najczęściej realizowanych zadań w cyklu życia bazy danych jest definiowanie i wykonywanie zapytań. Umiejętność ich poprawnego sformułowania, na

podstawie analizy planu wykonania, pozwala zoptymalizować składnię polecenia selekcji i znacznie skrócić czas otrzymania odpowiedzi.

Zapewnienie ciągłości pracy bazy danych to jeden z najistotniejszych obowiązków administratora. Jego realizacja wymaga posiadania kopii zarówno struktur danych, jak i samych danych. W zależności od przyjętej strategii tworzenia backupów, konieczne jest generowanie różnych rodzajów kopii plików danych oraz wszystkich innych plików, które są niezbędne do prawidłowego funkcjonowania bazy danych.

Potrzeba efektywnego zaspokajania potrzeb informacyjnych użytkowników, przy wykorzystaniu posiadanych zasobów danych, wymusza często wprowadzenie zmian w ich strukturach, a niekiedy nawet przejścia na nowszą wersję środowiska bazy danych. Działania te skutkują koniecznością migracji struktur danych oraz danych między poprzednim i nowym środowiskiem pracy. W tej sytuacji umiejętność prawidłowego wykonania eksportu i importu danych gwarantuje poprawność dalszego funkcjonowania systemu informatycznego.

Książka prezentuje zagadnienia związane z eksploatacją bazy danych na platformie Oracle. Jej zakres tematyczny obejmuje instalację serwera bazy danych Oracle 11g, tworzenie bazy danych, administrowanie strukturami przechowywania danych i zarządzanie obiektami bazy danych, zapewnienie bezpieczeństwa składowania i dostępu do danych, eksport i import danych, a także optymalizację zapytań. Czytelnik znajdzie w niej wiele praktycznych wskazówek i przykładów, które pozwolą stworzyć optymalne środowisko pracy.

Ze względu na ograniczoną objętość książki, niektóre zagadnienia zostały przedstawione w skrócie. Jednak kompletność prezentowanych treści pozwala zdobyć podstawową wiedzę, jaka jest niezbędna w eksploatacji bazy danych Oracle.

*Autorzy*



---

## Instalowanie systemu Oracle 11g

---

### **Cel**

Celem rozdziału jest zapoznanie Czytelnika z podstawowymi pojęciami i informacjami związanymi z instalowaniem systemu bazodanowego Oracle 11g. Procesy instalacji zostały wykonane na najnowszej wersji systemu Oracle – Release 2. Instalacja została przeprowadzona na platformie operacyjnej Windows.

### **Plan**

1. Wprowadzenie do instalowania środowiska Oracle 11g.
2. Instalacja systemu bazodanowego Oracle 11g.

### 1.1. OGÓLNA CHARAKTERYSTYKA PROCESU INSTALACJI

Podstawowa umiejętność administrowania bazą danych jest związana ze znajomością systemu bazodanowego oraz jego poprawną instalacją. Jest to bardzo ważna czynność, która niekiedy rzutuje na przyszłą pracę z bazą danych. Administrowanie systemem Oracle różni się w zależności od systemu operacyjnego. Przykładowo, instalując system Oracle na platformie Linux, należy wykonać wiele czynności przygotowawczych (m.in. tworzenie użytkownika i przydzielanie go do grupy używanej w trakcie instalacji, czy też uzupełnienie parametrów systemowych), zwanych pre-instalacją. Dodatkowo po zainstalowaniu systemu bazodanowego, wykonywana jest post-instalacja, która zapewnia m.in. automatyczne uruchamianie instancji Oracle. Instalacja systemu Oracle 11g na platformie Windows jest prostsza, gdyż nie wymaga czynności przygotowawczych. Instancję systemu należy uruchamiać z wiersza poleceń.

Administrator powinien przygotować się do instalacji poprzez wybranie odpowiedniej wersji systemu (w zależności od platformy operacyjnej, nałożonych ograniczeń związanych z pamięcią RAM czy liczbą procesorów). Należy także szczegółowo przeanalizować poszczególne wersje systemu pod kątem zawartych w nich pomocniczych narzędzi bazodanowych takich jak: *Enterprise Manager*, *Server Managed Backup and Recovery*, *Oracle Advanced Security*, *Data Guard*, *Oracle Real Applications Cluster* i wiele innych.

Także rodzaj instalacji, tj. desktopowa czy serwerowa, ma znaczenie dla przyszłej pracy z systemem. Instalacja systemu Oracle różni się w zależności od platformy, na której będzie on działał. W celu dokładnego poznania szczegółów, związanych z tym procesem, najlepiej jest zapoznać się z instrukcją instalacji systemu Oracle na wybranej platformie. Jest ona dostępna na stronie korporacyjnej Oracle wraz z innymi podręcznikami, wspomagającymi późniejsze zarządzanie tym systemem.

## 1.2. INSTALACJA SYSTEMU ORACLE 11G

Wersję instalacyjną systemu Oracle 11g Release 2 można pobrać ze strony producenta z sekcji *Downloads* (Oracle, 2011n). Są tam dostępne wersje dla różnych systemów operacyjnych, takich jak: Windows 32 i 64 bitowy, Linux, Solaris oraz innych. W celu uzyskania dostępu do plików, niezbędne jest posiadanie konta w portalu Oracle (rejestracja i późniejsze korzystanie z zasobów portalu są darmowe). Po skopiowaniu ze strony producenta i rozpakowaniu plików można przystąpić do instalacji systemu bazodanowego Oracle.

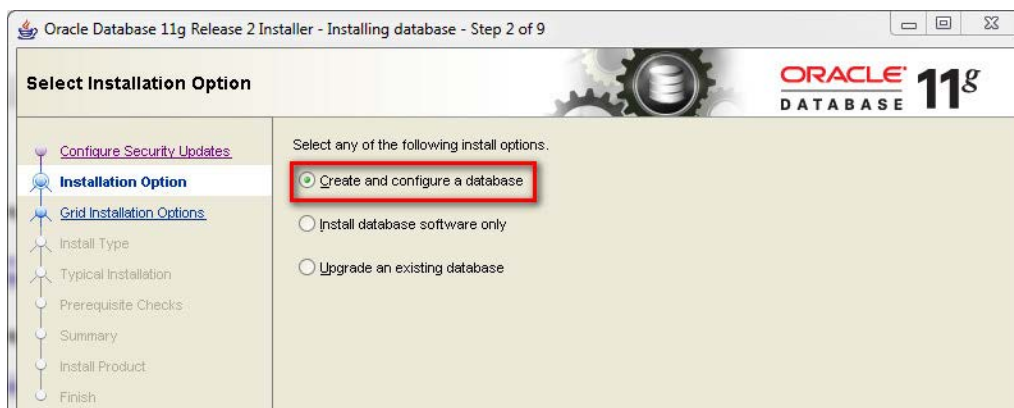
Instalacja na platformie Windows, najbardziej popularnej platformie operacyjnej, składa się z dziewięciu etapów. Pierwszy krok instalacji systemu bazodanowego Oracle 11g rozpoczyna się wyświetleniem okna przedstawionego na rys. 1.1. Jeżeli administrator chce otrzymywać powiadomienia związane z bezpieczeństwem systemu i możliwych uaktualnieniach, należy pozostawić domyślnie zaznaczoną opcję (*I wish to receive security updates via My Oracle Support*).



Rys. 1.1. Pierwszy etap instalacji systemu Oracle 11g na platformie Windows

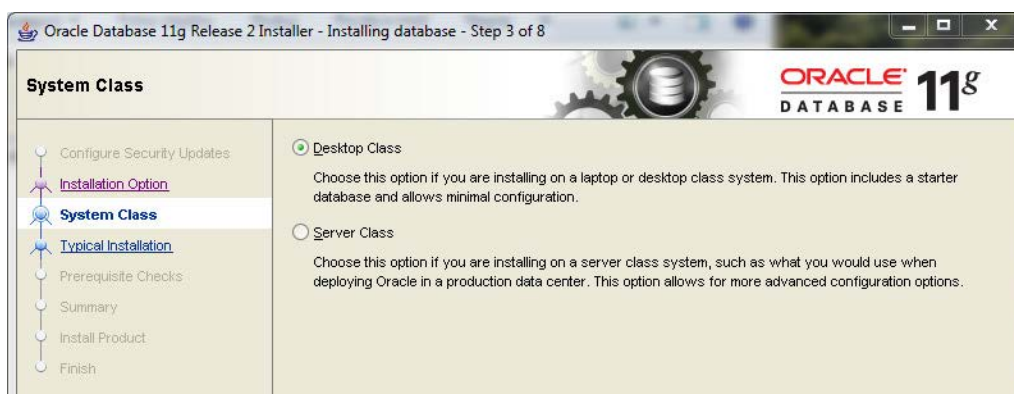
Drugi etap polega na wyborze jednej z trzech opcji, która określa rodzaj instalacji systemu. Pierwszy rodzaj (*Create and configure a database*), wskazany na rysunku 1.2. jako domyślny, pozwala utworzyć i skonfigurować bazę danych. Drugi rodzaj (*Install database software only*) polega na zainstalowaniu jedynie oprogramowania serwera bazy danych. Ostatni typ instalacji (*Upgrade an existing database*) polega na

aktualizacji wersji systemu. Po wyborze języka, który będzie używany w eksploatacji bazy danych, użytkownik jest przekierowany do okna wyboru wersji systemu.



Rys. 1.2. Wybór rodzaju instalacji systemu bazodanowego Oracle na platformie Windows

Trzecim etapem jest wybranie jednego z dwóch rozwiązań systemu Oracle 11g (System Class) – rys. 1.3. Pierwsze jest związane z instalacją systemu jako wersji desktopowej (ang. *Desktop Class*) i obejmuje minimalną konfigurację (Gerard, 2011). W tym przypadku razem z serwerem bazy danych instalowana jest startowa baza danych. Opcję tę należy wybrać, gdy system jest instalowany na laptopie lub na komputerze stacjonarnym. Rozwiązanie to pozwala na szybką instalację i uruchomienie systemu.

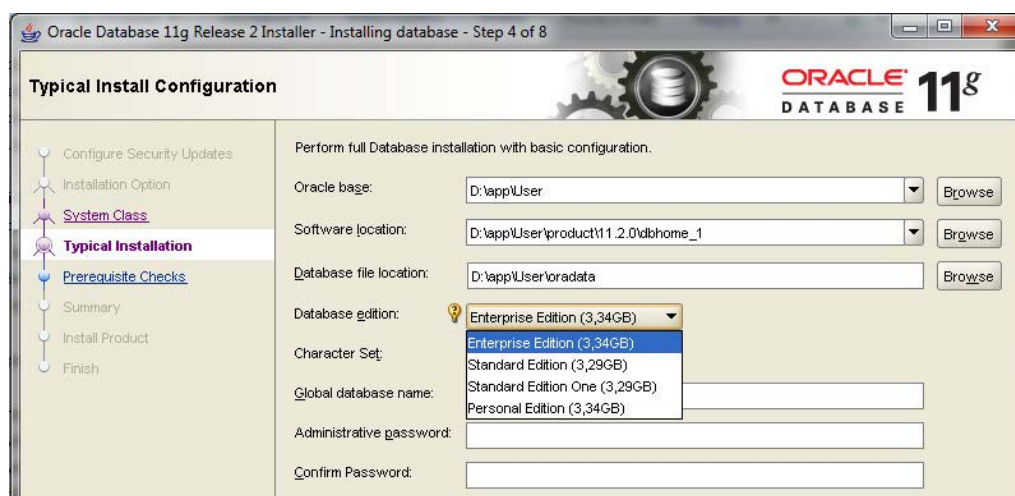


Rys. 1.3. Wybór rodzaju systemu bazodanowego Oracle na platformie Windows

Druga opcja to instalacja systemu w wersji serwerowej (ang. *Server Class*) (Gerard, 2011). Jest to wersja dla systemu Oracle, który ma zostać wdrożony w środowisku produkcyjnym. Posiada ona bardziej rozbudowane i nowoczesne opcje instalacyjne takie jak: *Oracle RAC* (ang. *Real Application Cluster*), *ASM* (ang. *Automatic Storage Management*), konfiguracja tworzenia i wykonywania kopii bezpieczeństwa bazy danych i inne.

Czwartym etapem instalacji systemu jest wybór podstawowych opcji konfiguracji takich jak: lokalizacja systemu Oracle oraz bazy danych, wersja systemu bazodanowego, sposób kodowania znaków, nazwa bazy danych oraz hasło administratora (rys. 1.4).

Istnieje pięć wersji instalacji systemu Oracle, bez względu na typ platformy systemowej. Są to: *Enterprise Edition*, *Standard Edition*, *Standard Edition One*, *Personal Edition* oraz *Express Edition* (Bryla, Loney, 2010; Oracle, 2011m).



Rys. 1.4. Konfiguracja systemu bazodanowego Oracle na platformie Windows

Najbardziej rozbudowaną wersją jest *Enterprise Edition*. Posiada ona wiele pomocniczych funkcji np. *Flashback Database*, która pozwala na przywrócenie bazy danych bez korzystania z fizycznych kopii zapasowych (Bryla, Loney, 2010). Jest to wersja, którą można rozszerzać poprzez dodawanie kolejnych narzędzi, takich jak: *Real Application Cluster* (umożliwia używanie tych samych plików bazy danych

umieszczonych na różnych serwerach przez więcej niż jedną instancję (Bryla, Loney, 2010) czy *Oracle Data Mining* do budowania zintegrowanych aplikacji typu business intelligence (Oracle, 2011l).

Najbardziej uproszczoną wersją jest *Express Edition*, która posiada odrębną wersję instalacyjną. Cechuje ją łatwa instalacja oraz znaczne ograniczenia nałożone na bazę danych w postaci zawężenia wykorzystywanych parametrów sprzętowych (np. liczba procesorów, pamięć RAM) oraz braku narzędzi występujących w innych wersjach. Niektóre z ograniczeń zostały przedstawione w tabeli 1.1.

Wersja *Standard Edition* posiada podstawowe narzędzia występujące w wersji *Enterprise*. Nie ma w niej jednak możliwości dodatkowego zainstalowania pozostałych narzędzi.

Wersja *Standard Edition One* oferuje takie same funkcje, co poprzednio opisana wersja. Zasadnicza różnica występująca między nimi polega na możliwości użycia wersji *One* na jednym serwerze z maksymalnie czterema procesorami.

Ostatnią wersją jest *Personal Edition*, której nie można używać w środowiskach produkcyjnych. Wersja ta pozwala na rozwijanie aplikacji działających na wersjach *Standard* oraz *Enterprise*.

Zestawienie wybranych cech poszczególnych wersji instalacyjnych zostało zamieszczone w tabeli 1.1 (Oracle, 2011m).

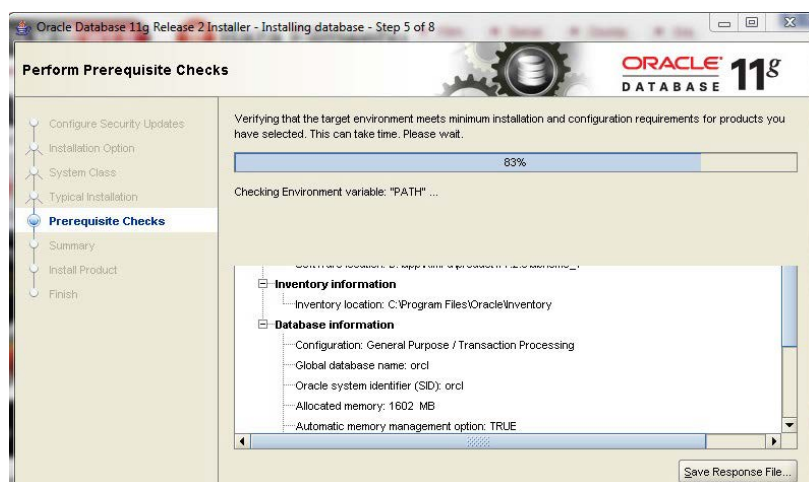
Tabela 1.1. Wersje instalacyjne systemu Oracle 11g

	<b>Enterprise Edition</b>	<b>Standard Edition</b>	<b>Standard Edition One</b>	<b>Express Edition</b>
Maksymalnie	Bez ograniczeń	4 gniazda	2 gniazda	1 procesor
RAM	Bez ograniczeń	Bez ograniczeń	Bez ograniczeń	1 GB
Baza danych	Bez ograniczeń	Bez ograniczeń	Bez ograniczeń	4 GB
Enterprise Manager	Jest	Jest	Jest	Brak
Server Managed Backup and Recovery	Jest	Jest	Jest	Brak

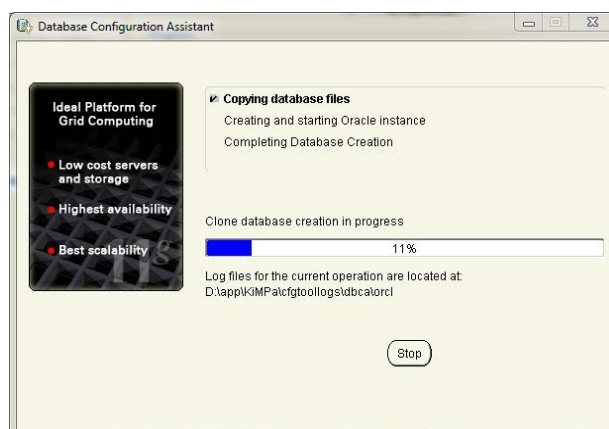
Źródło: (Bryla, Loney, 2010; Oracle, 2011m)

Po wyborze wersji oraz sposobu kodowania, należy podać hasła do kont użytkowników z uprawnieniami administratora. Dobrą praktyką jest stworzenie, poza kontami *SYS* oraz *SYSTEM*, dodatkowego konta z uprawnieniami administratora.

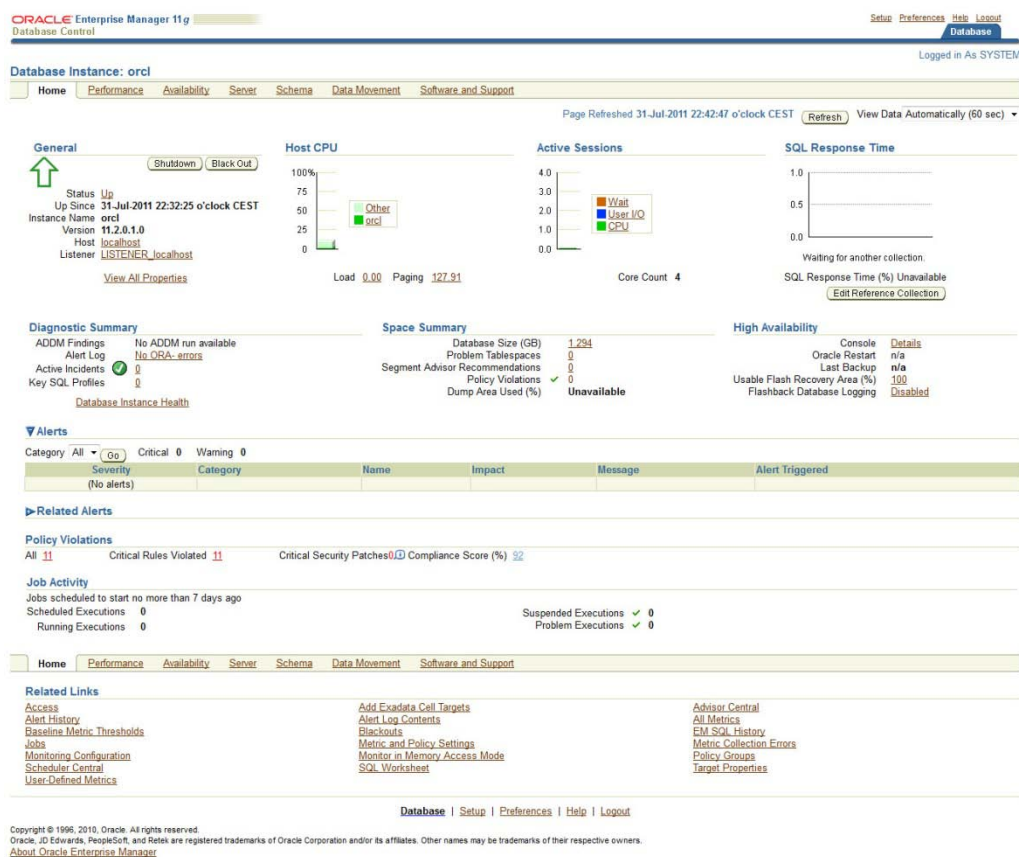
Kolejnymi etapami instalacji środowiska Oracle są: weryfikacja, czy wybrane opcje spełniają wymagania systemu operacyjnego (rys. 1.5), instalacja systemu (rys. 1.6) i zakończenie procesu instalowania. Po zakończeniu wszystkich etapów, możliwe jest uruchomienie narzędzia *Oracle Enterprise Manager* (rys. 1.7), które wspomaga pracę administratora w zarządzaniu bazą danych.



Rys. 1.5. Sprawdzenie minimalnych wymagań systemowych dla instalowanego systemu Oracle



Rys. 1.6. Instalacja systemu Oracle 11g



Rys. 1.7. Strona główna narzędzia Enterprise Manager

Po instalacji systemu, można przystąpić do tworzenia baz danych oraz dalszych czynności, związanych z zarządzaniem bazą danych.

### 1.3. PYTANIA KONTROLNE

1. Wymień kolejne kroki instalacji systemu Oracle na platformie Windows.
2. Jakie występują wersje instalacyjne systemu Oracle 11g i jakie posiadają ograniczenia?
3. Jakie narzędzie jest wykorzystywane do sprawdzenia poprawności instalacji systemu Oracle?



## Tworzenie bazy danych systemu Oracle 11g

---

### Cel

Celem rozdziału jest zapoznanie Czytelnika z podstawowymi pojęciami i informacjami związanymi z tworzeniem bazy danych w systemie Oracle 11g. Proces ten został przeprowadzony na najnowszej wersji systemu Oracle – Release 2. Zostanie on przedstawiony w środowisku systemu operacyjnego Windows. Szczegółowo zostaną zaprezentowane dwa rodzaje tworzenia bazy: z użyciem narzędzia *DBCA* (ang. *Database Configuration Assistant*) oraz manualne przez administratora, przeprowadzone z wiersza poleceń.

### Plan

1. Czynności przygotowawcze.
2. Tworzenie bazy danych przy użyciu narzędzia *DBCA*.
3. Manualne tworzenie bazy danych.

## 2.1. CZYNNOŚCI PRZYGOTOWAWCZE

Tworzenie bazy danych jest jedną z podstawowych umiejętności administratora. Powinien on umieć zainstalować i skonfigurować ją w taki sposób, aby baza była niezawodna oraz jak najbardziej wydajna. Osoba taka powinna czuwać także nad bezawaryjnością systemu, wykonując odpowiednie kopie bezpieczeństwa takie jak: import, eksport bazy danych, kopie zapasowe *offline* i *online* czy kopie utworzone narzędziem *RMAN* (Bryla, Loney, 2010).

Przed przystąpieniem do tworzenia bazy danych należy wykonać pewne czynności przygotowawcze, które obejmują (Bryla, Loney, 2010):

- Określenie nazwy bazy danych, która może być związana z jej funkcjonalnością. Powinno się także sprecyzować domenę, w której będzie ona się znajdować. Ustawienia te są dostępne w parametrach instalacyjnych *DB\_NAME* oraz *DB\_DOMAIN*.
- Oszacowanie liczby tabel, indeksów, a także ich rozmiaru. Można wtedy lepiej dopasować obszar przestrzeni na dysku do wymagań przyszłej bazy danych oraz przestrzeni dodatkowej.
- Zaplanowanie lokalizacji plików bazy danych na dysku fizycznym w taki sposób, aby wydajność przyszłej bazy była jak największa. Należy rozważyć zastosowanie narzędzia *OMF* (ang. *Oracle Manager Files*) do zarządzania plikami lub narzędzia *ASM* (ang. *Automatic Storage Management*) do obsługi dodatkowej pamięci.
- Wybór zestawu znaków, odpowiadającego narodowości przyszłych użytkowników bazy danych. Po instalacji bazy danych zestaw znaków można zmienić jedynie na taki, który będzie rozszerzeniem zestawu wybranego podczas instalacji.
- Zdefiniowanie wielkości obszaru bloku bazy danych. Wartość, opisująca wielkość tego bloku, przechowywana jest w parametrze *DB\_BLOCK\_SIZE*. Podczas instalacji bazy tworzone są przestrzenie tabel *SYSTEM*, *TEMP*, *SYSAUX* z uwzględnieniem zdefiniowanego rozmiaru bloku. Zaleca się, aby rozmiar ten był równy wielkości bloku systemowego, bądź też jego wielokrotności. Nieprawidłowe

ustalenie wielkości obszaru bloków może wpłynąć negatywnie na wydajność bazy danych.

- Zaplanowanie przestrzeni tabel (trwałych i tymczasowych), które zostaną utworzone i przydzielone użytkownikom bazy danych w trakcie ich definiowania. Nie zaleca się przydzielania domyślnej przestrzeni tabel *SYSTEM* tym użytkownikom, którzy nie posiadają statusu administratora.
- Zapoznanie się z narzędziem *Automatic Undo Management*, które ułatwia zarządzanie wycofywaniem transakcji bazy danych. Należy rozważyć przydzielenie dodatkowej przestrzeni dla tabel wycofania, przez co zwiększy się produktywność bazy.
- Zaplanowanie strategii wykonywania kopii zapasowych oraz ich odtwarzania. Należy podjąć decyzje dotyczące m.in. częstotliwości wykonywania kopii, wyboru rodzaju i metody tworzenia kopii oraz sprecyzować maksymalny czasu, w którym baza może pozostawać niedostępna.

Tworzenie bazy danych może odbywać się na dwa sposoby. Pierwszy polega na wykorzystaniu narzędzia *DBCA* (ang. *Database Configuration Assistant*). Sposób jego użycia zostanie opisany w rozdziale 2.2. Drugim rozwiązaniem jest stworzenie bazy danych manualnie, bez użycia narzędzia *DBCA*. Kolejne etapy tworzenia bazy danych są przedstawione w rozdziale 2.3.

## 2.2. TWORZENIE BAZY DANYCH PRZY UŻYCIU NARZĘDZIA DBCA

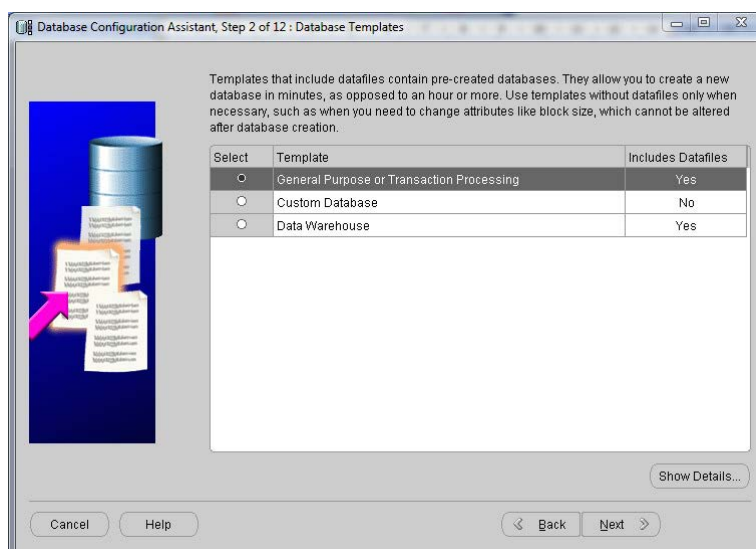
Bardzo wygodnym i przyjaznym użytkownikowi narzędzie tworzenia bazy danych jest *Database Configuration Assistant*, które stanowi część systemu bazodanowego Oracle.

W systemie operacyjnym Windows należy uruchomić program *Database Configuration Assistant* (DBCA). Natomiast w systemie operacyjnym Linux wystarczy wpisać *dbca* w wierszu poleceń.

Za pomocą tego narzędzia można utworzyć nową bazę danych, zmienić opcje konfiguracyjne istniejącej bazy danych, usunąć bazę danych bądź też dodawać, modyfikować i usuwać szablony bazy. Po zdefiniowaniu wszystkich parametrów bazy,

ustawienia konfiguracyjne można zapisać jako szablon, który może być wykorzystany w przyszłości w procesie tworzenia kolejnych baz danych.

Pierwszym etapem tworzenia nowej bazy danych jest wybór szablonu, na podstawie którego zostanie ona utworzona. W przypadku, gdy szablon taki już istnieje, jego nazwa zostanie wyświetlona jako składowa listy w części *Template*. Wybierając szablon, wystarczy zaznaczyć go w części *Select*. Okno dostępnych szablonów, wyświetlane podczas tworzenia bazy danych, zostało przedstawione na rysunku 2.1.



Rys. 2.1. Szablony bazy danych

Domyślnie dostępne są trzy szablony (Bryla, Loney, 2010):

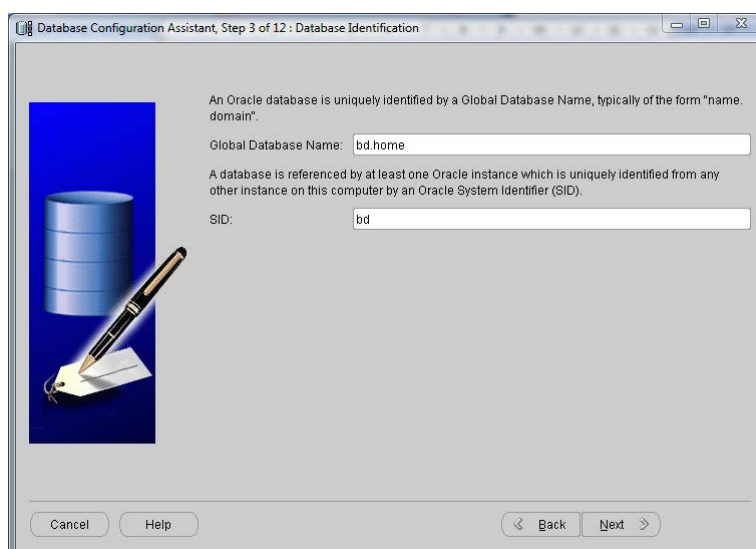
- *General Purpose or Transaction Processing*,
- *Custom Databases*,
- *Data Warehouse*.

Pierwsza opcja (*General Purpose or Transaction Processing*) dotyczy bazy danych ogólnego przeznaczenia, z zapewnieniem ciągłości dostępności danych. Szablon ten dedykowany jest do tworzenia niewielkich baz danych, w których zachodzą złożone transakcje, ale o krótkim czasie trwania, obejmujące głównie operacje modyfikowania i wprowadzania danych.

Opcja *Custom Database* jest przeznaczona dla osób, posiadających zaawansowane umiejętności tworzenia bazy danych. W tym przypadku należy samodzielnie definiować parametry bazy, co wymaga posiadania wiedzy, niezbędnej do tego rodzaju konfiguracji.

Ostatnia opcja (*Data Warehouse*) wykorzystywana jest w przypadku, gdy w tworzonej bazie danych będą wykonywane wielokrotnie złożone zapytania, szczególnie przy zaawansowanym raportowaniu czy podczas analizy danych.

Kolejnym etapem instalowania bazy danych jest jej identyfikacja. W tym miejscu należy podać nazwę bazy danych oraz jej identyfikator systemowy Oracle SID (ang. *Oracle System Identification*). Jeśli nazwa bazy danych będzie zawierała kropkę, wówczas automatycznie wszystkie znaki do tego separatora zostaną uznane za SID bazy. Okno przedstawiające sposób identyfikacji bazy zostało przedstawione na rysunku 2.2.



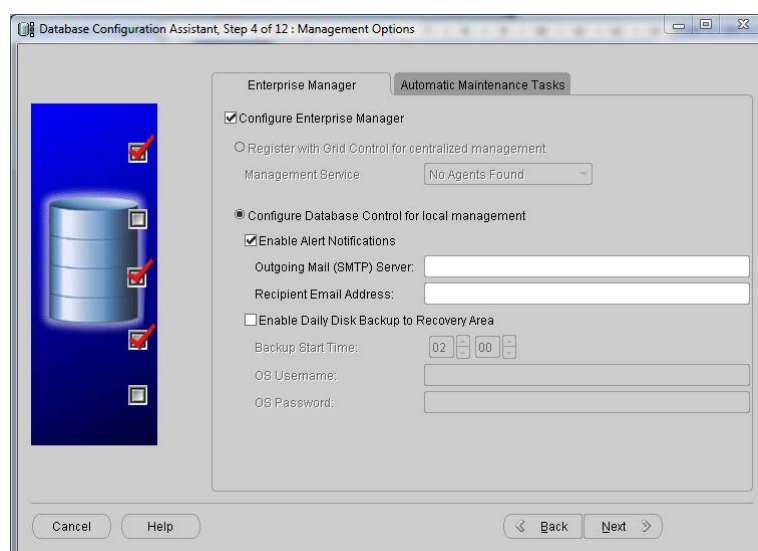
Rys. 2.2. Identyfikacja bazy danych

Następnym etapem tworzenia bazy danych jest określenie opcji związanych z konfiguracją narzędzia *Enterprise Manager (Configure Enterprise Manager)* i zarządzania bazą, wykorzystywaną przez to narzędzie (rys. 2.3).

W celu skonfigurowania usług sieciowych, należy zaznaczyć opcję *Enable Alert*

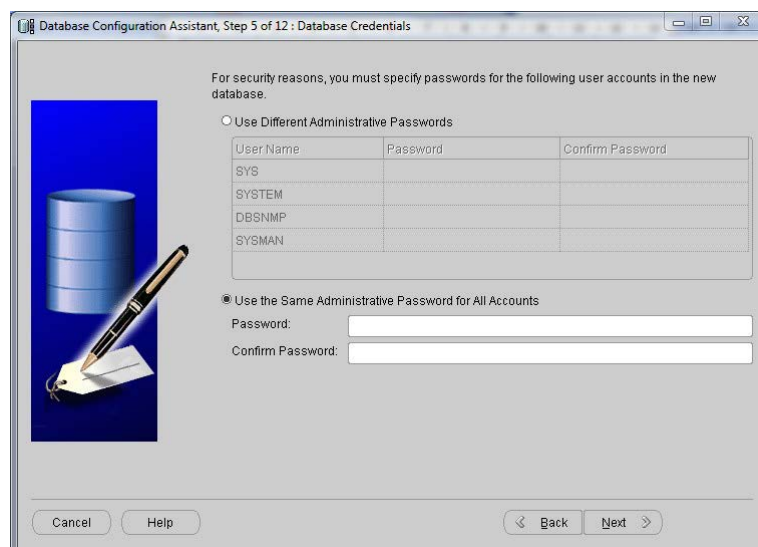
*Notification.* Na podany adres poczty elektronicznej będą przesyłane informacje zawierające ostrzeżenia i powiadomienia związane z eksploatacją bazy danych.

Istotnym elementem tworzenia bazy danych jest określenie czy i kiedy będą wykonywane automatyczne kopie bezpieczeństwa. Są one niezwykle ważne w przypadku awarii, gdyż dzięki nim można odzyskać zapisane dane. Polecenie wykonywania dziennych kopii bezpieczeństwa wymaga zaznaczenia pola *Enable Daily Disk Backup to Recovery Area*. Ponadto należy określić godzinę, w której będzie wykonywana zapasowa kopia oraz nazwę i hasło użytkownika systemowego.



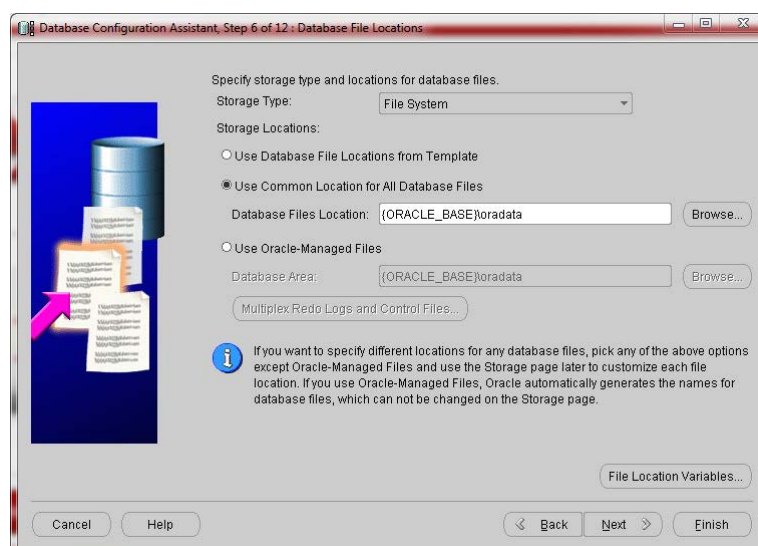
Rys. 2.3. Zarządzanie bazą danych

Przy tworzeniu bazy danych należy podać także dane uwierzytelniające (rys. 2.4). Są to początkowe hasła dla użytkowników administrujących bazą danych, takich jak SYS, SYSTEM, DBSNMP i SYSMAN. Można zdefiniować indywidualne hasła dla poszczególnych kont (*Use Different Administrative Passwords*) lub jedno hasło dla wszystkich (*Use the Same Administrative Password for All Accounts*). W celu zapewnienia wysokiego poziomu bezpieczeństwa, hasło administratora bazy danych powinno spełniać standardy systemu Oracle, tzn. zawierać minimum 8 znaków, przynajmniej jedną cyfrę oraz duże i małe litery.



Rys. 2.4. Dane uwierzytelniające bazy danych

W kolejnym etapie instalacji bazy danych należy zdefiniować opcje, dotyczące sposobu przechowywania plików bazy oraz lokalizacji miejsca ich składowania (rys. 2.5).



Rys. 2.5. Typ i lokalizacja plików bazy danych

Baza danych Oracle składa się z wielu plików, dla których należy określić tzw. rodzaj przechowywania (ang. *storage type*). Jeśli dysponuje się kilkoma dyskami, na których będą składowane dane, proponowanym rozwiązaniem jest system ASM (ang. *Automatic Storage Management*). Jednak należy pamiętać, że ASM wymaga dodatkowej instalacji. W przeciwnym razie pliki bazy wystarczy przechowywać w istniejącym systemie plików (ang. *file system*) danego systemu operacyjnego.

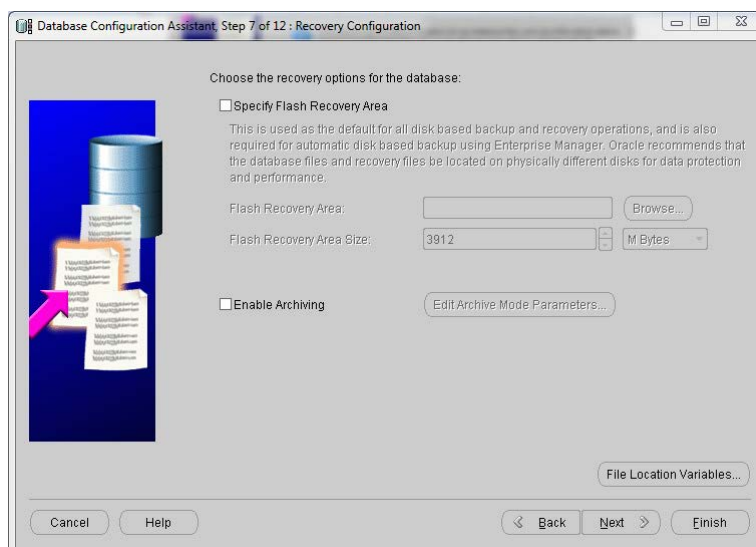
Następnie należy określić lokalizację dla plików bazy danych, tj. plików danych, kontrolnych, dziennika powtórzeń, kopii zapasowych oraz plików odtwarzania. Do wyboru jest jedna z następujących opcji (Bryła, Loney, 2010):

- lokalizacja zdefiniowana w szablonie (opcja *Use Database File Locations from Template*),
- podanie własnej lokalizacji (opcja *Use Common Location for All Database Files*),
- skonfigurowanie dla bazy danych mechanizmu OMF (opcja *Use Oracle-Managed Files*). W podanej lokalizacji przechowywane są pliki, które będą zarządzane przez system Oracle.

Istotnym elementem konfiguracji bazy danych jest ustawienie opcji dotyczących kopii bezpieczeństwa. Podczas tworzenia bazy danych można zdefiniować specjalny obszar na twardym dysku, który zostanie odseparowany od plików bazy i będzie zawierał pliki kopii bezpieczeństwa, utworzone przy pomocy narzędzia *RMAN* (ang. *Recovery Manager*) (Bryła, Loney, 2010). Należy zwrócić uwagę, aby określony obszar był wystarczająco duży do przechowania przynajmniej dwóch kopii bazy.

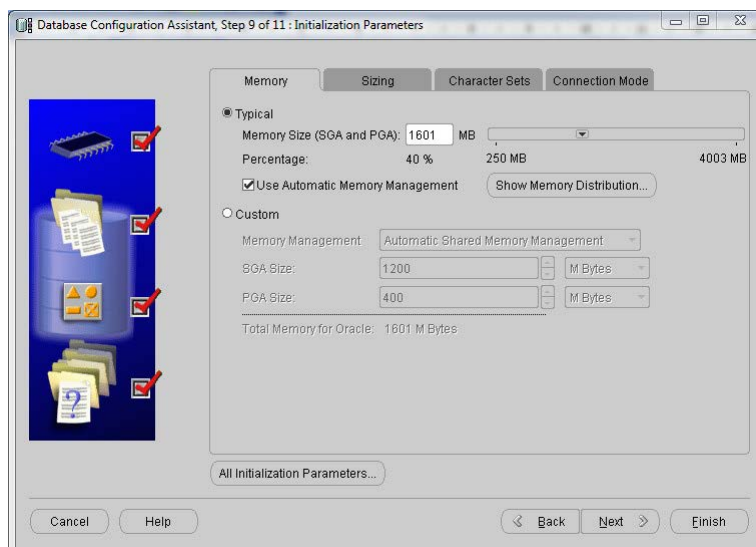
Utworzenie obszaru wymaga zaznaczenia opcji *Specify Flash Recovery Area*, podania jego lokalizacji (*Flash Recovery Area*) oraz maksymalnego rozmiaru (rys. 2.6). Dodatkowo możliwe jest wybranie opcji archiwizacji plików bazy danych poprzez włączenie trybu *ARCHIVELOG*, który umożliwia przechowywanie wypełnionego pliku dziennika powtórzeń w określonej lokalizacji. Plik ten można użyć w przyszłości do odtworzenia bazy danych w przypadku awarii (Bryła, Loney, 2010).





Rys. 2.6. Opcje wykonywania kopii zapasowych

W kolejnym etapie tworzenia bazy danych można wybrać opcję instalacji dodatkowych schematów (rys. 2.7).

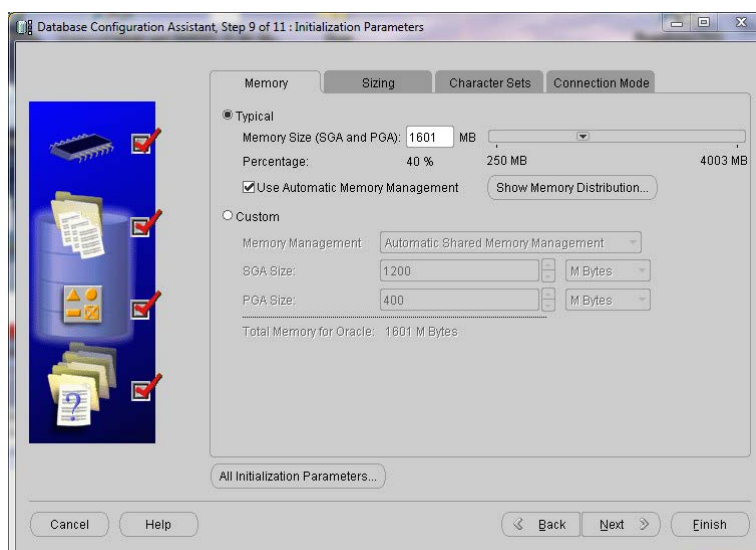


Rys. 2.7. Instalacja przykładowych schematów

Są one przydatne m.in. do testowania bazy oraz identyfikowania jej struktury. Na ich podstawie można generować materiały pomocnicze. Dodatkowe schematy są dobrym źródłem informacji o większości typów danych oraz konstrukcji bazy danych. Ponadto w zakładce *Custom Script* można dołączyć skrypty, które zostaną wykonane po utworzeniu bazy danych.

W środowiskach produkcyjnych instalowanie dodatkowych schematów nie jest zalecane ze względu na bezpieczeństwo oraz wydajność bazy danych. Pominięcie instalacji schematów nie oznacza jednak ich braku, gdyż można je także utworzyć w istniejącej bazie danych, korzystając z dostępnych skryptów (Bryla, Loney, 2010).

W celu zapewnienia prawidłowego działania bazy danych, należy skonfigurować jej parametry inicjalizacyjne. Kolejne okno, zaprezentowane na rysunku 2.8, umożliwia przydział odpowiedniej ilości pamięci dla systemu bazodanowego.

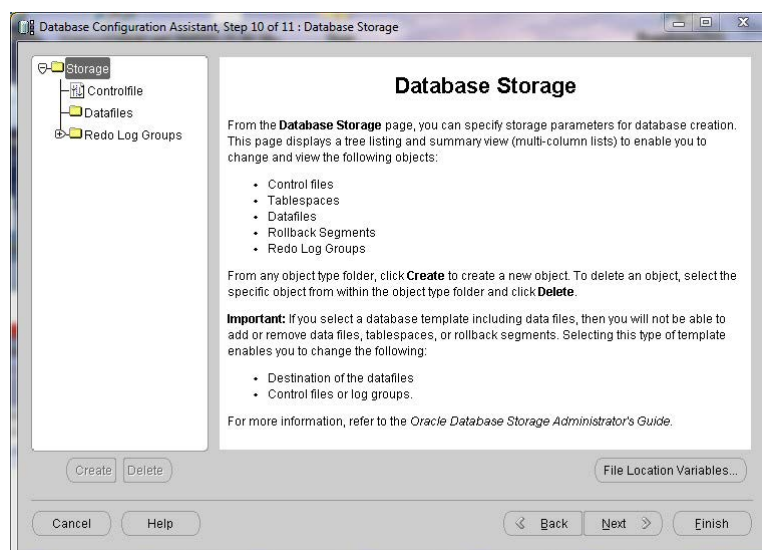


Rys. 2.8. Konfigurowanie parametrów inicjalizacyjnych

Jeśli nie występują szczególne wymagania związane z pamięcią, wystarczy zaznaczyć opcję *Typical*, która określi procentowy przydział dla obszarów *PGA* (ang. *Program Global Area*) i *SGA* (ang. *System Global Area*) w wysokości 40% pamięci systemu. W przeciwnym razie należy samodzielnie zdefiniować wielkość przydziału pamięci dla tych obszarów. Wybranie opcji *Use Automatic Memory Management*

optymalizuje wydajność systemu Oracle w obrębie obszarów *PGA* i *SGA*. Zostanie wtedy przeprowadzona ponowna alokacja pamięci w obrębie obszaru *SGA* oraz pomiędzy *PGA* i *SGA* (Bryla, Loney, 2010). Pozostałe parametry inicjalizacyjne można zweryfikować poprzez wybór opcji *All Initialization Parameters*.

Kolejny krok tworzenia bazy danych dotyczy organizacji przechowywania jej plików. Rysunek 2.9 przedstawia informacje o plikach kontrolnych, plikach danych i plikach dziennika powtórzeń. Dane te składają się na wartość parametru *CONTROL\_FILE* w pliku parametrów inicjalizacyjnych lub w pliku konfiguracyjnym *SPFILE*. Etap ten umożliwia także zweryfikowanie oraz zmianę lokalizacji tych plików, a także utworzenie grupy dziennika powtórzeń (Bryla, Loney, 2010).

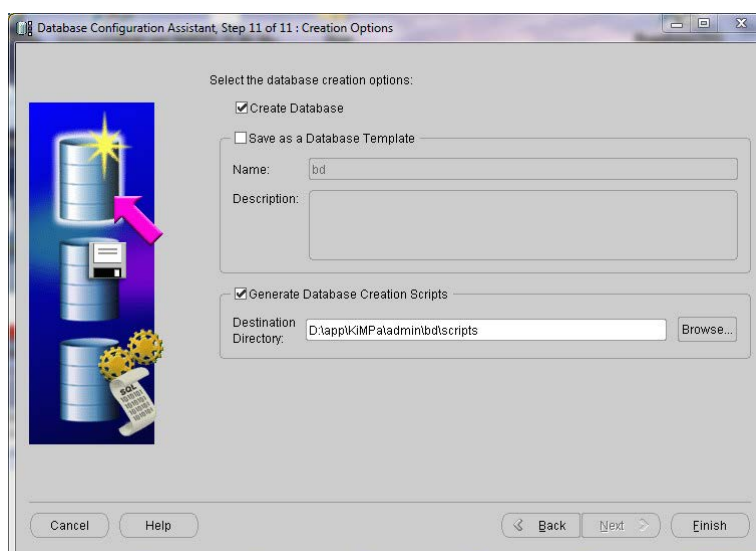


Rys. 2.9. Okno Database Storage

Ostatnim etapem, bezpośrednio poprzedzającym proces fizycznego tworzenia bazy danych, jest wybór jednej z czynności: utworzenie bazy danych, zapisanie szablonu bazy o podanej nazwie z uwzględnieniem opisu oraz wygenerowanie i zapisanie skryptu bazy danych (rys. 2.10).

Zarówno szablon jak i skrypt warto utworzyć. Nie zajmują one wiele miejsca na dysku i w razie potrzeby mogą być usunięte w późniejszym czasie. Ich istnienie przynosi administratorowi bazy wiele korzyści, takich jak możliwość wykorzystania

zapisanej konfiguracji do tworzenia lub zmiany bazy danych w przyszłości. Szablon taki może także posłużyć jako dokumentacja opcji konfiguracyjnych aktualnej bazy danych (Bryla, Loney, 2010).

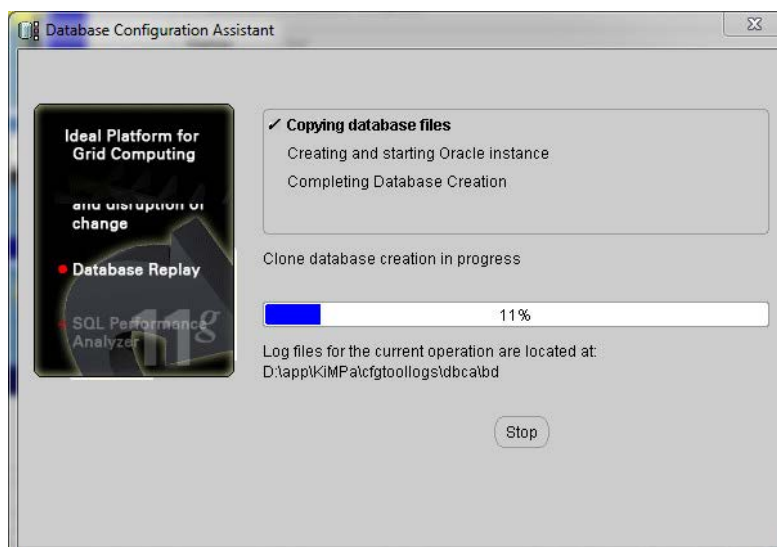


Rys. 2.10. Wybór opcji tworzenia bazy danych oraz szablonu

Po wybraniu opcji utworzenia bazy danych (rys. 2.10) i jej zatwierdzeniu, generowany jest raport, który można zapisać w pliku o rozszerzeniu *html*. Po zatwierdzeniu tego raportu rozpoczyna się proces tworzenia bazy danych przez narzędzie *DBCA*, które wykorzystuje zebrane informacje konfiguracyjne (rys. 2.11). Proces ten obejmuje utworzenie fizycznej i logicznej struktury bazy danych, uruchomienie jej instancji i wykonanie standardowego zestawu skryptów (Bryla, Loney, 2010).

Wygenerowanie bazy danych potwierdzone jest wyświetleniem podsumowania, w którym przedstawiona jest jej nazwa, SID oraz nazwa pliku serwera. W tym momencie możliwa jest także zmiana hasła (lub haseł) kont administratora.

Po zakończeniu instalacji można utworzyć pozostałe konta dla użytkowników z uprawnieniami oraz ograniczeniami nałożonymi przez administratora, niezbędnymi do pracy z bazą danych. W szczególności powinno zostać utworzone odrębne konto z uprawnieniami administratora, które umożliwi późniejsze zarządzanie bazą.



Rys. 2.11. Proces instalacji bazy danych

Narzędzie *DBCA* pozwala tworzyć bazę danych w sposób intuicyjny, w związku z czym działanie to może być wykonane nawet przez początkującego administratora. Jednak wadą *DBCA* są ograniczenia funkcjonalne, które nie pozwalają wykorzystać opcji dostępnych w natywnych poleceniach, realizujących to samo zadanie. Dlatego też koniecznym uzupełnieniem dotychczas przedstawionych treści będzie prezentacja manualnego tworzenia bazy danych przy użyciu języka SQL.

## 2.3. MANUALNE TWORZENIE BAZY DANYCH

W przypadku braku graficznego interfejsu, umożliwiającego tworzenie bazy danych, administrator musi samodzielnie zrealizować kolejne, uprzednio opisane etapy, wykorzystując do tego celu polecenia języka SQL, wydawane przy pomocy konsoli. Manualny sposób tworzenia bazy danych przynosi jednak wiele korzyści, gdyż w ten sposób można zdefiniować więcej opcji niż przy użyciu narzędzia *DBCA*. Instalacja tego rodzaju różni się w zależności od używanej platformy operacyjnej (Bryła, Loney, 2010). Poniżej zostanie przedstawiony proces tworzenia bazy danych w systemie operacyjnym Windows.

Pierwszym poleceniem, jakie należy wykonać, jest nadanie identyfikatora SID dla tworzonej bazy danych. Na platformie Windows odbywa się to poprzez wpisanie w wierszu poleceń komendy przedstawionej na listingu 2.1. W miejscu *nazwa* należy podać nowy, unikalny identyfikator bazy danych, który pozwoli rozróżnić definiowaną instancję bazy od innych już istniejących. W celu wyświetlenia nadanego identyfikatora, należy wykonać polecenie przedstawione na listingu 2.2. Analogicznie do nadania nazwy SID, należy sprawdzić i ewentualnie nadać wartość zmiennej *ORACLE\_HOME* (listing 2.3). Zmienna ta określa pełną nazwę i lokalizację katalogu głównego systemu bazodanowego Oracle.

*Listing 2.1. Utworzenie identyfikatora bazy danych*

```
set ORACLE_SID=nazwa
```

*Listing 2.2. Wyświetlenie identyfikatora bazy danych*

```
echo %ORACLE_SID%
```

*Listing 2.3. Utworzenie zmiennej ORACLE\_HOME*

```
set ORACLE_HOME=ścieżka
```

Podczas manualnego tworzenia bazy danych, administrator musi być odpowiednio uwierzytelniony oraz posiadać odpowiednie uprawnienia (Oracle, 2011d). Istnieją dwie metody uwierzytelnienia użytkownika, tj. plik haseł oraz uwierzytelnienie systemowe. Pierwsze oznacza przypisanie do użytkownika hasła, które jest zapisane w pliku haseł. Przy użyciu drugiej metody należy pamiętać, aby być zalogowanym na hoście komputera, który należy do grupy użytkowników systemowych. W trakcie instalacji systemu bazodanowego Oracle na platformie Windows, użytkownik automatycznie jest dodawany do wymaganej grupy (Oracle, 2011d).

Kolejnym etapem jest utworzenie pliku zawierającego parametry inicjalizacyjne

(ang. *initialization parametr file*) (Bryla, Loney, 2010; Oracle, 2011d). Jest to plik wymagany przy każdym uruchomieniu instancji bazy danych. Może on mieć format pliku tekstowego lub binarnego. Istnieją dwa rodzaje plików: *PFILE* oraz *SPFILE*. Pierwszym z nich jest plik *init.ora*. Jego nazwa powinna być utworzona jako *init<SID>.ora*, czyli zawierać SID bazy. Drugi z plików jest plikiem parametrów serwera o nazwie *spfile<SID>.ora*.

Po uruchomieniu instancji bazy danych, szukany jest plik *SPFILE* w domyślnej lokalizacji (unikalnej dla platformy systemowej), która została stworzona podczas instalacji systemu Oracle. Jeśli plik ten nie zostanie znaleziony, szukany jest plik *PFILE* (Bryla, Loney, 2010).

W plikach parametrów inicjalizacyjnych znajdują się informacje o lokalizacji plików śladu, plików sterowania, plików dziennika powtórzeń i innych, niezbędnych do prawidłowego działania bazy danych. Zawierają one także informacje o ograniczeniach rozmiarów struktur danych w obszarze SGA oraz określają maksymalną liczbę użytkowników, którzy mogą jednocześnie łączyć się z bazą danych.

Zawartość pliku *PFILE* musi być aktualizowana przez administratora, który przeprowadza ich edycję, szczególnie w sytuacji, gdy zostały wprowadzone zmiany w ustawieniach bazy danych poprzez polecenie *ALTER SYSTEM*. Dopiero manualne odzworowanie tych zmian w pliku gwarantuje ich zastosowanie także przy kolejnych uruchomieniach instancji bazy danych. Plik *SPFILE* jest wygodniejszy w użyciu, gdyż administrator nie musi nanosić w nim zmian, spowodowanych wykonaniem polecenia *ALTER SYSTEM*. Polecenie to automatycznie modyfikuje zawartość tego pliku.

O znaczeniu tych plików świadczy także fakt, iż administrator powinien tworzyć ich kopie zapasowe. Będą one wykorzystane na etapie poprawnego odtwarzania bazy danych. Jeśli do tworzenia bazy danych zostało użyte narzędzie *DBCA*, plik *SPFILE* zostanie utworzony automatycznie.

Aby zapewnić dostęp do bazy danych, należy jeszcze utworzyć jej wystąpienie, czyli instancję. Dopiero po jej stworzeniu można będzie połączyć się z bazą danych. Instancja (ang. *instance*) składa się z jednego obszaru pamięci, zlokalizowanego w obszarze SGA. Obszar ten komunikuje się z bazą danych przez procesy działające w tle (Bryla, Loney, 2010).

Utworzenie nowej instancji na platformie Windows, przy zastosowaniu wiersza poleceń, wymaga użycia polecenia *oradim* (Oracle, 2011d). Przykład takiej instrukcji przedstawiono na listingu 2.4. Parametr *sid* oznacza identyfikator SID, który został nadany bazie danych. Parametr *pfile* oznacza pełną ścieżkę do pliku zawierającego parametry inicjujące.

Listing 2.4. Tworzenie instancji bazy danych

```
oradim -NEW -SID sid -STARTMODE MANUAL -PFILE pfile
```

Polecenie z listingu 2.4 tworzy instancję bazy danych, ale jej nie uruchamia. Na tym etapie nie należy nadawać parametrowi uruchamiania *STARTMODE* wartości *AUTO*, gdyż baza danych w tym momencie nie została jeszcze utworzona. Dopiero po jej utworzeniu można dokonać zmiany trybu uruchamiania (Oracle, 2011p).

Polecenie *oradmin*, wykorzystywane w dotychczasowych operacjach, jest dostępne po zainstalowaniu systemu Oracle. Jest ono wykorzystywane jedynie przy ręcznym tworzeniu, modyfikowaniu i usuwaniu bazy danych. Wszystkie dostępne parametry tego polecenia zostały przedstawione na listingu 2.5 (Oracle, 2011p).

Listing 2.5. Polecenie ORADIM

```
oradim [-NEW -SID SID] | -SRVC service_name |  
-ASMSID SID | -ASMSRVC service_name  
[-SYSPWD password] [-STARTMODE auto | manual]  
[-SRVCSTART system | demand]  
[-PFILE filename | -SPFILE] [-SHUTMODE normal |  
immediate | abort] [-TIMEOUT secs] [-RUNAS osusr/ospass]
```

Znaczenie poszczególnych parametrów jest następujące:

- *-NEW* – utworzenie nowej instancji bazy danych;
- *-SID* – identyfikator SID bazy danych, który będzie nazwą tworzonej instancji;
- *-SRVC* – nazwa usługi do utworzenia (*OracleServiceSID*);
- *-ASMSID SID* – nazwa instancji *Automatic Storage Management* do utworzenia;



- `-ASMSRVC service_name` – nazwa tworzonej usługi *Automatic Storage Management Service*;
- `-SYSPWD password` – systemowe hasło do uwierzytelnienia;
- `-STARTMODE auto | manual` – wskazuje na rodzaj uruchamiania instancji bazy danych, domyślnie ustawiona jest wartość *manual*;
- `-SRVSTART system | demand` – definiuje, czy baza danych Oracle będzie uruchamiana podczas restartu systemu operacyjnego, domyślnie włączona jest opcja *demand*;
- `-PFILE filename` – wskazanie pliku inicjalizacyjnego, używanego przez instancję bazy danych, należy podać pełną ścieżkę do pliku i jego nazwę;
- `-SPFILE` – wymuszenie korzystania przez instancję bazy danych z pliku *SPFILE* zamiast z *PFILE*;
- `-SHUTDOWN normal | immediate | abort` – definiuje sposób zatrzymywania instancji bazy danych, jest to opcjonalny parametr (domyślnie *normal*);
- `-TIMEOUT secs` – określa maksymalny czas (w sekundach) oczekiwania przed zatrzymaniem usługi SID; domyślnie wynosi on 90 sekund; parametr ten może być użyty nawet wtedy, gdy nie został zdefiniowany parametr *SHUTDOWN*;
- `-RUNAS osusr/ospass` – uruchamia usługę jako podany użytkownik i działa z jego uprawnieniami; jest to parametr opcjonalny.

Po utworzeniu instancji bazy danych, można się z nią połączyć przy użyciu narzędzia SQL\*Plus. W celu zalogowania się z uprawnieniami użytkownika *SYSDBA*, należy wykonać polecenie z listingu 2.6. Jeśli użytkownik jest uwierzytelniany hasłem, powinien wykonać pierwsze 2 linijki listingu 2.6. W przypadku uwierzytelnienia przez system operacyjny, należy wykonać dwa ostatnie polecenia.

Listing 2.6. Zalogowanie się do SQL\*Plus przez użytkownika SYSDBA

```
sqlplus /nolog
SQL> CONNECT SYS AS SYSDBA
sqlplus /nolog
SQL> CONNECT / AS SYSDBA
```

Źródło: (Oracle, 2011d)

Przed logowaniem trzeba upewnić się, że zmienna `ORACLE_HOME` jest zdefiniowana. Jeśli zalogowanie powiodło się, oznacza to, że instancja została uruchomiona poprawnie.

Kolejnym etapem manualnego tworzenia bazy danych może być utworzenie pliku `SPFILE` na podstawie istniejącego pliku `PFILE` (Oracle, 2011d). Wykonanie odpowiedniego polecenia (listing 2.7) powoduje odczytanie zawartości pliku `PFILE` z domyślnej lokalizacji, utworzenie pliku `SPFILE` oraz zapisanie utworzonego dokumentu w domyślnej lokalizacji o domyślnej nazwie. Istnieje także możliwość podania lokalizacji oraz nazw tych dwóch plików innych niż domyślnie stosowane.

Listing 2.7. Utworzenie pliku `SPFILE`

```
CREATE SPFILE FROM PFILE;
```

W celu sprawdzenia poprawności działania bazy danych, po utworzeniu pliku `SPFILE`, konieczne jest ponowne jej uruchomienie.

Pomimo, że krok dotyczący utworzenia pliku `SPFILE` może być pominięty, zalecane jest jego stworzenie przez administratora.

Niekiedy zachodzi potrzeba uruchomienia instancji bazy bez jej włączania, czyli przykładowo podczas jej tworzenia lub wykonywania na niej czynności konserwacyjnych. W tym celu należy wydać polecenie `STARTUP` (listing 2.8). Jeśli plik `PFILE` znajduje się w domyślnej lokalizacji, nie trzeba podawać do niego ścieżki.

Listing 2.8. Uruchomienie instancji bazy danych

```
STARTUP NOMOUNT
```

Po przeprowadzeniu wszystkich powyższych operacji, można przystąpić do tworzenia bazy danych z użyciem polecenia `CREATE DATABASE`. Polecenie to wykorzystuje się do realizacji wielu zadań (Oracle, 2011r). Za jego pomocą można:

- utworzyć pliki danych bazy danych,
- utworzyć pliki kontrolne dla bazy danych,

- utworzyć pliki *redo logs* oraz pliki archiwizacyjne,
- utworzyć przestrzeń tabel dla użytkownika *SYSTEM*,
- utworzyć przestrzeń tabel dla użytkownika *SYSAUX*,
- utworzyć dane słownikowe (tzw. metadane),
- ustawić strefę czasową bazy danych,
- zamontować i otworzyć bazę danych do użytku.

Listing 2.9 przedstawia przykładowe polecenie tworzenia nowej bazy danych.

Listing 2.9. Polecenie *CREATE DATABASE*

```
CREATE DATABASE nowa1
  USER SYS IDENTIFIED BY usrsys_pass
  USER SYSTEM IDENTIFIED BY usrsystem_pass
  LOGFILE
    GROUP 1 ('D:\app\oradata\nowa1\redo01.log') SIZE 100M,
    GROUP 2 ('D:\app\oradata\nowa1\redo02.log') SIZE 100M,
    GROUP 3 ('D:\app\oradata\nowa1\redo03.log') SIZE 100M
  MAXLOGFILES 5
  MAXLOGMEMBERS 5
  MAXLOGHISTORY 1
  MAXDATAFILES 100
  MAXINSTANCES 1
  CHARACTER SET US7ASCII
  NATIONAL CHARACTER SET AL16UTF16
  DATAFILE 'D:\app\oradata\nowa1\system01.dbf'
    SIZE 325M REUSE EXTENT MANAGEMENT LOCAL
  SYSAUX DATAFILE 'D:\app\oradata\nowa1\sysaux01.dbf'
    SIZE 325M REUSE
  DEFAULT TABLESPACE USERS
  DATAFILE 'D:\app\oradata\nowa1\users01.dbf' SIZE 50M REUSE
  DEFAULT TEMPORARY TABLESPACE tempts
  TEMPFILE 'D:\app\oradata\nowa1\temp01.dbf' SIZE 20M REUSE
  UNDO TABLESPACE undotbs
  DATAFILE 'D:\app\oradata\nowa1\undo01.dbf' SIZE 20M REUSE
  AUTOEXTEND ON MAXSIZE UNLIMITED;
```

Źródło: (Bryla, Loney, 2010; Oracle, 2011d)

Wykonanie polecenia z listingu 2.9 spowoduje utworzenie bazy danych o nazwie *nowa1*. Druga i trzecia linia kodu definiują hasła dostępu dla użytkowników *SYS* oraz *SYSTEM* (należy pamiętać, że w środowisku Oracle rozróżniane są wielkie i małe

liter, co jest szczególnie ważne w tym przypadku). Definicja tych haseł ma charakter opcjonalny, ale jeżeli jedno z nich zostanie podane, wymagane jest również podanie drugiego (Oracle, 2011q).

Następnie definiowane są trzy pliki dziennika powtórzeń (ang. *redo log file*), każdy o rozmiarze 100 MB. Parametry *MAXLOGFILES*, *MAXLOGMEMBERS* oraz *MAXLOGHISTORY* nakładają ograniczenia na tworzone dzienniki powtórzeń, określając odpowiednio: maksymalną liczbę grup plików dla bazy danych, maksymalną liczbę użytkowników grupy oraz maksymalną liczbę archiwizowanych plików dziennika powtórzeń, które mogą zostać zapisane w historii logów plików kontrolnych (Oracle, 2011j). Parametr *MAXDATAFILES* precyzuje maksymalną liczbę plików bazy danych, które mogą być otwarte.

Kolejny parametr określa sposób kodowania znaków w bazie danych. W powyższym przykładzie kodowanie zostało ustalone jako *AL16UTF16*. Oznacza to, że wybrany sposób kodowania będzie używany do zapisu danych w kolumnach bazy danych oraz dla takich typów danych jak *NCHAR*, *NCLOB* czy *NVARCHAR2*.

Parametr *DATAFILE* pozwala utworzyć plik danych o podanej nazwie w lokalizacji wskazanej przez ścieżkę dostępu. Jeśli taki plik już istnieje, zostanie on nadpisany.

Następnie, przy pomocy parametru *TABLESPACE*, tworzone są przestrzenie tabel: domyślne, tymczasowe oraz wycofania. Dla każdej przestrzeni podana jest nazwa pliku wraz ze ścieżką dostępu oraz jej rozmiar. Opcja *AUTOEXTEND* zapewnia możliwość automatycznego powiększania obszaru przestrzeni, jeśli zabraknie w niej miejsca na obiekty lub dane, tworzone przez użytkowników bazy danych. Można tu podać konkretną wartość, o którą zostanie powiększony obszar pamięci, np. 25 MB. Należy jednak pamiętać o ograniczeniach wynikających z dostępnej przestrzeni dyskowej. W przykładzie 2.10 ograniczenie związane z maksymalnym rozmiarem pliku zostało zniesione.

Po utworzeniu bazy danych, można definiować dodatkowe przestrzenie tabel w celu poprawnego działania bazy danych. Listing 2.10 ilustruje sposób utworzenia przestrzeni tabel o nazwie *apps\_tbs* z jednym plikiem danych, który powstanie we wskazanej lokalizacji. Rozmiar pliku został ustalony na 500 MB. W sytuacji, w której zabraknie miejsca w przestrzeni, możliwe będzie powiększanie rozmiaru pliku o kolejne 1280 KB. Maksymalny rozmiar pliku nie został ograniczony parametrami

polecenia. Ponadto utworzona przestrzeń jest lokalnie zarządzana, dzięki czemu możliwe jest ręczne sterowanie jej obszarami.

*Listing 2.10. Utworzenie dodatkowych przestrzeni tabel*

```
CREATE TABLESPACE apps_tbs
  DATAFILE 'D:\app\oradata\nowa1\apps01.dbf'
  SIZE 500M REUSE AUTOEXTEND ON NEXT 1280K
  MAXSIZE UNLIMITED
  EXTENT MANAGEMENT LOCAL;
```

*Źródło: (Oracle, 2011d; Oracle, 2011g)*

Po zainstalowaniu bazy danych można także wywołać skrypty, które utworzą perspektywy, synonimy, pakiety i inne obiekty potrzebne do poprawnego działania bazy (Oracle, 2011p).

Dobłą praktyką, po pełnym zainstalowaniu bazy danych ze wszystkimi opcjami, jest wykonanie kopii bezpieczeństwa tej bazy (Oracle, 2011o). Działanie takie daje pewność, że w przypadku awarii będzie można odzyskać strukturę bazy danych oraz wszystkie pliki, niezbędne do prawidłowej eksploatacji bazy.

Innym działaniem, realizowanym po zakończeniu instalacji bazy danych, jest ustawienie automatycznego włączania instancji po uruchomieniu lub restarcie komputera. Aby włączyć tę opcję, należy wykonać polecenia podane na listingu 2.11 (w miejsce *sid* należy wstawić identyfikator bazy danych). Dodatkowo w poleceniu tym można podać, jako parametr, plik *SPFILE*, jeśli został on wcześniej utworzony.

*Listing 2.12. Ustawienie automatycznego uruchamiania instancji bazy danych*

```
oradadmin -EDIT -SID sid -STARTMODE AUTO
          -SRVSTART SYSTEM [-SPFILE]
```

*Źródło: (Oracle, 2011p)*

Po utworzeniu bazy danych można przystąpić do dalszych czynności związanych z wprowadzaniem, przechowywaniem i przetwarzaniem danych użytkowników oraz aplikacji.

## 2.4. PYTANIA KONTROLNE

1. Jakie aspekty powinny być uwzględnione przez administratora w trakcie przygotowywania się do tworzenia bazy danych?
2. W jaki sposób można utworzyć bazę danych w systemie Oracle 11g?
3. Wymień podstawowe szablony bazy danych przy tworzeniu bazy danych narzędziem *DBCA*.
4. Co to jest SID bazy danych?
5. Jakie parametry należy podać w celu konfiguracji kopii zapasowej bazy danych?
6. Jakie są rodzaje uwierzytelnienia użytkowników bazy danych?
7. Wymień pliki inicjalizacyjne bazy danych Oracle.
8. Co to jest instancja bazy danych?
9. Jak można uruchomić i edytować opcje instancji bazy danych?
10. Omów polecenie *ORADIM*.
11. W jaki sposób można uruchomić SQL\*Plus?
12. W jaki sposób można utworzyć plik *SPFILE*?
13. Opisz polecenie *CREATE DATABASE*.

## Administrowanie bazą danych Oracle 11g

---

### Cel

Celem rozdziału jest przedstawienie złożonej struktury bazy danych systemu Oracle 11g. Poprzez jej poznanie użytkownik zostanie zaznajomiony z wybranymi zagadnieniami administrowania bazą danych: obszarami pamięci *SGA* i *PGA*, obiektami bazy danych oraz zapewnieniem jej bezpieczeństwa poprzez definiowanie kont użytkowników oraz zarządzaniem ich przywilejami.

### Plan

1. Wprowadzenie.
2. Architektura bazy danych Oracle.
3. Logiczne struktury przechowywania danych systemu Oracle.
4. Fizyczne struktury przechowywania danych systemu Oracle.
5. Struktury pamięci systemu Oracle.
6. Strojenie użycia pamięci systemu Oracle.
7. Zarządzanie obiektami bazy danych Oracle.
8. Mechanizmy bezpieczeństwa bazy danych Oracle.

### 3.1. WPROWADZENIE

Praca administratora jest odpowiedzialnym zajęciem, wymagającym doskonałej znajomości struktury systemu Oracle oraz systemu operacyjnego, na którym system bazodanowy jest zainstalowany. Niekiedy administrator bierze także czynny udział w tworzeniu aplikacji opartej na systemie Oracle, gdyż jego wiedza i doświadczenie w tym obszarze pozwalają na optymalną konfigurację bazy danych.

Główne działania wykonywane przez administratora to (Oracle, 2008):

- tworzenie i konfigurowanie baz danych Oracle;
- monitorowanie i strojenie baz danych Oracle;
- tworzenie schematów obiektów, takich jak: tabele, indeksy i widoki;
- diagnozowanie i rozwiązywanie problemów związanych z raportowaniem;
- tworzenie kopii zapasowych bazy danych;
- odtworzenie kopii danych;
- zapewnienie bezpieczeństwa danych, poprzez definiowanie i przydzielanie praw dostępu, co zapobiega nieautoryzowanemu dostępowi do danych;
- zapewnienie bezawaryjnej pracy bazy danych;
- gwarantowanie wysokiej wydajności bazy danych.

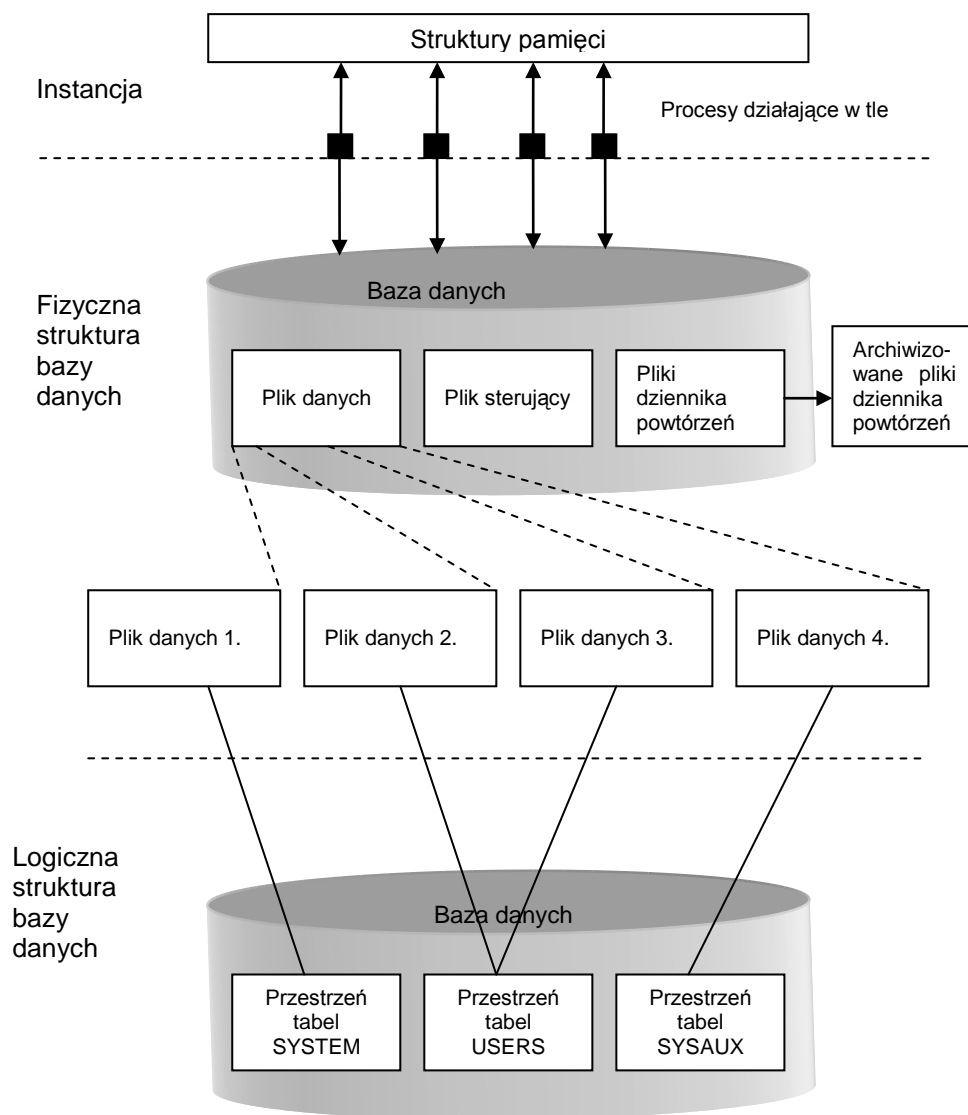
Administrator musi posiadać teoretyczną i praktyczną wiedzę, dzięki której może wykonywać wiele operacji zarządzających, definiujących oraz monitorujących pracę bazy danych.

### 3.2. ARCHITEKTURA BAZY DANYCH ORACLE

Baza danych (ang. *database*) jest to zbiór danych na dysku, przechowywanych w co najmniej jednym pliku na serwerze odpowiedzialnym za zbieranie i obsługę informacji (Bryla, Loney, 2010).

Architektura bazy danych systemu Oracle jest złożona (rys. 3.1). Składa się z trzech podstawowych części: instancji, fizycznych i logicznych struktur.





Rys. 3.1. Architektura bazy danych Oracle 11g

Źródło: (Bryla, Loney, 2010)

Instancja bazy danych (ang. *instance*) składa się z jednego bloku pamięci, który umieszczony jest w globalnym obszarze systemu SGA (ang. *System Global Area*)

(Bryla, Loney, 2010). Poprzez procesy działające w tle, zachodzi komunikacja pomiędzy instancją a bazą danych. Należy wyraźnie podkreślić różnicę między tymi dwoma pojęciami. W konfiguracji RAC (ang. *Real Application Cluster*) z danej bazy danych może korzystać więcej niż jedna instancja (Bryla, Loney, 2010).

Logiczne struktury bazy danych grupują dane w postaci przestrzeni tabel (Bryla, Loney, 2010). Każda z nich przetrzymuje takie struktury danych jak tabele i indeksy. Każda przestrzeń podzielona jest na segmenty, które z kolei składają się z obszarów. Taka struktura przechowywania danych ma na celu zapewnienie lepszego wykorzystania przestrzeni na dysku oraz stwarza większe możliwości zarządzania nimi przez administratora, np. poprzez ustalenie ich rozmiarów.

Fizyczne struktury bazy danych służą do przechowywania danych, które są związane z transakcjami użytkowników bazy danych (Bryla, Loney, 2010). Są to przede wszystkim takie pliki jak: pliki danych, pliki sterujące, pliki dziennika powtórzeń, archiwizowane pliki dziennika powtórzeń, pliki parametrów inicjujących, pliki alertów i dziennika śladu, pliki kopii zapasowych oraz pliki haseł.

### 3.3. LOGICZNE STRUKTURY PRZECHOWYWANIA DANYCH

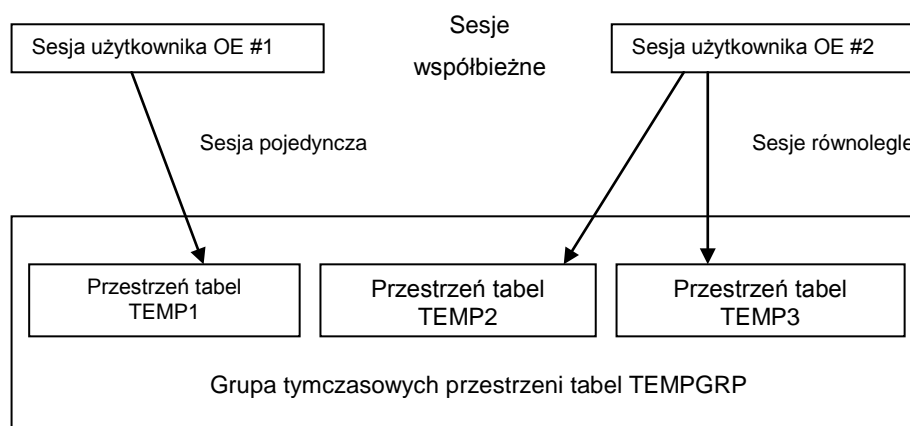
Baza danych Oracle przechowuje dane w jednej lub większej liczbie przestrzeni tabel. Przestrzeń tabel (ang. *tablespace*) systemu Oracle składa się z co najmniej jednego pliku umieszczonego na dysku twardym (Bryla, Loney, 2010). Każda przestrzeń może zawierać wiele plików, podczas gdy dany plik może należeć tylko do jednej przestrzeni tabel.

W systemie Oracle wyróżnia się trzy rodzaje przestrzeni: trwałe, wycofania oraz tymczasowe (Bryla, Loney, 2010). Przestrzeń tabel można także podzielić na wielkoplikowe (ang. *Bigfile tablespace*) oraz przestrzeń małych plików (ang. *Small tablespace*) (Bryla, Loney, 2010; Oracle, 2011e).

Przestrzeń trwałe posiadają segmenty, które muszą być dostępne także po zakończeniu sesji lub transakcji. Przykładami takich przestrzeni są *SYSTEM* oraz *SYSAUX*. Administrator może także tworzyć inne trwałe przestrzenie tabel po zakończeniu procesu tworzenia bazy danych.

Przestrzenie tabel wycofania, podobnie jak przestrzenie trwałe, mogą przechowywać segmenty po zakończeniu sesji. Jest to związane z obsługą wycofywania transakcji, zapewnienia spójności danych, ich odczytów podczas wykonywania instrukcji *SELECT* równoległe z instrukcjami *DML* na tych samych tabelach. W bazie danych może być zdefiniowanych wiele takich przestrzeni, ale tylko jedna z nich może być aktywna w danym momencie. Administrator powinien przydzielić im odpowiedni rozmiar do poprawnego działania bazy danych.

W Oracle 11g może być aktywnych więcej niż jedna przestrzeń tabel tymczasowych (Bryla, Loney, 2010). W poprzednich wersjach systemu bazodanowego w danym momencie mogła być używana tylko jedna przestrzeń tego rodzaju dla użytkownika bazy danych. Dlatego zostało wprowadzone pojęcie grupy przestrzeni tabel. Ich działanie zostało pokazane na rysunku 3.2.



Rys. 3.2. Grupa przestrzeni tabel tymczasowych TEMPGRP

Źródło: (Bryla, Loney, 2010)

Grupa tabel tymczasowych składa się z przynajmniej jednej przestrzeni. Jej główną zaletą jest fakt, że sesje tego samego użytkownika mogą używać różnych przestrzeni tabel. Na rysunku 3.2 przedstawione są dwie sesje jednego użytkownika. Pierwsza jest sesją pojedynczą, a druga równoległą. Obie korzystają z przestrzeni tymczasowych.

Przestrzenie tabel wielkoplikowych (Bryla, Loney, 2010; Oracle, 2011e) ułatwiają zarządzaniem bazą danych przez administratora. Taka przestrzeń zawiera tylko jeden plik, za to o dużych rozmiarach aż do 128 TB, przy rozmiarze bloku przestrzeni 32 KB. Wprawdzie administrowanie pojedynczym plikiem może wydawać się łatwiejsze, to jednak rozwiązanie takie posiada także pewne wady. Sporą niedogodnością jest tutaj czas trwania wykonywania kopii tak dużego pliku.

Przestrzenie tabel tworzone są już podczas instalacji systemu Oracle 11g. W jej trakcie zostają utworzone dwie domyślne przestrzenie o nazwach: *SYSTEM* oraz *SYSAUX*. Jednak już w trakcie domyślnej instalacji Oracle 11g powstaje aż sześć różnych przestrzeni tabel, które zostały przedstawione w tabeli 3.1.

*Tabela 3.1. Charakterystyka przestrzeni tabel tworzonych podczas standardowej instalacji bazy danych systemu Oracle 11g*

<b>Przestrzeń</b>	<b>Typ</b>	<b>Zarządzanie</b>	<b>Alokowany rozmiar (MB)</b>
SYSTEM	stała	Automatyczne	450
SYSAUX	stała	Automatyczne	350
TEMP	tymczasowa	Ręczne	20
UNDOTBS1	stała	Ręczne	30
USERS	stała	Automatyczne	5
EXAMPLE	stała	Automatyczne	150

*Źródło: (Bryla, Loney, 2010)*

Przestrzeń tabel *SYSTEM* jest pierwszą przestrzenią tworzoną w trakcie instalacji systemu Oracle (Bryla, Loney, 2010; Oracle, 2011e). Jest to trwała przestrzeń, która zawiera podstawowe informacje związane z funkcjonalnością serwera bazy danych, takie jak: dane słownikowe oraz segment wycofania systemu. Nie powinna ona zawierać segmentów użytkownika. Dodatkowo posiada ona także pewne ograniczenia. Nie jest możliwe usunięcie tej przestrzeni ani zmiana jej nazwy. Nie można też ustawić jej statusu na offline.

Przestrzeń *SYSAUX* (Bryla, Loney, 2010; Oracle, 2011h) także nie powinna

zawierać żadnych segmentów użytkownika. Jest ona pomocnicza w stosunku do przestrzeni *SYSTEM*. Przechowuje dane dla wybranych komponentów bazy Oracle takich jak: repozytorium *Enterprise Manager Repository*, *LogMiner*, *Oracle Spatial* i *Oracle Text*. Możliwe jest sprawdzenie aktualnych komponentów, które należą do tej przestrzeni. Należy wykonać polecenie przedstawione na listingu 3.1. Polecenie to wyświetla nazwę, opis oraz zajęta przestrzeń w kilobajtach. Wszystkie te dane są przechowywane w widoku *V\$SYSAUX\_OCCUPANTS*.

*Listing 3.1. Polecenie wyświetlające listę aktualnych komponentów przestrzeni SYSAUX*

```
SELECT occupant_name, occupant_desc, space_usage_kbytes  
FROM v$sysaux_occupants;
```

*Źródło: (Bryla, Loney, 2010; Oracle, 2011h)*

W przypadku awarii tej przestrzeni, wszystkie jej komponenty nie będą dostępne. Natomiast praca bazy danych pozostanie niezakłócona. Przestrzeń *SYSAUX* odciąża przestrzeń *SYSTEM* podczas pracy bazy danych.

Przestrzeń *TEMP* (Bryla, Loney, 2010) jest przestrzenią tymczasową, tworzoną w trakcie instalacji bazy danych Oracle 11g. Administrator decyduje także o dodatkowym stworzeniu kolejnych przestrzeni tabel tymczasowych lub o utworzeniu grupy takich przestrzeni.

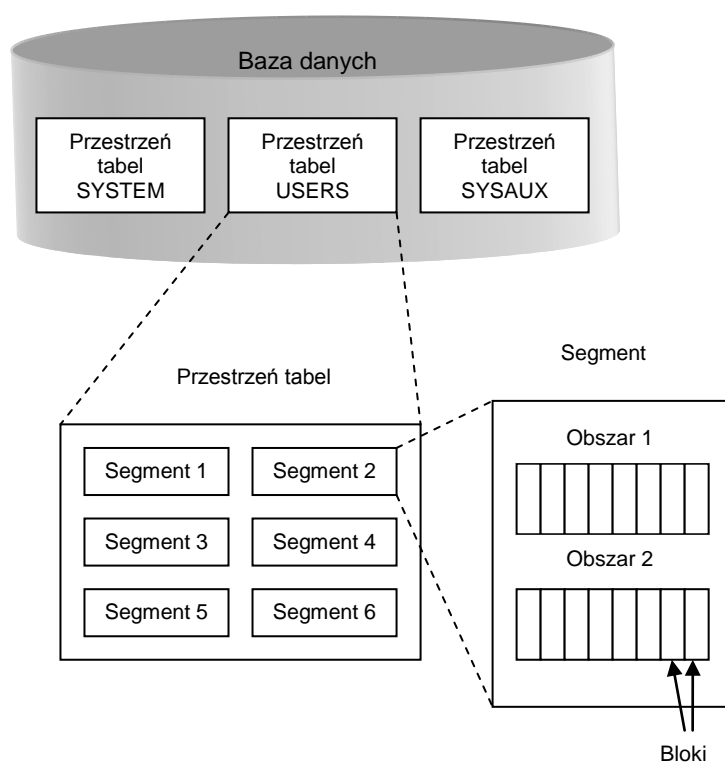
Przestrzeń *UNDOTBS1* (Bryla, Loney, 2010; Oracle, 2011i) jest przestrzenią wycofania. Administrator może tworzyć kolejne przestrzenie tego typu, biorąc pod uwagę, że w danym momencie może być aktywna tylko jedna z nich.

Kolejną przestrzenią stałą, tworzoną podczas instalacji bazy danych, jest przestrzeń *USERS* (Bryla, Loney, 2010). Jest to przestrzeń dedykowana dla użytkowników bazy danych. Posiada ona segmenty tworzone przez użytkowników. Nie powinna być stosowana w aplikacjach produkcyjnych, gdyż dla każdej aplikacji i typu segmentów powinny być utworzone oddzielne przestrzenie tabel.

Przestrzeń *EXAMPLE* zawiera przykłady wszystkich typów segmentów bazy danych Oracle oraz struktur danych (Bryla, Loney, 2010). Przestrzeń ta jest zbędna w środowisku produkcyjnym, dlatego powinno się ją usunąć.

Do prawidłowego zarządzania przestrzeniami danych, administrator musi posiadać pełną wiedzę o ich strukturze.

Każda przestrzeń tabel składa się z segmentów, obszarów i bloków. Logiczna struktura przechowywania danych została przedstawiona na rysunku 3.3.



Rys. 3.3. Logiczna struktura przechowywania danych

Źródło: (Bryla, Loney, 2010)

Najmniejszą jednostką przestrzeni jest blok (ang. *block*) (Bryla, Loney, 2010). Służy on do przechowywania danych bazy. Rozmiar bloku ustawiany jest przez administratora poprzez parametr *DB\_BLOCK\_SIZE*. Zazwyczaj blok posiada rozmiar równy wielokrotności rozmiaru bloków systemowych, co umożliwia efektywne wykorzystanie operacji wejścia-wyjścia.

Bloki bazy danych pogrupowane są w logiczne struktury zwane obszarami (ang. *extent*) (Bryla, Loney, 2010). Obszar składa się z przynajmniej jednego bloku. Przy alokowaniu większej ilości pamięci, dodawany jest kolejny obszar.

Jeszcze większą logiczną strukturą bazy danych jest segment (ang. *segment*) (Bryla, Loney, 2010). Segment jest grupą obszarów. Tworzy on obiekt bazy danych, np. tabelę lub indeks. Jest to też najmniejsza porcja danych, która będzie widoczna dla użytkownika bazy danych. Występują cztery rodzaje segmentów: danych, indeksów, tymczasowe oraz wycofania.

### 3.4. FIZYCZNE STRUKTURY PRZECHOWYWANIA DANYCH

Fizyczne struktury dyskowe służą do przechowywania danych, które związane są z transakcjami wykonywanymi przez użytkowników, plikami przechowującymi informacje o stanie obiektów bazy danych oraz tymi, które informują o zdarzeniach oraz błędach zaistniałych w bazie danych (Bryla, Loney, 2010). Główne pliki, składające się na fizyczną strukturę danych, to: pliki danych, pliki dziennika powtórzeń, pliki sterujące, archiwizowane pliki dziennika, pliki parametrów inicjujących, pliki alertów i dziennika śladu, pliki kopii zapasowych oraz pliki haseł.

Pliki danych (ang. *datafiles*) zawierają dane systemu Oracle oraz struktury wspomagające (Bryla, Loney, 2010). Baza Oracle posiada co najmniej jeden plik danych. Każdy plik należy do jednej przestrzeni tabel. Rozmiar takiego pliku można kontrolować. Parametr *MAXSIZE* określa jego maksymalny rozmiar. Możliwe jest także automatyczne rozszerzanie jego rozmiaru, jeśli zostanie on utworzony z parametrem *AUTOEXTENT*. Oczywiście rozmiar pliku danych nie może być większy niż pojemność dysku danych, na którym się znajduje.

W bazie danych każde działanie i zdarzenie jest zapisywane w specjalnych plikach zwanych plikami dziennika powtórzeń (ang. *redo log files*) (Bryla, Loney, 2010). Muszą istnieć przynajmniej dwa takie pliki, które działają naprzemiennie. Ale w danej chwili tylko jeden z nich jest bieżący, czyli ma przypisany status *CURRENT*. Najpierw w jednym pliku zapisywane są zdarzenia, a po jego wypełnieniu dane zapisywane są w kolejnym pliku. Jeśli plik jest jeszcze niezbędny do odtwarzania

danych instancji, posiada status *ACTIVE*. W przeciwnym razie przypisywany mu jest status *INACTIVE*. Dane zapisane w plikach dziennika powtórzeń są niezbędne w przypadku awarii bazy danych, szczególnie podczas aktualizacji danych, gdy zmiany nie zostały jeszcze całkowicie zapisane w plikach danych. Administrator powinien zabezpieczyć się przed ewentualną utratą chociaż jednego pliku dziennika powtórzeń poprzez tworzenie ich kopii zapasowej.

Plik sterujący (ang. *control file*) przechowuje informacje o całej fizycznej strukturze bazy danych (tzw. metadane). Zawiera on m. in.: nazwę bazy danych, informacje o utworzeniu bazy, nazwy i lokalizacje wszystkich plików danych i plików dziennika powtórzeń, trwałe ustalenia menedżera *RMAN* oraz typy kopii zapasowych. Jest to bardzo ważny plik związany z poprawnym działaniem bazy danych. System zarządzania bazą umożliwia analizowanie tylko jednego pliku. Dobrą praktyką jest wykonanie kopii zapasowej tego pliku, aby w razie awarii móc odtworzyć konfigurację takiej bazy.

Istnieją dwa tryby działania bazy danych: *ARCHIVELOG* oraz *NOARCHIVELOG* (Bryła, Loney, 2010; Wrembel, 1998). Jeśli baza pracuje w trybie *NOARCHIVELOG*, dane zapisywane są cyklicznie do plików dziennika powtórzeń online (ang. *online redo log files*). Jeśli zdarzy się awaria, np. dysku, nie będzie możliwe przywrócenie danych, ponieważ nie ma dostępu do poprzednich wpisów dziennika. Tryb ten natomiast zapewnia integralność danych, gdyż zatwierdzone transakcje, których dane nie zostały jeszcze zapisane w pliku danych, będą dostępne w plikach dziennika powtórzeń online. Jeśli baza danych pracuje w trybie *ARCHIVELOG*, plik dziennika powtórzeń jest przesyłany do wcześniej ustalonej lokalizacji (Bryła, Loney, 2010). Taki plik można użyć do rekonstrukcji bazy danych w przypadku jej awarii. Przechowywanie pliku dziennika powtórzeń pozwala na zachowanie wysokiej dostępności danych.

Kluczowym elementem, niezbędnym do prawidłowego działania bazy danych, jest plik parametrów inicjujących (ang. *initialization parameter files*). Podczas uruchamiania instancji bazy danych Oracle, wyszukiwany i analizowany jest jeden plik inicjujący. Plik ten zawiera m. in.: lokalizację plików śladu, plików sterowania, plików dziennika powtórzeń. Występują dwa rodzaje tych plików: *PFILE* oraz *SPFILE*. Pierwszy z nich jest plikiem tekstowym lub binarnym, zwanym także *init.ora*. Drugi jest



plikiem parametrów serwera. W pierwszej kolejności wyszukiwany jest plik *SPFILE*. Jeśli nie zostanie on znaleziony, wyszukiwany jest plik *PFILE*. Dla administratora wygodniejszym w użyciu jest plik serwera, gdyż w przypadku wykonania instrukcji *ALTER SYSTEM* zmiany automatycznie są odzwierciedlane w tym pliku. Natomiast plik *PFILE* trzeba edytować ręcznie.

Praca bazy danych jest monitorowana w sposób ciągły. Jeśli wystąpi jakikolwiek błąd, komunikaty o tym zdarzeniu umieszczane są w specjalnych plikach dziennika alertów (ang. *alert log*). W przypadku kłopotów z procesami działającymi w tle lub sesjami użytkowników, informacje takie umieszczane są w plikach dziennika śladów (ang. *trace log*).

Plik dziennika alertów umieszczony jest w katalogu. Można określić jego pełną nazwę za pomocą parametru inicjującego *BACKGROUND\_DUMP\_DEST*. Plik ten zawiera takie dane jak (Bryla, Loney, 2010):

- rutynowe komunikaty o stanie bazy danych;
- informacje o błędach, np. brak miejsca w przestrzeni tabel, uszkodzenie w pliku dziennika powtórzeń;
- listę parametrów podczas włączania i wyłączania bazy danych;
- zapisy o wszystkich poleceniach *ALTER DATABASE* oraz *ALTER SYSTEM*, wykonywanych przez administratora bazy danych;
- wszystkie operacje związane z przestrzeniami tabel i ich plikami danych, np. dodawanie przestrzeni tabel czy też dodawanie plików do przestrzeni tabel.

Lokalizację pliku śladu, monitorującego działanie instancji bazy danych, można także ustawić w parametrze *BACKGROUND\_DUMP\_DEST*. Dla sesji użytkownika oraz połączeń z bazą danych także tworzone są pliki śladu. Lokalizacje tych plików przetrzymywane są w parametrze *USER\_DUMP\_DEST*.

Pliki śladu dla procesów użytkownika tworzone są w przypadkach, gdy (Bryla, Loney, 2010):

- wystąpi błąd związany z niewłaściwymi uprawnieniami czy brakiem wolnego miejsca;
- ustawiona zostanie opcja ręcznego ich tworzenia (listing 3.2) – pliki tworzone będą dla każdej instrukcji SQL wykonywanej przez użytkownika.

Listing 3.2. Polecenie ręcznego tworzenia pliku śladu

```
ALTER SESSION SET SQL_TRACE=TRUE
```

Możliwe jest także tworzenie i zarządzanie plikami alertów. Oznacza to, że pliki takie można usuwać, zmieniać ich nazwę oraz przenosić do innej lokalizacji. Administrator może utworzyć zadanie wsadowe, dzięki któremu pliki alertów będą archiwizowane każdego dnia (Bryla, Loney, 2010).

Bardzo istotnym zadaniem administratora jest tworzenie kopii zapasowych, które wykonywane są przy pomocy poleceń systemowych lub narzędzia *RMAN* (ang. *Oracle Recovery Manager*). Pliki kopii obejmują pliki danych, dziennika powtórzeń, pliki sterujące, archiwizowane pliki dziennika powtórzeń, itp.

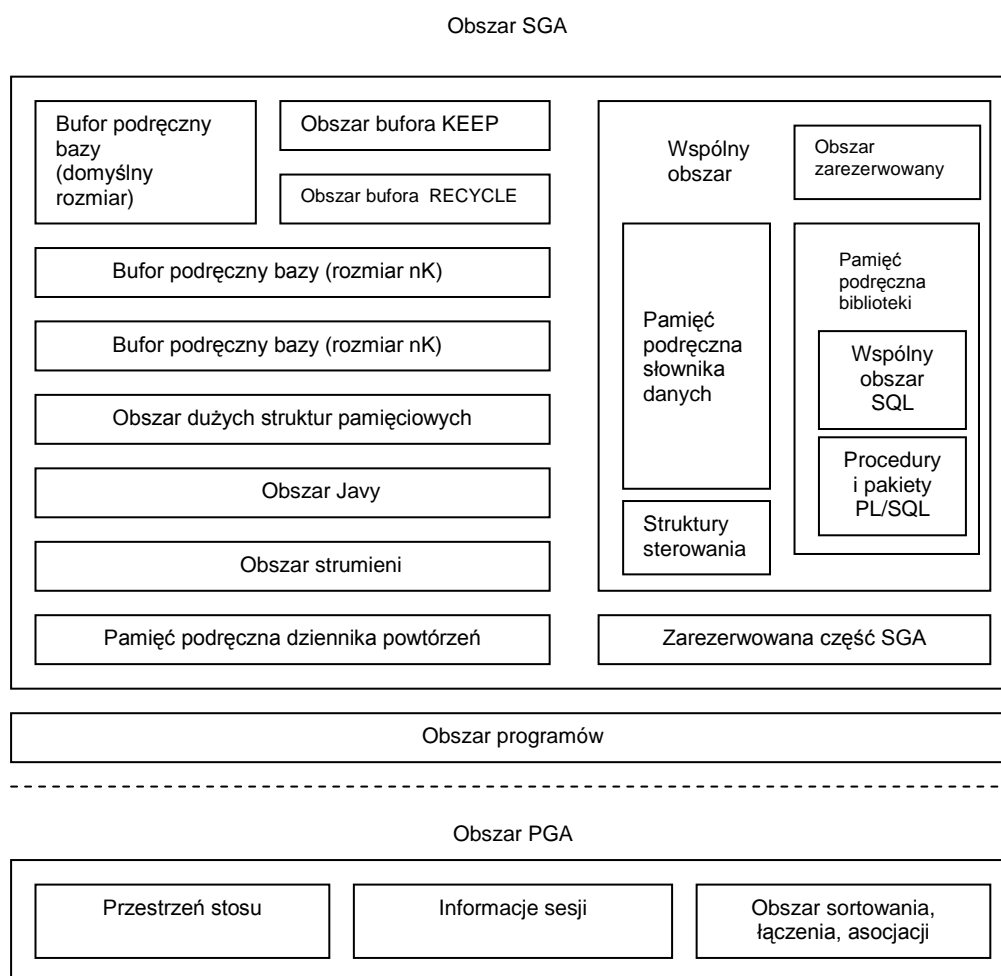
W systemie Oracle tworzony jest plik haseł (ang. *password file*). Znajduje się on w administracyjnej strukturze systemu Oracle lub w strukturze katalogów na dysku. Plik ten jest używany do uwierzytelniania administratorów systemu Oracle, tzn. nadawania uprawnień *SYSDBA* oraz *SYSOPER*. Jeśli plik haseł zostanie uszkodzony, można go odtworzyć przy pomocy narzędzia *orapwd*, uruchamianego z wiersza poleceń. Każdy administrator powinien posiadać własne konto z nadanymi wysokimi uprawnieniami. Należy także pamiętać, aby nie wykorzystywać powszechnie kont *SYS* oraz *SYSTEM*.

### 3.5. STRUKTURY PAMIĘCI SYSTEMU ORACLE

Baza danych Oracle używa pamięci do przechowywania informacji takich jak (Bryla, Loney, 2010; Oracle, 2011k):

- kod wykonywalny;
- informacje o sesjach;
- indywidualne procesy związane z bazą danych;
- informacje dzielone pomiędzy procesami bazy danych Oracle; np. blokady założone na obiektach bazy;
- informacje o instrukcjach SQL użytkownika, słownika danych;

- informacje w tymczasowej pamięci podręcznej, które w późniejszym etapie zostaną zapisane na dysku (transakcje, bloki danych z segmentów bazy danych).
- Pamięć systemu Oracle jest podzielona na trzy główne części: *PGA*, *SGA* oraz obszar kodu wykonywalnego. Struktura pamięci została przedstawiona na rysunku 3.4.



Rys. 3.4. Logiczne struktury pamięci systemu Oracle

Źródło: (Bryla, Loney, 2010)

Pierwszy obszar, *PGA* (ang. *Program Global Area*), globalny obszar programów, jest zaalokowany i dostępny wyłącznie dla jednego procesu (Bryla, Loney, 2010; Oracle, 2011k). Konfiguracja obszaru *PGA* zależy od rodzaju ustawień połączenia z bazą danych: jako serwer współdzielony (ang. *shared server*) lub serwer dedykowany (ang. *dedicated server*). Jeśli połączenie ustawione jest jako serwer współdzielony, wówczas dane związane z sesjami użytkownika przechowywane są w obszarze *SGA*. W drugiej konfiguracji, dane te znajdują się w pamięci *PGA*. W obszarze tym wykonywane są m.in. operacje sortowania, scalenia na podstawie mapy bitowej lub złączenia funkcją mieszającą (Bryla, Loney, 2010). W Oracle 11g administrator może ustawić rozmiar dla obszarów roboczych, a system będzie rozdzielał tę pamięć na poszczególne obszary.

Drugi obszar, *SGA* (ang. *System Global Area*), to zbiór struktur we wspólnej pamięci, wykorzystywanych przez instancję Oracle i wspólnie przez użytkowników instancji bazy danych (Bryla, Loney, 2010; Oracle, 2011k). Podczas uruchamiania instancji bazy danych alokowana jest pamięć dla obszaru *SGA*. Jej rozmiar jest uzależniony od parametrów zapisanych w plikach inicjujących lub zdefiniowanych ręcznie. Obszar *SGA* jest złożony, a rozmiary jego składników mogą być ustalane dynamicznie. Administrator może wpływać na całkowity rozmiar obszaru *SGA* (przypisując wartość do parametru *SGA\_MAX\_SIZE*), a także może ustawić opcję optymalizacji wydajności pamięci między obszarami *SGA* i *PGA* (parametr *MEMORY\_TARGET*). Pamięć w obszarze *SGA* alokowana jest w granulach (ang. *granules*), czyli jednostkach o rozmiarze 4 lub 16 MB (Bryla, Loney, 2010). Rozmiar granuła jest uzależniony od całkowitego rozmiaru obszaru *SGA*. Dla obszaru, który jest mniejszy niż 128 MB, granuła przyjmuje rozmiar 4 MB, w przeciwnym razie – 16 MB.

Ostatnią przestrzenią jest obszar kodu wykonywalnego (ang. *software code area*). Przechowuje on pliki wykonywalne Oracle, które działają jako część instancji Oracle (Bryla, Loney, 2010; Oracle, 2011k). Jest to obszar statyczny, a jego rozmiar jest zmieniany podczas instalowania nowej wersji oprogramowania. Obszar kodu wykonywalnego jest odseparowany od programów użytkowników. Jest on przeznaczony tylko do odczytu.

### 3.6. STROJENIE UŻYCIA PAMIĘCI SYSTEMU ORACLE

Administrator bazy danych powinien monitorować użycie pamięci przez system Oracle (Bryła, Loney, 2010). Poprzez wykonywanie zapytań SQL na odpowiednich widokach może on uzyskać informacje związane z zapytaniami wykonywanymi przez użytkowników czy też z obiektami, których bloki znajdują się w pamięci podręcznej. Administrator może także zarządzać pamięcią SGA oraz PGA przy pomocy narzędzia ASMM (ang. *Automatic Shared Memory Management*), które dostępne jest w systemie Oracle od wersji 10g. Możliwe jest także manualne ustawienie rozmiaru SGA przy pomocy instrukcji SQL.

Bufor pamięci podręcznej bloków danych oraz obszar wspólny zarządzane są przy pomocy algorytmu LRU (ang. *least recently used*), który zapewnia, że w przypadku przepełnienia obszaru, dane najstarsze zostają z niej usunięte i zapisane na dysku (Bryła, Loney, 2010). Obszar taki, który został poprawnie zdefiniowany przez administratora, przechowuje aktualne, najczęściej używane dane. W celu sprawdzenia danych rzadziej używanych, należy odczytać je z dysku.

System Oracle umożliwia odczytywanie danych związanych z zajętością pamięci przy pomocy odpowiednich zapytań SQL, bazujących na wybranych widokach. Przykładowo, widok V\$SQL może być użyty do pozyskania informacji związanych z liczbą operacji logicznego i fizycznego odczytu, wykonanych przez zapytania aktualnie znajdujących się w obszarze wspólnym oraz ile razy takie zapytanie zostało wykonane (Bryła, Loney, 2010). Na listingu 3.3 przedstawiono przykładowe polecenie, które odczytuje zapytania przechowywane w obszarze wspólnym (liczbę czytanych bloków pamięci, liczbę odczytów z dysku dla danego zapytania, liczbę jego wywołań, tekst zapytania – pierwszych 1000 znaków) (Bryła, Loney, 2010; Pelikant, 2009; Magee, 2003). Zbiór wynikowy zapytania został posortowany malejąco.

Listing 3.3. Przykładowe zapytanie na widok V\$SQL

```
SELECT Buffer_Gets, Disk_Reads, Executions, SQL_Text
FROM   V$SQL
WHERE  executions != 0 ORDER BY Disk_Reads DESC;
```

Źródło: (Bryła, Loney, 2010)

Listing 3.4. Przykładowe zapytanie na widoku V\$SESS\_IO

```
SELECT  SESS.Username,  
        SESS_IO.Block_Gets,  
        SESS_IO.Block_Changes  
FROM    V$SESSION SESS,  
        V$SESS_IO SESS_IO  
WHERE   sess_io.sid = sess.sid  
        AND SESS.Username != NULL;
```

Źródło: (Bryla, Loney, 2010; Session Statistic, 2011)

Korzystając z widoku V\$SESS\_IO można odczytać informacje, związane z danymi przechowywanymi w pamięci, o sesjach użytkowników bazy danych (Bryla, Loney, 2010). Zapytanie przedstawione na listingu 3.4 wykonywane jest na dwóch widokach V\$SESS\_IO oraz na V\$SESSION, z których można odczytać nazwę sesji użytkownika (*sid*), liczbę bloków pamięci niezbędnych dla danej sesji oraz liczbę bloków, które sesja zmienia (Bryla, Loney, 2010; Magee, 2003).

Administrator może także sprawdzić obiekty, których bloki znajdują się aktualnie w buforze pamięci podręcznej. W tym celu należy wykonać zapytanie na tabeli X\$BH w schemacie SYS (listing 3.5). Zaprezentowany przykład wyklucza obiekty użytkowników SYS oraz SYSTEM. Dane te zostały pogrupowane według nazwy obiektu oraz jego typu. Zapytanie zwraca nazwę obiektu, jego typ (np. tabela czy indeks) oraz zliczoną liczbę obiektów danego typu.

Listing 3.5. Przykładowe zapytanie na tabeli X\$BH

```
SELECT  Object_Name, Object_Type, COUNT(*) Num_Buff  
FROM    X$BH x, SYS.DBA_OBJECTS o  
WHERE   X.Obj = O.Object_Id  
        AND Owner NOT IN ('SYS', 'SYSTEM')  
GROUP BY Object_Name, Object_Type;
```

Źródło: (Bryla, Loney, 2010)

W pamięci podręcznej bloków danych wyróżnione są trzy obszary: *DEFAULT*, *KEEP*, *RECYCLE* oraz pamięć podręczna dla bloków o określonych rozmiarach (Bryla, Loney, 2010). Każdy z tych obszarów powinien mieć odpowiedni rozmiar, aby

pomieścić dane, które są w nich przetrzymywane. Odpowiednie ustawienie obszarów należących do SGA zapewnia zwiększenie wydajności pojedynczych zapytań bazy danych. Przydzielenia jednego z trzech obszarów do tabeli należy dokonać w momencie jej tworzenia lub edycji, odpowiednio za pomocą poleceń *CREATE TABLE* (listing 3.6) lub *ALTER TABLE* (listing 3.7).

*Listing 3.6. Polecenie tworzenia tabeli tab1*

```
CREATE TABLE tab1
  ( id      NUMBER,
    nazwa   VARCHAR(10) )
  STORAGE (buffer_pool keep);
```

*Listing 3.7. Polecenie modyfikujące tabelę tab1*

```
ALTER TABLE tab1
  STORAGE (buffer_pool default);
```

Administrator ma możliwość ręcznego zarządzania pamięcią SGA. Parametry modyfikujące oraz ich opis zostały przedstawione w tabeli 3.2.

*Tabela 3.2. Parametry rozmiarów pamięci podręcznych*

Parametr	Opis
SGA_TARGET	Parametr dynamiczny określający całkowity rozmiar komponentów obszaru SGA.
SGA_MAX_SIZE	Maksymalny przydzielony rozmiar dla SGA.
SHARED_POOL_SIZE	Rozmiar obszaru wspólnego.
DB_BLOCK_SIZE	Domyślny rozmiar bloku bazy danych.
DB_CACHE_SIZE	Rozmiar pamięci podręcznej (w bajtach).
DB_nK_CACHE_SIZE	Rozmiar chache dla bloku o n KB (DB_8K_CHACE_SIZE).

Źródło: (Bryla, Loney, 2010)

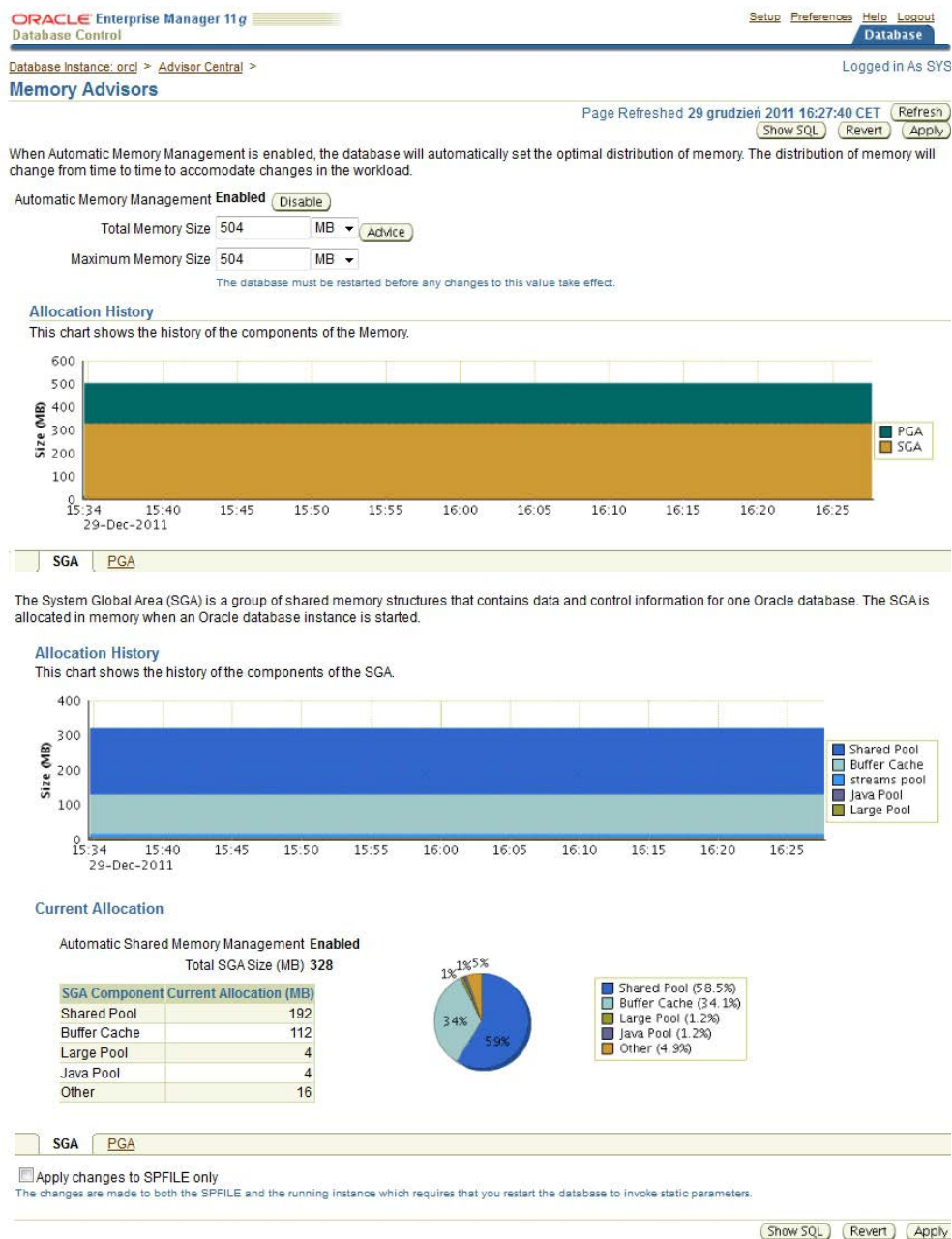
W celu automatycznego zarządzania pamięcią SGA oraz PGA można skorzystać z narzędzia ASMM. Uruchomienie tego narzędzia odbywa się dwójako: za pomocą polecenia SQL oraz za pomocą narzędzia OEM (ang. *Oracle Enterprise Manager*). Należy pamiętać, aby parametrowi *SGA\_TARGET* przypisać wartość różną od zera. Następnie parametrom, opisującym pamięć podręczną (*DB\_CACHE\_SIZE*, *SHARED\_POOL\_SIZE*, *JAVA\_POOL\_SIZE*, *LARGE\_POOL\_SIZE*), należy przypisać wartość zero. W przeciwnym razie nadane wartości będą traktowane jako minimalny rozmiar każdego obszaru. Po wprowadzeniu takich zmian wymagane jest zrestartowanie bazy danych, by zarządzanie tymi obszarami mogło odbywać się w sposób automatyczny. Monitorowanie rozmiarów pamięci podręcznych można przeprowadzić z użyciem widoku *V\$GASTAT*. W Oracle 11g można także ustawić wartość parametru *MEMORY\_TARGET*, która będzie oznaczała ilość pamięci dostępnej dla systemu Oracle, alokowanej pomiędzy obszary SGA i PGA.

W narzędziu OEM można także ustawić automatyczne zarządzanie pamięcią SGA i PGA. W celu włączenia tej opcji, należy w głównym oknie narzędzia przejść do zakładki *Server*, gdzie w sekcji *Database Configuration*, wybiera się opcję *Memory Advisors*. Pojawi się wówczas okno (rysunek 3.5), w którym opcję *Automatic Memory Management* należy ustawić na *Enabled*. W tym samym oknie wyświetlane są także inne opcje, związane z ustawieniem rozmiaru obszaru składającego się z sumy SGA oraz PGA oraz całkowitego rozmiaru przestrzeni (*Total Memory Size*), która będzie automatycznie zarządzana.

Kolejne elementy, przedstawione w zakładce SGA, informują o historii alokowanych komponentów pamięci SGA, których wielkość wyrażona jest w MB. Następna sekcja (*Current Allocation*) prezentuje informacje o bieżącej alokacji komponentów obszaru SGA zarówno w postaci tabeli, gdzie widnieją wartości alokacji w MB, jak i w formie wykresu kołowego, przedstawiającego tę samą alokację w postaci wartości wyrażonych w procentach.

Przedostatnia opcja stwarza możliwość zmiany całkowitego rozmiaru obszaru SGA, natomiast ostatnia – na zapisanie wprowadzonych zmian do pliku *SPFILE*.





Rys. 3.5. Formatka zarządzania i monitorowania pamięci obszaru SGA i PGA

Analogicznie do obszaru SGA, istnieją opcje związane z obszarem PGA, które można ustawić, przechodząc do zakładki PGA (rysunek 3.5). Są to informacje

dotyczące: maksymalnego i bieżącego rozmiaru tego obszaru, rozmiaru pamięci dostępnej dla wszystkich procesów serwera, związanych z daną instancją bazy (*Aggregate PGA Target*) oraz możliwości zapisania zmian w pliku *SPFILE*.

Opisane powyżej funkcjonalności narzędzia *OEM* pozwalają użytkownikowi w wygodny i szybki sposób zarządzać bazą danych. Jednak bardziej zaawansowane techniki administrowania bazą danych wymagają posługiwania się językiem SQL.

### 3.7. ZARZĄDZANIE OBIEKTAMI BAZY DANYCH ORACLE

Administrowanie obiektami bazy danych to proces, który w szczególności dotyczy zarządzania przestrzeniami tabel, segmentami oraz blokami danych.

Poza przestrzeniami tabel, które tworzone są podczas standardowej instalacji lub poleceniem *CREATE DATABASE*, administrator może tworzyć kolejne przestrzenie tabel, co pozwala na grupowanie danych w tych miejscach oraz oddzielenie ich od przestrzeni systemowych (*SYSTEM*). W tworzonych przestrzeniach tabel gromadzone są: dane użytkowników, indeksy, segmenty wycofań i segmenty tymczasowe (Banachowski, 2006).

Utworzenie nowej przestrzeni tabel wymaga podania jej typu. Może ona być przestrzenią trwałą, wycofania lub tymczasową. Przestrzenie tabel mogą zostać zdefiniowane poprzez wykonanie polecenia *CREATE TABLESPACE* lub przy pomocy narzędzia *OEM*. Zarządzanie przestrzeniami odbywa się przy użyciu poleceń *ALTER TABLESPACE* (modyfikacja) i *DROP TABLESPACE* (usuwanie) bądź też z poziomu *OEM*.

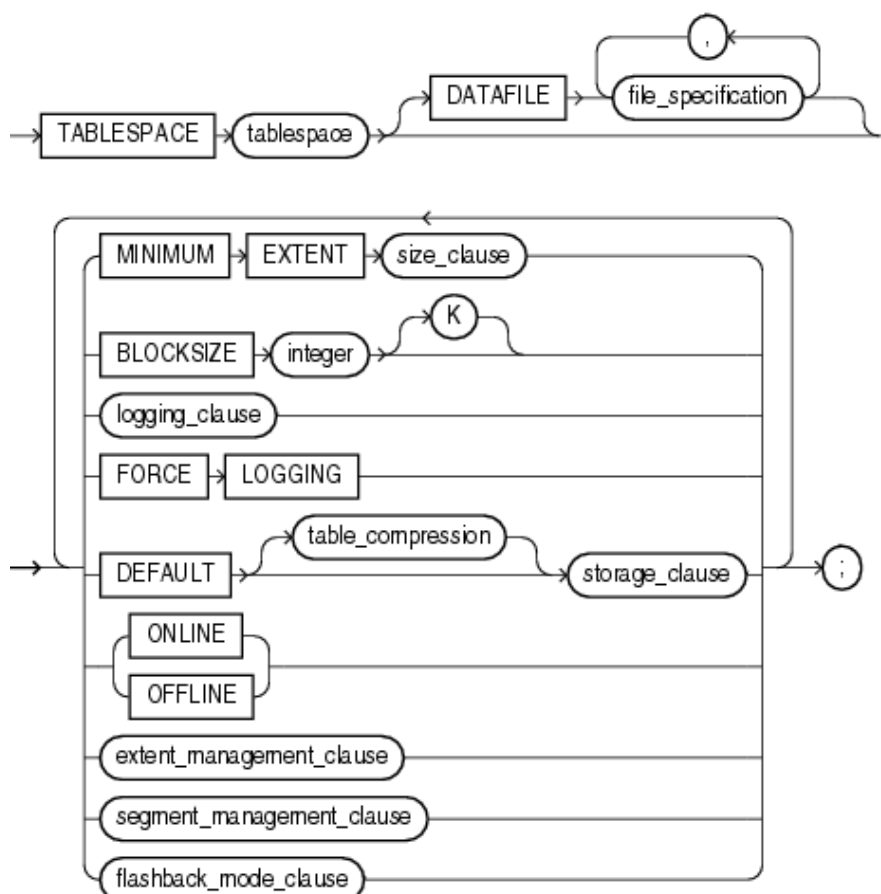
Polecenie *CREATE TABLESPACE* (Oracle, 2011e) pozwala na utworzenie nowej tabeli wielkoplikowej lub małych plików. W definicji niezbędne jest podanie nazwy przestrzeni oraz przynajmniej jednego pliku danych. Polecenie z dostępnymi opcjami zostało pokazane na rysunku 3.6. Po klauzuli *DATAFILE* podawana jest nazwa pliku i pełna ścieżka dostępu – plik ten nie może istnieć w systemie (Pelikant, 2009). Można dodać jego specyfikacje takie jak (Oracle, 2011e): rozmiar (*SIZE*), opcja generowania dziennika powtórzeń (*LOGGING*), możliwość jego kompresji (*table\_compression*), opcje przechowywania danych (*storage*), sposoby zarządzania przestrzenią tabel

(*extent management*), zarządzanie segmentami danych (*segment management*) oraz włączenie lub wyłączenie trybu flashback (*flashback mode*).

Przykład tworzenia przestrzeni tabel został przedstawiony na listingu 3.8. Przestrzeń ta początkowo ma rozmiar 500 MB, ale dozwolone jest jej zwiększenie.

Listing 3.8. Polecenie tworzenia trwałej przestrzeni tabel

```
CREATE TABLESPACE tabs_01
  DATAFILE 'D:\app\oradata\orcl\tabs_01.dbf'
  SIZE 500 M REUSE AUTOEXTEND ON;
```



Rys. 3.6. Polecenie CREATE TABLESPACE i jego opcje

Źródło: (Oracle, 2011e)

W administrowaniu bazami danych wyróżnia się dwie metody zarządzania jej obiektami: zarządzanie lokalne (ang. *Management Local*) oraz zarządzane słownikiem danych (Banachowski, 2006). Pierwsza metoda jest domyślna w systemie Oracle. Przy tworzeniu przestrzeni tabel należy zdecydować i podać rodzaj zarządzania. Listing 3.9 przedstawia tworzenie przestrzeni tabel zarządzanych lokalnie. Dla przestrzeni tabel zarządzanych w ten sposób istnieją dwa parametry konfiguracyjne (Banachowski, 2006):

- *UNIFORM* – obiekty bazy danych posiadają taki sam rozmiar, określony za pomocą parametru *SIZE* lub mającą rozmiar domyślny 1 MB;
- *AUTOALLOCATE* – rozmiar obiektów ustawiany jest przez system.

Listing 3.9. Polecenie tworzenia trwałej przestrzeni tabel zarządzanej lokalnie

```
CREATE TABLESPACE tbsusr
  DATAFILE 'D:\app\oradata\orcl\user_01.dbf'
  SIZE 250M
  EXTENT MANAGEMENT LOCAL UNIFORM SIZE 128K;
```

W przestrzeniach tabel zarządzanych lokalnie można definiować sposób użycia wolnego miejsca w ich blokach. Administrator ma tutaj do wyboru dwie metody (Banachowski, 2006):

- *MANUAL* – używanie listy wolnych bloków (opcja domyślna);
- *AUTO* – używanie mapy bitowej, wskazującej status bloku (np. podczas wykonywania instrukcji *INSERT*).

Przykład tworzenia tabeli z możliwością zarządzania blokami został przedstawiony na listingu 3.10.

Listing 3.10. Polecenie tworzenia trwałej przestrzeni tabel z opcją zarządzania blokami

```
CREATE TABLESPACE tbsusr
  DATAFILE 'D:\app\oradata\orcl\user_01.dbf'
  SIZE 250M
  EXTENT MANAGEMENT LOCAL AUTOALLOCATE
  SEGMENT SPACE MANAGEMENT AUTO;
```

Dla przestrzeni tabel zarządzanych słownikowo każdemu segmentowi można przypisać klauzulę *STORAGE*, określającą sposób użycia obszarów. Przykład tworzenia tak zarządzanej przestrzeni tabel ilustruje listing 3.11. Klauzula *DEFAULT STORAGE* posiada następujące dodatkowe parametry (Banachowski, 2006):

- *INITIAL* – rozmiar pierwszego alokowanego obszaru (domyślnie 5 bloków);
- *NEXT* – rozmiar drugiego alokowanego obszaru (domyślnie 5 bloków);
- *PCTINCREASE* – procentowy wzrost rozmiaru kolejnego obszaru w stosunku do drugiego;
- *MINEXTENTS* – minimalna liczba alokowanych obszarów podczas tworzenia segmentu (domyślnie 1, a dla segmentu wycofań – 2);
- *MAXEXTENTS* – maksymalna liczba alokowanych obszarów dla obiektu.

*Listing 3.11. Polecenie tworzenia trwałej przestrzeni tabel z klauzulą DEFAULT STORAGE*

```
CREATE TABLESPACE tbsusr
  DATAFILE 'D:\app\oradata\orcl\user_01.dbf'
  SIZE 250M
  EXTENT MANAGEMENT DISTIONARY
  DEFAULT STORAGE (INITIAL 2M NEXT 1M);
```

Przestrzenie tabel wycofań w istniejącej bazie danych tworzone są poleceniem *CREATE UNDO TABLESPACE* (Banachowski, 2006; Bryła, Loney, 2010; Oracle, 2011e). Zawierają one tylko segmenty wycofań, dlatego jedyną opcją jest klauzula *DATAFILE* z parametrami (listing 3.12). Dla tego rodzaju przestrzeni, segmenty zarządzane są lokalnie. W celu manualnego zarządzania segmentami, należy ustawić parametr *UNDO\_MANAGEMENT = MANUAL* (domyślnie wartość jest równa *AUTO*) oraz podać klauzulę *UNIFORM SIZE*.

*Listing 3.12. Polecenie tworzenia przestrzeni tabel wycofania*

```
CREATE UNDO TABLESPACE undo_02
  DATAFILE 'D:\app\oradata\orcl\undo_02.dbf'
  SIZE 100M;
```

Tworzenie tymczasowych przestrzeni tabel wykonywane jest przy pomocy polecenia *CREATE TEMPORARY TABLESPACE* (Banachowski, 2006; Oracle, 2011e). Są one zarządzane lokalnie oraz słownikowo. Przykład tworzenia przestrzeni zarządzanej lokalnie został przedstawiony na listingu 3.13. Takie przestrzenie nie mogą zawierać trwałych obiektów, ani posiadać statusu read-only (Banachowski, 2006).

*Listing 3.13. Polecenie tworzenia tymczasowej przestrzeni tabel*

```
CREATE TEMPORARY TABLESPACE temp_02
    tempfile 'D:\app\oradata\orcl\temp_02.dbf'
    size 40M extent management local
    uniform size 10M;
```

Definiowanie tymczasowych przestrzeni tabel zarządzanych słownikowo ilustruje listing 3.14.

*Listing 3.14. Polecenie tworzenia tymczasowej przestrzeni tabel zarządzanych słownikowo*

```
CREATE TABLESPACE temp_03
    datafile 'D:\app\oradata\orcl\temp_03.dbf'
    size 300M
    default storage (initial 4M next 2M
    pctincrease 0 maxextents unlimited)
    temporary;
```

Operacje modyfikacji przestrzeni tabel odbywają się przy pomocy polecenia *ALTER TABLESPACE* (Banachowski, 2006; Oracle, 2011e). W ten sposób można zmienić status utworzonej przestrzeni na domyślny (listing 3.15) lub na read-only (listing 3.16).

*Listing 3.15. Modyfikacja przestrzeni tabel na domyślną*

```
ALTER DATABASE DEFAULT TEMPORARY TABLESPACE temp_02;
```

*Listing 3.16. Modyfikacja przestrzeni tabel na tylko do odczytu.*

```
ALTER DATABASE DEFAULT TEMPORARY TABLESPACE temp_02;
```

W trakcie tworzenia przestrzeni tabel istnieje możliwość ustawienia jednej z dwóch opcji zarządzania blokami danych. Mogą one być zarządzane albo lokalnie albo manualnie. Pierwsza opcja dostępna jest tylko dla przestrzeni tabel zarządzanych lokalnie, natomiast druga zarówno dla przestrzeni zarządzanych lokalnie jak i słownikowo. W opcji automatycznego zarządzania blokami wykorzystywane są mapy bitowe, które określają status bloku w segmencie na podstawie wolnego miejsca (Banachowski, 2006). Mapa ta znajduje się w ostatnim zbiorze bloków *BMB* (ang. *bitmapped blocks*). Podczas wstawiania nowego wiersza mapa jest analizowana i szukany jest blok o odpowiedniej ilości miejsca. W przypadku zmiany miejsca w blokach, informacja ta jest także aktualizowana w mapie bitowej.

Automatyczne zarządzanie blokami danych zostało pokazane na listingu 3.17. Dotyczy ono zarządzania wszystkimi segmentami przestrzeni tabel, indeksów, tabel połączonych z indeksami i dużych obiektów LOB (Banachowski, 2006).

*Listing 3.17. Tworzenie przestrzeni tabel z automatycznym zarządzaniem blokiem danych*

```
CREATE TABLESPACE data_01
  DATAFILE 'D:\app\oradata\orcl\data_01.dbf'
  SIZE 10M
  EXTENT MANAGEMENT LOCAL UNIFORM SIZE 32K
  SEGMENT SPACE MANAGEMENT AUTO;
```

Ustawienie ręcznego zarządzania blokami zostało przedstawione na listingu 3.18.

*Listing 3.18. Tworzenie przestrzeni tabel z automatycznym zarządzaniem blokiem danych*

```
CREATE TABLESPACE data_01
  DATAFILE 'D:\app\oradata\orcl\data_01.dbf'
  SIZE 10M
  EXTENT MANAGEMENT LOCAL UNIFORM SIZE 32K
  SEGMENT SPACE MANAGEMENT MANUAL;
```

Przestrzeń tabel można utworzyć także za pomocą narzędzia *OEM* (rys. 3.7). W tym celu należy podać nową nazwę przestrzeni tabel, zaznaczyć jej rodzaj (*Permanent*, *Temporary* lub *Undo*), określić jej status (*Read Write*, *Read Only* lub *Offline*), a następnie wskazać opcję dodania pliku danych (*ADD*).

The screenshot shows the 'Create Tablespace' dialog in Oracle Enterprise Manager 11g. The 'General' tab is selected, and the 'Storage' tab is also visible. The 'Name' field is set to 'Testowa'. Under 'Extent Management', 'Locally Managed' is selected. Under 'Type', 'Permanent' is selected. Under 'Status', 'Read Write' is selected. The 'Datafiles' section shows a table with columns 'Select Name', 'Directory', and 'Size (MB)'. The 'Storage' tab is also visible, showing options for 'Use bigfile tablespace' and 'Tablespace can have only one datafile with no practical size limit'.

Rys. 3.7. Tworzenie nowej przestrzeni tabel

W nowym oknie (rys. 3.8) podaje się nazwę pliku, który będzie należał do tworzonej przestrzeni, ścieżkę dostępu do niego, początkowy rozmiar oraz ewentualnie opcje w sekcji *STORAGE*. Opcje te określają możliwość automatycznego zwiększania rozmiaru pliku w przypadku zapelnienia dotychczasowej przestrzeni (*Automatically extend datafile when full*), wielkość przyrostu pliku (*Increment*) oraz maksymalny jego rozmiar (*Maximum File Size*).

Bezpośrednio po utworzeniu przestrzeni tabel można już z niej korzystać, bez konieczności ponownego uruchamiania bazy danych.



ORACLE Enterprise Manager 11g  
Database Control

Setup Preferences Help Logout  
Database

Database Instance: orcl > Tablespaces > Logged in As SYS

**Add Datafile**

File Name: testowa\_01.dbf

File Directory: /opt/oracle/oradata/orcl/

Tablespace: TESTOWA

File Size: 200 MB

☐ Reuse Existing File

**Storage**

☒ Automatically extend datafile when full (AUTOEXTEND)

Increment: 250 KB

Maximum File Size: ☒ Unlimited ☐ Value: MB

TIP Changes made on this page will NOT take effect until you click "OK" button on the Tablespace page.

Cancel Continue

Rys. 3.8. Tworzenie nowego pliku danych podczas definiowania przestrzeni tabel

W trakcie eksploatacji bazy danych, może zaistnieć potrzeba zmiany rozmiaru przestrzeni tabel, dodania, usunięcia lub przeniesienia pliku do/z przestrzeni. Procedury postępowania w takich sytuacjach mogą różnić się w zależności, czy modyfikowane są pliki wielkoplikowe czy też małe pliki (Bryla, Loney, 2010).

Zmiana rozmiaru istniejących plików może zostać wykonana poleceniem *ALTER DATABASE* z podaniem konkretnego pliku oraz nowego rozmiaru. Jeśli nowy rozmiar jest zbyt mały lub zbyt duży, zostanie wyświetlony odpowiedni komunikat. Przykład polecenia zmiany rozmiaru pliku *data\_1.dbf* na 25MB został pokazany na listingu 3.19. W poleceniu tym można także użyć klauzul *AUTOEXTEND*, *NEXT* oraz *MAXSIZE*, które umożliwią odpowiednio automatyczne zwiększanie rozmiaru pliku, określenie przyrostu pliku oraz maksymalny jego rozmiar.

Listing 3.19. Zmiana rozmiaru pliku danych

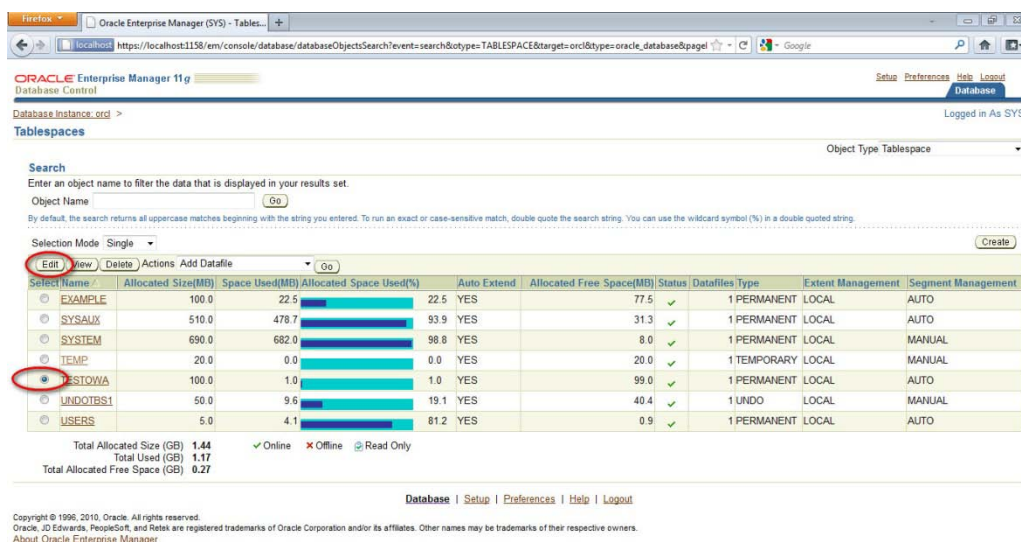
```
ALTER DATABASE
  DATAFILE 'D:\app\oradata\orcl\data_1.dbf'
  RESIZE 25M;
```

Dodanie nowego pliku do istniejącej przestrzeni tabel ilustruje listing 3.20. Należy użyć klauzuli *ADD DATAFILE*, po której podaje się nazwę pliku wraz z pełną ścieżką dostępu do niego.

Listing 3.20. Dodanie pliku do przestrzeni tabel

```
ALTER TABLESPACE data_01
ADD DATAFILE 'D:\app\oradata\orcl\data_2.dbf'
RESIZE 35M
AUTOEXTEND ON
MAXSIZE 500M;
```

Modyfikowanie rozmiaru plików danych można także przeprowadzić przy użyciu narzędzia *OEM*. W zakładce *Server*, w sekcji *Storage*, należy przejść do danych przestrzeni tabel (*Tablespaces*). Wyświetla się wówczas lista wszystkich istniejących przestrzeni tabel (rys. 3.9). Zaznaczając wybraną przestrzeń oraz wybierając opcję edycji, przechodzi się do jej parametrów (rys. 3.10). Opcji sposobu zarządzania oraz typu przestrzeni tabel nie można już modyfikować. Można natomiast zmienić status przestrzeni, wybierając jedną z opcji: odczyt-zapis, tylko do odczytu lub offline.



Select	Name	Allocated Size(MB)	Space Used(MB)	Allocated Space Used(%)	Auto Extend	Allocated Free Space(MB)	Status	Datafiles Type	Extent Management	Segment Management
<input type="radio"/>	EXAMPLE	100.0	22.5	22.5	YES	77.5	✓	1 PERMANENT	LOCAL	AUTO
<input type="radio"/>	SYSAUX	510.0	478.7	93.9	YES	31.3	✓	1 PERMANENT	LOCAL	AUTO
<input type="radio"/>	SYSTEM	690.0	682.0	98.8	YES	8.0	✓	1 PERMANENT	LOCAL	MANUAL
<input type="radio"/>	TEMP	20.0	0.0	0.0	YES	20.0	✓	1 TEMPORARY	LOCAL	MANUAL
<input checked="" type="radio"/>	TESTOWA	100.0	1.0	1.0	YES	99.0	✓	1 PERMANENT	LOCAL	AUTO
<input type="radio"/>	UNDOTBS1	50.0	9.6	19.1	YES	40.4	✓	1 UNDO	LOCAL	MANUAL
<input type="radio"/>	USERS	5.0	4.1	81.2	YES	0.9	✓	1 PERMANENT	LOCAL	AUTO

Total Allocated Size (GB) 1.44  
Total Used (GB) 1.17  
Total Allocated Free Space (GB) 0.27

Online Offline Read Only

Database | Setup | Preferences | Help | Logout

Copyright © 1996, 2010, Oracle. All rights reserved.  
Oracle, JD Edwards, PeopleSoft, and Retek are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.  
About Oracle Enterprise Manager

Rys. 3.9. Edytowanie przestrzeni tabel TESTOWA

ORACLE Enterprise Manager 11g Database Control

Database Instance: orcl > Tablespaces > Edit Tablespace: TESTOWA

Actions: Add Datafile Go Show SQL Revert Apply

General Storage Thresholds

Name: TESTOWA

Bigfile tablespace No

Extent Management: Locally Managed (selected), Dictionary Managed

Type: Permanent (selected), Set as default permanent tablespace, Encryption, Temporary, Set as default temporary tablespace, Undo

Status: Read Write (selected), Read Only, Offline, Offline Mode: Normal

Datafiles

Name	Directory	Size (MB)	Used (MB)
testowa_01.dbf	/opt/oracle/oradata/orcl/	200,00	1,00

General Storage Thresholds

Actions: Add Datafile Go Show SQL Revert Apply

Rys. 3.10. Parametry edycji przestrzeni tabel TESTOWA

W sekcji *Datafiles* (rys. 3.10) wyświetlane są nazwy plików, które należą do danej przestrzeni tabel. W tym miejscu można zainicjować proces edycji ich parametrów, których wartości będą definiowane w nowym oknie (rys. 3.11).

ORACLE Enterprise Manager 11g Database Control

Database Instance: orcl > Tablespaces > Edit Tablespace: TESTOWA: Edit Datafile

Cancel Continue

File Name: testowa\_01.dbf

File Directory: /opt/oracle/oradata/orcl/

Tablespace: TESTOWA

Status: Online (selected), Offline

File Size: 50 MB

Storage

☒ Automatically extend datafile when full (AUTOEXTEND)

Increment: 256 KB

Maximum File Size: Unlimited (selected), Value: 32767 MB

TIP: Changes made on this page will NOT take effect until you click "Apply" button on the Tablespace page.

Cancel Continue

Rys. 3.11. Parametry edycji pliku przestrzeni tabel TESTOWA

W przykładzie rozmiar pliku został zmieniony na 50 MB. Po zakończeniu edycji parametrów pliku, wyświetli się okno z nową charakterystyką przestrzeni tabel (rys. 3.12), w którym należy zatwierdzić dokonane zmiany. Można także wyświetlić kod SQL, obrazujący polecenie niezbędne do wykonania w celu utrwalenia wprowadzonych zmian.

ORACLE Enterprise Manager 11g  
Database Control

Database Instance: orcl > Tablespaces >  
Edit Tablespace: TESTOWA

Actions: Add Datafile Go Show SQL Revert Apply

**Information**  
Modification to the datafile will not take effect until you click "Apply" button.

General Storage **Thresholds**

Name: TESTOWA  
Bigfile tablespace: No

**Extent Management**  
☒ Locally Managed  
☐ Dictionary Managed

**Type**  
☒ Permanent  
☐ Set as default permanent tablespace  
☐ Encryption [Encryption Options](#)  
☐ Temporary  
☐ Set as default temporary tablespace  
☐ Undo

**Status**  
☒ Read Write  
☐ Read Only  
☐ Offline  
 Offline Mode: Normal

**Datafiles**

Select	Name	Directory	Size (MB)	Used (MB)
<input checked="" type="radio"/>	testowa_01.dbf	/opt/oracle/oradata/orcl/	50.00	1.00

General Storage **Thresholds**

Actions: Add Datafile Go Show SQL Revert **Apply**

Rys. 3.12. Zatwierdzenie zmian edycji pliku przestrzeni TESTOWA

Do istniejącej przestrzeni tabel można także dodać kolejne pliki. W narzędziu *Enterprise Manager* zadanie to wykonuje się w oknie analogicznym jak przy tworzeniu nowej przestrzeni tabel (rys. 3.8), w którym należy wprowadzić dane nowego pliku (nazwę, początkowy rozmiar oraz parametry do rozszerzenia). Po zatwierdzeniu zmian zostanie wyświetlone okno z parametrami przestrzeni i dwoma jej plikami (rys. 3.13).

ORACLE Enterprise Manager 11g  
Database Control

Database Instance: orcl > Tablespaces >  
Edit Tablespace: TESTOWA

Actions: Add Datafile Go Show SQL Revert Apply

**Update Message**  
Tablespace TESTOWA has been modified successfully

General Storage Thresholds

Name: TESTOWA  
Bigfile tablespace: No

**Extent Management**  
☒ Locally Managed  
☐ Dictionary Managed

**Type**  
☒ Permanent  
☐ Set as default permanent tablespace  
☐ Encryption (Encryption Options)  
☐ Temporary  
☐ Set as default temporary tablespace  
☐ Undo

**Status**  
☒ Read Write  
☐ Read Only  
☐ Offline  
 Offline Mode: Normal

**Datafiles**

Select	Name	Directory	Size (MB)	Used (MB)
<input checked="" type="radio"/>	testowa_02.dbf	/opt/oracle/oradata/orcl/	20,00	-48,00
<input type="radio"/>	testowa_01.dbf	/opt/oracle/oradata/orcl/	50,00	-18,00

General Storage Thresholds

Actions: Add Datafile Go Show SQL Revert Apply

Rys. 3.13. Dane przestrzeni TESTOWA po dodaniu drugiego pliku testowa\_02.dbf

Zarządzanie przestrzenią tabel obejmuje także usunięcie pliku należącego do niej. Istnieją jednak pewne ograniczenia wykonania tej operacji, wśród których należy wymienić (Oracle, 2011a):

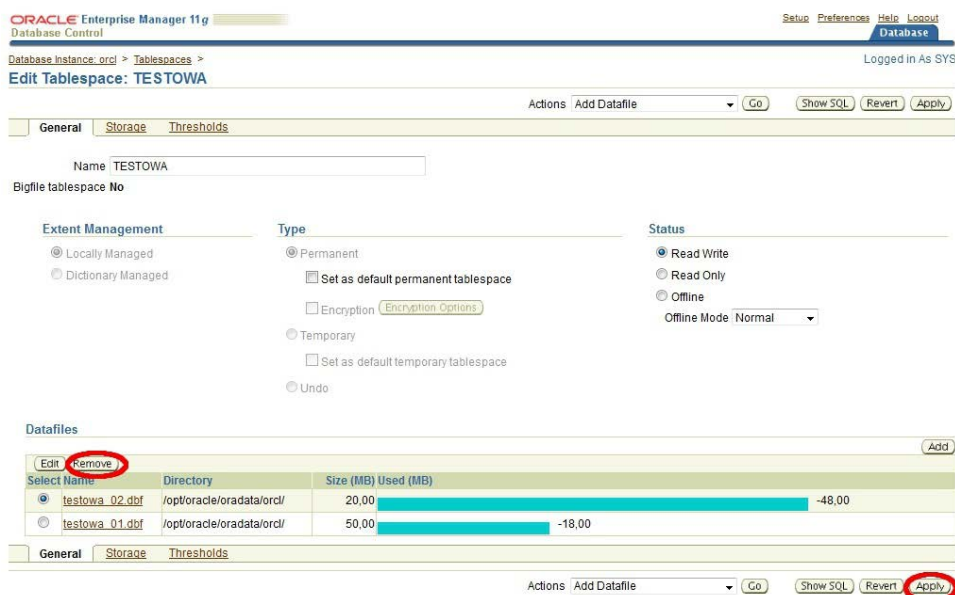
- plik danych musi być pusty;
- plik danych nie może być pierwszym plikiem utworzonym w przestrzeni tabel;
- plik danych nie może znajdować się w przestrzeni tabel tylko do odczytu (*read-only*), która zmieniła status z zarządzanej słownikowo na zarządzaną lokalnie;
- przestrzeń tabel nie może posiadać statusu offline;

Usunięcia pliku można dokonać przy użyciu polecenia *ALTER TABLESPACE* (Oracle, 2011a). Sposób wykonania tej operacji prezentuje listing 3.21. Plik przestrzeni tabel może być usunięty również za pomocą narzędzia *OEM*. Należy wówczas wskazać plik przestrzeni danych oraz wybrać opcję *Remove* (rys. 3.14). Następnie konieczne jest potwierdzenie zamiaru usunięcia pliku (rys. 3.15).

Listing 3.21. Usunięcie pliku danych z przestrzeni (Oracle, 2011f)

```
ALTER TABLESPACE data_01
  DROP DATAFILE 'D:\app\oradata\orcl\testowa_02.dbf';
```

Po przeprowadzonych operacjach administrator jest kierowany do strony, na której wyświetlane są szczegóły modyfikowanej przestrzeni tabel.



Rys. 3.14. Usunięcie pliku testowa\_02.dbf



Rys. 3.15. Potwierdzenie usunięcia pliku testowa\_02.dbf

W przypadku wieloplikowej przestrzeni danych, jej tworzenie odbywa się analogicznie jak w przypadku przestrzeni małych plików (listing 3.22). Zmiana

rozmiaru pliku danych takiej przestrzeni jest znacznie prostsza, ponieważ przestrzeń ta składa się tylko z jednego pliku (listing 3.23).

*Listing 3.22. Tworzenie wielkoplikowej przestrzeni tabel*

```
CREATE BIGFILE TABLESPACE tbs_bigdata_01
  DATAFILE 'D:\app\oradata\orcl\bigdatafile 01.dbf'
  SIZE 750M
  AUTOEXTEND ON NEXT 1024K
  MAXSIZE UNLIMITED
  EXTEND MANAGEMENT AUTO
  SEGMENT SPACE MANAGEMENT AUTO;
```

*Listing 3.23. Zmiana rozmiaru pliku danych przestrzeni wielkoplikowej*

```
ALTER TABLESPACE tbs_bigdata_01 RESIZE 2G;
```

W trakcie działania bazy danych może zaistnieć konieczność zmiany lokalizacji pliku danych, należącego do przestrzeni tabel. W systemie Oracle 11g występują trzy metody przenoszenia plików (Bryła, Loney, 2010):

- za pomocą polecenia *ALTER DATABASE*;
- za pomocą polecenia *ALTER TABLESPACE*;
- za pomocą narzędzia *Enterprise Manager*.

Spośród w/w technik, tylko dwie pierwsze zapewniają rozbudowane możliwości wykonania tego zadania, poprzez zestawy dodatkowych klauzul. Narzędzie *Enterprise Manager* nie oferuje w tym obszarze tak szerokiej funkcjonalności.

Przeniesienie pliku danych przestrzeni tabel, przy pomocy polecenia *ALTER DATABASE*, wymaga realizacji następujących czynności (Bryła, Loney, 2010):

1. Zalogowanie się do bazy jako użytkownik z uprawnieniami *SYSDBA* (listing 3.24), sprawdzenie nazwy pliku (listing 3.25) i wyłączenie instancji bazy (listing 3.26).

*Listing 3.24. Logowanie do bazy danych użytkownika SYSDBA*

```
sqlplus / as sysdba
```

*Listing 3.25. Wyświetlenie plików danych przestrzeni TESTOWA*

```
SELECT d.name  
FROM V$datafile d  
JOIN v$tablespace t USING(ts#)  
WHERE t.name = 'TESTOWA' ;
```

*Listing 3.26. Wyłączenie instancji bazy danych*

```
SHUTDOWN IMMEDIATE;
```

2. Przeniesienie pliku danych do nowej lokalizacji (w systemie operacyjnym).
3. Otworzenie bazy danych w trybie *MOUNT* (listing 3.27).

*Listing 3.27. Włączenie bazy danych w trybie MOUNT*

```
STARTUP MOUNT;
```

4. Zmiana odwołania do pliku danych, za pomocą polecenia *ALTER DATABASE*, na nową lokalizację (listing 3.28 obrazuje zmianę lokalizacji w pliku kontrolnym).

*Listing 3.28. Zmiana lokalizacji pliku*

```
ALTER DATABASE RENAME FILE  
'D:\app\oradata\orcl\testowa_02.dbf' TO  
'D:\app2\oradata\orcl\testowa_02.dbf';
```

5. Otwarcie bazy danych (listing 3.29).

*Listing 3.29. Otworzenie bazy danych*

```
ALTER DATABASE OPEN;
```

6. Wykonanie przyrostowej lub pełnej kopii zapasowej bazy danych, zawierającej pliki kontrolne, np. za pomocą polecenia przedstawionego na listingu 3.30.



Listing 3.30. Utworzenie kopii zapasowej bazy danych

```
ALTER DATABASE BACKUP CONTROLFILES TO TRACE;
```

Jeśli plik danych nie należy do przestrzeni tabel *SYSTEM* lub *SYSAUX*, aktywnej przestrzeni tabel wycofania i tymczasowej przestrzeni tabel, wówczas można go przenieść poleceniem *ALTER TABLESPACE*. Podczas tej operacji nie trzeba wyłączać bazy danych, dlatego też ta metoda jest wygodniejsza w użyciu. Jej zastosowanie obejmuje następujące czynności (Bryła, Loney, 2010):

1. Zmiana trybu przestrzeni tabel na *offline* (listing 3.31) – do wykonania tego polecenia należy posiadać uprawnienie systemowe *ALTER TABLESPACE*.

Listing 3.31. Zmiana statusu przestrzeni TESTOWA na offline

```
ALTER TABLESPACE testowa OFFLINE;
```

2. Przeniesienie pliku danych przy użyciu poleceń systemu operacyjnego.
3. Zmiana odwołania do pliku danych za pomocą polecenia *ALTER TABLESPACE* na nową lokalizację (listing 3.32).

Listing 3.32. Zmiana lokalizacji pliku

```
ALTER TABLESPACE TESTOWA RENAME DATAFILE  
'D:\app\oradata\orcl\testowa_02.dbf' TO  
'D:\app2\oradata\orcl\testowa_02.dbf';
```

4. Zmiana trybu przestrzeni na *online* (listing 3.33).

Listing 3.33. Zmiana statusu przestrzeni TESTOWA na online

```
ALTER TABLESPACE testowa ONLINE;
```

### 3.8. MECHANIZMY BEZPIECZEŃSTWA BAZY DANYCH ORACLE

Jednym z głównych zadań administratora jest zapewnienie bezpieczeństwa bazy, które przede wszystkim polega na ochronie danych przed niepożądanym dostępem osób trzecich. W związku z tym tak ważnym aspektem jego pracy jest szczegółowe poznanie mechanizmów oraz narzędzi dostępnych w systemie Oracle, dzięki którym będzie możliwa właściwa ochrona bazy i danych przed szkodliwą ingerencją z zewnątrz. Zapewnienie bezpieczeństwa danych dotyczy trzech aspektów administrowania bazą danych (Bryła, Loney, 2010):

- uwierzytelniania (ang. *authentication*),
- autoryzacji (ang. *authorization*),
- monitorowania (ang. *auditing*).

Uwierzytelnienie polega na identyfikowaniu użytkowników, którzy posiadają dostęp do bazy danych. Autoryzacja związana jest z udostępnieniem obiektów bazy danych, po uwierzytelnieniu użytkownika. Monitorowanie polega na sprawdzaniu i rejestrowaniu wszelakich prób logowania, zarówno tych skutecznych jak i kończących się niepowodzeniem, a także prób uzyskaniu dostępu do obiektu oraz wykonywania na nim operacji.

Pierwszym etapem zapewnienia bezpieczeństwa bazy danych jest tworzenie jej użytkowników oraz przydzielenie im odpowiednich uprawnień, które umożliwią dostęp do określonych zasobów. Każdy użytkownik musi posiadać unikalną nazwę i hasło do uwierzytelnienia. Nie musi on być właścicielem obiektów bazy danych, gdyż może mieć dostęp do obiektów innych użytkowników. Podczas tworzenia pierwszego obiektu, staje się jego właścicielem, a obiekt wchodzi w skład jego schematu (ang. *schema*).

Nowy użytkownik może być utworzony przez administratora lub innego użytkownika, który ma nadane uprawnienie systemowe *CREATE USER*. W trakcie tej operacji definiowane są pewne parametry, takie jak: hasło, domyślna przestrzeń tabel użytkownika, ewentualna blokada konta i inne (Oracle, 2011c). Tworzenie nowego użytkownika jest wykonywane przy pomocy polecenia *CREATE USER*, którego opcje zostały przedstawione w tabeli 3.3.

Tabela 3.3. Opcje polecenia CREATE USER

Opcja	Sposób użycia
<i>nazwa użytkownika</i>	Nazwa schematu i użytkownika; nazwa może zawierać do 30 znaków, nie powinna zawierać słów zarezerwowanych.
IDENTIFIED BY { <i>By hasło</i>   EXTERNALLY   GLOBALLY AS ' <i>nazwa zewnętrzna</i> ' }	Sposób uwierzytelnienia użytkownika: przez bazę danych na podstawie hasła, przez system operacyjny (lokalny lub zdalny) lub przez usługę (np. Oracle Internet Directory).
DEFAULT TABLESPACE <i>nazwa_przestrzeni_tabel</i>	Przestrzeń, w której będą tworzone obiekty stałe użytkownika, jeśli nie zostanie jawnie podana inna lokalizacji w trakcie ich tworzenia.
TEMPORARY TABLESPACE <i>nazwa_przestrzeni_tabel</i>	Przestrzeń, w której będą tworzone tymczasowe obiekty w trakcie operacji sortowania, tworzenia indeksów itp.
QUOTA { <i>rozmiar</i>   UNLIMITED ON <i>nazwa_przestrzeni_tabel</i> }	Wielkość przestrzeni na dysku (w kilobajtach – K lub megabajtach – M) dla obiektów użytkownika dla wskazanej przestrzeni tabel.
PROFILE <i>profil</i>	Nazwa profilu przypisana do użytkownika; jeśli nie zostanie wybrany żaden profil, automatycznie zostanie przydzielony profil <i>DEFAULT</i> .
PASSWORD EXPIRE	Wymuszenie zmiany hasła przy pierwszym logowaniu.
ACCOUNT {LOCK   UNLOCK}	Opcja zablokowania konta lub jego odblokowania; domyślnie konto jest odblokowane.

Źródło: (Bryla, Loney, 2010)

Przykład utworzenia nowego użytkownika za pomocą polecenia *CREATE USER* został przedstawiony na listingu 3.34. Utworzony zostanie użytkownik o nazwie *student*, z hasłem *student* oraz wymuszeniem jego zmiany przy pierwszym logowaniu, z przydzieloną domyślną i tymczasową przestrzenią tabel. Jednak samo przypisanie przestrzeni tabel do użytkownika nie pozwala mu jeszcze tworzyć w nich obiektów, gdyż do tego celu niezbędne są odpowiednie uprawnienia oraz określenie rozmiaru obszaru w tych przestrzeniach (*QUOTA*), w którym będą tworzone obiekty.

Listing 3.34. Utworzenie nowego użytkownika

```
CREATE USER student IDENTIFIED BY student  
  DEFAULT TABLESPACE users  
  TEMPORARY TABLESPACE temp  
  PASSWORD EXPIRE;
```

Dodanie użytkownika można zrealizować także przy użyciu narzędzia *OEM*. W zakładce *Server*, w sekcji *Security*, występują opcje: *użytkowników* (ang. *user*), *roli* (ang. *role*) oraz *profilów* (ang. *profile*). Okno dodawania nowego użytkownika pojawi się po wybraniu opcji *Create* (rys. 3.16). Kolejne zakładki pozwalają na przyznanie m.in. ról czy przestrzeni tabel. W trakcie tworzenia użytkownika można podejrzeć kod SQL, odpowiadający operacjom wykonywanym w oknie. Za pomocą narzędzia *OEM* można także edytować opcje kont użytkowników oraz je usunąć.

ORACLE Enterprise Manager 11g  
Database Control  
Database Instance: home > Users > Logged in As SYS  
Create User  
Show SQL Cancel OK

General Roles System Privileges Object Privileges Quotas Consumer Group Privileges Proxy Users

\* Name student  
Profile DEFAULT  
Authentication Password  
\* Enter Password  
\* Confirm Password  
For Password choice, the role is authorized via password.  
☒ Expire Password now  
Default Tablespace USERS  
Temporary Tablespace TEMP  
Status ☐ Locked ☒ Unlocked

General Roles System Privileges Object Privileges Quotas Consumer Group Privileges Proxy Users  
Show SQL Cancel OK

Rys. 3.16. Okno tworzenia nowego użytkownika

Modyfikowanie użytkownika przeprowadza się poleceniem *ALTER USER* (Bryla, Loney, 2010). Jego składnia jest podobna do polecenia *CREATE USER*. Dodatkowo polecenie modyfikowania umożliwia przypisywanie ról oraz nadawanie praw aplikacjom warstwy pośredniej. Przykład modyfikacji użytkownika *student* został

przedstawiony na listingu 3.35, w którym przypisano mu obszar o rozmiarze 250 MB dla obiektów tworzonych w przestrzeni *users*.

*Listing 3.35. Edycja parametrów użytkownika student*

```
ALTER USER student  
    QUOTA 250M ON users;
```

Usuwanie użytkownika odbywa się przy użyciu polecenia *DROP USER*, w którym należy podać nazwę usuwanego użytkownika. Dodatkową opcją jest *CASCADE*, która automatycznie usuwa wszystkie obiekty utworzone przez danego użytkownika. Jeśli opcja *CASCADE* zostanie pominięta, obiekty użytkownika należy usunąć manualnie lub przenieść do innego schematu. Jeśli w bazie danych istnieją widoki lub pakiety, korzystające z obiektów usuniętych wraz z użytkownikiem, wówczas otrzymają one status *INVALID*. Dla poprawnego działania bazy czy aplikacji, konieczne będzie przypisanie tych obiektów do innych użytkowników. Ponadto po usunięciu użytkownika anulowane są wszystkie uprawnienia, które nadał on innym użytkownikom. Przykład usunięcia użytkownika *student* został przedstawiony na listingu 3.36.

*Listing 3.36. Usunięcie użytkownika z opcją CASCADE*

```
DROP USER student CASCADE ;
```

Po utworzeniu użytkownika, administrator może przypisać mu uprawnienia systemowe (ang. *system privilege*) i uprawnienia do obiektów (ang. *object privilege*).

Uprawnienia systemowe określają prawa wykonywania operacji na wskazanym rodzaju obiektu bazy danych oraz czynności systemowych, np. tworzenie zadań wsadowych, zmiana parametrów systemowych, tworzenie ról czy łączenie się z bazą (Bryła, Loney, 2010). Wszystkie uprawnienia dostępne w systemie Oracle 11g są dostępne w tabeli słownikowej *SYSTEM\_PRIVILEGE\_MAP* (listing 3.37). W tabeli 3.4 zostały zaprezentowane podstawowe uprawnienia systemowe.

Listing 3.37. Wyświetlenie uprawnień systemowych

```
SELECT * FROM SYSTEM_PRIVILEGE_MAP;
```

Tabela 3.4. Wybrane uprawnienia systemowe

Uprawnienia systemowe	Opis
ALTER DATABASE	Zmiany w bazie danych (np. statusu bazy).
ALTER SYSTEM	Wykonywanie instrukcji <i>ALTER SYSTEM</i> , przełączanie na następną grupę dzienników powtórzeń, zmiana parametrów inicjalizacyjnych systemu w pliku <i>SPFILE</i> .
CREATE ANY INDEX	Tworzenie indeksów w dowolnym schemacie.
CREATE PROFILE	Tworzenie profilu zasobów i haseł.
CREATE PROCEDURE	Tworzenie funkcji, procedur i pakietów we własnym schemacie.
CREATE ANY PROCEDURE	Tworzenie funkcji, procedur i pakietów w dowolnym schemacie.
CREATE SESSION	Nawiązywanie połączenia z bazą danych.
CREATE TABLE	Tworzenie tabeli we własnym schemacie.
CREATE ANY TABLE	Tworzenie tabeli w dowolnym schemacie.
CREATE TABLESPACE	Tworzenie przestrzeni tabel w bazie danych.
CREATE USER	Tworzenie konta użytkownika (schematu).
ALTER USER	Modyfikowanie kontami użytkowników.
SYSDBA	Dopisywanie pozycji w zewnętrznym pliku haseł, włączanie/wyłączanie, tworzenie, modyfikowanie, przywracanie bazy danych, tworzenie pliku <i>SPFILE</i> , łączenie z bazą pozostającą w trybie <i>RESTRICTED SESSION</i> .
SYSOPER	Dopisywanie pozycji w zewnętrznym pliku haseł, włączanie/wyłączanie, modyfikowanie, przywracanie bazy danych, tworzenie pliku <i>SPFILE</i> , łączenie z bazą pozostającą w trybie <i>RESTRICTED SESSION</i> .

Źródło: (Bryła, Loney, 2010)

Wybrane uprawnienia systemowe można przypisywać użytkownikowi, roli lub grupie *PUBLIC* (grupa wszystkich użytkowników bazy danych) za pomocą polecenia *GRANT*. Jeśli administrator zamierza uprawnienia odebrać, wykonuje wówczas polecenie *REVOKE* (Bryła, Loney, 2010). Przykład nadania użytkownikowi *student* uprawnień do łączenia się z bazą danych, tworzenia tabel i indeksów został przedstawiony na listingu 3.38. Listing 3.39 ilustruje usunięcie uprawnień do tworzenie podprogramów wbudowanych.

*Listing 3.38. Przyznanie użytkownikowi student uprawnień systemowych*

```
GRANT CREATE SESSION, CREATE ANY TABLE, CREATE ANY INDEX  
TO student ;
```

*Listing 3.39. Odebranie użytkownikowi student uprawnień systemowych*

```
REVOKE CREATE ANY PROCEDURE  
FROM student ;
```

Podczas przyznawania uprawnień systemowych istnieje możliwość użycia opcji *WITH ADMIN OPTION*, pozwalającej użytkownikowi, któremu przyznano uprawnienia, na przekazanie tych samych uprawnień innym użytkownikom (Bryła, Loney, 2010).

W zarządzaniu uprawnieniami systemowymi bardzo pomocne są widoki, w których zebrane są wszelkie informacje związane z tym aspektem zapewniania bezpieczeństwa bazy danych. Podstawowe widoki wraz z opisem zawartych w nich danych zostały przedstawione w tabeli 3.5.

*Tabela 3.5. Widoki danych słownikowych uprawnień systemowych*

Widok	Opis
DBA_SYS_PRIVS	Uprawnienia systemowe przypisane rolom i użytkownikom.
SESSION_PRIVS	Wszystkie uprawnienia systemowe dla danego użytkownika w danej sesji, przyznane bezpośrednio lub za pośrednictwem roli.

Widok	Opis
ROLE_SYS_PRIVS	Uprawnienia w bieżącej sesji przyznane użytkownikowi za pośrednictwem roli.

Źródło: (Bryla, Loney, 2010)

Nadawanie uprawnień systemowych jest również możliwe w narzędziu *OEM*. Edycja wybranego użytkownika umożliwia dodanie nowych lub usunięcie istniejących uprawnień systemowych. W tym celu należy edytować listę nadanych już uprawnień. Na rysunku 3.17 przedstawiono listę przywilejów, nadanych użytkownikowi *student*.



Rys. 3.17. Lista nadanych przywilejów systemowych użytkownikowi *student*

Natomiast rysunek 3.18 obrazuje sposób nadania uprawnienia *CREATE TABLESPACE* oraz *ALTER TABLESPACE*, poprzez wybór tych przywilejów z listy *Available System Privileges* i umieszczenie ich na liście *Selected System Privileges*, będącej zbiorem uprawnień systemowych danego użytkownika.

W tym samym oknie może odbywać się odbieranie uprawnień systemowych użytkownikowi. Operacja ta może dotyczyć pojedynczych praw (*Remove*) bądź też wszystkich posiadanych uprawnień (*Remove All*). Należy jednak pamiętać, że pozbawienie użytkownika uprawnień w tym miejscu nie gwarantuje, iż nie będzie on mógł wykonywać operacji, których te uprawnienia dotyczyły. Może on nadal korzystać z tego rodzaju praw, jeśli zostały one przyznane w ramach roli.





Rys. 3.18. Edycja przywilejów systemowych użytkownika student

Analogicznie do przywilejów systemowych, użytkownik może mieć przyznane uprawnienia do obiektów baz danych (np. tabeli, sekwencji). Nadawanie tych uprawnień odbywa się przy pomocy polecenia *GRANT* (listing 3.40), a ich odbieranie – polecenia *REVOKE* (listing 3.41). Można także nadać uprawnienia tego rodzaju grupie *PUBLIC*. Użytkownik, któremu nadano przywileje z opcją *WITH GRANT OPTION*, może przekazywać je kolejnym użytkownikom (Bryła, Loney, 2010). Uprawnienia te można także nadawać w zakładce *Object Privileges* w narzędziu *Enterprise Manager*.

Listing 3.40. Przyznanie użytkownikowi student uprawnień do obiektów

```
GRANT INSERT, UPDATE ON wykłady
  TO student;
```

Listing 3.41. Odebranie użytkownikowi student uprawnień do obiektów

```
REVOKE INSERT ON wykłady
  FROM student;
```

Użytkownik, który sam utworzył obiekty w danym schemacie, może także sam nadawać uprawnienia do nich innym użytkownikom. Uprawnienia te są podzielone według typu obiektów, do których się odnoszą. Przykładowo, dla tabel są to przywileje: *INSERT*, *UPDATE*, *DELETE*, a więc są one zgodne z operacjami, jakie można wykonać na tego rodzaju obiekcie.

Informacje o uprawnieniach, związanych z operacjami wykonywanymi na obiektach, można wyświetlić, korzystając z widoków wymienionych w tabeli 3.6.

Tabela 3.6. Widoki danych słownikowych uprawnień do obiektów

Widok	Opis
DBA_TAB_PRIVS	Uprawnienia do tabel, przyznawane rolom i użytkownikom.
DBA_COL_PRIVS	Uprawnienia do tabel nadane rolom i użytkownikom; dane zawierają nazwy kolumny i typ uprawnienia na niej.
SESSION_PRIVS	Wszystkie uprawnienia do obiektów dla danego użytkownika w danej sesji, przyznane bezpośrednio lub za pośrednictwem roli.
ROLE_TAB_PRIVS	Uprawnienia w bieżącej sesji przyznane tabeli za pośrednictwem roli.

Źródło: (Bryla, Loney, 2010)

Zarządzanie pojedynczymi uprawnieniami systemowymi oraz do obiektów nie należy do łatwych zadań, gdyż wymaga wielokrotnego powtarzania tych samych czynności w odniesieniu do różnych użytkowników, np. nadawanie tego samego zbioru praw wielu użytkownikom. Dlatego też działania administratora są wspomagane w tym zakresie poprzez wykorzystanie tzw. ról.

Rola jest to grupa uprawnień systemowych lub uprawnień do obiektów bądź też grupa łącząca oba tego rodzaju uprawniania (Bryla, Loney, 2010). Posługiwanie się rolami sprawia, że zarządzanie uprawnieniami staje się wygodniejsze i szybsze. W przypadku konieczności zmiany uprawnień wielu użytkowników, wystarczy zmodyfikować przyznane im role, zamiast zmieniać pojedyncze uprawnienia kolejnych osób. Niektóre role mogą być także zabezpieczane hasłem lub włączane automatycznie podczas logowania.

W systemie Oracle istnieją predefiniowane role, które są dostępne w bazie danych

bezpośrednio po jej utworzeniu. Administrator może także tworzyć własne role, modyfikować je lub usuwać przy pomocy poleceń SQL i narzędzia *OEM*.

Do tworzenia i zarządzania rolami służy polecenie *CREATE ROLE* (Bryla, Loney, 2010; Oracle, 2011b). Wykonanie tej instrukcji uwarunkowane jest posiadaniem uprawnienia *CREATE ROLE*. Przykład utworzenia roli o nazwie *rola\_st*, bez podawania hasła do niej, ilustruje listing 3.42. Możliwe jest także nadanie hasła do roli poprzez użycie klauzuli *IDENTIFIED BY hasło*. Opcja ta spowoduje konieczność wprowadzenia hasła w chwili aktywacji roli poleceniem *SET ROLE*.

Listing 3.42. Utworzenie roli o nazwie *rola\_st*

```
CREATE ROLE rola_st NOT IDENTIFIED;
```

Usunięcie roli następuje po wydaniu polecenia *DROP ROLE* (listing 3.43).

Listing 3.43. Usunięcie roli o nazwie *rola\_st*

```
DROP ROLE rola_st;
```

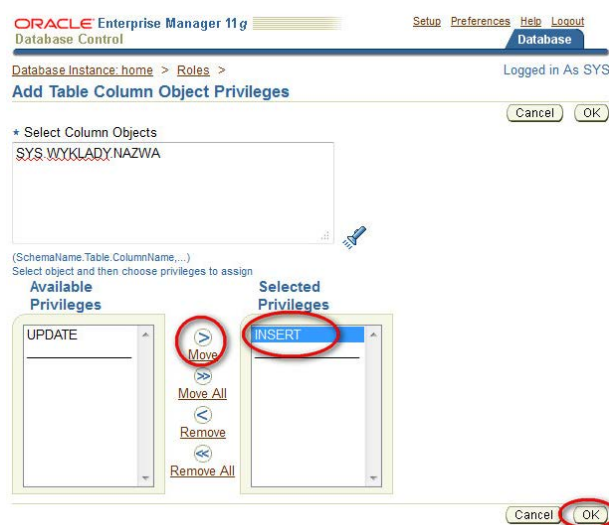
Rola, bezpośrednio po jej utworzeniu, nie zawiera żadnych uprawnień. Konieczne jest zatem przypisanie odpowiednich przywilejów do roli w taki sam sposób, w jaki są one nadawane użytkownikom. Listing 3.44 ilustruje przykład dodania do roli *rola\_st* uprawnienia systemowego *CREATE ANY TABLE* oraz uprawnienia *SELECT* do wybierania danych z tabeli *wyklady*.

Listing 3.44. Dodanie uprawnień do roli *rola\_st*

```
GRANT CREATE ANY TABLE TO rola_st;  
GRANT SELECT ON wykłady TO rola_st;
```

Uprawnienia takie można także przydzielić za pomocą narzędzia *OEM*. Edytując wybraną rolę, w kolejnych zakładkach można dodawać nowe uprawnienia. Dla

przykładu (rys. 3.19), nadanie uprawnienia *INSERT* do wykonania operacji wstawienia wartości w kolumnie *nazwa* tabeli *wykłady* wymaga wskazania tej kolumny w obszarze *Select Column Objects*, a następnie wybrania właściwego uprawnienia na liście *Available Privileges*.



Rys. 3.19. Dodanie uprawnienia *INSERT* do kolumny *nazwa*

Po zdefiniowaniu w ramach roli stosownych uprawnień, administrator może ją przypisać użytkownikowi za pomocą polecenia *GRANT* (listing 3.45). Rolę można także przypisać innej roli, tworząc w ten sposób hierarchię ról. Odbieranie roli, zarówno użytkownikowi jak i roli nadrzędnej, odbywa się przy wykorzystaniu polecenia *REVOKE* (listing 3.46).

Listing 3.45. Przypisanie roli *rola\_st* do użytkownika *student*

```
GRANT rola_st TO student ;
```

Listing 3.46. Odebranie roli *rola\_st* użytkownikowi *student*

```
REVOKE rola_st FROM student ;
```

Oprócz uprzednio wymienionych poleceń SQL, do zarządzania rolami można także wykorzystać narzędzie *OEM*. W tym celu należy wskazać użytkownika, któremu będzie przypisana rola, następnie przejść do zakładki *Role* oraz dokonać edycji listy ról. Lista dostępnych ról do przypisania wyświetlana jest w obszarze *Available Roles*, natomiast lista ról już przyznanych – w obszarze *Selected Roles* (rys. 3.20).



Rys. 3.20. Przypisywanie roli użytkownikowi student

W przypadku konieczności uzyskania informacji związanych z rolami za pomocą polecenia *SELECT*, można wykorzystać widoki systemowe, których charakterystykę zamieszczono w tabeli 3.7.

Tabela 3.7. Widoki danych słownikowych związanych z rolami

Widok	Opis
DBA_ROLES	Informacje o wszystkich rolach oraz czy wymagają podania hasła.
DBA_ROLE_PRIVS	Role przypisane użytkownikom lub inne role.
ROLE_ROLE_PRIVS	Role przypisane innym rolom.
ROLE_SYS_PRIVS	Uprawnienia systemowe nadane rolom.

Widok	Opis
ROLE_TAB_PRIVS	Uprawnienia do tabel i kolumn tabel przyznano rolom.
SESSION_ROLES	Role obowiązujące w danej sesji (wszystkich sesji użytkowników).

Źródło: (Bryla, Loney, 2010)

### 3.9. PYTANIA KONTROLNE

1. Jakie są podstawowe zadania administratora bazy danych?
2. Omów architekturę bazy danych Oracle.
3. Co to jest instancja bazy danych?
4. Wymień i omów przestrzenie tabel, tworzone przy domyślnej instalacji systemu Oracle 11g.
5. Jakie rodzaje przestrzeni tabel istnieją w bazie danych Oracle?
6. Wymień obszary pamięci systemu Oracle.
7. Na czym polega strojenie pamięci systemu Oracle? Jakimi narzędziami można nią zarządzać?
8. Jakie polecenie służy do tworzenia przestrzeni tabel?
9. W jaki sposób można edytować i usunąć przestrzeń tabel?
10. Jak modyfikuje się rozmiar przestrzeni tabel?
11. W jaki sposób można przenosić pliki danych?
12. Jakie polecenie SQL służy do tworzenia konta użytkownika? Omów jego parametry.
13. Jakie uprawnienia można nadawać użytkownikowi bazy danych? W jaki sposób nadaje się te uprawnienia?
14. Co to jest rola?
15. W jaki sposób można tworzyć, modyfikować i usuwać role?
16. Jak można przypisywać role do użytkowników i innych ról?
17. W jakich widokach można uzyskać informacje o wszystkich rolach?

## Wirtualne bazy danych

---

### Cel

Wirtualne prywatne bazy danych to mechanizm stosowany we współczesnych bazach danych, zapewniający bezpieczeństwo danych. Rozdział ten przedstawia sposoby tworzenia i wdrożenia polityk bezpieczeństwa danych z wykorzystaniem wirtualnych prywatnych baz danych.

### Plan

1. Mechanizm wirtualnych prywatnych baz danych.
2. Wstępna konfiguracja i tworzenie kontekstu.
3. Tworzenie strategii bezpieczeństwa.
4. Testowanie konfiguracji.

#### 4.1. MECHANIZM VPD

Wirtualne prywatne bazy danych (ang. *Virtual Private Database, VPD*) to jedno z częściej stosowanych zabezpieczeń w środowisku Oracle. Tradycyjne sposoby zapewniania bezpieczeństwa danych (jak uprawnienia, role, perspektywy, itd.) mają zwykle działanie zbyt rozległe, w wyniku czego dostęp użytkowników do danych może być bardziej ograniczony, niż jest to pożądane. Wirtualne prywatne bazy danych zawierają narzędzia, które pozwalają na określanie bezpieczeństwa na niższym poziomie danych.

Z łatwością można ograniczyć pojedynczemu użytkownikowi dostęp do wybranych wierszy w tabeli, spełniających zadany przez administratora warunek (np. wierszy dotyczących jedynie wybranego działu firmy). Co więcej, ograniczenia takie są zupełnie transparentne dla użytkownika bazy danych. Takie podejście powoduje, że polityka bezpieczeństwa danych stosowana jest zawsze w sposób spójny, niezależnie od poziomu dostępu. Nie ma więc potrzeby stosowania w aplikacjach specjalnie przystosowanych tabel, perspektyw czy innych obiektów, aby być w zgodzie z polityką bezpieczeństwa.

Dla tej samej bazy danych można istnieć równolegle kilka polityk VPD – nie trzeba tworzyć kilku oddzielnych baz np. dla poszczególnych klientów. Dostęp do każdego z obiektów bazy danych może być w łatwy sposób kontrolowany poprzez odpowiednie procedury i funkcje. Z danym obiektem, w tym samym czasie, można bowiem związać więcej niż jedną funkcję.

VPD umożliwiają szybkie zdejmowanie ograniczeń. Istnieje również możliwość nadania określonej użytkownikowi pełnego dostępu do danych, bez względu na obowiązujące ograniczenia. Do wad stosowania VPD można zaliczyć możliwe pogorszenie wydajności zapytań, które poddawane są modyfikacjom. Sytuacja taka ma miejsce w przypadku konieczności przeprowadzenia ponownej analizy składniowej i semantycznej zapytania. VPD powodują także dodatkowe utrudnienia podczas eksportu lub importu danych. W szczególności należy unikać eksportu ścieżkami bezpośrednimi, który omija warstwę VPD.



Wirtualne prywatne bazy danych posiadają serwerowe mechanizmy precyzyjnej kontroli dostępu, które są połączone z bezpiecznym kontekstem aplikacji (Loney, 2005; Loney, 2009). Zapewniają one bezpieczeństwo dostępu do danych na poziomie pojedynczych rekordów zamiast całej tabeli, jak to ma miejsce w przypadku tradycyjnych zabezpieczeń. Głównym komponentem mechanizmu VPD jest zabezpieczanie na poziomie wiersza (ang. *row – level security, RLS*), zwane również precyzyjną kontrolą dostępu (ang. *fine – grained access control, FGAC*) (Loney, Bryla, 2008). Strategie zabezpieczeń (tzn. reguły opisujące, jakie dane mają być udostępniane poszczególnym użytkownikom) są w mechanizmach VPD dołączane bezpośrednio do kolumn, tabel, perspektyw i synonimów, co uniemożliwia użytkownikom ich ominięcie. Taka drobnoziarnista kontrola dostępu do danych może być wykorzystana do następujących celów:

- Wymuszanie kontroli dostępu na poziomie wiersza poprzez instrukcje *SELECT*, *INSERT*, *UPDATE* i *DELETE*.
- Utworzenie polityki bezpieczeństwa, która kontrolować będzie dostęp oparty o określone wartości kolumn.
- Utworzenie polityk, które będą aplikowane zawsze w ten sam sposób oraz takich, które zmieniają się dynamicznie podczas wykonywania zapytania.
- Utworzenie zbioru (grupy) polis bezpieczeństwa.

Zabezpieczenia na poziomie wiersza, udostępniane przez VPD, są wymuszane poprzez zastosowanie polityki bezpieczeństwa bezpośrednio do obiektu bazy danych (jak tabela czy perspektywa). Dlatego niezależnie od narzędzia dostępowego do bazy danych użytkownik nie będzie w stanie ominąć nałożonych ograniczeń.

Instrukcje SQL, stosowane w VPD, są poddawane dynamicznym modyfikacjom tak, aby użytkownik miał dostęp jedynie do pewnych wierszy tabel i perspektyw. Specjalnie tworzone procedury składowane, dotyczące określonych obiektów (np. tabel), dodają do instrukcji SQL użytkownika odpowiednie warunki. Instrukcje takie jak *SELECT*, *INSERT*, *UPDATE*, *DELETE*, *INDEX* mają dołączany predykat (czyli odpowiedni warunek dodawany po klauzuli *WHERE*), ograniczający działanie instrukcji jedynie do tych wierszy tabeli, które są określone w polityce bezpieczeństwa. Predykat taki jest generowany przez funkcję PL/SQL w przypadku, gdy zapytanie dotyczy określonego obiektu. Modyfikacje takie są transparentne dla użytkownika.

Przykładem może być zapytanie pobierające wielkości pensji pracowników (listing 4.1), które w rzeczywistości jest modyfikowane tak, aby uzyskane rekordy dotyczyły jedynie pracowników z działu *Sales* (listing 4.2).

*Listing 4.1. Przykładowe zapytanie*

```
SELECT Name, Surname, Salary FROM Employee;
```

*Listing 4.2. Przykładowe zapytanie z listingu 4.1 z dodanym predykatem*

```
SELECT Name, Surname, Salary FROM Employee WHERE dept='Sales';
```

Dla każdego rodzaju polecenia można definiować inne strategie zabezpieczeń. Jednocześnie wprowadzenie dynamicznych modyfikacji zapytań sprawia, że sama aplikacja nie musi korzystać z konkretnych tabel, widoków lub synonimów, aby być w zgodzie z polityką bezpieczeństwa.

Mechanizm VPD umożliwia zdefiniowanie sposobu, w jaki tabela, a nawet pojedyncza kolumna, będą postrzegane przez różnych użytkowników. Istnienie mechanizmu VPD na poziomie kolumn pozwala na maskowanie kolumn, w których zawarte są wrażliwe dane. W ten sposób łatwo jest ograniczyć określonemu użytkownikowi dostęp do kolumny bądź kolumn tabeli.

Administrator bazy danych może obsługiwać wiele polityk VPD dla jednej dużej bazy danych. Może też określać statyczne polityki VPD w zależności od kontekstu aplikacji, który ułatwia tworzenie polityk bezpieczeństwa, dotyczących poszczególnych aspektów działań użytkownika (np. uprawnień nadawanych podczas logowania). Kontekst taki umożliwia definiowanie zestawu atrybutów (jak np. zmiennych środowiskowych), przydatnych podczas określania kontroli dostępu do bazy danych. Posługiwanie się tego rodzaju rozwiązaniem ułatwiają konteksty predefiniowane, zawierające zestawy przypisanych atrybutów. Oracle korzysta z kontekstu *USERENV*, którego predefiniowane atrybuty domyślnie używane są w zarządzaniu dostępem do bazy danych.

Obsługiwane przez VPD polityki bezpieczeństwa to: polityki dynamiczne, polityki

statyczne oraz polityki zależne od kontekstu. Dla polityk dynamicznych charakterystyczny jest fakt wykonywania ich funkcji zawsze podczas parsowania instrukcji SQL, zawierającej docelową tabelę VPD. Polityki statyczne oraz polityki zależne od kontekstu wprowadzono po raz pierwszy w wersji Oracle 10g. Ich istotną zaletą jest to, że nie muszą być wywoływane za każdym razem, gdy jest wykonywane zapytanie, ponieważ są one buforowane w pamięci podręcznej. Polityki statyczne wykonywane są tylko raz podczas logowania się, a następnie pozostają w pamięci podręcznej na czas trwania sesji (niezależnie od kontekstu) w razie, gdyby zaszła potrzeba ponownego ich użycia. Takie podejście znacznie podnosi wydajność bazy danych. Polityki zależne od kontekstu wywołują funkcję polityki w fazie parsowania jedynie w przypadku zmiany kontekstu aplikacji.

Zastosowanie mechanizmu VPD umożliwia określenie sposobu postrzegania tego samego obiektu przez różnych użytkowników. Utworzenie VPD odbywa się w następujących etapach:

- wstępna konfiguracja (w tym tworzenie użytkowników i nadawanie podstawowych uprawnień),
- utworzenie własnego kontekstu aplikacji,
- utworzenie wyzwalaczy logowania,
- utworzenie strategii bezpieczeństwa,
- zastosowanie strategii bezpieczeństwa do określonych obiektów,
- testowanie mechanizmu VPD.

Czynności te omówione zostaną w kolejnych podrozdziałach.

## 4.2. WSTĘPNA KONFIGURACJA I TWORZENIE KONTEKSTU

Definiowanie mechanizmu VPD należy rozpocząć od wstępnej konfiguracji, która obejmuje utworzenie użytkowników z uprawnieniami *CREATE ANY CONTEXT* i *CREATE PUBLIC SYNONYM* do określonych tabel (Loney, 2009).

Kontekst aplikacji tworzy się przy pomocy polecenia *CREATE CONTEXT* (listing 4.3). Umożliwia ono zdefiniowanie reguł dotyczących tworzonych polityk bezpieczeństwa, w szczególności związanych z uprawnieniami użytkowników.

Polecenie to można także użyć do stworzenia nazw atrybutów definiowanych dla polityki bezpieczeństwa oraz stworzenia pakietu, który będzie używany do definiowania kontekstu bezpieczeństwa dla funkcji i procedur (Urman i in, 2007).

*Listing 4.3. Tworzenie kontekstu*

```
CREATE CONTEXT nazwa_kontekstu USING schemat.nazwa_pakietu
      CREATE OR REPLACE PACKAGE nazwa_pakietu AS
          PROCEDURE nazwa_procedury;
END;
```

Kontekst *nazwa\_kontekstu* jest powiązany z pakietem *nazwa\_pakietu*, dzięki czemu żadna inna procedura nie będzie mogła zmienić atrybutów kontekstu dla każdej ustanowionej przez użytkownika sesji aż do jego wylogowania się. Umożliwia to zabezpieczenie kontekstu aplikacji, którego użytkownik po zalogowaniu nie może zmienić. Pakiet składający się z jednej procedury należy wypełnić treścią, korzystając z polecenia *CREATE PACKAGE BODY* (listing 4.4).

*Listing 4.4. Utworzenie ciała pakietu*

```
CREATE OR REPLACE PACKAGE BODY nazwa_pakietu IS
    PROCEDURE nazwa_procedury IS
    ...
    END nazwa_procedury ;
END nazwa_pakietu;
```

Każdy pakiet, implementujący kontekst aplikacji, korzysta z wbudowanego kontekstu *USERENV* (Urman i in, 2007). Kontekst ten jest domyślnym kontekstem Oracle. Posiada on wiele parametrów dotyczących sesji użytkownika, które warto wykorzystać, tworząc własny kontekst dla VPD. Jego najważniejsze parametry to: *CURRENT\_SCHEMA* (domyślny schemat), *DB\_NAME* (nazwa bazy danych), *HOST* (nazwa komputera, z którego użytkownik się łączy), *OS\_USER* (konto systemu operacyjnego, z którego użytkownik się łączy), *SESSION\_USER* (nazwa zalogowanego użytkownika).

Zmiana parametrów kontekstu aplikacji jest możliwa poprzez wykorzystanie funkcji `DBMS_SESSION.SET_CONTEXT` lub `DBMS_SESSION.SET_IDENTIFIER`. Aby utworzyć pakiet zawierający procedurę, która będzie nadawać parametrom określone wartości, należy użyć polecenia określającego ograniczenia mechanizmu VPD (listing 4.5).

Listing 4.5. Określanie parametrów kontekstu

```
DBMS_SESSION.SET_CONTEXT('nazwa_kontekstu',  
'zmienna_kontekstu', 'wartosc');
```

Zmienne kontekstu (np. nazwa użytkownika) mogą zostać pobrane przy pomocy instrukcji `SELECT` i funkcji `SYS_CONTEXT` (listing 4.6).

Listing 4.6. Pobranie wartości zmiennych kontekstu

```
SELECT SYS_CONTEXT('USERENV', 'SESSION_USER')  
FROM DUAL ;
```

Najważniejsze zmienne kontekstu przedstawiono w tabeli 4.1.

Tabela 4.1. Opis znaczenia wybranych zmiennych kontekstu

Parametr	Opis
AUTHENTICATION_DATA	Dane identyfikacyjne logującego się użytkownika.
AUTHENTICATION_TYPE	Metoda identyfikacji użytkownika.
BG_JOB_ID	Identyfikator zadania (zwrócony, jeśli bieżącą sesję ustanowił proces drugoplanowy Oracle).
CLIENT_IDENTIFIER	Identyfikator użytkownika (określany przez funkcję <code>DBMS_SESSION.SET_IDENTIFIER</code> ).
CLIENT_INFO	Informacje o sesji użytkownika.
CURRENT_SCHEMA	Nazwa domyślnego schematu, używanego w bieżącym schemacie.

Parametr	Opis
CURRENT_SQL	Pierwsze 4 kilobajty aktualnie wykonywanego wyrażenia SQL.
CURRENT_USER	Nazwa użytkownika, z którego uprawnieniami działa bieżąca sesja.
CURRENT_USERID	Identyfikator użytkownika, z którego uprawnieniami działa bieżąca sesja.
DB_DOMAIN	Nazwa domeny bazy danych.
DB_NAME	Nazwa bazy danych.
EXTERNAL_NAME	Zewnętrzna nazwa użytkownika bazy danych.
FG_JOB_ID	Identyfikator zadania (zwrócony, jeśli bieżącą sesję ustanowił proces pierwszoplanowy Oracle).
HOST	Nazwa komputera, z którego połączył się klient.
INSTANCE	Numer identyfikacyjny bieżącej instancji.
ISDBA	Jeśli rola DBA jest włączona, zwrócona zostanie wartość TRUE. W przeciwnym wypadku FALSE.
NETWORK_PROTOCOL	Protokół sieciowy używany do połączenia.
OS_USER	Nazwa użytkownika systemu operacyjnego, który zapoczątkował sesję.
PROXY_USER	Nazwa użytkownika bazy danych, który rozpoczął sesję jako SESSION_USER.
PROXY_USER_ID	Identyfikator użytkownika bazy danych, rozpoczynającego sesję jako SESSION_USER.
SESSION_USER	Nazwa użytkownika bazy danych, pod którą użytkownik jest identyfikowany.
SESSION_USERID	Identyfikator użytkownika bazy danych, pod którą bieżący użytkownik jest identyfikowany.
SESSIONID	Identyfikator sesji.
TERMINAL	Identyfikator systemu operacyjnego dla klienta w bieżącej sesji.

Źródło: (Loney, 2005)

Po utworzeniu pakietu konieczne jest przypisanie do niego odpowiednich uprawnień, zazwyczaj o charakterze publicznym (listing 4.7).

Listing 4.7. Przypisanie uprawnień do pakietu

```
GRANT EXECUTE ON schemat.nazwa_pakietu TO PUBLIC;  
CREATE PUBLIC SYNONIM nazwa_pakietu FOR schemat.nazwa_pakietu;
```

W celu ustawienia w sesji użytkownika odpowiedniego kontekst należy utworzyć odpowiedni wyzwalacz. Powinien on uruchamiać, podczas logowania, procedurę odpowiedzialną za definiowanie ustawień kontekstu dla odpowiednich obiektów (listing 4.8).

Listing 4.8. Utworzenie wyzwalacza dla procesu logowania

```
CREATE OR REPLACE TRIGGER schemat.ustaw_kontekst  
AFTER LOGON DATABASE  
BEGIN  
    schemat.nazwa_pakietu.procedura_ustawiania_kontekstu;  
END;
```

Procedura *procedura\_ustawiania\_kontekstu* powinna wywoływać odpowiednią procedurę *DBMS\_SESSION.SET\_CONTEXT*, która umożliwi automatyczne ustawienie określonych ograniczeń dla danego użytkownika po jego zalogowaniu się do bazy danych. Przykładowo, procedura umieszczona w takim wyzwalaczu mogłaby sprawdzać tożsamość zalogowanego użytkownika i na jej podstawie ograniczyć jego dostęp do tabel, np. jedynie do informacji dotyczących jego działu w firmie. Jeśli użytkownik nie zostałby poprawnie zweryfikowany, jego dostęp powinien zostać ograniczony w całości.

### 4.3. TWORZENIE STRATEGII BEZPIECZEŃSTWA

Po utworzeniu kontekstu i wyzwalaczy logowania kolejnym krokiem jest określenie strategii bezpieczeństwa. W tym celu należy zdefiniować funkcje, które będą generować predykaty dodawane do poleceń *SELECT* i operacji DML (Alapati, 2009; Loney, 2005). Predykat ten będzie wykorzystywany zawsze, gdy użytkownik wykona

określone zapytanie lub instrukcję DML. Jedna funkcja powinna generować predykat tylko dla jednego rodzaju operacji. Funkcja zwracająca predykat powinna posiadać przynajmniej dwa argumenty: właściciela chronionego obiektu i nazwę obiektu w schemacie użytkownika. Przykładowy szkielet pakietu, zawierającego dwie funkcje odpowiedzialne za kontrolę dostępu odpowiednio dla instrukcji *SELECT* i DML, przedstawiono na listingu 4.9. Funkcje te powinny zwracać ciąg znaków zawierający wyrażenie dodawane do klauzuli *WHERE* polecenia *SELECT* lub instrukcji DML (Urman i in, 2007).

Listing 4.9. Przykładowy szkielet pakietu

```
CREATE OR REPLACE PACKAGE BODY pakiet_bezpieczenstwa IS
    FUNCTION kontrola_select
        (uzytkownik1 VARCHAR2, obiekt1 VARCHAR2)
    RETURN VARCHAR2 IS
        predykat VARCHAR2(100) ;
    BEGIN
        -- ciąg instrukcji pozwalający na przeglądanie wierszy
        -- tabeli odpowiednim użytkownikom,
        -- sprawdzanie zmiennych kontekstowych
        -- utworzenie predykatu;
        RETURN predykat;
    END kontrola_select;

    FUNCTION kontrola_dml
        (uzytkownik1 VARCHAR2, obiekt1 VARCHAR2)
    RETURN VARCHAR2 IS
        predykat VARCHAR2(100);
    BEGIN
        -- ciąg instrukcji pozwalający na wprowadzanie zmian
        -- do tabeli odpowiednim użytkownikom,
        -- sprawdzane zmiennych kontekstowych
        -- utworzenie predykatu;
        RETURN predykat;
    END kontrola_dml;
END; --koniec pakietu
```

Do utrzymania polityk bezpieczeństwa, dotyczących kolumn, tabel, widoków i synonimów, można także użyć wbudowanego pakietu *DBMS\_RLS* (Kuhn, 2010). Zawiera on podprogramy, które są przydatne w zarządzaniu politykami



bezpieczeństwa, jak np.: *ADD\_POLICY* (dodanie polityki kontroli dostępu do obiektu), *DROP\_POLICY* (usunięcie polityki bezpieczeństwa), *REFRESH\_POLICY* (ponowne parsowanie instrukcji z pamięci podręcznej, powiązanych z polityką bezpieczeństwa), *ENABLE\_POLICY* (włączenie lub wyłączenie polityki bezpieczeństwa), *CREATE\_POLICY\_GROUP* (tworzenie grupy polityk), *DELETE\_POLICY\_GROUP* (usunięcie grupy polityk), *ADD\_GROUPED\_POLICY* (dodanie polityki do grupy polityk), *DROP\_GROUPED\_POLICY* (usunięcie polityki z grupy polityk), *ADD\_POLICY\_CONTEXT* (dodanie kontekstu do bieżącej aplikacji), *DROP\_POLICY\_CONTEXT* (usunięcie kontekstu z bieżącej aplikacji).

Zastosowanie polityki bezpieczeństwa do obiektu (np. do tabel) realizuje się przez użycie procedury *ADD\_POLICY*. Przykład jej użycia prezentuje listing 4.10.

*Listing 4.10. Zastosowanie polityki bezpieczeństwa*

```
DBMS_RLS.ADD_POLICY('uzytkownik1', 'obiekt1',  
                    'nazwa_polityki_bezpieczenstwa','schemat',  
                    'pakiet_bezpieczenstwa.kontrola_select', 'SELECT', TRUE);
```

Zastosowanie mechanizmów VPD w praktyce zaprezentowane zostanie na przykładzie, w którym w schemacie użytkownika *admin* zostaną stworzone dwie tabele (*Pracownicy*, *Pracownicy\_tr*), przechowujące dane o pracownikach i ich transakcjach. Aby możliwe było użycie mechanizmu VPD, konieczny jest dostęp do pakietu *DBMS\_RLS*, do którego użytkownicy powinni mieć uprawnienia *EXECUTE*. Należy też nadać użytkownikom uprawnienia systemowe *CREATE ANY CONTEXT*, *CREATE PUBLIC SYNONYM*, *CREATE ANY TABLE* oraz uprawnienia do tabel. Warto jednak pamiętać, że nadanie tych uprawnień możliwe jest tylko z konta użytkownika z uprawnieniami *SYSDBA* (listing 4.11).

*Listing 4.11. Konfiguracja wstępna*

```
CONNECT system/haslo AS SYSDBA  
  
GRANT EXECUTE ON DBMS_RLS TO PUBLIC;  
CREATE USER nowak IDENTIFIED BY now45;
```

```
CREATE USER kowalski IDENTIFIED BY kowal12;
CREATE USER admin IDENTIFIED BY admin;

GRANT CREATE SESSION TO nowak, kowal, admin;
GRANT CREATE ANY CONTEXT, CREATE PUBLIC SYNONYM,
      CREATE ANY TABLE TO admin;

CONNECT admin/admin

CREATE TABLE Pracownicy(
      id_pracownik NUMBER(8),
      nazwa_p VARCHAR2(50),
      nr_dokumentu VARCHAR2(30));

INSERT INTO Pracownicy VALUES (1, 'Nowak', 1, 'DS/55');
INSERT INTO Pracownicy VALUES (2, 'Kowalski', 1, 'DS/12');

CREATE TABLE Pracownicy_tr(
      id_pracownik NUMBER(8),
      nr_dokumentu VARCHAR2(30),
      cena NUMBER(6,2),
      flaga_tr VARCHAR2(1));

INSERT INTO Pracownicy_tr VALUES (1, 'FV01/2/15', 25.6, 1);
INSERT INTO Pracownicy_tr VALUES (1, 'FV03/2/31', 45.1, 2);
INSERT INTO Pracownicy_tr VALUES (2, 'FV01/22/1', 233.1, 1);

GRANT SELECT, INSERT ON Pracownicy TO nowak, kowalski;
GRANT SELECT ON Pracownicy_tr TO nowak, kowalski;
```

Kolejnym etapem jest zdefiniowanie kontekstu aplikacji (listing 4.12). Polecenie *CREATE CONTEXT* pozwoli na określenie nazwy pakietu (*pakiet\_kontekstu*) służącego do określenia reguł. Następnie tworzony jest sam pakiet, który składa się z wywołania procedury *SET\_CONTEXT*. Procedura ta umożliwi ustawienie kontekstu dla każdej sesji użytkownika. Zawiera ona zapytanie, które porównuje nazwę zalogowanego użytkownika z nazwą pracownika z tabeli *Pracownicy*. W zależności od zwróconego wyniku ustawiana jest zmienna *User\_Id* sesji. Przyjmuje ona wartość identyfikatora użytkownika lub 0, jeśli odpowiednik nie zostanie znaleziony. Po utworzeniu pakietu należy przypisać do niego odpowiednie uprawnienia.

Listing 4.12. Tworzenie kontekstu aplikacji

```
CONNECT admin/admin

CREATE CONTEXT admin USING admin.pakiet_kontekstu;

CREATE OR REPLACE PACKAGE pakiet_kontekstu AS
    PROCEDURE set_context;
END;

CREATE OR REPLACE PACKAGE BODY pakiet_kontekstu IS
    PROCEDURE set_context IS
        nazwa_u VARCHAR2(50);
        id_u NUMBER;
    BEGIN
        SELECT SYS_CONTEXT('USERENV','SESSION_USER')
            INTO nazwa_u FROM DUAL;
        DBMS_SESSION.SET_CONTEXT('admin','username',nazwa_u);
        BEGIN
            SELECT id_pracownik INTO id_u FROM Pracownicy
            WHERE nazwa_p = nazwa_u;
            DBMS_SESSION.SET_CONTEXT('admin','user_id',id_u);
        EXCEPTION
            WHEN NO_DATA_FOUND THEN
                DBMS_SESSION.SET_CONTEXT('admin','user_id',0);
        END;
    END set_context;
END pakiet_kontekstu;

GRANT EXECUTE ON admin.pakiet_kontekstu TO PUBLIC;
CREATE PUBLIC SYNONYM pakiet_kontekstu
    FOR admin.pakiet_kontekstu;
```

Aby reguły kontekstu były ustawiane w sesji dla każdego użytkownika, należy zdefiniować wyzwalacz uruchamiany przy każdej próbie logowania (listing 4.13). Zostanie wówczas wykonana procedura *SET\_CONTEXT* z pakietu *pakiet\_kontekstu*. Procedura ta dokona sprawdzenia, czy konto o podanej nazwie istnieje w tabeli *Pracownicy*. Jeśli konto istnieje, to kontekst sesji zostanie odpowiednio zmodyfikowany. Dzięki temu każdy zalogowany użytkownik, którego nazwa znajduje się w tabeli *Pracownicy*, może mieć wprowadzone odpowiednie ograniczenia dostępu do wierszy w innych tabelach.

Listing 4.13. Tworzenie wyzwalacza logowania

```
CONNECT system/haslo AS SYSDBA

CREATE OR REPLACE TRIGGER admin.set_context
  AFTER LOGON ON DATABASE
BEGIN
  admin.pakiet_kontekstu.set_context;
END;
```

Po utworzeniu kontekstu i wyzwalacza logowania należy utworzyć pakiet zabezpieczeń. Będzie on przetwarzał wyniki pakietu kontekstu i umożliwi wpływ pakietu kontekstu na interakcję z użytkownikami. Listing 4.14 prezentuje przykład zdefiniowania nagłówka pakietu oraz utworzenia ciała pakietu, zawierającego dwie wewnętrzne funkcje, obsługujące dwa rodzaje operacji: *INSERT* i *SELECT*. Funkcje te są odpowiedzialne za generowanie predykatu, który zostanie użyty za każdym razem, gdy wykonywana będzie odpowiednia instrukcja. W omawianym przykładzie, jeśli w/w instrukcje będą wykonywane przez użytkownika *admin*, wówczas wartość predykatu będzie równa NULL. Natomiast predykat zostanie wygenerowany, jeśli użytkownik wykonujący zapytanie nie jest właścicielem tabeli. Wówczas będzie wyznaczona wartość zmiennej *User\_Id*, która zostanie przekazana do zapytania jako warunek ograniczający. Należy też nadać wszystkim użytkownikom, którzy będą korzystać z tabeli *Pracownicy\_tr*, uprawnienia do wykonywania pakietu oraz utworzyć synonim publiczny (listing 4.14).

Listing 4.14. Tworzenie pakietu zabezpieczeń

```
CONNECT admin/admin

CREATE OR REPLACE PACKAGE pakiet_security IS
  FUNCTION pracow_tr_select_security
    (owner IN VARCHAR2,
     objname IN VARCHAR2) RETURN VARCHAR2;
  FUNCTION pracow_tr_insert_security
    (owner IN VARCHAR2,
     objname IN VARCHAR2) RETURN VARCHAR2;
END pakiet_security;
```

```
CREATE OR REPLACE PACKAGE BODY pakiet_security IS
    FUNCTION pracow_tr_select_security(owner IN VARCHAR2,
        objname IN VARCHAR2) RETURN VARCHAR2 IS
        predykat VARCHAR2(3000);
    BEGIN
        IF(SYS_CONTEXT('USERENV','SESSION_USER')='ADMIN')
            THEN predykat:= NULL;
        ELSE
            predykat:='Id_pracownik = SYS_CONTEXT
                (''admin'', ''user_id'')';
        END IF;
        RETURN predykat;
    END pracow_tr_select_security;

    FUNCTION pracow_tr_insert_security(owner IN VARCHAR2,
        objname IN VARCHAR2) RETURN VARCHAR2 IS
        predykat VARCHAR2(3000);
    BEGIN
        IF(SYS_CONTEXT('USERENV','SESSION_USER')='ADMIN')
            THEN predykat:= NULL;
        ELSE
            predykat:='Id_pracownik = SYS_CONTEXT
                (''admin'', ''user_id'')';
        END IF;
        RETURN predykat;
    END pracow_tr_insert_security;
END pakiet_security;

GRANT EXECUTE ON admin.pakiet_security TO PUBLIC;
CREATE PUBLIC SYNONYM pakiet_security
    FOR admin.pakiet_security;
```

Zastosowanie pakietu zabezpieczeń do tabel prezentuje listing 4.15. Przedstawia on wykorzystanie procedury *ADD\_POLICY* pakietu *DBMS\_RLS*, która uruchamiana jest dwa razy: przy próbie dodawania rekordów do tabeli oraz wykonywania zapytań.

*Listing 4.15. Zastosowanie strategii bezpieczeństwa do tabel*

```
BEGIN
    DBMS_RLS.ADD_POLICY('ADMIN','pracownicy_tr',
        'strategia_insert','ADMIN',
        'pakiet_security.pracow_tr_insert_security',
        'INSERT', TRUE);
```

```
DBMS_RLS.ADD_POLICY('ADMIN', 'pracownicy_tr',  
    'strategia_select', 'ADMIN',  
    'pakiet_security.pracow_tr_select_security',  
    'SELECT', TRUE);  
END;
```

Poza implementacją mechanizmów VPD dla tabel istnieje także możliwość zastosowania ich na poziomie kolumn, co pozwoli ukryć wybrane kolumny. Przykład takiego działania dla kolumny *Cena* z tabeli *Pracownicy\_tr* ilustruje listing 4.16.

*Listing 4.16. Zastosowanie strategii bezpieczeństwa do kolumn*

```
BEGIN  
    DBMS_RLS.ADD_POLICY  
        (object_schema=>'ADMIN',  
         object_name=>'pracownicy_tr',  
         policy_name=>'strategia_select',  
         function_schema=>'ADMIN',  
         policy_function=>  
             'pakiet_security.pracow_tr_insert_security',  
         sec_relevant_cols=>'cena');  
END;
```

W praktyce dla wierszy, do których użytkownik nie powinien mieć dostępu, w ukrytej kolumnie pojawiają się wartości NULL. Wartości w maskowanej kolumnie wyświetlą się jedynie dla wierszy, do których użytkownik ma prawo. Warto także zwrócić uwagę, że maskowanie kolumn można zastosować jedynie do zapytań, natomiast możliwość ta nie dotyczy instrukcji DML.

#### 4.4. TESTOWANIE KONFIGURACJI

Zapewnienie skuteczności zastosowania VPD jako mechanizmu bezpieczeństwa bazy danych wymaga przeprowadzenia testów, które zweryfikują poprawność przyjętych polityk bezpieczeństwa.

Listing 4.17 przedstawia przykład sprawdzania poprawności działania kontekstu

z wykorzystaniem procedury `SYS_CONTEXT`. Przy pomocy zapytania, zawartego w procedurze, można sprawdzić, czy w tabeli *Pracownicy* istnieje konto o podanej nazwie użytkownika i jeśli tak jest, to kontekst sesji zostanie zmodyfikowany zgodnie z wartością pola *Id\_pracownik*.

Listing 4.17. Sprawdzenie poprawności działania kontekstu

```
CONNECT nowak/now45
```

```
SELECT SYS_CONTEXT('USERENV','SESSION_USER') Nazwa_p,  
       SYS_CONTEXT('admin','user_id') Id_pracownik FROM DUAL;
```

Wynik zapytania:

```
NAZWA_P
```

```
ID_PRACOWNIK
```

```
Nowak
```

```
1
```

Efekty wprowadzonej strategii bezpieczeństwa można sprawdzić, wykonując odpowiednie zapytanie, po zalogowaniu się jako jeden z użytkowników odpowiednio uprawnionych i nieuprawnionych. Uprawnieni użytkownicy, nawet bez podawania warunku, powinni otrzymać jedynie dane, które mają prawo oglądać (w tle do zapytania będzie dodany odpowiedni predykat).

Listing 4.18 przedstawia zapytanie i jego wynik uzyskany dla użytkownika *admin*, który jest właścicielem tabel i ma dostęp do wszystkich ich rekordów, natomiast listing 4.19 – dla użytkownika z nałożonymi ograniczeniami, który może pobrać jedynie te rekordy, które są powiązane z jego identyfikatorem. Pozostałe rekordy tabeli *Pracownicy\_tr* będą dla niego niewidoczne.

Listing 4.18. Sprawdzenie poprawności działania strategii bezpieczeństwa dla użytkownika *bez ograniczeń*

```
CONNECT admin/admin
```

```
SELECT * FROM Pracownicy_tr;
```

Wynik zapytania:

ID_PRACOWNIK	NR_DOKUMENTU	CENA	FLAGA_TR
-----	-----	-----	-----
1	'FV01/2/15 '	25.6	1
1	'FV03/2/31 '	45.1	2
2	'FV01/22/1 '	233.1	1

*Listing 4.19. Sprawdzenie poprawności działania strategii bezpieczeństwa dla użytkownika z ograniczeniami*

```
CONNECT nowak/now45
```

```
SELECT * FROM Pracownicy_tr;
```

Wynik zapytania:

ID_PRACOWNIK	NR_DOKUMENTU	CENA	FLAGA_TR
-----	-----	-----	-----
1	'FV01/2/15 '	25.6	1
1	'FV03/2/31 '	45.1	2

Jeśli to samo zapytanie zostanie wykonane przez użytkownika nieuprawnionego, wówczas wynikowy zbiór rekordów będzie zbiorem pustym (listing 4.20).

*Listing 4.20. Sprawdzenie poprawności działania strategii bezpieczeństwa dla użytkownika bez uprawnień*

```
CONNECT adam/adam
```

```
SELECT * FROM Pracownicy_tr;
```

Wynik zapytania:

nie wybrano żadnych wierszy



Podobnie w przypadku operacji DML (np. *INSERT*), uprawnieni użytkownicy powinni mieć prawo dodawania danych zgodnych z kontekstem. Próba wprowadzenia wierszy, które zawierają dane niezgodne z polityką bezpieczeństwa, określoną dla danego użytkownika, zakończy się niepowodzeniem. Listing 4.21 prezentuje próbę dodania wiersza z danymi, które dotyczą innego użytkownika. Operacja taka jest niezgodna z przyjętą polityką bezpieczeństwa.

*Listing 4.21. Sprawdzenie poprawności działania strategii bezpieczeństwa dla użytkownika z ograniczonymi uprawnieniami*

```
CONNECT nowak/now45
```

```
INSERT INTO pracownicy_tr VALUES (2, 'FV11/5/31', 831.5, 'S');
```

Wynik zapytania:

Błąd w linii 1:

ORA-28115: naruszenie strategii z opcją sprawdzającą

Błąd wygenerowany przy próbie wykonania operacji z listingu 4.21 wskazuje, że użytkownik próbował wstawić rekord, do którego nie miałby dostępu ze względu na politykę bezpieczeństwa.

Podsumowując zaprezentowane zagadnienia, dotyczące mechanizmu VPD, należy stwierdzić, iż jest on wygodnym narzędziem do tworzenia polityk bezpieczeństwa, omijającym ograniczenia standardowych sposobów zapewniania bezpieczeństwa. Wirtualna prywatna baza danych może być utworzona dla każdego użytkownika, zapewniając w ten sposób indywidualizację oraz separację dostępu do danych. Jednocześnie odpowiednimi uprawnieniami poszczególnych użytkowników, w ramach przyjętych dla nich polityk bezpieczeństwa, można zarządzać oddzielnie, ponieważ są one niezależne od siebie.

**4.5. PYTANIA KONTROLNE**

1. Wskaż obszary zastosowań wirtualnych baz danych.
2. Wymień etapy tworzenia wirtualnych baz danych.
3. Wymień sposoby definiowania strategii bezpieczeństwa.
4. Co to jest kontekst? Wymień znane Ci konteksty wbudowane.
5. Co to jest predykat i jakie jest jego zastosowanie?
6. Scharakteryzuj funkcjonalność pakietu *DBMS\_RLS*.

---

## Import i eksport danych

---

### Cel

Zadania importu i eksportu danych w bazie danych Oracle można wykonywać przy użyciu różnych technik i narzędzi. Do standardowych rozwiązań w tym zakresie należą: *SQL Loader*, *Data Pump Import* i *Export* oraz jego starsza wersja, a także oryginalne narzędzia *Import* i *Export*. W tym rozdziale zostaną szczegółowo omówione zarówno proste jak i bardziej złożone sposoby migracji danych.

### Plan

1. Oryginalne narzędzia *Import* i *Export*.
2. Mechanizmy importu i eksportu *Data Pump*.
3. Ładowanie danych przy pomocy *SQL\*Loader*.

## 5.1. ORYGINALNE NARZĘDZIA EXPORT I IMPORT

Istnieje kilka rodzajów narzędzi wspomagających import i eksport danych. Mogą one być wykorzystane do migracji zarówno samych danych jak i całych obiektów bazy danych.

Narzędzia *Import/Export* używa się w praktyce do (Billings, Keesling, 2007):

- tworzenia kopii zapasowych i odzyskiwania (z nałożonym ograniczeniem, że eksportowane dane zajmują mniej niż 50GB);
- przenoszenia danych pomiędzy różnymi platformami Oracle;
- reorganizacji danych, w szczególności eliminacji fragmentacji danych (reorganizacja ta jest uzyskiwana po usunięciu i załadowaniu na nowo danych do bazy danych);
- migracji danych z bardzo starych wersji (takich, które nie obsługują *Database Upgrade Assistant*);
- wykrywania uszkodzeń danych, które zostają ujawnione podczas operacji importu/eksportu.

W zależności od potrzeb, operację eksportu można przeprowadzić na wszystkich obiektach bazy danych lub jedynie wybranych. Podobnie operację importu można wykonać na wszystkich wcześniej wyeksportowanych danych lub jedynie na części z nich. Do wbudowanych narzędzi migracji danych w środowisku Oracle należą: mechanizmy *Import* i *Export*, mechanizmy *Data Pump* oraz narzędzie *SQL\*Loader*.

*Import* i *Export* są starszymi wersjami narzędzia *Data Pump*, które dostępne jest od wersji Oracle 10g (Loney, 2005). Więc w praktyce ich użycie ma sens jedynie w przypadku wykonywania migracji danych z wcześniejszych wersji baz danych.

Narzędzia *Import* i *Export* wykorzystują do działania tzw. plik zrzutowy eksportu. Narzędzie *Export* umożliwia pobranie obiektów i danych z bazy danych i zapis ich do pliku binarnego (pliku zrzutowego eksportu). Plik ten zawiera polecenia tworzące struktury danych oraz wstawiające dane, takie jak *CREATE TABLE* czy *INSERT*. Eksportowane definicje struktur danych i dane mogą być importowane jedynie przez bazy danych Oracle. Eksport może objąć całą bazę danych lub tylko wybrane obiekty.

Operację tę można także ograniczyć do definicji konkretnych użytkowników. Poza danymi pobranymi z tabel można eksportować również dane ze słownika danych. W szczególności istnieje możliwość określenia, czy eksportowane mają być prawa dostępu, ograniczenia czy indeksy. Eksport można przeprowadzać na poziomie tabel oraz podzbiorów tabel.

Dane wyeksportowane przy pomocy narzędzia *Export* można następnie importować. Narzędzie *Import* wczytuje dane z pliku będącego rezultatem operacji eksportu. Polecenia z pliku zrzutowego eksportu są wykonywane we wskazanej, docelowej bazie danych lub schemacie. Istnieje możliwość wyboru między importem całkowitym a częściowym. Podczas importu całkowitego wszelkie przestrzenie tabel, pliki użytkowników czy danych są tworzone automatycznie na podstawie pliku zrzutowego. Import częściowy to operacja, podczas której importowana jest jedynie część wyeksportowanych uprzednio danych. Wymaga on przeprowadzenia przez administratora pewnych dodatkowych działań przygotowawczych. Przed rozpoczęciem importu częściowego należy samodzielnie utworzyć potrzebne struktury danych oraz zdefiniować użytkowników. Do tak przygotowanej bazy danych można bezpiecznie importować dane. Należy podkreślić fakt, że import częściowy dotyczy jedynie samych danych. Uważa się, że tego rodzaju operacja jest bezpieczniejsza niż import całkowity, ponieważ zapewnia lepszą kontrolę nad przeprowadzanymi operacjami (Loney, 2005).

### Narzędzie *Export*

Narzędzie *Export* uruchamiane jest poprzez wydanie polecenia *exp*. Przykład użycia narzędzia prezentuje listing 5.1.

*Listing 5.1. Przykład użycia narzędzia Export*

```
exp scott/shop file=users.dmp tables=(users,products)
    indexes=yes
```

Eksport może być wykonany na jednym z czterech poziomów (Loney, Bryła, 2008):

- Tryb pełny (*full*) to tryb eksportu całej bazy danych wraz z pełnym odczytem słowników danych. Plik zrzutowy zawiera polecenia tworzenia przestrzeni tabel, obiektów bazy danych, użytkowników oraz ich uprawnień.
- Tryb przestrzeni tabel (*tablespace*) to tryb, w którym eksportowi podlegają obiekty ze wskazanych przestrzeni tabel wraz z indeksami.
- Tryb użytkownika (*user*) to tryb eksportu wszystkich obiektów określonego użytkownika. Eksportowane są jedynie obiekty, indeksy oraz uprawnienia zdefiniowane dla użytkownika.
- Tryb tabeli (*table*) to eksport struktury, indeksów, uprawnień i danych pojedynczej tabeli. Same dane z tabeli mogą być eksportowane opcjonalnie. Jeśli zamiast nazwy tabeli podana zostanie nazwa właściciela schematu, wówczas wyeksportowany będzie pełny zestaw tabel należących do użytkownika. Istnieje też możliwość wskazania do eksportu konkretnych partycji tabeli.

Listę parametrów polecenia eksportu można otrzymać wykonując komendę *exp HELP=YES*. Najczęściej stosowane parametry przedstawia tabela 5.1.

Tabela 5.1. Wybrane parametry operacji eksportu

Parametr	Opis
FULL	Określenie zakresu eksportu (wartość Y – eksport całej bazy, wartość N – tylko wskazane obiekty bazy danych). Domyślna wartość – N.
TABLESPACE	Lista przestrzeni tabel przeznaczonych do eksportu.
TABLE	Lista eksportowanych tabel (możliwe jest użycie symboli wieloznacznych).
OWNER	Lista użytkowników, których schematy będą eksportowane.
USERID	Nazwa i hasło użytkownika, który wykonuje eksport.
FILE	Nazwa pliku(ów) zrzutowego.
LOG	Nazwa pliku dziennika eksportu.
TRIGGERS	Określenie możliwości eksportu wyzwalaczy (domyślnie – Y).

Parametr	Opis
CONSTRAINTS	Określenie możliwości eksportu definicji warunków integralności danych (domyślnie – Y).
QUERY	Treść warunku <i>WHERE</i> ograniczającego zakres danych z każdej eksportowanej tabeli.
PARFILE	Nazwa pliku zawierającego zestaw parametrów dla operacji eksportu.
FILESIZE	Maksymalny rozmiar pliku zrzutu. Jeśli wartość ta zostanie przekroczona, dalsza część treści eksportu zostanie przeniesiona do kolejnych plików, których nazwa określona jest parametrem FILE.
COMPRESS	Określenie, czy zdefragmentowane segmenty mają być skompresowane w jeden pojedynczy obszar (o tej samej wielkości pamięci).

Źródło: (Loney, 2005)

Przykłady wykorzystania różnych parametrów eksportu przedstawia listing 5.2.

Listing 5.2. Przykłady użycia narzędzia *Export* z różnymi parametrami

```
exp scott/shop tables=users query="where city='Lublin'"
exp scott/shop file='users.dat' full=Y
exp scott/shop file='schemat1.dat' owner=scott
exp scott/shop file=users.dmp tables=users:Partition1
```

Podczas eksportu danych istotną kwestią jest zadbanie o ich spójność. Polecenie *exp* posiada pewne mechanizmy, zapewniające zachowanie takiej spójności. Jednak tabele podczas eksportu są zapisywane pojedynczo, co niestety zwiększa prawdopodobieństwo niespójności danych, jeśli baza jest intensywnie używana (Preston, 2008). Najlepszym sposobem uniknięcia problemów ze spójnością jest eksport danych w czasie, gdy nie są wykonywane żadne ich modyfikacje. Innym sposobem jest tzw. wyciszenie bazy danych, które polega na ograniczeniu dostępu do bazy jedynie do użytkowników z uprawnieniami *SYSDBA*. Kolejną możliwością zapewniającą zachowanie spójności danych jest wykonanie eksportu z parametrem *CONSISTENT=Y*. Jest to jednak możliwe tylko w przypadku eksportu pełnego.

Niezależnie od wartości parametru *CONSISTENT* istnieje ryzyko, że baza danych nie będzie w stanie utworzyć spójnej wersji danych. Jeśli taka sytuacja będzie miała

miejsce, wówczas powstanie błąd oraz będzie wyświetlony komunikat o treści *Snapshot too old*. W celu uniknięcia takiego zdarzenia, należy tworzyć duże segmenty wycofania lub też duże przestrzenie tabel wycofania (Kuhn, 2010).

### Narzędzie *Import*

Narzędzie *Import* uruchamiane jest przy pomocy polecenia *imp*. Operacja importu realizowana jest przy wykorzystaniu pliku zrzutu, wygenerowanego podczas eksportu (Loney, 2005). W razie potrzeby import może zostać ograniczony jedynie do wybranych obiektów. Tryby pracy narzędzia *Import* są analogiczne jak narzędzia *Export*.

Przykład użycia narzędzia *Import* prezentuje listing 5.3.

*Listing 5.3. Przykład użycia narzędzia Import*

```
imp scott/shop file= users.dmp full=yes
imp scott/shop file=users.dmp fromuser=scott touser=scott
    tables=users
```

Podczas operacji importu możliwe jest określenie pewnych parametrów, które będą wpływać na sposób i zakres przeprowadzanych działań. Wybrane parametry importu przedstawiono w tabeli 5.2.

*Tabela 5.2. Wybrane parametry operacji importu*

Parametr	Opis
FULL	Określenie zakresu importu (cała baza danych – wartość Y, tylko wskazane obiekty bazy danych – wartość N). Domyślna wartość – N.
TABLESPACE	Lista przestrzeni tabel, które będą importowane.
TABLE	Lista importowanych tabel (możliwe użycie symboli wieloznacznych).
FROMUSER	Lista użytkowników, których obiekty będą importowane.
TOUSER	Lista użytkowników, których obiekty w bazie docelowej zostaną zasilone danymi wczytanymi z pliku zrzutu.



Parametr	Opis
USERID	Nazwa i hasło użytkownika, który wykonuje import.
FILE	Nazwa pliku(ów) zrzutowego, który ma być zaimportowany.
LOG	Nazwa pliku dziennika importu.
INDEXES	Określenie możliwości importu indeksów (domyślnie – Y).
CONSTRAINTS	Określenie możliwości importu definicji warunków integralności danych (domyślnie – Y).
SHOW	Wyświetlenie zawartości pliku zrzutu (domyślnie – N).
IGNORE	Ignorowanie ostrzeżeń podczas wykonywania instrukcji <i>CREATE</i> (domyślnie – N).
DESTROY	Określenie możliwości wykonywania instrukcji <i>CREATE TABLESPACE</i> podczas importu (domyślnie – N).
PARFILE	Nazwa pliku zawierającego zestaw parametrów dla operacji importu.
FILESIZE	Maksymalny rozmiar pliku zrzutu (parametr wymagany, jeśli eksport również był przeprowadzany z tą opcją).
COMMIT	Zatwierdzenie operacji importu instrukcją <i>COMMIT</i> (domyślnie – N).

Źródło: (Loney, 2005)

Zatwierdzenie importu odbywa się przy użyciu polecenia *COMMIT*. Domyślnie polecenie to jest uruchamiane automatycznie po imporcie każdej tabel. Jednak jeśli ustawi się opcję *COMMIT=Y* oraz odpowiedni rozmiar bufora (parametr *BUFFER*), wtedy zatwierdzenie importu odbywać się będzie po wstawieniu określonej w buforze ilości danych. Wyjątek stanowią kolumny *BFILE*, *LONG* i *LOB*, dla których import odbywa się po jednym wierszu. Dzięki temu nie musi być przeprowadzana kontrola, czy wartości tych typów mieszczą się w przestrzeni bufora.

Operacje eksportu i importu zazwyczaj dotyczą znacznej liczby obiektów bazy danych jak i samych danych. Fakt ten sprawia, że czas wykonania tych operacji ulega znacznemu wydłużeniu. Można jednak skorzystać z kilku wskazówek, które przyczynią się do zwiększenia wydajności poleceń *imp/exp*.

Dla operacji eksportu cel ten zostanie osiągnięty poprzez następujące działania (Alapati, 2009):

- Ustawienie dużej wartości bufora pobierania danych (parametr *BUFFER*) – powinna być ona przynajmniej tak duża, jak najdłuższy z importowanych wierszy. Jeśli wartość ta nie jest znana, można ją określić empirycznie, przyjmując początkowo wartość powyżej 100 000 bajtów.
- Ustawienie dużej długości rekordu pliku eksportu (parametr *RECORDLENGTH*) – zaleca się, aby wartość ta była wielokrotnością wartości parametru, określającego rozmiar bloku systemu operacyjnego.
- Używanie eksportu bezpośredniego (parametr *DIRECT*).
- Zatrzymanie niepotrzebnych operacji wykonywanych na bazie danych.
- Zapis równoległe wykonywanych eksportów do różnych lokalizacji fizycznych.
- Unikanie eksportu do systemu plików NTFS.

Wyższą wydajność operacji importu osiąga się poprzez wykonanie następujących czynności (Alapati, 2009):

- Tworzenie plików zawierających wszystkie polecenia *CREATE TABLE*, *CREATE CLUSTER* oraz *CREATE INDEX* (parametr *INDEXFILE*); w pliku polecenia te (poza *CREATE INDEX*) będą ujęte w komentarz, więc w razie potrzeby będzie można ich użyć.
- Umieszczenie importowanego pliku poza przestrzenią plików Oracle.
- Zwiększenie wartości rozmiaru bufora danych obszaru wspólnego dla wszystkich użytkowników bazy (*DB\_CACHE\_SIZE*). Zaleca się, aby wartość ta, w zależności od potrzeb, wynosiła od 20 do 80% pamięci dostępnej dla bazy danych.
- Ustawienie dużej wartości bufora logów w bazie (parametr *LOG\_BUFFER*) – zaleca się, aby wartość ta była wielokrotnością wartości parametru, opisującego rozmiar bloku systemu operacyjnego.
- Utworzenie tabel z dużą przestrzenią wycofania. Ich rozmiar powinien być przynajmniej tak duży, aby mógł pomieścić treść długiego zapytania. W praktyce przestrzenie wycofania są regulowane automatycznie przez system zarządzania bazą danych.
- Ustawienie parametru *COMMIT* na wartość równą N.
- Wyłączenie statystyk importu (parametr *STATISTICS* powinien mieć w takim przypadku wartość *NONE*).
- Ustanowienie pliku indeksu (*INDEXFILE*).

Odpowiednie określenie parametrów importu jest bardzo istotne. Jednak warto zwrócić uwagę na fakt, iż niektóre parametry narzędzia *Import* są ze sobą sprzeczne i ich jednoczesne użycie mogłoby zakończyć się błędem. Przykładowo, nieprawidłowe jest jednoczesne użycie parametrów *FULL=Y* oraz *TABLES=Sales*, ponieważ pierwszy z nich wskazuje na pełny import, drugi zaś – na import jedynie jednej tabeli.

## 5.2. MECHANIZMY IMPORTU I EKSPORTU DATA PUMP

*Data Pump* jest mechanizmem służącym do importu i eksportu danych po stronie serwera bazy danych Oracle. Został on wprowadzony w wersji Oracle 10g i jako nowocześniejsze rozwiązanie w zasadzie zastąpił starsze mechanizmy *Import* i *Export*. Wprawdzie są one jeszcze dostępne w nowszych wersjach Oracle, ale nie mają wsparcia dla nowych cech, które zostały wprowadzone w tychże wersjach (Loney, 2009). Oznacza to, że ich funkcjonalność jest ograniczona i są one dostępne przede wszystkim po to, aby możliwy był import danych ze starszych wersji baz danych Oracle. Dla wersji Oracle 10g i wyżej zaleca się korzystanie z mechanizmu *Data Pump*. Jednocześnie *Data Pump* może z powodzeniem korzystać z plików, które zostały wygenerowane przez narzędzie *Export*. Jednak pliki wygenerowane przez *Data Pump* nie mogą być wykorzystane przez narzędzie *Import*.

Wśród cech, które odróżniają narzędzie *Data Pump* od narzędzi *Import/Export*, można wymienić: wstrzymywanie i wznowianie zadań, podgląd statusu postępu zadań, ograniczenia importowanych lub eksportowanych danych. Ze względu na to, że *Data Pump* jest procesem klienckim, uruchamianym na serwerze, operacje importu i eksportu danych realizowane są w sposób bardziej wydajny niż w przypadku narzędzi *Import/Export*. Z polepszeniem wydajności związane są dodatkowo wprowadzone zmiany, do których należy rezygnacja z obsługi przyrostowego zatwierdzania (*COMMIT*, *BUFFER*) oraz automatycznego scalania wielu obszarów (*COMPRESS*). Kolejną różnicą jest sposób zachowania się narzędzia podczas importu danych, naruszających więzy integralności. *Data Pump* przerywa wówczas działanie w celu uniknięcia wczytania danych, które doprowadziłyby do niespójnego stanu bazy danych.

Operacje *Data Pump* uruchamia się poleceniami *expdp* i *impdp*. Większość operacji takich jak zapis i odczyt plików jest oparta o katalogi, które w systemie Oracle są tworzone poleceniem *CREATE DIRECTORY*. Listing 5.4 prezentuje przykład użycia poleceń odpowiedzialnych za tworzenie katalogu i nadających prawa dostępu do niego.

Listing 5.4. Tworzenie katalogu

```
CREATE DIRECTORY dmpdir AS '/c/dmpdir';  
GRANT read, write ON DIRECTORY dmpdir TO scott;
```

Od wersji Oracle 10g istnieje domyślny katalog (*DATA\_PUMP\_DIR*), który może być użyty jako katalog roboczy dla narzędzia *Data Pump*. Określenie jego lokalizacji możliwe jest poprzez wydanie polecenia przedstawionego na listingu 5.5.

Listing 5.5. Identyfikacja lokalizacji domyślnego katalogu dla narzędzia *Data Pump*

```
SQL> SELECT directory_path FROM dba_directories  
       WHERE directory_name = 'DATA_PUMP_DIR';  
  
DIRECTORY_PATH  
-----  
/app/oracle/product/10.2.0/rdbms/log/
```

Narzędzie *Data Pump Export* (instrukcja *expdp*) może działać w jednym z pięciu trybów (Loney, 2009):

- tryb pełny (*Full*) – eksport wszystkich danych (włączając metadane), wymaga uprawnień systemowych *EXP\_FULL\_DATABASE*;
- tryb schematu (*Schema*) – eksport danych i metadanych dla zadanych schematów użytkownika (tryb domyślny);
- tryb przestrzeni tabel (*Tablespace*) – eksport danych i metadanych dla zadanej przestrzeni tabel;
- tryb tabel (*Table*) – eksport wskazanych tabel i partycji;

- tryb przenośnej przestrzeni tabel (*Transportable Tablespace*) – eksport metadanych dla zadanej przestrzeni tabel (wymaga uprawnień systemowych *EXP\_FULL\_DATABASE*).

Zestawienie najczęściej wykorzystywanych parametrów narzędzia *Data Pump Export* przedstawiono w tabeli 5.3.

Tabela 5.3. Wybrane parametry narzędzia *Data Pump Export*

Parametr	Opis
FULL	Eksport całej bazy (wartość Y).
SCHEMAS	Lista nazw eksportowanych schematów.
TABLESPACES	Lista nazw przestrzeni tabel, które mają być eksportowane.
TABLES	Lista tabel i ich partycji przeznaczonych do eksportu.
TRANSPORT_TABLESPACES	Lista przestrzeni tabel, z których mają być eksportowane metadane.
DUMPFILE	Nazwa pliku(ów) i lokalizacja katalogów zrzutowych.
LOGFILE	Nazwa pliku (i katalogu) dziennika eksportu.
FILESIZE	Maksymalny rozmiar pliku zrzutu. Jeśli wartość ta zostanie przekroczona, dalsza część treści eksportu zostanie przeniesiona do kolejnych plików (z parametru <i>DUMPFILE</i> ).
ATTACH	Ustanowienie połączenia sesji klienckiej z istniejącym zadaniem eksportu.
INCLUDE	Definicja kryteriów dołączania obiektów do listy eksportowej.
EXCLUDE	Definicja kryteriów wyłączania obiektów z listy eksportowej.
CONTENT	Rodzaj eksportowanych danych (tylko dane – <i>DATA_ONLY</i> , tylko metadane – <i>METADATA_ONLY</i> , dane i metadane – <i>ALL</i> ).
JOB_NAME	Określenie nazwy zadania eksportu (jeśli nie podano, nazwa generowana automatycznie).
QUERY	Treść warunku <i>WHERE</i> ograniczającego zakres danych z każdej eksportowanej tabeli.
PARFILE	Nazwa pliku zawierającego zestaw parametrów dla eksportu.

Źródło: (Loney, 2009)

W momencie zlecenia zadania eksportu automatycznie nadawana jest jego nazwa, o ile nie została ona zdefiniowana przy użyciu parametru *JOB\_NAME*. Nadanie własnej nazwy zadaniu eksportu pozwala na łatwiejsze i bardziej intuicyjne zarządzanie tym procesem. Jednak nadając mu nazwę należy sprawdzić, czy nie koliduje ona przypadkiem z nazwami innych obiektów w schemacie.

Narzędzie *Data Pump* podczas wykonywania zadania eksportu pozwala użytkownikowi na wprowadzanie dodatkowych poleceń w trybie interaktywnym. Dzięki temu możliwa jest zmiana parametrów, naprawa realizowanego zadania i jego wznowienie w przypadku problemów. Tabela 5.4 prezentuje dodatkowe parametry, które mogą być wykorzystane w trybie interaktywnym.

Tabela 5.4. Dodatkowe parametry narzędzia *Data Pump Export*

Parametr	Opis
ADD_FILE	Dodanie dodatkowych plików zrzutowych.
CONTINUE_CLIENT	Zmiana trybu z interaktywnego na tryb rejestrowania.
EXIT_CLIENT	Wyjście z sesji klienckiej (sesja po stronie serwera nadal aktywna).
STATUS	Wyświetlenie statusu zadania.
START_JOB	Ponowne uruchomienie zadania.
STOP_JOB	Zatrzymanie zadania (z możliwością ponownego uruchomienia przy pomocy <i>START_JOB</i> ).
KILL_JOB	Zakończenie zadania wraz z zatrzymaniem sesji klienckich.
PARALLEL	Określenie liczby procesów dla eksportu.

Źródło: (Loney, 2009)

Instrukcja uruchamiająca zadanie eksportu (*expdp*) może zawierać, jako parametr, nazwę pliku definiującego parametry operacji eksportu (typ eksportu, zakres, nazwa pliku zrzutowego, katalog, itd.). Przykładowy rezultat wykonania takiej operacji prezentuje listing 5.6. Przedstawia on informacje o przebiegu eksportu poszczególnych głównych typów obiektów. Możliwy jest również zapis danych wynikowych do pliku, który zostanie utworzony w katalogu podanym w parametrze.

Listing 5.6. Przykład wykonania operacji Data Pump Export

```
$ expdp scott/sklep DIRECTORY=dmpdir DUMPFILE=scottdmp.dmp
CONTENT=METADATA_ONLY
Export: Release 10.2.0.1.0 - 64bit Production on Monday, 18
July, 2010 12:15:02
Copyright (c) 2003, 2005, Oracle. All rights reserved.

Connected to: Oracle Database 10g Enterprise Edition Release
10.2.0.1.0 - 64bit Production
With the Partitioning, OLAP and Data Mining options
Starting "SCOTT"."SYS_EXPORT_SCHEMA_01":
scott/***** DIRECTORY=dmpdir DUMPFILE=scottdmp.dmp
Estimate in progress using BLOCKS method...
Processing object type SCHEMA_EXPORT/TABLE/TABLE_DATA
Processing object type
  SCHEMA_EXPORT/PRE_SCHEMA/PROCACT_SCHEMA
Processing object type SCHEMA_EXPORT/TYPE/TYPE_SPEC
Processing object type SCHEMA_EXPORT/TABLE/TABLE
Processing object type SCHEMA_EXPORT/TABLE/INDEX/INDEX
Processing object type
  SCHEMA_EXPORT/TABLE/CONSTRAINT/CONSTRAINT
Processing object type
  SCHEMA_EXPORT/TABLE/INDEX/STATISTICS/INDEX_STATISTICS
Processing object type SCHEMA_EXPORT/TABLE/COMMENT
Processing object type SCHEMA_EXPORT/PACKAGE/PACKAGE_SPEC
Processing object type SCHEMA_EXPORT/PROCEDURE/PROCEDURE
Processing object type
  SCHEMA_EXPORT/PACKAGE/COMPILE_PACKAGE/PACKAGE_SPEC/ALTER_PACKAG
  E_SPEC
Processing object type
  SCHEMA_EXPORT/PROCEDURE/ALTER PROCEDURE
Processing object type SCHEMA_EXPORT/PACKAGE/PACKAGE_BODY
Processing object type
  SCHEMA_EXPORT/TABLE/CONSTRAINT/REF_CONSTRAINT
Processing object type
  SCHEMA_EXPORT/TABLE/STATISTICS/TABLE_STATISTICS
Master table "SCOTT"."SYS_EXPORT_SCHEMA_01" successfully
loaded/unloaded
*****
*****
Dump file set for SCOTT.SYS_EXPORT_SCHEMA_01 is:
/app/oracle/scott.dmp
Job "SCOTT"."SYS_EXPORT_SCHEMA_01" successfully completed at
12:17:45
```

W wynikowym pliku zrzutowym znajdują się zapisy XML, które umożliwią późniejsze odtworzenie wyeksportowanych struktur i/lub danych.

Raz uruchomione zadanie eksportu nie zostanie przerwane po zamknięciu sesji klienckiej (sesję kliencką zamyka polecenie *EXIT\_CLIENT*), ponieważ działa ono na serwerze. Istnieje możliwość ponownego podłączenia się do działającego zadania, poprzez wydanie polecenia *expdp dmpdir/att1 attach*. Po tej operacji zostanie wyświetlony status zadania i jego podstawowe parametry.

Możliwość szczegółowego wglądu w zapisy dziennika operacji oferuje polecenie *CONTINUE\_CLIENT* (listing 5.7). Umożliwia ono nie tylko przegląd zapisów dziennika podczas ich generowania, ale także pozwala na modyfikację działającego zadania.

*Listing 5.7. Polecenie CONTINUE\_CLIENT*

```
Export> CONTINUE_CLIENT
```

W przypadku konieczności wprowadzenia poważniejszych zmian w sposobie realizacji zadania eksportu (np. dołączenia dodatkowego pliku zrzutu), można je tymczasowo zatrzymać poleceniem *STOP\_JOB* (listing 5.8). Po zdefiniowaniu dodatkowego pliku zrzutu (opcja *ADD\_FILE*), wznowienie zadania następuje po wydaniu polecenia *START\_JOB* (listing 5.9).

*Listing 5.8. Polecenie STOP\_JOB*

```
Export> STOP_JOB
```

*Listing 5.9. Polecenie START\_JOB*

```
Export> START_JOB
```

*Data Pump* umożliwia także eksport danych z zewnętrznych baz danych. Do tego celu stosuje się parametr eksportu *NETWORK\_LINK*, przechowujący nazwę łącza bazy danych, która będzie eksportowana.



Istotnymi funkcjami narzędzia *Data Pump Export* są ograniczenia, które pozwalają określić dane podlegające eksportowi. Zakres eksportowanych danych definiuje się przy pomocy parametrów *EXCLUDE*, *INCLUDE* i *QUERY*. Dwa pierwsze parametry używa się podczas eksportu obiektów bazy danych, natomiast parametr *QUERY* – podczas eksportu danych.

*EXCLUDE* to opcja wyłączająca z eksportu zadane obiekty, np. tabele, indeksy, uprawnienia, warunki integralności, użytkowników czy schematy. W opcji tej definiuje się warunek, określający typ i/lub nazwę obiektu, który zostanie pominięty w trakcie eksportu. Nie jest możliwe użycie tej opcji, jeśli w tym samym poleceniu *expdp* zastosowany będzie warunek *CONTENT=DATA\_ONLY*. Sposób użycia opcji *EXCLUDE* przedstawia listing 5.10.

Listing 5.10. Składnia opcji *EXCLUDE*

```
EXCLUDE = typ_obiektu [:nazwa]  
np.: EXCLUDE = table : "'USERS'"
```

*INCLUDE* to opcja pozwalająca wskazać te obiekty (np. tabele, indeksy, uprawnienia, warunki, użytkowników, schematy), które zostaną poddane eksportowi. Określa ona warunek, wskazujący typ i/lub nazwę obiektu, który będzie eksportowany. Opcji tej nie można jednak stosować, jeśli równolegle zdefiniuje się warunek *CONTENT=DATA\_ONLY* oraz gdy aktywna jest opcja *EXCLUDE*. Sposób użycia opcji *INCLUDE* prezentuje listing 5.11.

Listing 5.11. Składnia opcji *INCLUDE*

```
INCLUDE = typ_obiektu [:nazwa]  
np.: INCLUDE = table : "IN ('USERS', 'PRODUCTS')"
```

*QUERY* to opcja umożliwiająca definicję warunku, jaki powinny spełniać wiersze, będące rezultatem eksportu. Listing 5.12 przedstawia składnię dla opcji *QUERY*.

Listing 5.12. Składnia opcji QUERY

```

QUERY = [schemat.][nazwa_tabeli:] zapytanie
np.: QUERY = SCOTT.USERS : "WHERE Miasto Like 'W%'"

```

Do importu danych, wyeksportowanych uprzednio przez *Data Pump Export*, służy narzędzie *Data Pump Import* (polecenie *impdp*). Jest to narzędzie również działające po stronie serwera, z którym użytkownik nawiązuje połączenie przy pomocy wiersza poleceń. W poleceniu *impdp* określa się plik z parametrami oraz plik interaktywnego interfejsu. Podobnie jak *Data Pump Export*, *Data Pump Import* wykorzystuje istniejące katalogi dla plików zrzutu oraz plików dziennika.

Narzędzie importu również może działać w jednym z pięciu trybów (pełnym, schematu, przestrzeni tabel, tabel lub przenośnej przestrzeni tabel). Najczęściej wykorzystywane parametry *Data Pump Import* zostały przedstawione w tabeli 5.5.

Tabela 5.5. Wybrane parametry narzędzia *Data Pump Import*

Parametr	Opis
FULL	Import całej bazy (wartość Y).
SCHEMAS	Lista nazw schematu przeznaczonych do importu.
TABLESPACES	Listę nazw importowanych przestrzeni tabel.
TABLES	Lista importowanych tabel i ich partycji.
TRANSPORT_TABLESPACES	Lista przestrzeni tabel, z których mają być importowane metadane.
DUMPFILE	Nazwa pliku(ów) i katalogów zrzutowych.
LOGFILE	Nazwa pliku (i katalogu) dziennika importu.
PARFILE	Nazwa pliku zawierającego parametry importu.
SQLFILE	Nazwa pliku(ów) zawierającego instrukcje DDL dla operacji importu.
ATTACH	Ustanowienie połączenia sesji klienckiej z istniejącym zadaniem importu (tryb interaktywny).
INCLUDE	Definicja kryteriów dołączania obiektów do importu.

Parametr	Opis
EXCLUDE	Definicja kryteriów wyłączenia obiektów z importu.
QUERY	Definicja warunku <i>WHERE</i> ograniczającego zakres importowanych danych.
CONTENT	Rodzaj importowanych danych (tylko dane – <i>DATA_ONLY</i> , tylko metadane – <i>METADATA_ONLY</i> , dane i metadane – <i>ALL</i> ).
JOB_NAME	Nazwa zadania importu (jeśli parametr nie został określony, nazwa generowana automatycznie).
PARALLEL	Liczba procesów dla operacji importu

Źródło: (Alapati, 2009)

Podobnie jak w zadaniu eksportu, *Data Pump Import* umożliwia użytkownikowi działania w trybie interaktywnym. Parametry dla tego rodzaju trybu importu są podobne do tych, które wykorzystuje się w interaktywnym trybie eksportu (tabela 5.6).

Tabela 5.6. Dodatkowe parametry narzędzia *Data Pump Import*

Parametr	Opis
CONTINUE_CLIENT	Zmiana trybu z interaktywnego na tryb rejestrowania
EXIT_CLIENT	Wyjście z sesji klienckiej (sesja po stronie serwera nadal aktywna)
STATUS	Wyświetlanie statusu zadania importu
START_JOB	Ponowne uruchomienie zadania importu
STOP_JOB	Zatrzymanie zadania importu (z możliwością ponownego uruchomienia przy pomocy <i>START_JOB</i> )
KILL_JOB	Zakończenie importu wraz z zatrzymaniem sesji klienckich
PARALLEL	Określenie liczby procesów dla zadania importu

Źródło: (Alapati, 2009)

Działania w trybie interaktywnym dla importu uruchamia się analogicznie jak dla zadania eksportu. Można zatem zatrzymać i wznowić import, wyjść lub wznowić sesję importu i sprawdzić status działania.

Import obiektów umożliwia polecenie *impdp*. Ważne jest, aby podczas wydawania

tego polecenia od razu podać wszystkie pliki zrzutowe. Istnieje możliwość importu danych do innego schematu niż ten, z którego dane eksportowano. Służy do tego opcja `REMAP_SCHEMA=schemat_źródłowy:schemat_docelowy`. W przypadku zmiany schematu właściciela dla importowanych danych, konieczne jest posiadanie uprawnień `IMP_FULL_DATABASE`. Podczas importu wyświetlana jest informacja o postępie wczytywania danych oraz tworzony jest plik logów – domyślnie w katalogu, w którym znajduje się plik zrzutowy (listing 5.13).

Jeśli importowana tabela istnieje już w schemacie docelowym, wówczas nowe dane zostaną dołączone do niej, o ile nie wybrano opcji `METADATA_ONLY`. Ustawienie to można zmienić określając opcję `TABLE_EXISTS_ACTION`.

*Listing 5.13. Przykład operacji Data Pump Import*

```
impdp system/sklep DIRECTORY=dmpdir DUMPFILE=scottdmp.dmp  
CONTENT=METADATA_ONLY REMAP_SCHEMA=sklep:sklep_nowy
```

Istnieje możliwość określenia ograniczeń importu, dotyczących obiektów bazy danych oraz danych. Do tego celu należy wykorzystać parametry `EXCLUDE`, `INCLUDE`, `QUERY`. Ich działanie jest takie samo, jak to opisane dla zadania eksportu.

Alternatywnym sposobem dla importu danych jest wygenerowanie skryptu SQL na podstawie pliku zrzutowego. Import jest możliwy dzięki opcji `SQLFILE` (listing 5.14).

*Listing 5.14. Składnia opcji SQLFILE w operacji Data Pump Import*

```
SQLFILE=[obiekt_katalogu:] nazwa_pliku
```

Wartość *obiekt\_katalogu* jest opcjonalna. Jeśli nie zostanie podana, plik zostanie utworzony w katalogu zawierającym plik zrzutowy (listing 5.15).

*Listing 5.15. Przykład działania operacji Data Pump Import z opcją SQLFILE*

```
$ impdp directory=dmpdir dumpfile= scottdmp.dmp  
sqlfile= scottdmp.sql
```

### 5.3. ŁADOWANIE DANYCH PRZY POMOCY SQL\*LOADER

*SQL\*Loader* pozwala na wprowadzanie danych do tabel z zewnętrznego skryptu (Billings, Keesling, 2007). Jest to rozbudowane narzędzie, które ładuje dane, umożliwiając jednocześnie manipulowanie nimi. Oferuje ono również możliwość podziału zbiorów danych na mniejsze części, co pozwala zwiększyć wydajność procesu migracji danych. Narzędzie to obsługuje ładowanie rozszerzonych typów danych, kolumn XML oraz typów obiektowych.

*SQL\*Loader*, do prawidłowego działania, potrzebuje przynajmniej dwóch plików: pliku z danymi do załadowania oraz pliku sterującego, zawierającego takie informacje jak formaty danych, nazwy i lokalizacje plików przechowujących dane, sposoby odczytywania i zapisywania danych, dodatkowe reguły ładowania (ograniczenia danych). Dane zawarte w pliku sterującym można też umieścić bezpośrednio w pliku z danymi.

Pliki sterujące mają charakter pliku tekstowego. Przykład jego zawartości przedstawiono na listingu 5.16.

*Listing 5.16. Przykład zawartości pliku sterującego*

```
LOAD DATA
INFILE 'produkty.dat'
INTO TABLE PRODUKTY
      (Nazwa      POSITION(01:100)  CHAR,
       Producent  POSITION(101:150) CHAR,
       Kategoria  POSITION(151:160) CHAR)
```

W przykładzie tym, do tabeli *Produkty* zostaną załadowane dane z pliku *produkty.dat*. Określenie pozycji poszczególnych danych upraszcza i przyspiesza proces ładowania danych. Niewykorzystane znaki w każdym z pól będą uzupełniane spacjami. Kolejność kolumn w tak określonym pliku sterującym nie ma znaczenia. Ładowane dane nie muszą mieć stałej długości, gdyż poszczególne wartości mogą być odróżnione od innych np. znakiem separatora lub poprzez ujęcie w znaki specjalne (listing 5.17).

Listing 5.17. Przykład zawartości pliku sterującego

```
LOAD DATA
INFILE 'produkty.dat'
BADFILE '/bad/produkty.dat'
INTO TABLE PRODUKTY
    FIELDS TERMINATED BY ";"
    (Nazwa, Producent, Kategoria)
```

Składnia polecenia *INTO TABLE* może zostać rozszerzona o klauzulę *WHEN*, określając ograniczenia (filtr) dla danych, które mają być ładowane (np. *WHEN Kategoria<10 and Kategoria>0*). Wiersze, które zostaną odrzucone, umieszczane są w pliku odrzuconych rekordów. Do ładowania można także wykorzystać sekwencje (*Sequence\_nr SEQUENCE (10,1)*).

W trakcie pracy *SQL\*Loader* generuje plik dziennika (z rozszerzeniem *.log*), przechowujący takie dane jak status operacji i dane podsumowujące, np. liczba przetworzonych wierszy. Dodatkowo tworzony jest plik tzw. "złych danych" (z rozszerzeniem *.bad*), zawierający wiersze, których nie udało się załadować z powodu różnych błędów. Określenie lokalizacji pliku *BADFILE* nie jest konieczne. Podaje się ją wówczas, gdy plik ten ma być umieszczony w katalogu innym niż ten, w którym znajduje się plik z wczytywanymi danymi. Istnieje także plik tzw. „odrzuconych rekordów” (plik z rozszerzeniem *.dsc*), w którym zapisywane są wiersze niespełniające zadanych kryteriów ładowania.

Ładowanie danych wykonuje się przy użyciu polecenia *sqlldr* z odpowiednimi parametrami (tabela 5.7). Warto jednak pamiętać, że polecenia umieszczone w pliku sterującym mają wyższy priorytet niż te wpisane w wierszu polecenia.

Tabela 5.7. Parametry narzędzia *SQL\*Loader*

Parametr	Opis
USERID	Nazwa użytkownika i hasło.
CONTROL	Lokalizacja i nazwa pliku sterującego.
BAD	Lokalizacja i nazwa pliku złych danych.

Parametr	Opis
DISCARD	Lokalizacja i nazwa pliku odrzuconych danych.
LOG	Lokalizacja i nazwa pliku dziennika.
PARFILE	Lokalizacja i nazwa pliku z dodatkowymi parametrami.
SKIP	Liczba wierszy z pliku wejściowego do pominięcia.
LOAD	Liczba wierszy z pliku wejściowego do załadowania.
ROWS	Liczba wierszy ładowanych jednorazowo (domyślnie – 64).
DIRECT	Tryb ładowania Direct Path (domyślnie FALSE).
SILENT	Wyłączenie komunikatów informacyjnych ( <i>HEADER</i> – wyłączenie nagłówka SQL*Loader, <i>FEEDBACK</i> – wyłączenie komunikatów w punktach zatwierdzania, <i>ERRORS</i> – wyłączenie rejestrowania błędnych rekordów w pliku dziennika, <i>DISCARDS</i> – wyłączenie rejestrowania odrzuconych rekordów w pliku dziennika, <i>PARTITIONS</i> – wyłączenie rejestrowania statystyk partycji w pliku dziennika, <i>ALL</i> ).
RESUMABLE	Określenie możliwości włączania i wyłączania wznawianych operacji.
PARALLEL	Określenie możliwości ładowania równoległego (domyślnie – FALSE).

Źródło: (Fernandez, 2009)

Spośród wymienionych parametrów obowiązkowym jest parametr *USERID*. Jeśli nie będzie on umieszczony w poleceniu, wówczas zostanie wyświetlony monit, przypominający o konieczności autentykacji użytkownika wykonującego operację ładowania danych. Użycie znaku ukośnika (/) oznacza wykorzystanie konta identyfikowanego zewnętrznie, tj. konta użytkownika systemu operacyjnego.

Przykładową operację z użyciem polecenia *sqlldr* przedstawiono na listingu 5.18. Wykorzystano w niej plik z danymi *produkty.txt* (listing 5.19) oraz plik sterujący *produkty.ctl* (listing 5.20).

Listing 5.18. Przykład operacji *sqlldr*

```
sqlldr userid=user1/passw control=produkty.ctl
sqlldr / control=produkty.ctl
```

Listing 5.19. Przykład pliku z danymi (plik *produkty.txt*)

```
Produkt1.producentA.1  
Produkt2.producentA.12
```

Listing 5.20. Przykład pliku sterującego (plik *produkty.ctl*)

```
LOAD DATA  
INFILE 'produkty.txt'  
BADFILE '/bad/produkty.bad'  
DISCARDFILE 'produkty.dsc'  
APPEND  
INTO TABLE PRODUKTY  
    WHEN Producent IS NOT NULL  
    FIELDS TERMINATED BY ";"  
        (Nazwa, Producent, Kategoria)
```

Jeśli operacja ładowania danych zakończy się sukcesem, wówczas do tabeli *Produkty* zostaną dodane dwa rekordy. Zastosowanie opcji *APPEND* spowoduje dodanie tych wierszy do już istniejących w tabeli. Dostępne są także opcje: *INSERT* (wprowadzanie wierszy do pustej tabeli), *REPLACE* (wiersze zostaną wstawione do tabeli po uprzednim usunięciu dotychczas istniejących danych) oraz *TRUNCATE* (usunięcie istniejących rekordów, bez kontroli integralności danych, a następnie załadowanie nowych danych). W przytoczonym przykładzie wskazano również plik *produkty.bad*, jako miejsce zapisu rekordów, które nie zostały załadowane, gdyż ich wstawienie naruszyłoby reguły integralności, oraz plik *produkty.dsc*, w którym znajdują się rekordy odrzucone ze względu na niespełnienie przez nie warunku określonego po klauzuli *WHEN*.

Operacje ładowania mogą zakończyć się niepowodzeniem z wielu różnych powodów, takich jak problemy z formatem danych, niezgodność typu danych lub naruszenie reguł integralności danych. *SQL\*Loader* domyślnie kontynuuje operację ładowania, pomijając błędne lub odrzucone wiersze, które trafiają odpowiednio do plików złych danych oraz odrzuconych danych. Dane z tych plików można jednak wykorzystać podczas ponownego ładowania, dokonując uprzednio koniecznych zmian wartości w polach tych rekordów. Jednak im więcej błędów tego rodzaju zostanie



zarejestrowanych, tym łatwiej się pomylić, zgubić dane lub załadować je dwa razy. W celu kontrolowania procesu ładowania warto korzystać z takich opcji jak (Hart, Freeman, 2008): *errors* (przerwanie operacji ładowania po wystąpieniu określonej liczby błędów) lub *discardmax* (przerwanie operacji ładowania po wystąpieniu określonej liczby odrzuceń). Można także skorzystać z opcji *skip*, pozwalającej na pominięcie określonej liczby operacji z początku pliku z danymi. Jeśli zajdzie taka potrzeba, ładowanie można zawiesić, a następnie je wznowić (opcje *resumable*, *resumable\_name*, *resumable\_timeout*).

Ładowanie dużych plików wsadowych nie pozostaje bez wpływu na wydajność bazy danych. Warto w tym miejscu zwrócić uwagę na pewne zasady, które pozwolą uniknąć znacznego obciążenia bazy danych realizacją procesu wprowadzania danych (Kuhn, 2010):

- Operacje ładowania należy w miarę możliwości przeprowadzać w okresach zmniejszonej (lub najlepiej żadnej) aktywności bazy.
- Podczas ładowania danych powinno się ustawić bazę w tryb *NOARCHIVELOG* (bez archiwizacji plików dziennika powtórzeń), o ile nie są przeprowadzane na niej inne transakcje.
- Po powrocie do trybu *ARCHIVELOG* powinno się wykonać kopię zapasową bazy danych (przełączenie trybów wymaga restartu instancji bazy danych).
- Ładowanie można przeprowadzić z parametrem *UNRECOVERABLE*, wstrzymującym zapis dziennika powtórzeń, co jednak uniemożliwia późniejsze odtworzenie danych.
- Należy prowadzić kontrolę zapisów w dzienniku powtórzeń dla bloków (pojedyncze tabele, partycje).
- Zaleca się zrównoleglenie operacji ładowania (parametr *PARALLEL*).
- Należy pamiętać o dokonaniu odpowiedniego przydziału zasobów dyskowych i pamięciowych dla operacji ładowania.
- Zaleca się wyłączenie zbędnych ograniczeń (więzów integralności) dla tabel i wyzwalaczy dla operacji DML.
- Ładowanie powinno odbywać się w trybie Direct Path, który przed załadowaniem spowoduje utworzenie wstępnie sformatowanych bloków danych. Proces importu w tym trybie będzie realizowany wydajniej niż w przypadku braku tego parametru.

- Zaleca się podział tabeli na partycje, co umożliwi wykonanie kilku równoległych ładowań do pojedynczych partycji.

Podsumowując opis narzędzia *SQL\*Loader* należy podkreślić, iż oprócz importowania danych numerycznych, znakowych czy typu daty, umożliwia ono także wprowadzanie danych o bardziej złożonej strukturze jak dane XML, dane typu obiektowego, a także dane znakowe z uwzględnieniem obsługi Unicode.

#### 5.4. PYTANIA KONTROLNE

1. W jakich obszarach wykorzystuje się oryginalne narzędzia *Import/Export*.
2. Wymień i scharakteryzuj poziomy eksport w oryginalnym narzędziu *Export*.
3. Wskaż różnice pomiędzy *Data Pump Import Export* a oryginalnymi wersjami narzędzi *Import* i *Export*. Podaj przykłady użycia tych narzędzi.
4. Wymień i scharakteryzuj tryby pracy narzędzi *Data Pump Import Export*.
5. Do czego służy narzędzie *SQL\*Loader* i jakie operacje można wykonywać przy jego pomocy?
6. Do czego służy polecenie *CREATE DIRECTORY*?
7. Podaj najważniejsze polecenia narzędzia *Data Pump Export*.

---

## Optymalizacja dostępu do danych

---

### Cel

Celem niniejszego rozdziału jest prezentacja podstawowych błędów popełnianych podczas tworzenia zapytań w języku SQL oraz wskazanie sposobów, umożliwiających eliminowanie tych błędów w celu skrócenia czasu wykonywania zapytań i zwiększenia wydajności pracy całego systemu bazodanowego.

### Plan

1. Znaczenie strojenia zapytań w SQL.
2. Plan wykonania zapytania.
3. Optymalizatory.
4. Optymalizacja składni zapytań.

## 6.1. ZNACZENIE STROJENIA ZAPYTAŃ SQL

W eksploatacji współczesnych baz danych jednym z najczęściej realizowanych działań jest tworzenie i wykonywanie zapytań w języku SQL. Jest to operacja, która z punktu widzenia użytkownika końcowego, przynosi najbardziej widoczne efekty w postaci zbioru danych, niezbędnych do wygenerowania raportów czy przeprowadzenia analiz. Zazwyczaj użytkownicy wymagają, aby czas oczekiwania na te dane był możliwie krótki. O długości tego czasu decyduje wiele czynników, które powinny być uwzględniane na etapie zarówno projektowania bazy danych jak i późniejszej jej eksploatacji.

Jednym z istotnych aspektów, wpływających na wydajność pracy bazy danych, jest strojenie zapytań SQL, tzn. tworzenie zapytań w takiej postaci, która zapewni optymalne jego wykonanie. Celem strojenia jest osiągnięcie takiego stanu, w którym pożądaný wynik zapytania zostanie osiągnięty przy zużyciu minimalnej ilości zasobów środowiska bazy danych oraz czas uzyskania tego wyniku będzie możliwie najkrótszy dla danej konfiguracji bazy danych (Price, 2009).

Strojenie danego zapytania SQL nie jest procesem jednoetapowym, wykonywanym jednorazowo. Działanie to można porównać do długotrwałego procesu wprowadzania zmian w istniejącym zapytaniu, a niekiedy wręcz pisania polecenia od nowa tak, aby za każdym razem otrzymać kod bardziej optymalny od poprzedniego.

Strojenie zapytań SQL obejmuje działania, które można podzielić na dwie kategorie. Pierwsza kategoria obejmuje czynności optymalizujące składnię zapytania, druga – czynności związane z efektywnym wykorzystaniem architektury bazy danych. Użytkownik bazy danych ma największe możliwości strojenia zapytania poprzez wpływanie na jego postać np. poprzez ograniczanie listy kolumn w instrukcji *SELECT*, sposób zapisu warunków wyboru rekordów czy łączenia tabel. Niezbędnym uzupełnieniem optymalizacji składni zapytania powinno być także (Tow, 2004):

- wykorzystanie indeksów i funkcji haszujących;

- wykorzystanie PL/SQL do przesyłania wielu zapytań do serwera bazy danych (blok PL/SQL, zawierający zapytania, przesyłany jest jako całość, w przeciwnym wypadku każde zapytanie przesyłane jest oddzielnie);
- przechowywanie zapytań w procedurach składowanych po stronie serwera (ich wykonanie zmniejsza ruch sieciowy i zwiększa prawdopodobieństwo, że dana procedura znajduje się już w obszarze wspólnym, a przez to zostanie szybciej wykonana);
- wykorzystanie pakietów, w których umieszczono procedury z definicją zapytań (zwiększa to wydajność systemu, ponieważ pakiety są ładowane w całości przy pierwszym ich wywołaniu);
- wykorzystanie sekwencji buforowanych do generowania kluczy głównych w celu polepszenia sprawności generowania wartości tych kluczy;
- efektywne wykorzystanie przestrzeni dyskowej poprzez stosowanie typu danych *VARCHAR2* zamiast *CHAR*, dzięki czemu zyskuje się na końcowych spacjach;
- korzystanie z tzw. wskazówek, które pozwalają na przekazanie wiedzy użytkownika dotyczącej zawartości tabel do optymalizatora.

Wymienione powyżej różne aspekty, związane z wykonywaniem zapytań, pozwalają zrozumieć, jak wiele różnych czynników wpływa na czas otrzymania zbioru wyników. W dalszej części rozdziału uwaga zostanie skupiona na czynnościach, które pozwolą uzyskać możliwie najlepszą składnię zapytania, zapewniającą optymalne jego wykonanie.

Aby poznać i zrozumieć techniki optymalizacji składni zapytania, należy wiedzieć, w jaki sposób polecenia te są wykonywane w środowisku Oracle. Realizacja każdego polecenia SQL odbywa się w następujących etapach:

1. Utworzenie kursora – system tworzy kursor o strukturze zgodnej ze strukturą danych występującą w poleceniu.
2. Parsowanie zapytania – jeśli polecenie o podanej składni nie było dotychczas wykonywane, wówczas system dokonuje jego parsowania, które polega na wykonaniu następujących czynności:
  - kontrola poprawności składniowej (nazwy słów kluczowych, ich kolejność);
  - sprawdzenie poprawności odwołań do obiektów, których nazwy zostały wymienione w poleceniu;

- założenie blokad na obiekty, wykorzystywane w poleceniu;
- kontrola praw użytkownika, wydającego polecenie, do działań na podanych obiektach;
- wyznaczenie planu wykonania polecenia;
- umieszczenie polecenia w obszarze wspólnym.

Jeśli natomiast w obszarze wspólnym istnieje polecenie o strukturze zgodnej ze strukturą utworzonego kursora, wówczas ponowne parsowanie nie jest wykonywane. Jednak wykorzystanie rezultatów wcześniejszych parsowań poleceń SQL wymaga spełnienia następujących warunków:

- tekst bieżącego polecenia musi być identyczny z tekstem polecenia uprzednio sparsowanego;
  - aktualne polecenie musi się odwoływać do tych samych obiektów, do których odwoływało się polecenie już sparsowane;
  - zmienne wiązane muszą nazywać się tak samo i być tych samych typów;
  - parsowanie polecenia bieżącego i historycznego odbywa się w tym samym trybie optymalizatora Oracle (dla optymalizatora kosztowego konieczne jest zachowanie tego samego celu optymalizacji).
3. Przygotowanie dodatkowych zasobów do wykonania zapytania – obejmuje działania, które zapewnią:
    - spójność odczytów danych podczas wyboru danych;
    - zapewnienie przestrzeni pamięci dyskowej (segmentów tymczasowych) do wykonania ewentualnych operacji sortowania, konwersji, złączenia podczas wykonywania zapytania.
  4. Podwiązanie zmiennych do zapytania – program wywołujący powinien utworzyć te zmienne, a do Oracle podać ich adresy.
  5. Właściwe wykonanie polecenia, tj. pobranie danych z bazy danych.
  6. Przesyłanie rezultatów do użytkownika, który pobiera w pętli kolejne rekordy ze zbioru wyników.

Z punktu widzenia optymalizacji zapytań istotna jest także znajomość procesów zachodzących podczas właściwego wykonywania polecenia, a mianowicie metod dostępu do rekordów oraz metod łączenia zbiorów rekordów.

Wśród metod dostępu do zbiorów rekordów należy wymienić:

- metodę *full scan*,
- metodę *rowid access*,
- metodę *lookup index*,
- metodę *hash key access*.

*Full scan* to metoda, w której dostęp do rekordów odbywa się w sposób liniowy. Oznacza to, że podczas określania zbioru wynikowego każdy rekord tabeli jest pobierany i sprawdzany czy spełnia warunek zdefiniowany w klauzuli *WHERE*.

W metodzie *rowid access* dostęp do rekordów odbywa się z wykorzystaniem ich unikalnego identyfikatora, zawierającego informację o lokalizacji każdego rekordu. Identyfikator ten wskazuje na plik oraz blok danych, w których znajduje się dany rekord oraz na położenie rekordu wewnątrz bloku. Ze względu na tak dużą precyzję opisu położenia rekordu, metoda ta uznawana jest za najszybszą.

Metoda *index lookup* opiera się na wykorzystaniu indeksu w procesie selekcji rekordów. Każda wartość, znajdująca się w indeksie, jest opisana przez identyfikatory rekordów, które zawierają tę wartość w kolumnie tworzącej strukturę indeksu. Podczas wykonania zapytania najpierw przeglądany jest indeks w poszukiwaniu zadanej wartości, a następnie realizowany jest dostęp do rekordów, zawierających tę wartość we odpowiedniej kolumnie.

Metoda *hash key access* wykorzystuje tzw. klaster haszujący. Jest to obiekt bazy danych, utworzony w określonej przestrzeni tabel i zawierający bloki danych. W blokach tych znajdują się rekordy, którym przypisano tę samą wartość funkcji haszującej. Poszukiwana wartość jest przekształcana przez tę funkcję, a wynik haszowania jest porównywany z wartością, charakteryzującą blok danych. Na podstawie równości tych wartości określany jest blok danych, zawierający wynikowy zbiór rekordów.

Optymalne wykonanie zapytania uzależnione jest także od wybranej metody łączenia zbiorów rekordów, tj. tabel bazy danych lub wyników zapytań. W tym obszarze stosuje się:

- metodę *nested loop*,
- metodę *sort merge*,
- metodę *hash join*.

*Nested loop* jest najczęściej i domyślnie stosowaną metodą. W złączeniu tego rodzaju

jedna z tabel jest tzw. tabelą sterującą. Rekordy z tej tabeli są pobierane w pierwszej kolejności, a dopiero na ich podstawie dobierane są rekordy z drugiej tabeli, poprzez porównywanie wartości w kolumnach łączących.

Metoda *sort merge* realizowana jest dwuetapowo. W pierwszym etapie w każdym zbiorze odbywa się sortowanie rekordów według kolumny, której użyto w warunku złączenia. Drugi etap to odpowiednie złączenie rekordów z posortowanych zbiorów na podstawie równości wartości w kolumnie łączącej.

Ostatni rodzaj złączenia, tzw. *hash join*, wykorzystuje pamięć RAM do utworzenia w niej tzw. tabeli haszującej, zawierającej wartości z kolumny łączącej mniejszego zbioru, odpowiednio przetworzone przez funkcję haszującą. Wartości te są porównywane z wartościami kolumny łączącej z drugiego zbioru, przekształconymi przez tę samą funkcję haszującą. Równość tych wartości jest podstawą złączenia odpowiednich rekordów.

Tak szczegółowa znajomość kolejnych procesów, zachodzących od chwili wydania polecenia do momentu otrzymania wynikowego zbioru danych pozwala lepiej zrozumieć metody optymalizacji, które będą przedstawione w dalszej części rozdziału.

## 6.2. PLAN WYKONANIA ZAPYTANIA

Znajomość zasad właściwego konstruowania zapytań i ich stosowanie pozwalają szybciej uzyskać wynikowy zbiór danych. Jednak warto pamiętać o pewnego rodzaju kontroli, czy faktycznie kod SQL został poprawnie zoptymalizowany. Zauważalne skrócenie czasu odpowiedzi nie może spowodować zaniechania poszukiwania dalszych możliwości ulepszenia kodu.

W środowisku ORACLE istnieje polecenie *EXPLAIN PLAN*, które pozwala wyświetlić plan wykonania poleceń *SELECT*, *INSERT*, *UPDATE* oraz *DELETE*, wybrany przez optymalizator. Analiza takiego planu, uzupełniona znajomością charakteru gromadzonych danych, pozwala stwierdzić, czy optymalizator wybrał właściwy plan dla danego polecenia.

Instrukcja *EXPLAIN PLAN* jest wykonywana dla polecenia SQL, będącego jej parametrem. Przykład jej użycia oraz otrzymany wynik prezentuje listing 6.1.



Listing 6.1. Przykład użycia instrukcji EXPLAIN PLAN oraz planu wykonania

```
SQL> EXPLAIN PLAN FOR
2  SELECT DISTINCT s.Id_student, Nazwisko, Imie
3  FROM Studenci s
4  INNER JOIN Egzaminy e ON s.Id_student = e.Id_student ;

Plan wykonywania
-----
0      SELECT STATEMENT Optimizer=CHOOSE
1      0      SORT (UNIQUE)
2      1      NESTED LOOPS
3      2      TABLE ACCESS (FULL) OF 'EGZAMINY'
4      2      TABLE ACCESS (BY INDEX ROWID) OF 'STUDENCI'
5      4      INDEX (UNIQUE SCAN) OF 'ENY_ID_ST_PK' (UNIQUE)
```

Alternatywnym sposobem śledzenia sposobu wykonania poleceń jest wykonanie instrukcji *SET AUTOTRACE ON*, która spowoduje wyświetlanie planu oraz dodatkowo statystyk dla każdej z wyżej wymienionych operacji (listing 6.2).

Listing 6.2. Przykładowy rezultat uzyskany po zastosowaniu instrukcji SET AUTOTRACE ON

```
SQL> SELECT Id_student, Nazwisko, Imie
2  FROM Studenci
3  WHERE Id_student NOT IN (SELECT DISTINCT Id_student
4                          FROM Egzaminy);

Plan wykonywania
-----
0      SELECT STATEMENT Optimizer=CHOOSE
1      0      FILTER
2      1      TABLE ACCESS (FULL) OF 'STUDENCI'
3      1      TABLE ACCESS (FULL) OF 'EGZAMINY'

Statystyki
-----
0 recursive calls
0 db block gets
103 consistent gets
0 physical reads
0 redo size
813 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)
5 rows processed
```

Tworzenie planu wykonania wymaga istnienia tabeli systemowej o nazwie *PLAN\_TABLE*, w której przechowywane są rezultaty działania polecenia *EXPLAIN PLAN*. Tabelę tę można utworzyć, uruchamiając skrypt o nazwie *utlxplan.sql*, znajdujący się w folderze *\$ORACLE\_HOME/rdbms/admin/*. Dane zawarte w tej tabeli pozwalają uzyskać informacje m.in. na temat:

- indeksów wykorzystywanych podczas wykonywania zapytań;
- metod dostępu do tabel (*full scan*, *rowid access*, *index lookup*, *hash key access*);
- metod łączenia tabel (*sort merge*, *nested loop*, *hash join*).

Dzięki tym informacjom użytkownik może stwierdzić, czy zapytania odpowiednio wykorzystują istniejące indeksy, czy przeszukiwanie tabel odbywa się w efektywny sposób oraz która tabela jest tzw. tabelą sterującą w definicji złączenia.

Generowanie planu wykonania i posługiwanie się nim należy zaliczyć do kategorii najistotniejszych czynności, które umożliwią podjęcie działań optymalizujących kod SQL. Znajomość takiego planu stwarza również możliwość ingerowania przez użytkownika w sposób działania wbudowanego w system narzędzia, tzw. optymalizatora, które będzie wykorzystywane w trakcie wykonywania zapytań do bazy danych.

### 6.3. OPTIMALIZATORY

Optymalizator to narzędzie wykorzystywane do tworzenia najbardziej optymalnego planu wykonania zapytania SQL. Jego działanie polega na określeniu sekwencji czynności, które będą realizowane w celu uzyskania wynikowego zbioru danych. Podczas konstruowania planu wykonania optymalizator może wykorzystywać uprzednio zebrane dane statystyczne, dotyczące obiektów występujących w poleceniu.

W trakcie określania najlepszego sposobu wykonania zapytania, optymalizator bierze pod uwagę (Whalen, Schroeter, 2003):

- składnię polecenia SQL;
- warunki wyboru rekordów (klauszula *WHERE*);
- tabele, z których będą pobierane dane lub będą użyte do wykonania złączeń;
- indeksy, które mogą być wykorzystane do pobrania danych z tabeli;
- wskazówki polecenia SQL;
- statystyki dotyczące obiektów, używanych podczas wykonywania polecenia.

Efektywność wykonania zapytania zależy głównie od wybranej metody optymalizacji. Wyróżnia się dwa podstawowe rodzaje optymalizatorów: regułowy i kosztowy. Optymalizator regułowy wyznacza wszystkie możliwe plany wykonania, nadaje im odpowiednie rangi według przyjętego w systemie rankingu i wybiera plan z najniższą rangą. Wybór najlepszej ścieżki dostępu do danych opiera się tutaj wyłącznie na analizie składni polecenia oraz preferowaniu indeksów w procesie wybierania danych. Natomiast nie uwzględnia się w żaden sposób wielkości tabel, tzn. czy są one duże czy małe oraz statystyk dotyczących tabel i indeksów. Te cechy, a właściwie wady, optymalizatora regułowego spowodowały, że nie jest on już powszechnie wykorzystywany we współczesnych systemach baz danych.

Obecnie do wyznaczenia optymalnego planu wykonania zapytań stosuje się optymalizator kosztowy, który jest narzędziem bardziej zaawansowanym w porównaniu do uprzednio opisanego. Główną jego zaletą jest posługiwanie się statystykami zgromadzonymi w metadanych, dotyczącymi (Kawa, 2003):

- tabel – liczba wierszy w tabeli, liczba wykorzystanych i pustych bloków danych, średnia długość rekordu, średnia zajętość bloku danych;
- indeksów – liczba bloków liści, przypadająca na jedną wartość klucza, głębokość indeksu typu B-drzewo, liczba jednakowych wartości klucza (tzw. selektywność indeksu).

Odejście od sztywnych zasad wyboru planu wykonania, jakie obowiązywały w optymalizatorze regułowym, oraz uwzględnianie aktualnego charakteru gromadzonych danych, pozwoliło uzyskać wyższą efektywność zapytań. Należy jednak pamiętać, że to użytkownik, poprzez indywidualną konstrukcję składni zapytania (np. stosowane operatory porównania, kolumny w warunku *WHERE*),

wpływa bezpośrednio na czas otrzymania wynikowego zbioru danych, a optymalizator poszukuje jedynie najlepszego sposobu wykonania tak zdefiniowanego zapytania.

Jak już uprzednio wspomniano, optymalizator kosztowy korzysta z informacji statystycznych, dotyczących m.in. tabel. Uzyskanie tych informacji wymaga wykonania polecenia *ANALYZE* w odniesieniu do wskazanego obiektu (listing 6.3) lub wykorzystania pakietu *DBMS\_STATS*, znajdującego się w pliku *dbmsstat.sql* w folderze *\$ORACLE\_HOME/rdbms/admin*.

Listing 6.3. Przykłady użycia instrukcji *ANALYZE*

```
ANAYLZE TABLE Osrodki COMPUTE STATISTICS FOR TABLE ;
ANALYZE TABLE Egzaminy COMPUTE STATISTICS FOR ALL COLUMNS ;
ANALYZE TABLE Egzaminy COMPUTE STATISTICS FOR ALL INDEXES ;
```

Zebrane statystyki będą wyświetlane podczas prezentacji planu wykonania zapytania. Przykład takiego planu, zawierającego dodatkowe informacje statystyczne, prezentuje listing 6.4.

Listing 6.4. Przykład rezultatu stosowania statystyk dla tabel

```
SQL> SELECT DISTINCT o.Id_osrodek, Nazwa_o
  2 FROM Egzaminy e
  3 INNER JOIN Osrodki o ON o.Id_osrodek = e.Id_osrodek ;

Plan wykonywania
-----
  0      SELECT STATEMENT Optimizer=CHOOSE (Cost=7 Card=85
        Bytes=3655)
  1  0      SORT (UNIQUE) (Cost=7 Card=85 Bytes=3655)
  2  1      HASH JOIN (Cost=5 Card=85 Bytes=3655)
  3  2          TABLE ACCESS (FULL) OF 'OSRODKI' (Cost=2
        Card=15 Bytes=450)
  4  2          TABLE ACCESS (FULL) OF 'EGZAMINY' (Cost=2
        Card=85 Bytes=1105)
```

Na jego podstawie można dowiedzieć się, jaki rodzaj optymalizatora został wybrany do utworzenia planu (*OPTIMIZER*), jaki był koszt wykonania danej operacji

(*Cost*), jaka liczba rekordów została przetworzona w tej operacji (*Card*) oraz jaka była wielkość tych rekordów (*Bytes*).

Analizując plan wykonania zapytania przedstawiony na listingu 6.4 można stwierdzić, że:

- zapytanie zostało wykonane według planu wygenerowanego przez optymalizator kosztowy (*OPTIMIZER = CHOOSE*);
- całkowity koszt wykonania zapytania wynosił 7 jednostek, koszt operacji sortowania – 7 jednostek, koszt operacji złączenia – 5 jednostek, a koszt dostępu do tabeli *Osrodki* – 2 jednostki;
- podczas wykonania instrukcji *SELECT* przetworzono 85 rekordów, w operacji dostępu do tabeli *Osrodki* – 15 rekordów, a do tabeli *Egzaminy* – 85 rekordów;
- rozmiar rekordów, przetworzonych instrukcją *SELECT*, wynosił 3655 bajtów, podczas dostępu do tabeli *Osrodki* – 450 bajtów, a do tabeli *Egzaminy* – 1105 bajtów.

Tak otrzymane statystyki mają jednak pewną wadę. Mianowicie, nie informują one o rozkładzie statystycznym danych w poszczególnych kolumnach tabeli. Taka informacja byłaby użyteczna dla użytkownika podczas konstruowania warunku wyboru rekordów (wybór właściwego operatora porównania), a także dla samego optymalizatora, który mógłby wybrać lepszą metodę dostępu do danych w tabeli (np. *index lookup* zamiast *full scan*). Ponadto należy pamiętać o ciągłej aktualizacji statystyk oraz wykonywaniu tej operacji zarówno dla tabel jak i indeksów.

Wyznaczenie optymalnego planu wykonania zapytania, z użyciem optymalizatora kosztowego, odbywa się w następujących etapach:

1. Wygenerowanie zbioru możliwych planów (podobnie jak to ma miejsce w trybie regułowym).
2. Określenie kosztu realizacji każdego planu na podstawie dostępnych statystyk. Na ten koszt składa się czas procesora, liczba operacji wejścia-wyjścia i wielkość pamięci, niezbędnej do wykonania zapytania.
3. Porównanie otrzymanych kosztów i wybór tego planu, którego koszt realizacji jest najmniejszy.

W trakcie tych działań optymalizatora przyjmuje się następujące zasady (Whalen, Schroeter, 2003):

- Równość indeksu opartego na kluczu głównym i indeksu unikatowego – podstawą tej równości jest selektywność indeksu unikatowego (utworzonego poleceniem zawierającym klauzulę *UNIQUE*), która wynosi 100%. Dostęp do danych z użyciem takiego indeksu jest najszybszy. Zatem będzie on wykorzystany zawsze, gdy tylko jest utworzony dla danej tabeli.
- Równomierność indeksu nieunikatowego – optymalizator przyjmuje założenie o równomiernym rozłożeniu danych w indeksie (pod warunkiem, że w poleceniu *ANALYZE* nie użyto opcji *FOR ALL INDEXED COLUMNS*). Następnie oblicza selektywność dla takiego indeksu, która pozwala określić ewentualną asymetrię rozkładu wartości w indeksowanych kolumnach.
- Obliczenie zakresu wartości indeksów następuje poprzez określenie selektywności tych danych. Operacja ta wykorzystuje wartości maksymalne i minimalne ostatnio utworzonych statystyk kolumn. Wartości te można poznać, odwołując się odpowiednio do kolumn *HIVAL* oraz *LOWVAL* obiektu *SYS.HIST\_HEAD\$*.
- Aktualne zasoby systemowe są wykorzystywane tylko przez użytkownika, który wydaje zapytanie do bazy. Jest to jednak założenie bardzo restrykcyjne, gdyż w praktyce taka sytuacja rzadko ma miejsce. Potencjalne możliwości lepszego wykorzystania zasobów oferuje pakiet *DBMS\_STATS*, zawierający procedurę *GATHER\_SYSTEM\_STATS*.
- Aktualność statystyk – zakłada się, że statystyki dotyczące tabel oraz skojarzonych z nią indeksów są aktualne, tzn. otrzymano je w wyniku jednoczesnej analizy tabeli oraz jej indeksów (aktualna statystyka dla tabeli i nieaktualna dla indeksu nie zapewnią osiągnięcia celu optymalizatora kosztowego).

W celu uwzględnienia wpływu charakteru danych na plan wykonania, użytkownik ma możliwość definiowania tzw. wskazówek dla optymalizatora. Są to pewne klauzule, umieszczane w definicji polecenia, które pozwalają wymusić określony plan wykonania, inny od tego, który mógłby zaproponować optymalizator. Wskazówki optymalizatora mogą dotyczyć następujących obszarów (Nyffenegger, 2010):

- trybu pracy i celu optymalizacji np. *ALL\_ROWS*, *FIRST\_ROWS*, *CHOOSE*;

- sposobu dostępu do danych np. *FULL*, *HASH*, *CLUSTER*, *ROWID*, *INDEX*, *INDEX\_JOIN*, *INDEX\_COMBINE*;
- kolejności łączenia tabel np. *ORDERED*, *LEADING*;
- sposobu łączenia zbiorów wierszy np. *DRIVING\_SITE*, *USE\_HASH*, *USE\_MERGE*, *USE\_NL*;
- sposobu transformacji zapytania np. *FACT*, *MERGE*, *NO\_EXPAND*, *REWRITE*, *STAR\_TRANSFORMATION*, *USE\_CONCAT*;
- równoległego przetwarzania danych np. *PARALLEL*, *PARALLEL\_INDEX*, *PQ\_DISTRIBUTE*;
- dodatkowych działań np. *APPEND*, *BITMAP*, *BUFFER*, *CACHE*, *STAR*.

Wymienione powyżej przykłady wskazówek optymalizatora obrazują, jak wiele dodatkowych możliwości poprawy efektywności wykonania zapytania posiada użytkownik. Przykład zastosowania wybranych klauzul prezentuje listing 6.5.

Listing 6.5. Przykład zastosowania wskazówki optymalizatora

```
SQL> SELECT /*+ ROWID(p) FULL(e) */ DISTINCT Nazwa_p
2 FROM Przedmioty p
3 INNER JOIN Egzaminy e
4 ON e.Id_przedmiot = p.Id_przedmiot ;
```

Skutkiem użycia, w prezentowanym przykładzie, wskazówki *ROWID* będzie realizacja dostępu do rekordów tabeli *Przedmioty* z wykorzystaniem metody *rowid access*. Natomiast wskazówka *FULL* wymusza użycie metody typu *full scan* podczas dostępu do tabeli *Egzaminy*.

Stosowanie tego rodzaju wskazówek pozwala wykorzystać wiedzę użytkownika na temat danych w procesie optymalizacji zapytań. Przykładowo, podczas eksploracji danych w tabeli zawierającej miliardy rekordów, użytkownik może zdecydować, poprzez wskazówkę *PARALLEL* czy *PARALLEL\_INDEX*, aby zrównoleglić działania podczas wykonywania zapytania. Optymalizator ma dostęp do informacji o liczbie rekordów w tabeli, ale nie posiada wiedzy, czy jest to duża czy mała tabela.

Podczas korzystania z optymalizatora kosztowego użytkownik powinien mieć świadomość występowania pewnych jego ograniczeń, które będą wpływały na

efektywność wykonania zapytania. Ograniczenia te związane są z:

- asymetrią rozkładu danych,
- brakiem aktualnych statystyk tabel,
- niespójnością między statystykami tabel i statystykami odpowiadających im indeksów,
- wyborem niewłaściwego indeksu.

Problem asymetrii danych wynika z faktu, iż domyślnie optymalizator nie posiada informacji ile jest rekordów posiadających daną wartość w indeksowanej kolumnie. W związku z tym przyjmuje założenie o równomiernym rozkładzie każdej z wartości, która występuje w takiej kolumnie (tzn. każda wartość występuje w takiej samej liczbie rekordów). Takie założenie powoduje, że podczas selekcji rekordów przeglądana jest cała tabela wiersz po wierszu. Jeśli jednak zostanie wykonane polecenie *ANALYZE* z parametrem *FOR ALL INDEXED COLUMNS* lub uruchomiony będzie pakiet *DBMS\_STATS*, wówczas w systemie generowane będą histogramy rozkładu poszczególnych wartości kolumny. Mogą one być wykorzystane do wybrania właściwego indeksu tabeli, zamiast przeglądania jej w trybie *FULL*.

Innym ograniczeniem optymalizatora jest korzystanie z nieaktualnych statystyk tabel. Optymalizator, korzystając z takich statystyk, zakłada, że są one aktualne i dotyczą bieżącej zawartości tabeli. Jeśli jednak administrator bazy danych nie wykonuje regularnych analiz tabel (a także indeksów), wówczas uniemożliwia optymalizatorowi wygenerowanie najlepszego planu wykonania zapytania. Problem ten dotyczy szczególnie tych tabel, których wielkość i zawartość często ulega zmianie ze względu na charakter przetwarzania danych np. tabel tymczasowych. Jeśli, na podstawie istniejących statystyk, optymalizator będzie posiadał informację o małej liczbie rekordów w takiej tabeli, wówczas podejmie decyzję o liniowym dostępie do jej wierszy nawet wtedy, gdy tabela ta będzie zawierała nowy zestaw wierszy, który będzie bardziej liczny od poprzedniego. Dla uniknięcia problemu braku aktualności statystyk, wskazane jest regularne korzystanie z pakietu *DBMS\_STATS* lub polecenia *ANALYZE*, np. umieszczając ich wywołania jako zadania bazy danych (ang. *jobs*).

Z problemem aktualności statystyk związane jest zagadnienie dotyczące braku spójności pomiędzy statystykami tabel i statystykami indeksów. Domyślnie pakiet *DBMS\_STATS* umożliwia generowanie statystyk jedynie dla tabel (alternatywnie



można użyć polecenia *ANALYZE*). Fakt ten powoduje, że zachodzi wysokie prawdopodobieństwo wystąpienia niespójności informacji pochodzących ze statystyk tabel oraz indeksów. Brak statystyk dla indeksów (nie tylko aktualnych, ale w ogóle) sprawia, że optymalizator wybierze nieoptymalny plan wykonania. Rozwiązaniem tego problemu jest posługiwanie się, w procedurach *GATHER\_SCHEMA\_STATS* oraz *GATHER\_TABLE\_STATS* pakietu *DBMS\_STATS*, parametrem *CASCADE*, któremu należy nadać wartość równą *TRUE*. Działanie takie wymusza tworzenie statystyk dla wszystkich indeksów skojarzonych z tabelą wraz z każdorazowym tworzeniem statystyk tabeli.

Problem opisanej tu niespójności występuje również wtedy, gdy analizie statystycznej poddaje się wyłącznie indeksy, bez równoczesnego analizowania zawartości tabel. Można, za pomocą procedury *GATHER\_INDEX\_STATS* tego samego pakietu, tworzyć statystyki wyłącznie dla indeksów. Jednak ta operacja musi być uzupełniona wygenerowaniem statystyk dla tabel.

Ostatnim prezentowanym problemem, związanym z pracą optymalizatora kosztowego, jest wybór niewłaściwego indeksu. Sytuacja taka może mieć miejsce, jeśli po klauzuli *WHERE* zdefiniowano warunek zawierający wiele kolumn, a optymalizator wybiera indeks odwołujący się tylko do kilku z nich (czyli do indeksu podrzędnego) zamiast do indeksu, zawierającego wszystkie kolumny. Rozwiązaniem tego problemu jest jawne wskazanie indeksu, który powinien być użyty, poprzez wskazówki optymalizatora.

Znajomość znaczenia optymalizatora planu wykonania, zasad, jakimi się kieruje podczas wyznaczania takiego planu i sposobów wpływania na jego działanie pozwoli w dalszej części rozdziału zaprezentować problemy, jakie pojawiają się podczas tworzenia niewłaściwej składni zapytań oraz ich rozwiązania.

## 6.4. OPTYMALIZACJA SKŁADNI ZAPYTAŃ

Tworząc zapytania w języku SQL wiele osób zwraca szczególną uwagę wyłącznie na ich poprawność składniową. Niekiedy pojawiają się w nich wręcz elementy „proceduralne” wskazujące, jakie operacje mają być kolejno wykonywane. Takie

podejście jest niezgodne z charakterem języka SQL, którego instrukcje *SELECT*, *INSERT*, *UPDATE* i *DELETE* służą do określania wynikowego zbioru danych, a nie do określania sposobu wyboru tych danych.

Poniżej zostaną przedstawione przykłady błędnych konstrukcji zapytań SQL oraz wskazane zostaną poprawne postacie instrukcji, zapewniające optymalny plan ich wykonania. Prezentowane tutaj treści mają na celu uświadomienie Czytelnikowi faktu, że posiada on realny wpływ na zwiększenie efektywności wykonywania zapytań już w fazie ich definiowania. Podane tutaj rozwiązania mogą być stosowane nie tylko na platformie ORACLE, ale również w innych środowiskach bazodanowych.

Najczęściej popełniane błędy, dotyczące postaci zapytania, można podzielić na następujące kategorie:

- operowanie na zbyt dużych zbiorach danych,
- brak możliwości wykorzystania indeksów,
- nieefektywne wykorzystanie istniejących indeksów,
- zbędne operacje sortowania,
- inne działania.

Każda z tych kategorii błędów zostanie omówiona w sposób szczegółowy w celu uwidocznienia konkretnych problemów.

Podstawowym czynnikiem wpływającym na długi czas wykonania poleceń eksploracji jest ilość danych, które mają być przetworzone i przekazane do użytkownika jako rezultat zapytania. Problemy w tej kategorii związane są z nadmiernie rozbudowaną listą kolumn w poleceniu *SELECT* oraz używaniem w klauzuli *WHERE* kolumny o niskiej selektywności wartości. W pierwszym przypadku, o ile jest to możliwe, należy unikać zapisu listy kolumn w postaci przedstawionej na listingu 6.6, tj. z zastosowaniem symbolu "\*". Taka konstrukcja zapytania doprowadziła w przykładzie m.in. do dużej liczby dodatkowych żądań dostępu do danych (*recursive calls*), niezbędnych do otrzymania zbioru wynikowego.

*Listing 6.6. Przykład nadmiernej długości listy kolumn i statystyka wykonania polecenia*

```
SQL> SELECT * FROM Egzamin ;
```

Statystyki	
-----	
82	recursive calls
0	db block gets
21	consistent gets
0	physical reads
0	redo size
8373	bytes sent via SQL*Net to client
1158	bytes received via SQL*Net from client
7	SQL*Net roundtrips to/from client
0	sorts (memory)
0	sorts (disk)
85	rows processed

Alternatywnym rozwiązaniem jest jawna lista tych kolumn, które są niezbędne do wyświetlenia (listing 6.7). Można zauważyć, że podanie listy wszystkich kolumn tabeli będzie lepszym zapisem od użycia symbolu "\*". W przedstawionym przykładzie, dzięki takiemu rozwiązaniu, nastąpiło zmniejszenie liczby *recursive calls* z 82 do zera oraz liczby spójnych odczytów bloków danych z obszaru SGA (*consistent gets*) z 21 do 13.

Listing 6.7. Przykład poprawnego zapisu listy kolumn i statystyka wykonania polecenia

SQL> SELECT Nr_egz, Id_student, Id_osrodek, Id_egzaminator, 2 Id_przedmiot, Data_egz, Zdal FROM Egzamin;	
Statystyki	
-----	
0	recursive calls
0	db block gets
13	consistent gets
0	physical reads
0	redo size
8373	bytes sent via SQL*Net to client
1158	bytes received via SQL*Net from client
7	SQL*Net roundtrips to/from client
0	sorts (memory)
0	sorts (disk)
85	rows processed

Innym przykładem działania na zbyt dużym zbiorze danych jest użycie w warunku

wyboru rekordów kolumny, która przyjmuje wiele tych samych wartości. Niska selektywność takiego warunku zwiększa liczbę operacji odczytu z indeksu, co może spowodować, że optymalizator zrezygnuje z jego zastosowania.

Kolejna kategoria błędów związana jest z brakiem możliwości wykorzystania indeksów podczas selekcji rekordów, które mają spełniać warunek określony klauzulą *WHERE*. Najczęściej popełniane błędy w tym obszarze dotyczą operacji na kolumnach, które tworzą strukturę indeksu. Obejmują one:

- użycie niepoprawnych operatorów porównania,
- wykonywanie działań arytmetycznych,
- przeprowadzanie konwersji danych (niejawnej lub jawnej),
- stosowanie funkcji wbudowanych na kolumnach.

W języku SQL można wyróżnić dwie grupy operatorów porównania, z punktu widzenia możliwości wykorzystania indeksów w procesie porównywania wartości. Pierwsza grupa określana jest mianem *SARGABLE* (ang. *Search ARGument ABLE*) i obejmuje następujące operatory: *=*, *>*, *>=*, *<*, *<=*, *BETWEEN*, *LIKE* 'ciąg\_znaków%'. Umożliwiają one optymalizatorowi wykorzystanie indeksu, podnosząc tym samym efektywność wykonania zapytania.

W drugiej grupie operatorów, tzw. *NONSARGABLE*, należy wymienić: *<>*, *!=*, *!>*, *!<*, *NOT EXISTS*, *NOT IN*, *IN*, *OR*, *NOT LIKE*, *LIKE* '%ciąg\_znaków%'. Operatory *NOT IN*, *NOT EXISTS*, *IN*, *EXISTS* często są używane alternatywnie do łączenia tabel w zapytaniach wykorzystujących podzapytania. Praktyka pokazuje jednak, że tradycyjne złączenia tabel (z użyciem operatora *JOIN*) są wydajniejsze od stosowania operatorów *IN* czy *NOT IN* oraz podzapytań. Wynika to z faktu, iż tradycyjne złączenia tabel wykorzystują indeksy, podczas gdy dostęp do tabel w podzapytaniu odbywa się w trybie *FULL*. W przypadku konieczności stosowania podzapytań bardziej efektywnym rozwiązaniem będzie użycie operatora *EXISTS* zamiast *IN*, gdyż plan wykonania jest sterowany przez tabele pochodzące z zewnętrznej instrukcji *SELECT*. Obecność operatora *IN* powoduje, że najpierw przetwarzane jest podzapytanie, a dopiero jego wynik jest łączony z każdym wierszem zwróconym przez zapytanie zewnętrzne. Jednak i tu istnieje pewne odstępstwo od tej zasady, wynikające z charakteru tabel. Otóż, jeśli tabela w podzapytaniu zawiera małą liczbę wierszy, a zapytanie główne korzysta z tabeli o dużej liczbie wierszy, wówczas

powinno się użyć wskazówek optymalizatora *HASH\_SJ*, *MERGE\_SJ* oraz *NL\_SJ*. Takie działanie pozwala zmienić charakter złączenia tych zapytań i traktować podzapytanie jak odrębną tabelę, z której wszystkie wiersze zostaną zwrócone w jednej operacji odczytu danych.

Alternatywnym rozwiązaniem dla podzapytania jest także stosowanie złączenia tabel, używanych w zapytaniu nadrzędnym i podrzędnym. Analiza statystyki wykonania zapytania z listingu 6.8 wskazuje na znaczną liczbę odwołań rekursywnych (*recursive calls* = 477), wydłużających czas jego wykonania, dużą liczbę spójnych odczytów danych z obszaru SGA (*consistent gets* = 108) oraz konieczność 7 odczytów bloków danych z dysku (*physical reads* = 7).

Listing 6.8. Przykład użycia podzapytania i statystyka wykonania zapytania

```
SQL> SELECT Id_student, Nazwisko, Imie
2  FROM Studenci
3  WHERE Id_student NOT IN (
4  SELECT Id_student FROM Egzaminy) ;
```

Statystyki

```
-----
477 recursive calls
0 db block gets
108 consistent gets
7 physical reads
0 redo size
813 bytes sent via SQL*Net to client
503 bytes received via SQL*Net from client
2 SQL*Net roundtrips to/from client
6 sorts (memory)
0 sorts (disk)
5 rows processed
```

Zmodyfikowana postać zapytania (listing 6.9), w której użyto złączenia rozszerzającego typu *LEFT JOIN*, pozwoliła wyeliminować tego typu odwołania oraz zmniejszyć liczbę koniecznych odczytów.

Listing 6.9. Przykład poprawnego zastosowania złączenia table zamiast podzapytania

```
SQL> SELECT s.Id_student, Nazwisko, Imie
2  FROM Studenci s
3  LEFT JOIN Egzaminy e
4  ON s.Id_student = e.Id_student
4  WHERE Data_egz IS NULL ;

Statystyki
-----
          0  recursive calls
          0  db block gets
         14  consistent gets
          0  physical reads
          0  redo size
        813  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)
           5  rows processed
```

Innym przykładem braku możliwości wykorzystania indeksu będzie użycie operatora w postaci *LIKE* '%ciąg\_znaków%'. Umieszczenie symbolu % na początku wzorca ciągu znaków powoduje, że dostęp do danych będzie odbywał się przy użyciu metody full scan, czyli w sposób liniowy – wiersz po wierszu.

Kolejnym działaniem, którego należy unikać, jest wykonywanie operacji arytmetycznych z użyciem indeksowanej kolumny. Listing 6.10 prezentuje przykład, w którym dokonuje się konkatencji wartości z kolumn *Imie* i *Nazwisko*, a następnie wynik tego działania jest porównywany z poszukiwaną wartością.

Listing 6.10. Przykład błędnego zapisu operacji porównania na indeksowanych kolumnach

```
SQL> SELECT Id_student, Imie || ' ' || Nazwisko
2  FROM Egzaminy
3  WHERE Imie || ' ' || Nazwisko = 'Piotr Muryjas';
```

Poprawna postać zapisu warunku została przedstawiona na listingu 6.11.

*Listing 6.11. Przykład poprawnego zapisu operacji porównania na indeksowanych kolumnach*

```
SQL> SELECT Id_student, Imie || ' ' || Nazwisko
2  FROM Egzaminy
3  WHERE Imie = 'Piotr' AND Nazwisko = 'Muryjas';
```

Jednym z powodów uniemożliwiającym korzystanie z indeksu jest stosowanie funkcji konwersji danych podczas selekcji rekordów (listing 6.12). W prezentowanym przykładzie przed dokonaniem właściwego porównania, decydującego o wyborze danego rekordu, przeprowadzana jest operacja konwersja wartości, w celu zapewnienia zgodności typu danych. Poprawna postać zapytania, w której takowej konwersji się nie wykonuje, została przedstawiona na listingu 6.13.

*Listing 6.12. Przykład błędnego użycia funkcji konwersji na indeksowanej kolumnie*

```
SQL> SELECT DISTINCT Id_student, Id_przedmiot
2  FROM Egzaminy
3  WHERE TO_CHAR(Data_egz) > '26-12-1991' ;
```

*Listing 6.13. Przykład poprawnego użycia funkcji konwersji w klauzuli WHERE*

```
SQL> SELECT DISTINCT Id_student, Id_przedmiot
2  FROM Egzaminy
3  WHERE Data_egz > TO_DATE('26-12-1991', 'DD-MM-YYYY') ;
```

W konstrukcji warunku wyboru rekordów nie należy także stosować funkcji wbudowanych, których parametrem będzie indeksowana kolumna (listing 6.14). Każde dodatkowe działanie na takiej kolumnie jest źródłem wydłużenia czasu odpowiedzi. Dlatego też poprawny zapis takiego zapytania powinien wyglądać tak, jak na listingu 6.15.

Listing 6.14. Przykład błędnego użycia funkcji na indeksowanej kolumnie

```
SQL> SELECT Id_student, Imie, Nazwisko  
2 FROM Studenci  
3 WHERE SUBSTR(Id_student,1,2) = '05' ;
```

Listing 6.15. Przykład poprawnego zapisu zapytania z listingu 6.14

```
SQL> SELECT Id_student, Imie, Nazwisko  
2 FROM Studenci  
3 WHERE Id_student LIKE '05%' ;
```

Omówione dotychczas przykłady eliminowały możliwość wykorzystania indeksów podczas wykonywania zapytania. Jednak nawet konstrukcje zapytań, które pozwalają optymalizatorowi użyć indeksu, nie zawsze mają optymalną postać. Wśród błędów nieefektywnego wykorzystania indeksu należy wymienić:

- stosowanie operatorów *IN* lub *OR* zamiast *BETWEEN*,
- zbyt mała liczba znaków we wzorcu w operatorze *LIKE*,
- posługiwanie się indeksami jednokolumnowymi zamiast stosowanie indeksów wielokolumnowych.

Zapis warunku w klauzuli *WHERE* przy pomocy operatorów *IN* lub *OR* powoduje, że odczyt wartości z indeksu odbywa się dla każdej porównywanej wartości (listing 6.16). Wydłużenie czasu odpowiedzi spowodowane jest wielokrotnym wykonywaniem tej samej operacji porównywania z użyciem różnych wartości.

Listing 6.16. Przykład niepoprawnego zapisu warunku z wykorzystaniem operatora *IN*

```
SQL> SELECT DISTINCT Id_student, Id_przedmiot  
2 FROM Egzaminy  
3 WHERE Id_przedmiot IN (1,2,3,4,5) ;
```

Poprawny zapis warunku powinien zawierać operator *BETWEEN*, dla którego odczyt indeksu będzie wykonany jednorazowo (listing 6.17).



Listing 6.17. Przykład poprawnego zapisu warunku z wykorzystaniem operatora *BETWEEN*

```
SQL> SELECT DISTINCT Id_student, Id_przedmiot  
2   FROM Egzaminy  
3   WHERE Id_przedmiot BETWEEN 1 AND 5 ;
```

Inny przykład nieefektywnego wykorzystania indeksu to zbyt krótki ciąg znaków we wzorcu poszukiwanych wartości, zdefiniowanym dla operatora porównania *LIKE*. Zbyt mała liczba znaków w takim wzorcu powoduje, że liczba odwołań do tabeli i jej indeksu znacznie wzrasta, wydłużając czas odpowiedzi. Jeśli tylko jest to możliwe, należy szczegółowo precyzować wzorzec, aby w ten sposób efektywnie wykorzystać istniejący indeks.

Wśród często popełnianych błędów, związanych z nieumiejętnym stosowaniem indeksów, należy wymienić posługiwanie się indeksami jednokolumnowymi. Dostęp do danych w takim przypadku polega na dokonywaniu odczytu z kolejnych indeksów, wydłużając w ten sposób czas oczekiwania na odpowiedź. Zwielokrotnieniu operacji odczytu poszczególnych indeksów można zapobiec używając wskazówki */\*+ AND\_EQUAL \*/* dla optymalizatora (listing 6.18) bądź też definiując indeks wielokolumnowy, w którym odczyt taki odbywa się tylko jeden raz dla wybieranych wartości.

Listing 6.18. Przykład poprawnego zastosowania indeksów jednokolumnowych

```
SQL> SELECT /*+ AND_EQUAL(o, ikod, imiasto)*/  
2   Id_osrodek, Nazwa_o, Kod, Miasto  
3   FROM Osrodki o  
4   WHERE Kod = '20-618' AND Miasto = 'Lublin';
```

Zapytanie przedstawione na listingu 6.18 pozwala wykonać optymalne złączenie dwóch indeksów jednokolumnowych o nazwach *ikod* oraz *imiasto*, zamiast przeprowadzania pojedynczych odczytów z każdego z nich. Alternatywnym rozwiązaniem będzie zdefiniowanie dla tabeli *Osrodki* indeksu złożonego, opartego na kolumnach *Miasto* oraz *Kod*.

Kolejną kategorią błędów konstrukcji zapytania będzie zbędne sortowanie, które znacznie wpływa na koszt wykonania całego zapytania. Operacja ta jest wykonywana dla poleceń *DISTINCT*, *GROUP BY*, *ORDER BY*, *INTERSECT*, *MINUS*, *UNION* oraz złączeń nieindeksowanych tabel. Przykładowo, operacja *UNION* wymusza sortowanie i scalenie wszystkich wierszy, zwróconych przez różne zapytania w nim umieszczone tak, aby możliwe było odrzucenie wszelkich duplikatów, zanim do modułu wywołującego zostanie zwrócony pierwszy wiersz z grupy powtarzających się rekordów. Zmniejszenie kosztu takiego zapytania może nastąpić poprzez użycie *UNION ALL*, kiedy to rezultat zapytania nie jest sortowany. Jednak w tym przypadku w zbiorze wynikowym pojawiają się duplikaty, co może okazać się nie do zaakceptowania przez użytkownika.

Innym sposobem uniknięcia sortowania jest korzystanie z indeksów. Domyślnie przechowują one wartości w porządku rosnącym. Jednak przewidując wykorzystanie w praktyce danych w kolejności malejącej, warto stworzyć indeks z tak określonym porządkiem. Podstawowym założeniem możliwości wykorzystania indeksu do sortowania danych, zamiast klauzuli *ORDER BY*, jest obecność w indeksie tych samych kolumn, które pojawiłyby się po *ORDER BY*. Wymuszenie użycia indeksu może nastąpić poprzez wskazówkę */\*+ INDEX \*/* dla optymalizatora bądź też przez definicję warunku *WHERE* z wykorzystaniem indeksowanej kolumny, który zawsze jest spełniony. Obecność *WHERE* w zapytaniu spowoduje, że optymalizator będzie korzystał z istniejącego indeksu i w ten sposób rekordy będą zwrócone w uporządkowanej kolejności. Dla takiego zapytania nie jest już konieczne posługiwanie się klauzulą *ORDER BY*.

Podsumowując prezentację metod zwiększenia efektywności zapytań SQL należy jeszcze wspomnieć o wpływie rodzaju złączeń dwóch zbiorów wierszy na czas wykonania zapytania. Praktyka wskazuje, że użycie metody *nested loop* ma uzasadnienie, jeśli zapytanie może zwrócić mniej niż 10% wszystkich wierszy. W innych przypadkach powinno się wykorzystać złączenie haszujące (metoda *hash join*) lub złączenie przez sortowanie i scalanie (metoda *sort merge join*).

Złączenie haszujące należy zastosować, jeśli jeden z przetwarzanych zbiorów wierszy jest znacznie większy od drugiego. Ze względu na tworzenie tabeli haszującej dla mniejszego zbioru, należy pamiętać o zapewnieniu odpowiedniej ilości wolnej

pamięci komputera, w której możliwe będzie utworzenie takiej tabeli.

Zastosowanie złączenia haszującego uwarunkowane jest przede wszystkim rodzajem operatora porównania. Może ono być wykorzystane tylko dla złączeń opartych na operatorze „=”. W przypadku użycia w klauzuli *WHERE* operatorów „<”, „<=”, „>” i „>=” nastąpi złączenie przez sortowanie i scalenie, które może podnieść koszt wykonania zapytania, jeśli kolumny, których wartości są porównywane, nie występują w indeksie. W takiej sytuacji wiersze ze wszystkie łączonych zbiorów danych (np. tabel) podlegają sortowaniu. Obecność indeksu eliminuje konieczność takiego sortowania, gdyż wiersze są pobierane z tabeli z wykorzystaniem ROWID, znajdującego się w indeksie.

Złączenie przez sortowanie i scalenie odbywa się również wtedy, gdy operatorem porównania jest „=”. Jest ono bardziej efektywne od złączenia haszującego, jeśli istnieje już uprzednio posortowany zbiór wierszy, np. w wyniku zdefiniowania indeksu opartego na kolumnie, występującej w kryterium wyboru rekordu.

Decyzja o domyślnym wyborze sposobu złączenia należy do optymalizatora, który na podstawie rodzaju operatora porównania, wielkości tabel i struktury indeksów wybiera najlepszy sposób złączenia. Jednak użytkownik może sam zdecydować o typie złączenia, wykorzystując do tego celu wskazówki optymalizatora *USE\_NL*, *USE\_HASH* oraz *USE\_MERGE*.

Zaprezentowane w rozdziale zagadnienia pozwalają wykazać wpływ konstrukcji składni zapytania SQL na czas jego wykonania. Przykłady błędnych i poprawnych zapytań mogą stanowić wskazówki dla osób zamierzających podnieść efektywność działania aplikacji bazodanowej. Omówione metody ingerencji w sposób działania optymalizatora planu wykonania pozwolą uzyskać dodatkową korzyść w postaci uwzględnienia charakteru danych w procesie optymalizacji wykonania zapytania.

## 6.5. PYTANIA KONTROLNE

1. Przedstaw sposób wykonania zapytania w języku SQL.
2. Wymień i scharakteryzuj metody dostępu do danych.
3. Wymień i scharakteryzuj metody łączenia zbiorów rekordów.

4. Na czym polega strojenie zapytań w SQL?
5. Co to jest plan wykonania zapytania i jakie informacje on zawiera?
6. Na czym polega praca optymalizatora planu wykonania i jakie informacje on wykorzystuje?
7. Podaj przykłady wskazówek dla optymalizatora zapytań.
8. Wymień i scharakteryzuj sposoby optymalizacji składni zapytań.

## Kopie zapasowe i archiwizacja danych

---

### Cel

Celem rozdziału jest przedstawienie zagadnień związanych z tworzeniem kopii zapasowych i odzyskiwaniem danych. Przedstawione zostały rodzaje kopii zapasowych, w tym kopie tworzone online i offline. Scharakteryzowano także narzędzie *RMAN*, dedykowane do wykonywania archiwizacji i odzyskiwania danych.

### Plan

1. Rodzaje kopii zapasowych. Kopie online i offline.
2. Archiwizacja z wykorzystaniem narzędzia *RMAN*.

## 7.1. RODZAJE KOPII ZAPASOWYCH. KOPIE ONLINE I OFFLINE

Regularne wykonywanie kopii zapasowych aktywnej bazy danych jest koniecznością. Jest to podstawowy sposób wspomagający ochronę danych i zapobiegający ich utracie. Dzięki aktualnej kopii zapasowej można przywrócić utracone obiekty bazy danych oraz same dane. Ponadto, jeśli w bazie znajdują się stare, rzadko używane rekordy, to warto rozważyć ich archiwizację, czyli przeniesienie z bieżącej bazy do archiwum.

Zagadnienia tworzenia kopii zapasowych są skomplikowane i wymagają posiadania rozległej wiedzy oraz doświadczenia ze strony administratora. Powinien on opracować i przetestować odpowiednie procedury, które zapewnią poprawność przebiegu i rezultatu procesu archiwizacji oraz późniejszego odtwarzania kopii zapasowych.

Istnieją cztery sposoby tworzenia kopii zapasowych (Loney, 2005):

- tworzenie kopii logicznych przy użyciu narzędzi *Data Pump*,
- tworzenie kopii logicznych przy pomocy narzędzi *Import* i *Export*,
- tworzenie kopii fizycznych plików bazy danych dla bazy zamkniętej (offline) i otwartej (online),
- tworzenie przyrostowych kopii fizycznych przy pomocy narzędzia *RMAN*.

Mechanizmy *Data Pump* oraz jego starsze odpowiedniki (*Import* i *Export*) zostały już omówione w rozdziale 5. W tym miejscu zaprezentowane zostaną dwie pozostałe techniki archiwizacji.

Tworzenie fizycznych kopii plików może odbywać się w momencie, gdy baza danych jest zamknięta bądź też podczas jej bieżącego działania, czyli gdy jest otwarta. Kopie generowane na bazie zamkniętej (tzw. kopie offline), określane także jako zimne kopie, są pełnym obrazem bazy danych z momentu jej zamknięcia. Zawierają one wszystkie pliki danych, pliki kontrolne, pliki dziennika powtórzeń oraz opcjonalnie plik parametrów.

Bezpieczne utworzenie tak dokładnego obrazu wymaga całkowitego zamknięcia bazy danych. Można do tego celu użyć poleceń: *SHUTDOWN NORMAL*,

*SHUTDOWN IMMEDIATE* lub *SHUTDOWN TRANSACTIONAL*. Zamknięcie bazy umożliwi utworzenie jej wiernego obrazu, bez konieczności wprowadzania dodatkowych zabezpieczeń, np. przed aktualizacją danych ze strony potencjalnie zalogowanych użytkowników. Główną zaletą tego rodzaju kopii jest możliwość pełnego przywrócenia stanu bazy danych i szybkiego wznowienie jej poprawnego działania.

Ponieważ kopie zapasowe powinno się wykonywać często, warto określić wcześniej zasady dotyczące przechowywania plików kopii, tj. ich lokalizacji oraz jednolitej struktury katalogów. Takie podejście ułatwi w przyszłości szybkie odtworzenie potrzebnych danych po awarii.

W celu utworzenia kopii offline należy (Alapati, 2009):

1. Określić miejsce składowania plików kopii i rozmiar przestrzeni dyskowej, niezbędnej do ich przechowywania.
2. Określić lokalizację i nazwy plików do skopiowania.
3. Wyłączyć bazę danych, stosując jedną z opcji: *IMMEDIATE*, *TRANSACTIONAL*, *NORMAL*.
4. Wykonać fizyczne kopiowanie plików w systemie operacyjnym.
5. Ponownie uruchomić bazę danych.

Odtworzenie kopii polega na skopiowaniu uprzednio zarchiwizowanych plików do oryginalnych lokalizacji. Po zakończeniu kopiowania można otworzyć bazę danych z opcją *OPEN RESETLOGS*. Tworzenie kopii offline gwarantuje uzyskanie pełnego i dokładnego obrazu bazy danych z chwili jej zamknięcia. Obraz taki może być w łatwy sposób przywrócony w przypadku wystąpienia awarii bazy danych.

Innym rodzajem kopii jest tzw. kopia online. Jest ona tworzona w sytuacji, kiedy nie jest możliwe zamknięcie bazy danych, np. ze względu na konieczność ciągłej jej eksploatacji. Kopię tę tworzy się, gdy baza jest w trybie *ARCHIVELOG*, w którym pliki dziennika powtórzeń archiwizowane są na bieżąco, dzięki czemu możliwy jest zapis wszystkich przeprowadzonych transakcji. Tryb ten można określić dla całej bazy lub dla pojedynczych tabel. Procesy bazy, działającej w trybie archiwizacji, wykonują kopię każdego dziennika powtórzeń przed jego nadpisaniem. W normalnym trybie pracy dzienniki powtórzeń są nadpisywane cyklicznie, tzn. napełniane są kolejno, a po zapełnieniu ostatniego zapis dokonywany jest ponownie w pierwszym pliku dziennika.

Podczas pracy w trybie *ARCHIVELOG* pliki danych są archiwizowane przed ich nadpisaniem. Tworzenie tego rodzaju kopii obejmuje wszystkie pliki danych, archiwalne pliki dziennika powtórzeń oraz plik kontrolny.

Odtworzenie stanu spójnego bazy z kopii online polega na załadowaniu plików danych oraz ponownym wykonaniu operacji, zarejestrowanych w plikach dziennika powtórzeń. Dane z dziennika można załadować w całości lub jedynie do wybranego momentu czasowego.

Wykonanie kopii online odbywa się w następujących etapach:

1. Uruchomienie bazy danych w trybie *ARCHIVELOG* (listing 7.1).

*Listing 7.1. Uruchomienie bazy danych w trybie ARCHIVELOG*

```
STARTUP MOUNT baza1;  
ALTER DATABASE ARCHIVELOG;
```

2. Otworzenie bazy danych (listing 7.2).

*Listing 7.2. Otworzenie bazy danych*

```
ALTER DATABASE OPEN;
```

3. Utworzenie pliku parametrów, określających m.in. katalog docelowy kopii (*log\_archive\_dest\_1*) oraz wzorzec nazw plików (w przykładzie nazwy plików mają postać: *arch\_1.arc*, *arch\_2.arc*,...). W każdym pliku zapisana będzie zawartość jednego dziennika powtórzeń (listing 7.3).

*Listing 7.3. Plik parametrów*

```
log_archive_dest_1=/db1/oracle/arch  
log_archive_dest_state_1=ENABLE  
log_archive_format=arch_%s.arc
```

W definicji parametru *log\_archive\_format* można używać następujących znaków specjalnych:



- %s – numer kolejny dziennika,
  - %S – numer kolejny dziennika dopełniony zerami,
  - %t – numer wątku,
  - %T – numer wątku dopełniony zerami,
  - %d – identyfikator bazy danych,
  - %r – identyfikator zerowania dzienników.
4. Upewnienie się, że nie zostaną nadpisane wcześniejsze kopie zapasowe. W tym celu należy sprawdzić ich lokalizację i w razie potrzeby przenieść te pliki w inne miejsce.
  5. Skontrolowanie wartości parametrów, przy pomocy polecenia zamieszczonego na listingu 7.4.

Listing 7.4. Sprawdzenie parametrów

```
show parameter log_archive_dest
```

NAME	TYPE	VALUE
log_archive_dest	string	
log_archive_dest_1	string	location=/db1/oracle/arch

```
SELECT dest_name, destination, status, binding
FROM v$archive_dest;
```

DEST_NAME	DESTINATION	STATUS	BINDING
LOG_ARCHIVE_DEST_1	/db1/oracle/arch	VALID	OPTIONAL
LOG_ARCHIVE_DEST_2		INACTIVE	OPTIONAL

6. Jawne przywrócenie bazy danych do trybu pracy *NOARCHIVELOG*, po utworzeniu kopii (listing 7.5).

Listing 7.5. Przywrócenie trybu NOARCHIVELOG

```
STARTUP MOUNT baza1;
ALTER DATABASE NOARCHIVELOG;
ALTER DATABASE OPEN;
```

Wykonywanie kopii zapasowych typu online należy przeprowadzać w okresach najniższej aktywności użytkowników, gdyż realizacja operacji wejścia-wyjścia znacznie absorbuje system operacyjny i tym samym zmniejsza jego wydajność. Poza tym w trybie *ARCHIVELOG* zapisywane są na dysku wszystkie, znajdujące się w buforze danych, bloki wchodzące w skład aktualnie używanej przestrzeni tabel. Bloki te są kopiowane do bufora dziennika dopiero w momencie przeprowadzenia zmiany. Natomiast dane nie są kopiowane do dziennika powtórzeń dopóki znajdują się w buforze, co powoduje zwiększenie użycia pamięci serwera.

Przed rozpoczęciem procesu archiwizacji należy określić strategię zarządzania kopiami zapasowymi typu online. Strategia taka powinna obejmować m.in. konwencję nazewnictwa, częstotliwość tworzenia kopii i ich rodzaj oraz wskazanie miejsca ich przechowywania. Warto także sprawdzić, czy na dysku, przeznaczonym do zapisu kopii, jest wystarczająca ilość wolnej przestrzeni (należy pamiętać, że archiwalne pliki zazwyczaj mają duży rozmiar, a dodając kolejne łatwo jest zapełnić całą dostępną przestrzeń dyskową). Jeśli w trakcie archiwizacji zabraknie miejsca we wskazanej lokalizacji, wówczas proces ten zostanie wstrzymany. Wznowienie kopiowania danych nastąpi automatycznie po zapewnieniu odpowiedniej ilości wolnego miejsca na docelowym nośniku danych.

W celu określenia domyślnych obszarów przechowywania kopii zapasowych Oracle umożliwia zdefiniowanie *FRA* (ang. *Flash Recovery Area*). Jest to specjalny obszar na dysku, określany w parametrach bazy danych (Hart, Freeman, 2008). Wskazuje on domyślne miejsce składowania takich plików jak archiwa logów, kopie zapasowe *RMAN*, dzienniki powtórzeń czy pliki kontrolne. Utworzenie *FRA* wymaga podania dwóch parametrów tj. *DB\_RECOVERY\_FILE\_DEST\_SIZE*, określającego maksymalny rozmiar obszaru *FRA* oraz *DB\_RECOVERY\_FILE\_DEST*, wskazującego podstawowy katalog dla *FRA*. Wartość tych parametrów można ustawić np. w pliku *init.ora* (listing 7.6).

Listing 7.6. Przykładowy plik *init.ora*

```
SQL> ALTER SYSTEM SET db_recovery_file_dest_size=200g;  
SQL> ALTER SYSTEM SET db_recovery_file_dest='/ora02/fra';
```

Podstawową zaletą stosowania *FRA* jest możliwość łatwiejszego zarządzania obszarem przechowującym kopie zapasowe. Jednak należy także podkreślić pewną wadę takiego rozwiązania. Obszar *FRA* jest wykorzystywany przez procesy, związane z innymi metodami archiwizacji (kopie online czy procesy *RMAN*) oraz podczas tworzenia kopii dla różnych baz danych. W związku z tym istnieje możliwość utraty kontroli nad zarchiwizowanymi plikami oraz powstania zatorów w dostępie do danych w przypadku współdzielenia obszaru *FRA* przez kilka baz danych. Dlatego też warto rozważyć, czy dedykowane lokalizacje dla kopii krytycznych plików różnego typu nie okażą się lepszym rozwiązaniem od zdefiniowania i wykorzystywania obszaru *FRA*.

O wiele wygodniejszym i zautomatyzowanym narzędziem do tworzenia i zarządzania kopiami jest program *RMAN*. Jest to sposób archiwizacji zalecany przez Oracle. Narzędzie to zostanie opisane w kolejnym podrozdziale.

## 7.2. ARCHIWIZACJA Z WYKORZYSTANIEM NARZĘDZIA *RMAN*

Menedżer odzyskiwania *RMAN* (ang. *Recovery Manager*) to narzędzie, które automatyzuje procesy związane z tworzeniem i odtwarzaniem kopii zapasowych bazy danych (ang. *backup and recovery, B&R*). Jest ono dostępne od wersji Oracle 8. Narzędzie to można uruchomić z linii poleceń lub z menu *Oracle Enterprise Manager* (Preston, 2008).

*RMAN* oferuje bardziej rozbudowaną funkcjonalność w porównaniu do innych narzędzi wspomagających archiwizację, udostępnianych przez Oracle. Może on posługiwać się katalogiem odzyskiwania lub też zapisywać dane w pliku kontrolnym kopiowanej bazy. Katalog odzyskiwania (ang. *recovery catalog*) to schemat bazy danych opcjonalnie używany przez *RMAN* do przechowywania metadanych o jednej lub kilku bazach danych. Zawiera on te same dane, które znajdują się w pliku kontrolnym bazy danych. Spełnia on rolę zapasowego repozytorium metadanych.

*RMAN* umożliwia podgląd zaawansowania prac i śledzenia ich postępu, obsługuje wielowątkowość, kompresję i szyfrowanie oraz walidację. W skład *RMAN*-a wchodzi nawet kilka baz docelowych, baza danych katalogu odzyskiwania oraz oprogramowanie wspomagające zarządzanie nośnikami pamięci.

Architektura *RMAN* jest złożona i składają się na nią następujące komponenty (Urman i in, 2007):

- baza danych (ang. *target database*) – baza danych do archiwizacji;
- klient *RMAN* (ang. *RMAN Client*) – narzędzie umożliwiające wydawanie komend;
- procesy (kanały) serwera Oracle (*Oracle Server Processes*) – dwa procesy uruchamiane do koordynacji archiwizowania oraz aktualizacji dziennika powtórzeń;
- pakiety PL/SQL (ang. *PL/SQL packages*) – pakiety *DBMS\_RCMAN* i *DBMS\_BACKUP\_RESTORE*, służące do przeprowadzania podstawowych operacji *RMAN*;
- bufor pamięci (ang. *memory buffers*, PGA and SGA) – bufor wspomagające odczyt i kopiowanie bloków;
- zewnętrzna baza danych (ang. *auxiliary database*) – baza danych, do której kopiowane są dane;
- kanały (ang. *channels*) – procesy serwera, zarządzające operacjami wejścia/wyjścia do i z urządzeń przechowujących kopie zapasowe;
- zbiory kopii zapasowych (ang. *backup sets*) – logicznie uporządkowane zestawy plików wchodzących w skład kopii zapasowych;
- pliki kopii zapasowych (ang. *backup piece file*) – pliki binarne, zawierające kopie zapasowe;
- obraz kopii (ang. *image copy*) – wierna kopia plików danych, plików dziennika powtórzeń i/lub plików kontrolnych;
- katalog odzyskiwania (ang. *recovery catalog*) – opcjonalny schemat bazy danych, zawierający tabele przechowujące metadane;
- menedżer mediów (ang. *media manager*) – zewnętrzne narzędzie, pozwalające wykonywać backup bazy danych bezpośrednio na taśmę magnetyczną;
- obszar szybkiego odzyskiwania (ang. *fast-recovery area*) – opcjonalna przestrzeń dyskowa, przeznaczona do przechowywania kopii zapasowych;
- migawka pliku kontrolnego (ang. *snapshot control file*) – plik zawierający tymczasową kopię pliku kontrolnego (najważniejszego pliku bazy danych).

*RMAN* jest rozbudowanym narzędziem, oferującym wiele funkcjonalności. Umożliwia on tworzenie kilku rodzajów kopii zapasowych:

- pełnych (kopiowanie wszystkich bloków danych) – poziom 0;
- kumulacyjnych (kopiowanie bloków, które zostały zmienione od czasu wykonania pełnej kopii) – poziom 1;
- przyrostowych (kopiowanie bloków, które zostały zmienione od czasu wykonania kopii pełnej lub kumulacyjnej) – poziom 2.

Wykorzystując narzędzie *RMAN* należy pamiętać, iż jest ono uruchamiane jako klient w środowisku systemu operacyjnego, a nie z poziomu SQL\*Plus-a. W związku z tym, logując się do programu *RMAN* wykorzystuje się autentykację użytkownika systemu operacyjnego (zamiast użytkownika bazy danych). Wymagane jest tylko wcześniejsze ustawienie zmiennych środowiskowych takich jak *ORACLE\_HOME* czy *ORACLE\_SID* oraz posiadanie dostępu do bazy z uprawnieniami *SYSDBA*.

W celu uruchomienia narzędzia *RMAN* należy zalogować się do bazy jako użytkownik posiadający prawa *SYSDBA* i użyć komend podanych w listingu 7.7.

*Listing 7.7. Uruchamianie narzędzia RMAN*

```
$ rman target /  
$ rman target sys/foo@remote_db -- połączenie zdalne
```

Ideę działania narzędzia *RMAN* prezentuje listing 7.8. Przedstawione w nim polecenia pozwalają przeprowadzić podstawowe operacje związane z tworzeniem kopii zapasowej, odzyskiwaniem plików kopii oraz ponownym przywróceniem poprawnego działania bazy danych.

*Listing 7.8. Polecenia narzędzia RMAN*

```
-- Uruchomienie i prosty backup:  
$ rman target /  
RMAN> BACKUP DATABASE;  
-- Przywrócenie plików:  
RMAN> SHUTDOWN IMMEDIATE;  
RMAN> STARTUP MOUNT;  
RMAN> RESTORE DATABASE;  
-- Pełne przywrócenie działania bazy:  
RMAN> RECOVER DATABASE;  
RMAN> ALTER DATABASE open;
```

Pełne wykorzystanie funkcjonalności narzędzia *RMAN* wymaga zdefiniowania zasad tworzenia kopii zapasowych. Na zasady te składają się elementy i parametry przedstawione w tabeli 7.1. Określają one różne opcje pracy środowiska *RMAN* i procesów tworzenia kopii.

Tabela 7.1. Opcje tworzenia kopii zapasowych w narzędziu *RMAN*

Nazwa	Dostępne opcje	Uwagi
Sposób uruchamiania <i>RMAN</i>	Lokalnie lub zdalnie	Zaleca się korzystanie z połączeń lokalnych.
Typ backupów	Online lub offline	Zwykle korzysta się z kopii online (nie wymagają one zamykania bazy danych).
Określenie uprawnionego użytkownika	Konto SYS lub inne konto z odpowiednimi uprawnieniami	Jeżeli nie ma specjalnych restrykcji, można korzystać z konta SYS.
Określenie lokalizacji i formatu plików zarchiwizowanego dziennika powtórzeń	Zdefiniowanie FRA	np.: <code>&lt;db_name&gt;/archivelog/&lt;YYYY_MM_DD&gt;/o1_mf_1_51m_.arc)</code>
	Ustawienie opcji LOG_ARCHIVE_DEST_N	np.: <code>log_archive_dest_1='LOCATION=/db1/oracle/arch'</code>
	Wykorzystanie ustawień domyślnych	Domyślna lokalizacja: <code>ORACLE_HOME/dbs/arch</code> domyślny format danych: <code>%t_%s_%r.dbf</code>
Określenie lokalizacji i formatu plików <i>RMAN</i>	Zdefiniowanie FRA	Domyślna lokalizacja: <code>ORACLE_HOME/dbs/arch</code>
	Określenie opcji FORMAT lub CONFIGURE	np.: <code>RMAN&gt; backup database format 'ora01/O11R2/rman/rm_%U.bkp';</code> <code>RMAN&gt; configure device type disk parallelism 3;</code> <code>RMAN&gt; configure channel 1 device type disk format 'ora01/O11R2/rman/rm_%U.bk';</code>
	Wykorzystanie ustawień domyślnych	Domyślna lokalizacja: <code>ORACLE_HOME/dbs</code> domyślny format danych (określony przez <i>Oracle Managed File</i> )
Utworzenie (lub nie) automatycznej kopii zapasowej pliku kontrolnego	Tak/Nie	Zaleca się korzystanie z tej opcji.  Kontrola stanu: <code>RMAN&gt; show controlfile autobackup;</code>

Nazwa	Dostępne opcje	Uwagi
		<p><code>RMAN&gt; configure controlfile autobackup on;</code></p> <p>Jeśli opcja <i>AUTOBACKUP</i> jest wyłączona, plik kontrolny będzie kopiowany automatycznie.</p>
Określenie lokalizacji automatycznej kopii zapasowej pliku kontrolnego	Zdefiniowanie lokalizacji dla FRA	np.: <code>&lt;/fra&gt;/&lt;db_name&gt;/autobackup/&lt;YYYY_MM_DD&gt;/o1_mf_749_62_.bkp</code>
	Zdefiniowanie lokalizacji bez FRA	np.: <code>RMAN&gt; configure controlfile autobackup format for device type disk to 'ora01/O11R2/rman/rm_ct_%F.bk';</code>
	Wykorzystanie ustawień domyślnych	<code>ORACLE_HOME/dbs</code>
Określenie lokalizacji migawek pliku kontrolnego	Pozostawienie lokalizacji domyślnej lub określenie nowej	<p>Zwykle zostawia się lokalizację domyślną: <code>ORACLE_HOME/dbs/snappcf_@.f</code>. Lokalizację można sprawdzić poleceniem:  <code>RMAN&gt; show snapshot controlfile name;</code></p> <p>Ustawienie innej lokalizacji:  <code>RMAN&gt; configure snapshot controlfile name clear;</code></p>
Użycie(lub nie) katalogu odzyskiwania (ang. <i>recovery catalog</i> )	Tak/Nie	Oracle zaleca jego użycie.
Określenie minimalnego czasu zanim rekord w pliku kontrolnym może zostać nadpisany	Liczba dni	Za wartość tą odpowiada parametr <code>CONTROL_FILE_RECORD_KEEP_TIME</code> . Domyślnie jest to 7 dni.
Użycie(lub nie) menadżera mediów	Tak/Nie	Wymagane jest tylko w przypadku korzystania z taśm
Określenie stopnia równoległości zadań	Liczba procesów	<p>W przypadku kilku procesorów warto zwiększyć liczbę kanałów równoległych. Mogą one korzystać z jednej lokalizacji lub kilku:</p> <p><code>RMAN&gt; configure device type disk parallelism 4;</code>  <code>RMAN&gt; configure channel 1 device type disk format 'ora01/O11R2/rman/rman_%U.bk';</code></p>

Nazwa	Dostępne opcje	Uwagi
		<p>RMAN&gt; <i>configure channel 2 device type disk format</i>  '<i>ora02/O11R2/rman/rman_%U.bk</i>';</p> <p>Do czyszczenia kanałów służy polecenie:  RMAN&gt; <i>configure device type disk clear</i>;</p>
Użycie zbiorów kopii zapasowych lub obrazów kopii	Kopie zapasowe/obrazy kopii	Zbiory kopii zapasowych są mniejsze i są łatwiejsze do zarządzania, w szczególności do kompresji. Obrazy kopii są z kolei szybsze do przywrócenia, szczególnie w przypadku uszkodzenia nośnika.
Tworzenie kopii przyrostowych	Tak/Nie	Zalecane szczególnie dla dużych baz danych z niewielką ilością zmian. Opcję tę wspomaga opcja śledzenia zmian w blokach. Przyspiesza to tworzenie kopii i zwiększa wydajność, ponieważ nie jest już potrzebne sprawdzanie różnic we wszystkich obiektach bazy.
Użycie (lub nie) kompresji	Tak/Nie	<p>Binarną kompresję bloków można określić na dwa sposoby:</p> <ul style="list-style-type: none"> <li>komenda <i>BACKUP</i>:  RMAN&gt; <i>backup as compressed backupset database</i>;</li> <li>komenda <i>CONFIGURE</i>:  RMAN&gt; <i>configure device type disk backup type to compressed backupset</i>;</li> </ul> <p>Domyślnie dostępny jest podstawowy algorytm kompresji  RMAN&gt; <i>show compression algorithm</i>;  <i>configure compression algorithm 'basic' as of release 'default' optimize for load true; # default</i></p> <p>Można dokupić dostęp do bardziej zaawansowanych algorytmów kompresji:  RMAN&gt; <i>configure compression algorithm 'HIGH'</i>;  RMAN&gt; <i>configure compression algorithm 'MEDIUM'</i>;  RMAN&gt; <i>configure compression algorithm 'LOW'</i>;</p>



Nazwa	Dostępne opcje	Uwagi
Użycie (lub nie) szyfrowania	Tak/Nie	Włączenie szyfrowania wykonuje się poleceniem: RMAN> <i>configure encryption for database on;</i>

Źródło: (Alapati, 2009)

Podczas tworzenia kopii zapasowych Oracle wykorzystuje katalog odzyskiwania, służący między innymi do przechowywania metadanych (bez właściwych danych), związanych z kopiami zapasowymi. Ten oddzielny schemat zawiera obiekty bazy danych, przechowujące informacje o kopii zapasowej. Mechanizm taki stanowi dodatkowe zabezpieczenie przed utratą plików kontrolnych. Dlatego też zaleca się przechowywanie katalogu odzyskiwania na innym serwerze niż baza źródłowa. Warto jednak zauważyć, że katalog odzyskiwania to kolejna baza danych, którą trzeba utrzymywać i pielęgnować.

Wykorzystanie katalogu odzyskiwania, w celu podniesienia poziomu bezpieczeństwa generowania kopii, wymaga jego istnienia przed rozpoczęciem procesu archiwizacji. Definiowanie katalogu odzyskiwania odbywa się w następujących etapach (Alapati, 2009):

1. Utworzenie bazy danych o odpowiednich rozmiarach (przestrzenie tabel *SYSTEM*, *SYSAUX*, *TEMP* i *UNDO* powinny mieć min. 500 MB).
2. Stworzenie przestrzeni tabel (przy pomocy polecenia *CREATE TABLESPACE*). Nazwa przestrzeni tabel powinna wskazywać na jej przeznaczenie, np. *RECCAT*, a przykładowy jej początkowy rozmiar można określić np. na 128 KB.
3. Utworzenie użytkownika z uprawnieniami do zarządzania bazą, w szczególności nadanie jemu praw *RECOVERY\_CATALOG\_OWNER* i *CREATE\_SESSION*.
4. Połączenie się z utworzoną bazą przy użyciu narzędzia *RMAN* (listing 7.9).

Listing 7.9. Połączenie z bazą danych za pomocą *RMAN*

```
$ rman catalog rcat_user/rat_pass
```

## 5. Utworzenie katalogu odzyskiwania (listing 7.10).

Listing 7.10. Stworzenie katalogu dzyskiwania

```
RMAN> CREATE CATALOG ;
```

## 6. Sprawdzenie poprawności utworzenia katalogu odzyskiwania.

## 7. Zalogowanie się do bazy źródłowej (ang. *target database*) i rejestracja katalogu odzyskiwania w źródłowej bazie danych (listing 7.11).

Listing 7.11. Logowanie do bazy i rejestracja katalogu odzyskiwania

```
$ rman target / catalog rcat_user/rat_pass  
register database;
```

Po utworzeniu katalogu odzyskiwania można już przystąpić do tworzenia kopii zapasowych plików z danymi, plików kontrolnych, dziennika powtórzeń oraz pliku parametrów serwera (*spfile*). W dalszej części rozdziału zostaną przedstawione procesy tworzenia pełnego oraz inkrementacyjnego backupu bazy danych.

*RMAN* umożliwia tworzenie różnych rodzajów kopii zapasowych. Tabela 7.2 przedstawia najważniejsze polecenia przydatne do ich tworzenia i zarządzania nimi.

Tabela 7.2. Przydatne polecenia tworzenia i zarządzania kopiami zapasowymi

Nazwa	Polecenie	Opis
Tworzenie zbioru kopii zapasowych	RMAN> <i>backup database;</i> RMAN> <i>backup as backupset database;</i>	
Tworzenie obrazu kopii	RMAN> <i>backup as copy database;</i>	
Tworzenie kopii przestrzeni tabel	RMAN> <i>backup tablespace system, sysaux;</i>	<i>SYSTEM</i> i <i>SYSAUX</i> to nazwy przestrzeni tabel.
Tworzenie kopii pojedynczych plików	RMAN> <i>backup datafile</i> <i>'ora01/dbfile/O11R2/system_01.dbf';</i> RMAN> <i>backup as copy datafile 3;</i>	Pliki można wskazać przy pomocy nazw lub ich numerów.

Nazwa	Polecenie	Opis
	<code>RMAN&gt; backup incremental level 1 datafile 4;</code>	
Ręczne tworzenie kopii pliku kontrolnego	<code>RMAN&gt; backup current controlfile;</code>	Zaleca się ustawienie automatycznego tworzenia kopii pliku kontrolnego.
Tworzenie kopii pliku parametrów serwera	<code>RMAN&gt; backup spfile;</code>	Plik zostanie skopiowany automatycznie, jeśli ustawiona będzie opcja automatycznego tworzenia kopii pliku kontrolnego.
Tworzenie kopii zarchiwizowanych plików dziennika powtórzeń	<code>RMAN&gt; backup archivelog all;</code> <code>RMAN&gt; backup archivelog sequence 300;</code> <code>RMAN&gt; backup archivelog from time "sysdate-7" until time "sysdate-1";</code>	Istnieje możliwość utworzenia pełnej kopii lub takiej, która obejmuje wpisy w określonym czasie lub o określonych numerach.
Wyłączanie przestrzeni tabel z obszaru do skopiowania	<code>RMAN&gt; configure exclude for tablespace users;</code>	Wyświetlenie listy wykluczeń: <code>RMAN&gt; show exclude;</code>  Można też utworzyć kopię zapasową tylko wykluczonych przestrzeni tabel: <code>RMAN&gt; backup database noexclude;</code>
Tworzenie kopii zapasowych obszarów, które nie były włączone w obszar kopiowania	<code>RMAN&gt; backup database not backed up;</code> <code>RMAN&gt; backup database not backed up since time='sysdate-1';</code>	Polecenie przydatne w przypadku np. awaryjnego zatrzymania tworzenia kopii. Po jego wykonaniu skopiowane zostaną te obszary, które z powodu awarii nie zostały skopiowane.
Pominięcie przestrzeni tabel tylko do odczytu	<code>RMAN&gt; backup database skip readonly;</code>	
Pominięcie plików niedostępnych lub będących offline	<code>RMAN&gt; backup database skip inaccessible;</code> <code>RMAN&gt; backup database skip offline;</code>	Pliki takie mogą uniemożliwić tworzenie kopii. Można je zatem pominąć.
Równoległe tworzenie kopii	<code>RMAN&gt; configure device type disk parallelism 2;</code>	

Nazwa	Polecenie	Opis
zapasowych dużych plików	<pre>RMAN&gt; configure channel 1 device type disk format 'ora01/O11R2/rman/r1%U.bk'; RMAN&gt; configure channel 2 device type disk format 'ora02/O11R2/rman/r2%U.bk'; RMAN&gt; backup section size 2500M datafile 10;</pre>	

Źródło: (Alapati, 2009)

Wykonanie backupu całej bazy można przeprowadzić, wydając polecenie przedstawione na listingu 7.12.

Listing 7.12. Stworzenie kopii zapasowej całej bazy

```
RMAN> BACKUP incremental level=0 database plus archivelog ;
```

W tym przypadku zostaną skopiowane wszystkie pliki danych z bazy danych, pliki dziennika powtórzeń, wygenerowane od chwili wykonania poprzedniego backupu i powstałe w trakcie bieżącego procesu kopiowania. Takie działanie pozwoli utworzyć wszystkie pliki, które w przyszłości będą wymagane do odtworzenia bazy danych. Jeśli dodatkowo włączona jest opcja automatycznego tworzenia kopii pliku kontrolnego, zostanie on także dołączony do tworzonej kopii.

Pełną kopię zapasową można wygenerować przy pomocy dwóch, niemal równorzędnych, poleceń (listing 7.13).

Listing 7.13. Stworzenie pełnej kopii zapasowej

```
RMAN> BACKUP AS BACKUPSET FULL DATABASE;
RMAN> BACKUP AS BACKUPSET INCREMENTAL LEVEL=0 DATABASE;
```

W drugim poleceniu wykorzystano klauzulę *incremental level=0*. Jej użycie umożliwi w przyszłości utworzenie kopii w trybie inkrementacyjnym, a nie pełnym.

Kopie inkrementacyjne (przyrostowe) to kolejny sposób archiwizowania danych. Prawidłowe wykonanie tego rodzaju kopii wymaga stworzenia na początku kopii inkrementacyjnej na poziomie 0. Kolejne kopie mogą być tworzone już na poziomie 1. Poziom ten obejmuje archiwizację tylko tych bloków, które zmieniły się od poprzedniego backupu. Jeśli jednak pierwsza kopia bazy danych zostanie wykonana na poziomie 1, to w rzeczywistości będzie to działanie równoważne wykonaniu pełnego backupu na poziomie 0 (listing 7.14).

Listing 7.14. Stworzenie inkrementacyjnej kopii zapasowej

```
RMAN> BACKUP INCREMENTAL LEVEL=1 DATABASE;
```

Istnieją dwa rodzaje kopii inkrementacyjnych: różnicowe oraz kumulacyjne (Loney, 2005). Domyślnie wykonywane są kopie różnicowe (ang. *differential backups*), które są mniejsze i szybsze w tworzeniu, gdyż kopiowanie różnicowe polega na archiwizowaniu tylko tych bloków, które zmieniły się od poprzedniego backupu typu 0 lub 1.

Kopie kumulacyjne są większe, ale ich odtworzenie zajmuje mniej czasu. W tym przypadku kopiowane są wyłącznie bloki zmienione od czasu utworzenia backupu na poziomie 0. Backupy utworzone na poziomie 1 nie są brane pod uwagę. Przykład utworzenia kumulacyjnej kopii zapasowej przedstawia listing 7.15.

Listing 7.15. Stworzenie kumulacyjnej kopii zapasowej

```
RMAN> BACKUP INCREMENTAL LEVEL=1 CUMULATIVE DATABASE;
```

Tworzenie kopii inkrementacyjnych jest szybsze, jeśli włączy się opcję śledzenia zmian bloków (ang. *block change tracking*). Jednak takie działanie powoduje większą zajętość przestrzeni na twardym dysku.

W celu przeprowadzenia kontroli poprawności wykonania operacji archiwizacji konieczna jest walidacja wygenerowanych kopii zapasowych. Istnieją trzy sposoby jej realizacji (Hart, Freeman, 2008; Alapati, 2009):

1. Polecenie *VALIDATE* sprawdza, czy istnieją wszystkie wymagane pliki, a także czy nie ma fizycznych uszkodzeń w plikach danych, dziennikach powtórzeń lub plikach kontrolnych. Wszelkie dane o nieprawidłowościach z tym związanych (np. numery plików i bloków) zostaną wyświetlone na ekranie. Przykłady wykorzystania polecenia *VALIDATE* przedstawia listing 7.16.

*Listing 7.16. Przykład wykorzystania polecenia VALIDATE*

```
RMAN> VALIDATE DATABASE;  
RMAN> VALIDATE CURRENT CONTROLFILE;  
RMAN> VALIDATE ARCHIVELOG ALL;  
RMAN> VALIDATE DATABASE SKIP OFFLINE;  
RMAN> VALIDATE COPY OF DATABASE;  
RMAN> VALIDATE TABLESPACE SYSTEM;
```

2. Polecenie *BACKUP...VALIDATE* – jego działanie jest bardzo zbliżone do działania polecenia *VALIDATE* i polega na wyszukiwaniu fizycznych uszkodzeń plików z danymi. Może ono być też użyte do wyszukiwania uszkodzeń logicznych, np. struktury plików. Podobnie jak w przypadku polecenia *VALIDATE*, informacje o nieprawidłowościach (np. numery plików i bloków) są wyświetlane na ekranie. Przykład wykorzystania polecenia *BACKUP...VALIDATE* przedstawia listing 7.17.

*Listing 7.17. Przykład wykorzystania polecenia BACKUP..VALIDATE*

```
RMAN> BACKUP VALIDATE DATABASE;  
RMAN> BACKUP VALIDATE CHECK LOGICAL DATABASE;  
RMAN> BACKUP VALIDATE DATABASE CURRENT CONTROLFILE;
```

3. Polecenie *RESTORE...VALIDATE* używane jest do sprawdzenia, czy pliki kopii zapasowej mogą być użyte podczas realizacji zadania odtwarzania. Jednak żadne działania związane z właściwym odzyskiwaniem nie są prowadzone podczas wykonywania tego polecenia (listing 7.18).

Listing 7.18. Przykład wykorzystania polecenia *RESTORE..VALIDATE*

```
RMAN> RESTORE VALIDATE DATABASE;
```

Tworzenie kopii zapasowych powinno być dokumentowane raportami zawierającymi informacje o przebiegu całego procesu archiwizacji. *RMAN* udostępnia dwa polecenia służące do raportowania, a mianowicie *LIST* i *REPORT*.

Polecenie *LIST* (listing 7.19) wyświetla nazwy obiektów należących do zbioru kopii zapasowych, elementy backupów i nazwy innych plików, wchodzących w skład utworzonej kopii. Można je także użyć do uzyskania informacji na temat obrazu kopii czy archiwalnych plików dziennika powtórzeń, w szczególności o ich numerach, nazwach, statusie i datach utworzenia.

Listing 7.19. Przykład wykorzystania polecenia *LIST*

```
RMAN> LIST BACKUP;  
RMAN> LIST BACKUP SUMMARY;  
RMAN> LIST COPY;  
RMAN> LIST ARCHIVELOG ALL;
```

Polecenie *REPORT* pozwala wyświetlić informacje na temat szczegółów utworzenia różnych elementów kopii zapasowej, np. backupów o określonym typie, pojedynczych plików danych, plików uszkodzonych lub przestarzałych, a nawet plików, które nie zostały włączone do kopii zapasowej (listing 7.20). Szczegóły takie, w zależności od rodzaju wyświetlanych informacji, mogą dotyczyć daty, rodzaju i statusu utworzenia kopii, nazw lub numerów poszczególnych plików.

Listing 7.20. Przykład wykorzystania polecenia *REPORT*

```
RMAN> REPORT SCHEMA;  
RMAN> REPORT NEED BACKUP;  
RMAN> REPORT UNRECOVERABLE;
```

Listing 7.21 prezentuje przykład raportu wygenerowanego przy użyciu polecenia *Report*.

*Listing 7.21. Przykład raportu wygenerowanego przy użyciu polecenia REPORT*

```
RMAN> REPORT NEED BACKUP;  
  
RMAN retention policy will be applied to the command  
RMAN retention policy is set to redundancy 1  
Report of files with less than 1 redundant backups  
  
File #bkps Name  
-----  
1 0 /u01/app/oracle/product/11.1.0/oradata/db/system01.dbf  
2 0 /u01/app/oracle/product/11.1.0/oradata/db/undotbs01.dbf  
3 0 /u01/app/oracle/product/11.1.0/oradata/db/sysaux01.dbf  
4 0 /u01/app/oracle/product/11.1.0/oradata/db/users01.dbf
```

Narzędzie *RMAN* jest wykorzystywane nie tylko do tworzenia kopii bazy danych, ale także do odzyskiwania danych w przypadku ich utraty lub uszkodzenia, a także awarii nośników danych. Wystąpienie powyższych problemów powoduje wyświetlenie komunikatu, którego treść przedstawia listing 7.22.

*Listing 7.22. Błąd utraty lub uszkodzenia danych*

```
ORA-01157: cannot identify/lock data file 1 - see DBWR trace  
file
```

Komunikat taki pojawi się podczas zatrzymywania lub startu bazy danych, to jest w chwili żądania dostępu do zasobów danych. Jeśli start bazy danych nie powiedzie się, należy rozpatrzyć przeprowadzenie odzyskiwania danych. Sposób jego realizacji zależy od rozmiaru uszkodzeń oraz od rodzaju dostępnych kopii zapasowych.

Backupy utworzone przy pomocy narzędzie *RMAN* mogą być wykorzystane do wykonania odzyskiwania pełnego lub częściowego, które ogranicza się do wybranych fragmentów bazy danych. Odzyskiwanie można przeprowadzić na podstawie jednego z dostępnych typów kopii:



- pełnego backupu bazy danych;
- inkrementacyjnego backupu bazy danych na poziomie 0, z opcjonalnie dołączonymi backupami inkrementacyjnymi wyższych poziomów;
- obrazu kopii (ang. *image copy backup*).

Niezależnie od sposobu przeprowadzenia odzyskiwania, konieczne jest określenie plików, które wymagają odtworzenia. Narzędzie *RMAN* może wskazać je automatycznie bądź też administrator sam musi uzyskać taką informację. Jednym ze sposobów wykonania tego zadania jest sprawdzenie treści komunikatów o błędach, wyświetlanych w konsoli *RMAN* lub *SQL\*Plus* oraz treści znajdujących się w pliku *Alert.log*. Informacje o plikach wymagających odzyskiwania mogą być też pozyskane poprzez użycie instrukcji *SQL*, wykorzystujących perspektywy systemowe (ang. *data dictionary views*).

Podczas odtwarzania bazy danych można również posłużyć się raportem wygenerowanym przez narzędzie *Data Recovery Advisor*. Raport ten zawiera szczegółowe informacje dotyczące awarii i sugeruje, jakie działania są konieczne do odzyskania danych.

Odzyskiwanie bazy danych z backupów składa się z następujących etapów:

1. Określenie, jakie pliki mają być odzyskane.
2. Ustawienie bazy danych w jeden z trybów *NOMOUNT*, *MOUNT* lub *OPEN*, w zależności od rodzaju uszkodzeń lub zniszczeń.
3. Użycie polecenia *RESTORE* do odzyskania plików danych z kopii zapasowych *RMAN*.
4. Użycie polecenia *RECOVER* w celu odzyskania koniecznych plików danych.
5. Otworzenie bazy danych.

Podczas odzyskiwania danych przy pomocy polecenia *RESTORE*, program *RMAN* automatycznie wykryje, w jaki sposób i w jakim obszarze należy przeprowadzić odzyskiwanie. Może to być odzyskiwanie na podstawie pełnego backupu, backupu inkrementacyjnego na poziomie 0 lub obrazu kopii wygenerowanego za pomocą komendy *BACKUP AS COPY*.

Podstawowym sposobem odtwarzania danych jest tzw. pełne odzyskiwanie. Dotyczy ono wszystkich transakcji, które zostały zapisane przed awarią. Nie jest to jednak odzyskiwanie wszystkich plików, które zostały zapisane podczas tworzenia

kopii. Przywrócone zostaną jedynie te, które wymagane są dla poprawnego działania bazy danych i zachowania jej stanu spójnego. Aby pełne odzyskiwanie danych było możliwe, muszą być spełnione następujące warunki (Alapati, 2009):

- baza danych musi być w trybie *ARCHIVELOG*;
- istnieją odpowiednie kopie zapasowe;
- istnieją odpowiednie zarchiwizowane pliki dziennika powtórzeń;
- dostępne są odpowiednie pliki logów, zawierające informacje od ostatniego backupu online;
- dostępne są kopie inkrementacyjne (jeśli były tworzone);
- pliki logów zawierają transakcje, które nie były jeszcze zapisane.

Przed odzyskiwaniem danych można sprawdzić, które pliki będą wykorzystane przez narzędzie *RMAN* do przeprowadzenia tej operacji. W tym celu należy wydać polecenie przedstawione na listingu 7.23. Warto też przeprowadzić walidację przy pomocy poleceń przedstawionych uprzednio w tym rozdziale.

*Listing 7.23. Przygotowanie bazy danych do odzyskiwania*

```
RMAN> RESTORE DATABASE PREVIEW;  
RMAN> RESTORE DATABASE PREVIEW SUMMARY;
```

Kompletne odzyskiwanie danych można przeprowadzić na dwa sposoby:

1. Użycie aktualnego pliku kontroli – wymaga wprowadzenia bazy danych w tryb *MOUNT*, ponieważ przestrzeń tabel *SYSTEM* musi być w trybie offline podczas odzyskiwania. Tryb *MOUNT* gwarantuje, że użytkownicy nie będą łączyć się z bazą danych i żadne transakcje nie zostaną rozpoczęte ani zatwierdzone (listing 7.24).

*Listing 7.24. Przykład odzyskiwania danych z użyciem aktualnego pliku kontroli*

```
RMAN> CONNECT TARGET /  
RMAN> STARTUP MOUNT;  
RMAN> RESTORE DATABASE;  
RMAN> RECOVER DATABASE;  
RMAN> ALTER DATABASE OPEN;
```

2. Użycie kopii zapasowej pliku kontroli (listing 7.25) – plik kontroli pobierany jest z kopii zapasowej przed odzyskiwaniem bazy danych.

*Listing 7.25. Przykład odzyskiwania danych z użyciem kopii zapasowej pliku kontroli*

```
RMAN> CONNECT TARGET /  
RMAN> STARTUP NOMOUNT;  
RMAN> RESTORE CONTROLFILE FROM AUTOBACKUP;  
RMAN> ALTER DATABASE MOUNT;  
RMAN> RESTORE DATABASE;  
RMAN> RECOVER DATABASE;  
RMAN> ALTER DATABASE OPEN RESETLOGS;
```

Jeśli podczas przeprowadzania pełnego odzyskiwania konieczne jest przywrócenie pojedynczych przestrzeni tabel lub plików z danymi, można to zrobić w czasie, gdy baza danych jest otwarta lub też, gdy baza jest w trybie *MOUNT*. W pierwszym przypadku konieczne jest przejście danej przestrzeni tabel w tryb offline (można to zrobić z każdą przestrzenią tabel poza *SYSTEM* oraz *UNDO*). W drugim przypadku baza danych jest już w trybie offline. Dla przestrzeni tabel można np. użyć zestawu poleceń z listingu 7.26, dla plików – poleceń z listingu 7.27.

*Listing 7.26. Przykład odzyskiwania danych z pojedynczych przestrzeni tabel*

```
RMAN> CONNECT TARGET /  
RMAN> SQL 'alter tablespace users offline immediate';  
RMAN> RESTORE TABLESPACE users;  
RMAN> RECOVER TABLESPACE users;  
RMAN> SQL 'alter tablespace users online';
```

*Listing 7.27. Przykład odzyskiwania danych z pojedynczych plików*

```
RMAN> SQL 'alter database datafile 32, 33 offline';  
RMAN> RESTORE DATAFILE 32, 33;  
RMAN> RECOVER DATAFILE 32, 33;  
RMAN> SQL 'alter database datafile 32, 33 online';
```

Najczęściej jednak, podczas odtwarzania baza danych jest zamykana i restartowana w trybie *MOUNT*. Przykład z listingu 7.28 przedstawia polecenia dla przestrzeni tabel, a z listingu 7.29 – dla plików.

*Listing 7.28. Przykład odzyskiwania danych z pojedynczych przestrzeni tabel (tryb MOUNT)*

```
RMAN> CONNECT TARGET /  
RMAN> SHUTDOWN IMMEDIATE;  
RMAN> STARTUP MOUNT;  
RMAN> RESTORE TABLESPACE system;  
RMAN> RECOVER TABLESPACE system;  
RMAN> ALTER DATABASE OPEN;
```

*Listing 7.29. Przykład odzyskiwania danych z pojedynczych plików (tryb MOUNT)*

```
RMAN> CONNECT TARGET /  
RMAN> SHUTDOWN ABORT;  
RMAN> STARTUP MOUNT;  
RMAN> RESTORE DATAFILE '/ora01/dbfile/O11R2/system01.dbf';  
RMAN> RECOVER DATAFILE '/ora01/dbfile/O11R2/system01.dbf';  
RMAN> ALTER DATABASE OPEN;
```

Poza pełnym przywróceniem danych istnieje także możliwość niekompletnego odzyskiwania. Podejście takie obejmuje przywrócenie danych jedynie do pewnego punktu (np. czasowego) lub zwinięcie bazy do określonego punktu. Zwinięcie bazy danych oznacza cofnięcie wykonanych na niej transakcji. Takie odzyskiwanie może być przydatne w przypadku np. błędnego wprowadzenia danych lub niepoprawnej ich aktualizacji.

Proces niekompletnego odzyskiwania składa się z przywracania plików danych oraz odzyskiwania, tj. ponownego wykonania poleceń zapisanych w dzienniku powtórzeń. Przykład polecenia realizującego w/w czynności przedstawia listing 7.30.

Listing 7.30. Przykład niekompletnego odzyskiwania

```
RESTORE DATABASE UNTIL  
RESTORE TABLESPACE UNTIL  
FLASHBACK DATABASE
```

Wśród poleceń z listingu 7.30 najczęściej używanym jest *RESTORE DATABASE UNTIL*. Jest ono najbardziej popularne, ponieważ jego parametrem może być:

- czas, określający punkt do którego mają być przywrócone dane (listing 7.31);
- zakres wartości SCN<sup>1</sup> (listing 7.32);
- numer sekwencji logów (ang. *Log sequence number*) (listing 7.33);
- punkt odzyskiwania (listing 7.34).

Listing 7.31. Przykład odzyskania danych dla określonego okresu czasu

```
RMAN> CONNECT TARGET /  
RMAN> STARTUP MOUNT;  
RMAN> RESTORE DATABASE UNTIL TIME 2> "to_date('04-sep-2010  
14:00:00', 'dd-mon-rrrr hh24:mi:ss')";  
RMAN> RECOVER DATABASE UNTIL TIME 2> "to_date('04-sep-2010  
14:00:00', 'dd-mon-rrrr hh24:mi:ss')";  
RMAN> ALTER DATABASE OPEN RESETLOGS;
```

Listing 7.32. Przykład odzyskania danych dla określonych zmian i wartości SCN

```
RMAN> CONNECT TARGET /  
RMAN> STARTUP MOUNT;  
RMAN> RESTORE DATABASE UNTIL SCN 95019865425;  
RMAN> RECOVER DATABASE UNTIL SCN 95019865425;  
RMAN> ALTER DATABASE OPEN RESETLOGS;
```

---

<sup>1</sup> Wartość SCN (ang. *System Change Number*) jest unikalnym w ramach całej bazy danych numerem nadawanym każdej zatwierdzonej w bazie danych transakcji. Każda kolejna transakcja ma numer SCN większy od numeru transakcji poprzedniej.

*Listing 7.33. Przykład odzyskania danych dla określonego numeru sekwencji logów*

```
RMAN> CONNECT TARGET /  
RMAN> STARTUP MOUNT;  
RMAN> RESTORE DATABASE UNTIL SEQUENCE 45;  
RMAN> RECOVER DATABASE UNTIL SEQUENCE 45;  
RMAN> ALTER DATABASE OPEN RESETLOGS;
```

*Listing 7.34. Przykład odzyskania danych dla określonego punktu odzyskiwania*

```
SQL> CREATE RESTORE POINT MY_RP;  
RMAN> CONNECT TARGET /  
RMAN> STARTUP MOUNT;  
RMAN> RESTORE DATABASE UNTIL RESTORE POINT MY_RP;  
RMAN> RECOVER DATABASE UNTIL RESTORE POINT MY_RP;  
RMAN> ALTER DATABASE OPEN RESETLOGS;
```

Istotną kwestią podczas niepełnego odtwarzania bazy danych jest odzyskiwanie pliku kontrolnego, który jest jednym z najważniejszych plików bazy danych. Istnieją trzy podstawowe sposoby przeprowadzenia tej operacji:

1. Użycie katalogu odzyskiwania (listing 7.35) – po połączeniu z katalogiem odzyskiwania informacje dotyczące pliku kontrolnego są dostępne nawet wtedy, gdy docelowa baza danych jest w trybie *NOMOUNT*.

*Listing 7.35. Przykład odzyskiwania pliku kontrolnego z użyciem katalogu odzyskiwania*

```
RMAN> CONNECT TARGET /  
RMAN> CONNECT CATALOG rcat/rcat@recov  
RMAN> STARTUP NOMOUNT;  
RMAN> RESTORE CONTROLFILE;
```

2. Użycie opcji *AUTOBACKUP* – po połączeniu z docelową bazą danych należy wejść w tryb *NOMOUNT* i wykonać polecenie *RESTORE CONTROLFILE FROM AUTOBACKUP* (listing 7.36).

Listing 7.36. Przykład odzyskiwania pliku kontrolnego (opcja AUTOBACKUP)

```
RMAN> CONNECT TARGET /  
RMAN> STARTUP NOMOUNT;  
RMAN> RESTORE CONTROLFILE FROM AUTOBACKUP;
```

3. Wczytanie pliku z podanej ścieżki (listing 7.37) – polecenie przydatne jest np. podczas kopiowania pliku na zewnętrzny serwer. Po połączeniu się z serwerem i skopiowaniu do jego zasobów kopii zapasowej, plik kontrolny można odzyskać, wskazując jego nazwę i lokalizację.

Listing 7.37. Przykład odzyskiwania pliku kontrolnego wczytanego z określonej lokalizacji

```
RMAN> STARTUP NOMOUNT;  
RMAN> RESTORE CONTROLFILE FROM  
    '/ora01/O11R2/rman/c-345-20100923-07.bk';
```

Podsumowując przedstawione funkcjonalności programu *RMAN* należy stwierdzić, że jest to narzędzie bardzo przydatne w pracy administratora bazy danych. Z powodzeniem może być stosowane do tworzenia różnego rodzaju kopii zapasowych i archiwalnych. Wspomaga też przechowywanie i utrzymywanie wygenerowanych kopii, np. wykorzystując katalog odzyskiwania. Udostępnia także złożone mechanizmy pełnego i częściowego odzyskiwania danych. Z istotnych zalet *RMAN* należy także wymienić możliwość obsługi wielowątkowości, możliwość kompresji danych oraz ich szyfrowania.

### 7.3. PYTANIA KONTROLNE

1. Wymień i scharakteryzuj znane Ci rodzaje kopii zapasowych.
2. Podaj różnicę pomiędzy kopią inkrementacyjną a kumulacyjną.
3. Scharakteryzuj etapy tworzenia kopii offline.
4. Wymień i opisz etapy tworzenia kopii online.
5. Wskaż wady i zalety kopii online.

6. Co to jest tryb *ARCHIVELOG*?
7. Co to jest *FRA* i co zawiera?
8. Wymień najważniejsze funkcje narzędzia *RMAN*.
9. Co to jest katalog odzyskiwania i jakie jest jego znaczenie w procesie archiwizacji i odtwarzania bazy danych?
10. Opisz procedurę tworzenia katalogu odzyskiwania.



## Migracja danych

---

### **Cel**

Migracja danych z poprzednich wersji bazy danych Oracle do wersji 11g jest realizowana przy wykorzystaniu wielu technik. Można je podzielić na pośrednie oraz bezpośrednie. Celem tego rozdziału jest prezentacja obu rodzajów metod.

### **Plan**

1. Sposoby przeprowadzania migracji danych.
2. Asystent aktualizacji bazy danych.
3. Bezpośrednia aktualizacja ręczna.
4. Wykorzystanie mechanizmów importu i eksportu.
5. Metoda oparta o kopiowanie danych.

## 8.1. SPOSOBY PRZEPROWADZANIA MIGRACJI DANYCH

Migracja to proces przeniesienia danych pomiędzy różnymi systemami baz danych. Często jest ona określana mianem aktualizacji środowiska bazy danych. Dane podczas tego procesu są zazwyczaj standaryzowane oraz, jeśli zachodzi taka potrzeba, również konwertowane do jednolitej postaci. Migrację przeprowadza się zazwyczaj w przypadku unowocześniania istniejącego systemu informatycznego lub przejścia na jego nowszą wersję. W szczególności proces ten należy postrzegać jako istotny element aktualizacji bazy danych z wersji wcześniejszych do bardziej aktualnych.

Migracja do wersji Oracle 11g może być przeprowadzona z wykorzystaniem następujących technik (Loney, 2005; Loney, 2009):

- Asystent aktualizacji bazy danych (ang. *Database Upgrade Assistant, DBUA*), który przeprowadzi użytkownika przez proces aktualizacji. Narzędzie to sprawdza aktualną konfigurację i sugeruje odpowiednie modyfikacje, dotyczące np. rozmiaru plików czy specyfikacji tabel *SYSAUX* (podczas migracji z wersji Oracle 10g). *DBUA* posiada graficzny interfejs, umożliwia łatwe przekazywanie parametrów i pracę w tle.
- Aktualizacja przeprowadzana samodzielnie przez administratora. Jest to proces złożony, dlatego też jego realizacja wymaga zaawansowanej wiedzy i doświadczenia ze strony osoby wykonującej to zadanie. Jednak zapewnia ona korzyści w postaci lepszej kontroli nad przebiegiem jej poszczególnych etapów.
- Narzędzie *Data Pump (Import i Export)*, które może być użyte do przeniesienia danych z wcześniejszych wersji bazy. Operacja ta wymaga realizacji działań przygotowawczych, takich jak zabezpieczenie odpowiedniej ilości wolnej przestrzeni na dysku na pliki źródłowe bazy danych, plik zrzutu dla eksportu oraz docelową bazę danych. Główną zaletą wykorzystania tego narzędzia jest większa elastyczność wyboru danych do skopiowania (tylko wybrane wiersze, tabele, schematy, przestrzenie tabel).

- Kopiowanie danych ze starszych wersji bazy danych Oracle do wersji aktualnej poprzez wykorzystanie poleceń SQL. W tej metodzie administrator wydaje takie polecenia jak *CREATE TABLE AS SELECT* czy *INSERT AS SELECT*. Tabele w docelowej bazie danych są zatem tworzone jako rezultat wykonania zapytania do źródłowej bazy. Dane wprowadzane są przyrostowo, więc administrator ma nad nimi pełną kontrolę. Jednak trzeba zwrócić szczególną uwagę na konieczność odrębnego zdefiniowania koniecznych więzów integralności w nowej bazie danych, gdyż wymienione tutaj polecenia nie zapewniają automatycznego ich utworzenia (z wyjątkiem więzów typu *NOT NULL*).

Wybór metody aktualizacji jest zwykle uzależniony od obecnie posiadanej wersji bazy danych, od jej rozmiaru i rodzaju danych, a także od ilości czasu, jaki można przeznaczyć na realizację tego procesu.

W praktyce wyróżnia się migrację bezpośrednią oraz pośrednią. Pierwsza z nich polega na aktualizacji zawartości bazy danych z wykorzystaniem tylko jednej kopii bazy. Można ją przeprowadzić przy użyciu narzędzia *DBUA* lub samodzielnie, wydając kolejne polecenia. Natomiast migracja pośrednia wymaga utworzenia dodatkowej bazy danych, do której zostaną załadowane dane ze środowiska źródłowego, w celu ich ujednolicenia czy sprawdzenia z punktu widzenia integralności. Metody pośrednie, oparte o kopiowanie danych i wykorzystujące *Data Pump* lub łącze z bazą docelową, są czasochłonne, szczególnie w tych przypadkach, w których źródłowa baza danych ma duże rozmiary. Migracja bezpośrednia nie wymaga tworzenia dodatkowej bazy danych, więc zazwyczaj przebiega w znacznie krótszym czasie niż migracja pośrednia. Jej istotną zaletą jest także brak konieczności zapewnienia dodatkowych zasobów pamięci dyskowej, używanych do wstępnego przetworzenia danych. W środowisku Oracle bezpośrednia aktualizacja do wersji 11g możliwa jest tylko wówczas, gdy wersja źródłowej bazy danych to minimum 9.2.0.4, 10.1.0.2 lub 10.2.0.1 (Alapati, 2009). W przypadku posiadania wersji starszych niż uprzednio wymienione, konieczne jest zastosowanie pośredniej metody aktualizacji.

Niezależnie od wybranej metody migracji, przed jej rozpoczęciem należy zarchiwizować bazę danych, istniejącą w środowisku docelowym (jeśli zawiera ona dane), aby w razie potrzeby możliwe było jej odtworzenie. Warto również przygotować wcześniej skrypty, które umożliwiłyby określenie wydajności i poprawności działania

bazy w różnych warunkach po wykonaniu migracji. W celu zapewnienia wysokiego poziomu jakości procesu przeniesienia danych, proponuje się także wykonanie migracji na bazie testowej w celu identyfikacji i eliminacji potencjalnych błędów.

Przed wykonaniem migracji dowolnego typu, Oracle zaleca przeprowadzenie wstępnych testów przy użyciu tzw. *Pre-Upgrade Information Tool*. Narzędzie to dokonuje analizy źródłowej bazy danych (Loney, 2009). Jej rezultatem jest wyświetlenie informacji o możliwych problemach, które należy rozwiązać przed uruchomieniem właściwego procesu aktualizacji. Wykonanie tego rodzaju testu polega na uruchomieniu, w środowisku SQL\*Plus, skryptu, znajdującego się w pliku *utlu111i.sql*, który należy skopiować z katalogu *ORACLE\_HOME/rdbms/* do katalogu znajdującego się poza katalogiem domowym Oracle. Przed uruchomieniem należy również utworzyć, przy pomocy komendy *spool*, plik logów, do którego trafią wyniki analizy (listing 8.1).

*Listing 8.1. Przeprowadzenie wstępnej analizy bazy danych pod kątem aktualizacji*

```
SQL> spool upgrade_info.log
```

Jeżeli *Pre-Upgrade Information Tool* nie wykryje żadnych problemów, można przystąpić do realizacji migracji. Przed jej wykonaniem powinno się zamknąć bazę danych oraz wykonać kopię zapasową w trybie offline. Warto także zapoznać się z dokumentacją instalacji nowej bazy danych.

## 8.2. ASYSTENT AKTUALIZACJI BAZY DANYCH

Asystent aktualizacji bazy danych (*DBUA*) w systemie UNIX uruchamiany jest poleceniem *dbua*. W systemie Windows program ten można wybrać (jako polecenie *Database Upgrade Assistant*) z menu *Oracle Configuration and Migration Tool*. Po uruchomieniu narzędzia należy wskazać bazę danych do aktualizacji, określić miejsce przechowywania plików pomocniczych dla archiwizacji i odzyskiwania oraz podać dane konfiguracyjne, np. e-mail administratora bazy danych czy hasła dla kont

użytkowników. Kolejnym etapem jest wyświetlenie szczegółów konfiguracji sieciowej, którą na tym etapie można zmienić. Po zatwierdzeniu jej szczegółów zostanie wykonane wstępne sprawdzenie bazy docelowej. Wyszukiwane będą np. przestarzałe parametry konfiguracyjne i pliki o zbyt małym rozmiarze. *DBUA* pozwoli także stwierdzić, czy wymagana będzie ponowna kompilacja niepoprawnych obiektów, zawierających kod PL/SQL. Proces ten może być zrealizowany po uruchomieniu bazy danych.

Następnie użytkownik decyduje, czy będzie tworzona kopia zapasowa bazy danych. Jeśli tak, wówczas proces archiwizacji odbywa się w trybie offline, przed właściwą migracją. Katalog, w którym kopia zostanie zapisana, należy wskazać przed jej utworzeniem (Preston, 2008). Po zakończeniu migracji wyświetlane są informacje o jej rezultacie i statusie. W przypadku, gdy rezultaty przenoszenia danych nie są zadowalające, można automatycznie wykonać odzyskiwanie poprzedniej wersji bazy danych, o ile kopia zapasowa była uprzednio stworzona w środowisku *DBUA*. Jeśli migracja zakończyła się pomyślnie, wówczas przebudowany zostaje plik konfiguracyjny programu nasłuchującego sieć. Przebudowa ta polega na usunięciu informacji o starej bazie danych i zastąpieniu ich zapisami dotyczącymi nowej.

### **8.3. BEZPOŚREDNIA AKTUALIZACJA RĘCZNA**

Bezpośrednia aktualizacja ręczna zakłada przeprowadzenie przez administratora tych samych czynności, które są wspomagane przez *DBUA*. Przed aktualizacją konieczne jest użycie skryptu do wstępnego testowania bazy. Skrypt ten należy uruchomić jako użytkownik z uprawnieniami *SYSDBA*. Jeśli w pliku logów nie zostaną zapisane żadne informacje o problemach czy błędach, należy zamknąć bazę i utworzyć kopię zapasową w trybie offline. Właściwy proces migracji powinien być przeprowadzony w oparciu o wcześniej utworzone skrypty, odpowiadające za realizację poszczególnych jej etapów. Podczas tworzenia i uruchamiania skryptów należy uwzględnić specyfikę systemu operacyjnego oraz istniejącą konfigurację bazy danych.

Wykonanie tego rodzaju migracji danych odbywa się w następujących etapach (Urman i in, 2007):

1. Zamknięcie bazy danych oraz jej narzędzi dostępowych.
2. Utworzenie kopii zapasowej w trybie offline.
3. Przygotowanie nowego katalogu domowego Oracle dla bazy docelowej oraz edycja jej plików konfiguracyjnych.
4. Uruchomienie docelowej bazy danych przy użyciu polecenia *startup upgrade*.
5. W przypadku migracji z wersji Oracle 9.2 – utworzenie przestrzeni tabel *SYSAUX* w bazie docelowej.
6. Migracja tabel słownika danych przy użyciu skryptu *catupgrd.sql*.
7. Ponowne uruchomienie instancji docelowej bazy danych.
8. Uruchomienie skryptu wstępnego testowania *utlu111i.sql* (*Pre-Upgrade Information Tool*) do walidacji komponentów, pobranych ze źródłowej bazy danych.
9. Uruchomienie skryptu *catuppst.sql*, przeprowadzającego dalsze etapy migracji.
10. Ponowne skompilowanie pakietów i procedur (uruchomienie skryptu *utlrlp.sql*) w docelowej bazie danych.

Ręczny proces aktualizacji jest metodą bezpośrednią, ponieważ do jego przeprowadzenia nie jest konieczne utworzenie dodatkowej bazy danych. Jednak poprawne wykonanie wszystkich operacji, składających się na cały proces przeniesienia danych, wymaga zaawansowanej wiedzy i dużego doświadczenia administratora w realizacji tego rodzaju działań. Dlatego też przed rozpoczęciem procesu migracji należy zapoznać się z dokumentacją Oracle, dotyczącą tego tematu, ze szczególnym uwzględnieniem sekcji, poświęconej rozwiązywaniu problemów aktualizacji systemu bazy danych.

#### 8.4. WYKORZYSTANIE MECHANIZMÓW IMPORTU I EKSPORTU

Wykorzystanie mechanizmów importu i eksportu, takich jak *Data Pump* czy oryginalnych narzędzi *Import* i *Export*, to jedna z pośrednich metod migracji danych. W celu jej przeprowadzenia należy utworzyć nową, docelową bazę Oracle 11g i użyć narzędzi eksportujących oraz importujących dane pomiędzy źródłową a docelową bazą danych. W szczególnych przypadkach, gdy nie ma miejsca na dysku, po

wykonaniu eksportu można usunąć źródłową bazę danych. Jest to jednak działanie ryzykowne i powinno się unikać takiego sposobu postępowania.

Operacja eksportu musi być przeprowadzona w czasie, gdy baza danych nie jest dostępna do aktualizacji. Przed importem danych do bazy docelowej należy utworzyć w niej użytkowników i odpowiednie przestrzenie tabel, w których zapisane zostaną dane źródłowe. Podczas importu należy uważać, aby nie zostały nadpisane dane już istniejące, np. podczas uruchamiania poleceń typu *CREATE TABLESPACE*. Po załadowaniu danych należy sprawdzić pliki logów, zawierające informacje o obiektach, których nie udało się zaimportować.

Uzupełnieniem dotychczas zaprezentowanych działań jest takie skonfigurowanie aplikacji klienckich, aby mogły korzystać już z nowej bazy. Ponadto należy dokonać modyfikacji zawartości plików konfiguracyjnych (*init.ora*, *ocfs.conf*, *sqlnet.ora*), skryptów specyficznych dla określonych wersji bazy i sieciowych plików konfiguracyjnych (*tnsnames.ora*, *listener.ora*).

W zależności od wersji źródłowej bazy danych, dostępne są różne narzędzia do eksportu danych. Plik zrzutu eksportu danych, utworzony przez narzędzie *Export*, jest kompatybilny jedynie z wybranymi wersjami bazy danych. Związane jest to z faktem, iż narzędzia *Data Pump* wprowadzono dopiero w wersji Oracle 10g. Wcześniej dostępne były jedynie oryginalne narzędzia *Import* i *Export*. Są one także zaimplementowane w nowszych wersjach bazy danych Oracle, jednak od wersji 10g nie są już rozwijane. Przykładowe zestawienia kompatybilności poszczególnych wersji, z punktu widzenia operacji eksportu i importu danych, prezentuje tabela 8.1.

Tabela 8.1. Wybrane parametry eksportu

Eksport z	Import do	Narzędzie eksportu	Narzędzie importu
Wersja 10.2	Wersja 11.1	Data Pump Export 10.2	Data Pump Import 11.1
Wersja 10.1	Wersja 11.1	Data Pump Export 10.1	Data Pump Import 11.1
Wersja 9.2	Wersja 11.1	Narzędzie Export 9.2	Narzędzie Import 11.1

Źródło: (Kuhn, 2010)

## 8.5. METODA OPARTA O KOPIOWANIE DANYCH

Metoda oparta o kopiowanie danych wymaga jednoczesnego istnienia obu baz, tj. źródłowej oraz docelowej. Warto ją wykorzystać w sytuacji, kiedy kopiowanych jest niewiele tabel i nie mają one dużych rozmiarów. Dane są przenoszone wówczas za pomocą zapytań, jednak nie wszystkie więzy integralności zostaną uwzględnione podczas wykonywania tej operacji. Źródłowa baza danych powinna być niedostępna do aktualizacji na czas migracji (Loney, Bryla, 2008).

Proces migracji należy rozpocząć od utworzenia docelowej bazy danych. Po jej uruchomieniu konieczne jest zdefiniowanie przestrzeni tabel, użytkowników oraz tabel, które mają być wypełnione. Ponadto trzeba także określić związki między tabelami oraz inne więzy integralności. Podstawowym poleceniem używanym do przenoszenia danych w tej metodzie jest *INSERT AS SELECT*.

Główną zaletą metody kopiowania danych jest pełna kontrola nad przenoszonymi danymi. Można w niej ograniczyć zakres migrujących danych nawet do pojedynczych wierszy i kolumn. Dodatkowo, łatwo jest też zmodyfikować docelową strukturę danych. Należy także pamiętać, aby po wykonaniu migracji zostały zachowane relacje między tabelami oraz wszystkie wymagane więzy integralności.

Kopiowanie danych jest procesem bardzo czasochłonnym, ponieważ dane muszą być często alokowane do wielu miejsc. Jednak istnieją pewne techniki, których zastosowanie zwiększa wydajność tej metody. Należą do nich:

- wyłączenie indeksów i kontroli warunków integralności do czasu zakończenia importu danych (klauszula *DISABLE*),
- zrównoleglenie zadań kopiowania,
- wykorzystywanie równoległych zapytań.

Po zakończeniu migracji danych, zawsze należy sprawdzić pliki konfiguracyjne, w szczególności *tnsnames.ora* i *listener.ora*. Warto także skontrolować aplikacje, korzystające z nowej bazy danych, czy zostały w nich uaktualnione dane dotyczące nowego środowiska pracy, np. nazwa instancji.

Konieczne jest również zweryfikowanie parametrów inicjalizacyjnych bazy danych w celu sprawdzenia, czy stare parametry lub procedury nie pozostały aktywne. Także programy, wykorzystujące biblioteki serwera danych, powinny zostać ponownie



skompilowane. Taką kompilację można przeprowadzić podczas migracji lub też po uruchomieniu nowej bazy danych.

Nie należy zapomnieć o przeprowadzeniu testów funkcjonalnych i wydajnościowych. Jeżeli baza danych z nowymi danymi nie działa poprawnie lub nie zapewnia odpowiednio wysokiej wydajności wykonywania zapytań, wówczas konieczne będzie sprawdzenie wartości parametrów konfiguracyjnych bazy, obecności indeksów oraz innych jej obiektów. W praktyce obserwuje się przypadki, kiedy administrator nie potrafi rozwiązać problemów z nową bazą danych. W tej sytuacji należy rozważyć powrót do poprzedniej, z reguły starszej wersji bazy danych. Dlatego zawsze przed uruchomieniem procedury migracji należy utworzyć jej pełną kopię zapasową.

## 8.6. PYTANIA KONTROLNE

1. Wymień i scharakteryzuj techniki migracji danych do wersji Oracle 11g.
2. Podaj i scharakteryzuj metody pośrednie i bezpośrednie migracji danych.
3. Wymień etapy migracji metodą bezpośredniej aktualizacji ręcznej.
4. Jakie są wady i zalety metody bezpośredniej aktualizacji ręcznej?
5. Co to jest i do czego służy *Pre-Upgrade Information Tool*?
6. Jakie czynności należy wykonać po zakończeniu migracji?





## Literatura

---

- Alapati S. (2009), *Expert Oracle Database 11g Administration*, NY, Apres.
- Banachowski L. (2006), *Realizacja SZBD w Oracle*, dostępny 12.08.2011, <<http://edu.pjwstk.edu.pl/wyklady/szb/scb/wyklad14/w14.htm>>.
- Billings M., Keesling D. (2007), *Oracle Database 11g: High Availability Student Guide*, USA, Oracle.
- Bryla M., Loney K. (2010), *Oracle Database 11g. Podręcznik administratora baz danych*, Gliwice, Helion.
- Fernandez I. (2009), *Beginning Oracle Database 11g Administration*, NY, Apres.
- Gerard N. (2011), *Oracle Database - Installation 11g Release 2 (11.2) on Linux OEL 5 (X86)*, dostępny 9.08.2011, <[http://gerardnico.com/wiki/database/oracle/install\\_11gr2\\_oel\\_linux](http://gerardnico.com/wiki/database/oracle/install_11gr2_oel_linux)>.
- Hart M., Freeman R. (2008), *Oracle Database 10g RMAN. Archiwizacja i odzyskiwanie danych*, Gliwice, Helion.
- Kawa K. (2003), *Strojenie zapytań SQL czyli „można jeszcze szybciej”*, IX Konferencja PLOUG, Kościelisko.
- Kuhn D. (2010), *Pro Oracle Database 11g Administration*, NY, Apres.
- Loney K. (2005), *Oracle Database 10g. Kompendium administratora*. Oracle Press, Gliwice, Helion.
- Loney K. (2009), *Oracle Database. The complete reference*, NY, The McGraw-Hill Companies.
- Loney K., Bryla B. (2008), *Oracle Database 10g. Podręcznik administratora baz danych*, Gliwice, Helion.

- Magee B. (2003), *An Introduction to ORACLE SQL Statement Tuning*, dostępny 11.08.2011, <[http://www.billmagee.co.uk/oracle/sqltune/080\\_identify.html](http://www.billmagee.co.uk/oracle/sqltune/080_identify.html)>.
- Nyffenegger R. (2010), *Oracle SQL Hints*, dostępny 15.08.2011, <<http://www.adp-gmbh.ch/ora/sql/hints/index.html>>.
- Oracle Corporation (2008), *Oracle® Database*, w *2 Day DBA 11g Release 1 (11.1)*, Part No. B28301-03, dostępny 12.08.2011, <[http://download.oracle.com/docs/cd/B28359\\_01/server.111/b28301.pdf](http://download.oracle.com/docs/cd/B28359_01/server.111/b28301.pdf)>.
- Oracle Corporation (2011a), *Alter Tablespace*, w *Oracle® Database SQL Language Reference 11g Release 2 (11.2)*, Part Number E17118-04, dostępny 16.08.2011, <[http://download.oracle.com/docs/cd/E11882\\_01/server.112/e17118/statements\\_3002.htm#i2133310](http://download.oracle.com/docs/cd/E11882_01/server.112/e17118/statements_3002.htm#i2133310)>.
- Oracle Corporation (2011b), *Configuring Privilege and Role Authorization*, w *Oracle® Database Security Guide 11g Release 1 (11.1)*, Part Number B28531-14, dostępny 16.08.2011, <[http://download.oracle.com/docs/cd/B28359\\_01/network.111/b28531/authorization.htm#BABCADIB](http://download.oracle.com/docs/cd/B28359_01/network.111/b28531/authorization.htm#BABCADIB)>.
- Oracle Corporation (2011c), *Create User*, w *Oracle® Database SQL Language Reference 11g Release 1 (11.1)*, Part Number B28286-06, dostępny 17.08.2011, <[http://download.oracle.com/docs/cd/B28359\\_01/server.111/b28286/statements\\_8003.htm](http://download.oracle.com/docs/cd/B28359_01/server.111/b28286/statements_8003.htm)>.
- Oracle Corporation (2011d), *Creating a Database with the CREATE DATABASE Statement*, w *Oracle® Database Administrator's Guide 11g Release 1 (11.1)*, Part Number B28310-04, dostępny 7.08.2011, <[http://download.oracle.com/docs/cd/B28359\\_01/server.111/b28310/create003.htm#i1008816](http://download.oracle.com/docs/cd/B28359_01/server.111/b28310/create003.htm#i1008816)>.
- Oracle Corporation (2011e), *Creation Tablespaces* w *Oracle® Database Administrator's Guide 11g Release 1 (11.1)*, Part Number B28310-04, dostępny 16.08.2011, <[http://download.oracle.com/docs/cd/B28359\\_01/server.111/b28310/tspaces002.htm](http://download.oracle.com/docs/cd/B28359_01/server.111/b28310/tspaces002.htm)>.
- Oracle Corporation (2011f), *Dropping Datafiles*, w *Oracle® Database Administrator's Guide 11g Release 2 (11.2)*, Part Number E17120-07, dostępny 16.08.2011, <[http://download.oracle.com/docs/cd/E11882\\_01/server.112/e17120/dfiles006.htm#i1006556](http://download.oracle.com/docs/cd/E11882_01/server.112/e17120/dfiles006.htm#i1006556)>.
- Oracle Corporation (2011g), *Managing Tablespaces*, w *Oracle® Database Administrator's Guide 11g Release 1 (11.1)*, Part Number B28310-04, dostępny 10.08.2011, <[http://download.oracle.com/docs/cd/B28359\\_01/server.111/b28310/tspaces.htm#g1029288](http://download.oracle.com/docs/cd/B28359_01/server.111/b28310/tspaces.htm#g1029288)>.
- Oracle Corporation (2011h), *Managing the SYSAUX Tablespace*, w *Oracle® Database Administrator's Guide 11g Release 1 (11.1)*, Part Number B28310-04, dostępny 10.08.2011, <[http://download.oracle.com/docs/cd/B28359\\_01/server.111/b28310/tspaces010.htm](http://download.oracle.com/docs/cd/B28359_01/server.111/b28310/tspaces010.htm)>.

- Oracle Corporation (2011i), *Managing UNDO Tablespace*, w *Oracle® Database Administrator's Guide 11g Release 1 (11.1)*, Part Number B28310-04, dostępny 10.08.2011, <[http://download.oracle.com/docs/cd/B28359\\_01/server.111/b28310/undo005.htm](http://download.oracle.com/docs/cd/B28359_01/server.111/b28310/undo005.htm)>.
- Oracle Corporation (2011j), *Manually Creating Real Application Clusters Databases*, w *Oracle9i Real Application Clusters Setup and Configuration Release 2 (9.2)*, Part Number A96600-02, dostępny 20.08.2011, <[http://download.oracle.com/docs/cd/B10500\\_01/rac.920/a96600/mancrea.htm](http://download.oracle.com/docs/cd/B10500_01/rac.920/a96600/mancrea.htm)>.
- Oracle Corporation (2011k), *Memory Architecture*, w *Oracle® Database Concepts 11g Release 1 (11.1)*, Part Number B28318-06, dostępny 10.08.2011, <[http://download.oracle.com/docs/cd/B28359\\_01/server.111/b28318/memory.htm#i8451](http://download.oracle.com/docs/cd/B28359_01/server.111/b28318/memory.htm#i8451)>.
- Oracle Corporation (2011l), *Oracle Data Mining*, dostępny 9.08.2011, <<http://www.oracle.com/us/products/database/options/data-mining/index.html>>.
- Oracle Corporation (2011m), *Oracle Database 11g Editions*, dostępny 2.08.2011, <<http://www.oracle.com/us/products/database/product-editions-066501.html>>.
- Oracle Corporation (2011n), *Oracle Database Software Downloads*, dostępny 1.08.2011, <<http://www.oracle.com/technetwork/database/enterprise-edition/downloads/index.html?ssSourceSitel=ocomen>>.
- Oracle Corporation (2011o), *Part I Overview Backup and Recovery*, w *Oracle® Database Backup and Recovery User's Guide 11g Release 1 (11.1)*, Part Number B28270-03, dostępny 9.08.2011, <[http://download.oracle.com/docs/cd/B28359\\_01/backup.111/b28270/part\\_int.htm#sthref12](http://download.oracle.com/docs/cd/B28359_01/backup.111/b28270/part_int.htm#sthref12)>.
- Oracle Corporation (2011p), *Postinstallation Database Creation on Windows*, w *Oracle® Database Administrator's Guide 11g Release 1 (11.1)*, Part Number B28310-04, dostępny 7.08.2011, <[http://download.oracle.com/docs/cd/B28359\\_01/win.111/b32010/create.htm#NTQRF05011](http://download.oracle.com/docs/cd/B28359_01/win.111/b32010/create.htm#NTQRF05011)>.
- Oracle Corporation (2011q), *Protecting Your Database: Specifying Passwords For Users SYM and SYSTEM*, w *Oracle® Database Administrator's Guide 11g Release 1 (11.1)*, Part Number B28310-04, dostępny 8.08.2011, <[http://download.oracle.com/docs/cd/B28359\\_01/server.111/b28310/create004.htm#i1009226](http://download.oracle.com/docs/cd/B28359_01/server.111/b28310/create004.htm#i1009226)>.
- Oracle Corporation (2011r), *Specifying CREATE DATABASE Statement Clause*, w *Oracle® Database Administrator's Guide 11g Release 1 (11.1)*, Part Number B28310-04, dostępny 8.08.2011, <[http://download.oracle.com/docs/cd/B28359\\_01/server.111/b28310/create004.htm#i1009187](http://download.oracle.com/docs/cd/B28359_01/server.111/b28310/create004.htm#i1009187)>.
- Pelikant A. (2009), *Programowanie serwera Oracle 11g SQL i PL/SQL*, Gliwice, Helion.
- Preston W. (2008), *Archiwizacja i odzyskiwanie danych*, Gliwice, Helion.
- Price J. (2009), *Oracle Database 11g i SQL. Programowanie*, Gliwice, Helion.

- Session Statistic (2011), dostępny 11.08.2011, <<http://vsbabu.org/oracle/sect04.html>>.
- Tow D. (2004), *SQL. Optymalizacja*, Gliwice, Helion.
- Urman S., Hardman R., McLaughlin M. (2007), *Oracle Database 10g. Programowanie w języku PL/SQL*, Gliwice, Helion.
- Whalen E., Schroeter M. (2003), *Oracle. Optymalizacja wydajności*, Gliwice, Helion.
- Wilton P., Colby J. (2005), *SQL od podstaw*, Gliwice, Helion.
- Wrembel R. (1998), *Archiwizowanie danych i odtwarzanie bazy danych po awarii (cz. 1)*, Instytut Informatyki Politechniki Poznańskiej, dostępny 10.08.2011, <<http://www.ploug.org.pl/plougtki.php?action=read&p=7&a=6>>.

# Indeks

---

Administrowanie obszarami

pamięci, 53

Aktualizacja bezpośrednia, 189

Aktualizacja pośrednia, 190

ANALYZE, 140

Architektura bazy danych

Oracle, 41

Architektura RMAN, 164

ARCHIVELOG, 48

Asystent aktualizacji bazy

danych, 188

Błędy składni zapytań, 146

Błędy użycia indeksów, 152

CREATE CONTEXT, 92

CREATE DATABASE, 33

CREATE TABLESPACE, 59

CREATE USER, 74

Data Pump Export, 116

Data Pump Import, 122

Data Pump, 115

Database Configuration Assistant

(DBCA), 22

Database Upgrade Assistant

(DBUA), 186

DBMS\_RLS, 96

Dziennik alertów, 49

Dziennik powtórzeń, 47

Etapy instalacji systemu Oracle, 11

EXPLAIN PLAN, 136

FGAC, 89

Fizyczne struktury bazy danych, 47

FRA, 162

GRANT, 79

Instancja bazy danych, 30

Interaktywny Data Pump Export, 118

Interaktywny Data Pump Import, 123

Katalog odzyskiwania, 169

Kontekst – definicja, 91

Kontekst – tworzenie, 93

Kopia dziennik powtórzeń, 159

- Kopia inkrementacyjna, 173
- Kopia kumulacyjna, 165
- Kopia offline, 158
- Kopia online, 159
- Kopia pełna, 165
- Kopia różnicowa, 163
- Kopiowanie danych, 192
- Logiczne struktury bazy danych, 42
- Manualne tworzenie bazy danych, 28
- Metody dostępu do rekordów, 135
- Metody łączenia zbiorów rekordów, 135
- Migracja bezpośrednia, 187
- Migracja danych, 186
- Migracja pośrednia, 187
- Nadawanie uprawnień do obiektów, 81
- Nadawanie uprawnień systemowych, 79
- Narzędzie Export, 109
- Narzędzie Import, 112
- NOARCHIVELOG, 48
- Odbieranie uprawnień do obiektów, 81
- Odbieranie uprawnień systemowych, 79
- Odzyskiwanie bazy danych, 177
- Optymalizator kosztowy, 139
- Optymalizator planu wykonania, 138
- Optymalizator regułowy, 139
- ORADIM, 31
- Parametry Data Pump Export, 117
- Parametry Data Pump Import, 122
- Parametry eksportu, 110
- Parametry importu, 112
- Parametry SQL\*Loader, 126
- Parsowanie zapytania, 133
- Pełne odzyskiwanie bazy danych, 178
- PFILE, 30
- PGA, 51
- Plan wykonania zapytania, 136
- Plik haseł, 50
- Plik kontrolny, 48
- Plik sterujący, 48
- Plik śladu, 49
- Plik zrzutu, 108
- Pliki parametrów inicjalizacyjnych, 30
- Polityka bezpieczeństwa, 88
- Poziomy backupu, 165
- Poziomy eksportu, 110
- Precyzyjna kontrola dostępu, 89
- Predykat, 89
- Pre-Upgrade Information Tool, 188
- Przenoszenie plików danych pomiędzy przestrzeniami, 71
- Przestrzenie tabel, 42
- Przestrzenie trwałe, 42
- Przestrzenie tymczasowe, 43
- Przestrzenie wycofywania, 43
- Przydzielanie i odbieranie ról, 84
- Przygotowanie instalacji systemu, 18
- REVOKE, 79
- RLS, 89
- RMAN, 163
- Rodzaje instalacji systemu Oracle, 11
- Rodzaje polityk bezpieczeństwa, 90



- Rola, 82
- SGA, 52
- SID, 21
- SPFILE, 30
- Sposoby tworzenia kopii
  - zapasowych, 158
- Spójność eksportu danych, 111
- SQL\*Loader, 125
- Statystyki wykonania zapytania, 140
- Strategia bezpieczeństwa, 95
- Strojenie zapytania, 132
- Struktury logiczne pamięci, 51
- SYS\_CONTEXT, 93
- Szablony bazy danych, 20
- Techniki migracji, 186
- Tryby pracy Data Pump, 116
- Tworzenie kopii offline, 159
- Tworzenie kopii online, 160
- Tworzenie ról, 83
- Tworzenie użytkowników, 74
- Uprawnienia systemowe, 78
- USERENV, 92
- Virtual Private Database (VPD), 88
- VPD, tworzenie, 91
- Walidacja kopii zapasowych, 173
- Wersje instalacyjne systemu, 13
- Wirtualne prywatne bazy danych, 88
- Wskazówki eksportu, 113
- Wskazówki importu, 114
- Wskazówki optymalizatora, 142
- Zabezpieczenie na poziomie
  - wiersza, 89
- Zapytanie, sposób wykonania, 133
- Zarządzanie kontekstem, 93
- Zarządzanie obiektami bazy danych, 58
- Zarządzanie przestrzeniami tabel, 62
- Zarządzanie rolami, 84
- Zarządzanie użytkownikami, 76
- Zmienne kontekstu, 93