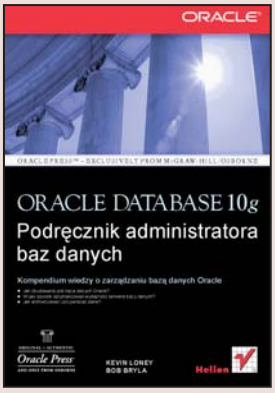
Oracle Database 10g. Podręcznik administratora baz danych



Autor: Kevin Loney, Bob Bryla
Tłumaczenie: Sławomir Dzieniszewski, Daniel Kaczmarek, Piotr Pilch
ISBN: 978-83-246-1205-5
Tytuł oryginału: Oracle Database 10g DBA Handbook
Format: B5, stron: około 750
Przykłady na ftp: 14 kB

Kompendium wiedzy o zarządzaniu bazą danych Oracle

- Jak zbudowana jest baza danych Oracle?
- W jaki sposób optymalizować wydajność serwera bazy danych?
- Jak archiwizować i przywracać dane?

Baza danych Oracle wykorzystywana jest wszędzie tam, gdzie kluczowymi aspektami są bezpieczeństwo danych, stabilność systemu oraz szybkość działania. Rozwiązania stosowane w tej niemal legendarnej już bazie danych gwarantują to wszystko. Każda nowa wersja bazy Oracle jest wyposażona w kolejne, niezwykle przydatne funkcje i wytycza nowe standardy w zakresie przechowywania informacji i zarządzania nimi.

Ogromne możliwości tego systemu wiążą się jednak z koniecznością przyswojenia przez jego użytkowników ogromnej wiedzy, zawartej w dziesiątkach tomów dokumentacji, z których każdy szczegółowo opisuje inne funkcje Oracla'a. Praca administratora, często wykonywana pod presją czasu, rzadko pozwala na wertowanie tysięcy stron w poszukiwaniu rozwiązania jednego problemu – w takich przypadkach o wiele bardziej przydatne okazuje się zestawienie najbardziej potrzebnych informacji.

Książka „Oracle Database 10g. Podręcznik administratora baz danych" to zbiór wszystkich informacji niezbędnych dla zarządzającego tym potężnym systemem. Opisano w niej architekturę bazy Oracle 10g, proces instalacji i aktualizacji oraz metody rozwiązywania zadań i problemów związanych z zarządzaniem, optymalizacją, skalowaniem i bezpieczeństwem bazy danych. Czytając ją, poznasz nowe narzędzia administracyjne, sposoby planowania przestrzeni tabel, zarządzania transakcjami, optymalizowania zapytań i archiwizowanie danych. Dowiesz się także, jak postępować podczas awarii serwera, jak zabezpieczać dane przed nieautoryzowanym dostępem i jak zarządzać rozproszonymi bazami danych.

- Architektura systemu Oracle
- Struktury przechowywania danych
- Instalacja bazy Oracle 10g
- Planowanie przestrzeni tabel
- Zarządzanie bazą danych
- Monitorowanie zużycia przestrzeni dyskowej
- Optymalizacja wydajności bazy
- Klastrowanie bazy za pomocą Real Application Clusters
- Archiwizacja i przywracanie danych
- Zarządzanie bazami rozproszonymi

Wydawnictwo Helion
ul. Kościuszki 1c
44-100 Gliwice
tel. 032 230 98 63
e-mail: helion@helion.pl

Spis treści

Rozdział 10.

Bezpieczeństwo i monitorowanie bazy danych

Aby skutecznie chronić dane — jedną z najważniejszych części majątku firmy — administrator bazy danych musi doskonale orientować się w sposobach, w jakie Oracle zabezpiecza dane firmowe, oraz umieć posługiwać się dostępnymi narzędziami. Narzędzia i mechanizmy udostępniane przez bazę Oracle można podzielić na trzy szerokie kategorie: uwierzytelnianie, autoryzacja i monitorowanie.

Uwierzytelnianie (ang. *authentication*) to zbiór metod, które służą do identyfikowania użytkowników uzyskujących dostęp do bazy danych oraz zapewnienia, że użytkownicy ci rzeczywiście są tymi, za kogo się podają bez względu na rodzaj wywoływanych przez nich zasobów bazy danych. Nawet jeśli celem użytkownika jest tylko odczytanie danych na temat menu restauracji, kluczowym punktem jest prawidłowe przedstawienie się bazie danych. Jeśli na przykład działająca w sieci WWW aplikacja bazodanowa prezentuje dane dostosowane do bieżącego konta użytkownika, trzeba zapewnić, że wyświetlone menu będzie dotyczyło restauracji w Gliwicach, a nie w Łodzi!

Autoryzacja (ang. *authorization*) odpowiada za udostępnianie różnorodnych obiektów bazy danych już po uwierzytelnieniu się w bazie. Niektórzy użytkownicy mogą posiadać autoryzację do uruchamiania raportów na podstawie tabeli z dziennymi danymi sprzedażowymi, zaś inni użytkownicy mogą być programistami i jako tacy muszą posiadać możliwość tworzenia tabel i raportów; jeszcze inni z kolei mogą być uprawnieni jedynie do przeglądania menu na dany dzień. Niektórzy użytkownicy mogą w ogóle nie potrzebować logować się do bazy danych, lecz ich schemat może posiadać szereg tabel używanych przez konkretne aplikacje, odpowiedzialne na przykład za naliczanie płac albo zarządzanie należnościami. Dodatkowe metody autoryzacji są używane w przypadku administratorów bazy danych ze względu na szerokie uprawnienia, jakie oni posiadają. Dodatkowy poziom autoryzacji jest wykorzystywany choćby dlatego, że administrator bazy danych może wyłączać i włączać bazę.

Autoryzacja dalece wykracza poza samo przyznawanie dostępu do tabeli albo raportu. Zakres autoryzacji obejmuje również prawa do używania w bazie danych zasobów systemowych oraz uprawnienia do wykonywania w bazie określonych czynności. Dany użytkownik bazy może być na przykład uprawniony do używania procesora jedynie przez 15 sekund w jednej sesji albo do pozostawania w trybie jałowym jedynie przez pięć minut, zanim nastąpi jego odłączenie od bazy danych. Inny użytkownik bazy danych może z kolei posiadać uprawnienie do tworzenia i usuwania tabel w dowolnym schemacie użytkownika, a jednocześnie może nie mieć możliwości tworzenia synonimów albo widoków na tabelach danych słownikowych. Dzięki szczegółowej kontroli dostępu administrator może odpowiednio sterować zakresem dostępu do obiektów bazy danych. Na przykład standardowe uprawnienia do obiektów pozwalają użytkownikowi uzyskiwać dostęp do całego wiersza tabeli albo w ogóle ten dostęp blokować. Kontrolowanie dostępu na różnych poziomach pozwala administratorowi bazy danych na tworzenie polityki implementowanej w postaci procedury składowanej, która ogranicza zakres dostępu zależnie od pory dnia, pochodzenia żądania, odczytywanej kolumny tabeli lub z uwzględnieniem wszystkich trzech kryteriów łącznie.

Na końcu punktu poświęconego autoryzacji w bazie danych przedstawiony zostanie krótki przykład dotyczący **wirtualnej prywatnej bazy danych** (ang. *Virtual Private Database — VPD*). Wirtualne prywatne bazy danych udostępniają metody definiowania, ustawiania i odczytywania atrybutów bazy danych z zastosowaniem predykatów (zwykle klauzul `WHERE`), dzięki którym można sprawować szczegółową kontrolę nad danymi odczytywanymi przez aplikację użytkownika lub zwracanymi do niej.

Monitorowanie (ang. *auditing*) w bazie danych Oracle obejmuje szereg różnych poziomów monitorowania bazy. Na poziomie ogólnym można powiedzieć, że monitorowanie polega na rejestrowaniu skutecznych i nieskutecznych prób zalogowania, uzyskania dostępu do obiektu czy wykonania jakiejś operacji. Począwszy od wersji Oracle9*i*, monitorowanie szczegółowe (ang. *fine-grained auditing — FGA*) polega nie tylko na rejestrowaniu dostępu do obiektów, ale również na rejestrowaniu kolumn tabeli udostępnianych w trakcie operacji wstawiania, uaktualniania lub usuwania danych. Monitorowanie szczegółowe ma się tak do monitorowania, jak szczegółowa kontrola dostępu ma się do standardowej autoryzacji: zapewnia o wiele bardziej precyzyjną kontrolę i bardziej szczegółowe informacje na temat odczytywanych obiektów lub wykonywanych czynności.

Administratorzy baz danych powinni korzystać z mechanizmów monitorowania z rozwagą, aby nie zagubić się pod nawałem danych generowanych w trakcie monitorowania ani nie obciążać zbytnio bazy danych zaimplementowanymi mechanizmami monitorującymi. Z drugiej strony dzięki monitorowaniu można lepiej chronić aktywa firmy przez kontrolowanie tego, kto używa poszczególnych zasobów, o jakiej porze i jak często, a także przez sprawdzanie, czy próba uzyskania dostępu zakończyła się powodzeniem, czy niepowodzeniem. Zatem monitorowanie jest kolejnym narzędziem, którego administrator bazy danych powinien używać na co dzień, aby kontrolować stan zabezpieczeń bazy.

Zabezpieczenia poza bazą danych

Wszystkie rozwiązana prezentowane w dalszej części niniejszego rozdziału na nic się nie zdadzą, jeśli dostęp do systemu operacyjnego nie zostanie odpowiednio zabezpieczony albo używany sprzęt nie będzie się znajdował w zabezpieczonym miejscu. W tym punkcie przedstawionych zostanie kilka elementów występujących na zewnątrz bazy danych, które muszą zostać odpowiednio zabezpieczone, zanim w ogóle będzie można zacząć myśleć o bezpieczeństwie bazy danych.

Na poniższej liście przedstawiono elementy spoza bazy danych, o których zabezpieczenie należy zadbać.

- **Bezpieczeństwo systemu operacyjnego.** Jeśli baza danych Oracle nie jest uruchamiana na przeznaczonej tylko dla niej infrastrukturze sprzętowej i wyłącznie na kontach użytkowników `root` i `oracle`, wówczas konieczne jest przeanalizowanie i wdrożenie mechanizmów zabezpieczających system operacyjny. Należy zapewnić, że oprogramowanie jest instalowane na koncie użytkownika `oracle`, a nie `root`. Właścicielem oprogramowania i plików bazy danych można również uczynić użytkownika innego niż `oracle`, aby utrudnić zadanie ewentualnemu włamywaczowi. Pliki oprogramowania i bazy danych powinny być dostępne do odczytu wyłącznie dla konta `oracle` oraz grupy, do której użytkownik ten należy. Dla wszystkich plików wykonywalnych, które tego nie potrzebują, należy wyłączyć bit SUID (set UID albo możliwość uruchamiania z uprawnieniem `root`). Haseł systemu operacyjnego ani bazy Oracle nie powinno się wysyłać do innych użytkowników pocztą elektroniczną zwykłym tekstem. Należy również wyłączyć wszystkie usługi systemowe, które nie są potrzebne do pracy bazy danych, takie jak telnet czy FTP.
- **Zabezpieczanie nośników z kopiami zapasowymi.** Należy zapewnić, by nośniki kopii zapasowych — nośniki taśmowe, dyski twarde czy płyty CD-ROM, DVD-ROM — były dostępne jedynie dla wąskiego kręgu osób. Odpowiednio zabezpieczony system operacyjny i trudne do złamania, zaszyfrowane hasła nie będą mieć większego znaczenia, jeśli potencjalny włamywacz będzie mógł zdobyć kopie zapasowe bazy danych i załadować je na innym serwerze. Ta sama zasada dotyczy wszystkich serwerów, na których znajdują zreplikowane dane z bazy danych.
- **Audyty bezpieczeństwa.** Monitorowanie pracy osób, które pracują na poufnych danych — bez względu na to, czy jest to administrator bazy danych, audytor czy administrator systemu operacyjnego — jest bezwzględnie konieczne.
- **Szkolenia z zakresu bezpieczeństwa.** Wszyscy użytkownicy powinni rozumieć polityki bezpieczeństwa oraz dozwolone metody korzystania z infrastruktury IT. Wymóg zaznajomienia się użytkowników z zasadami bezpieczeństwa opisanymi w politykach i konieczność stosowania ich na co dzień wskazuje jednocześnie na kluczową rolę i wartość danych posiadanych przez firmę, w tym również danych przechowywanych w bazie. Dobrze wyszkolony użytkownik będzie mniej podatny na udostępnienie systemu potencjalnemu włamywaczowi stosującemu sztuczki psychologiczne.
- **Kontrola dostępu do sprzętu.** Wszystkie komputery, na których pracuje baza danych, powinny znajdować się w bezpiecznym miejscu, dostępnym jedynie dla osób posiadających karty magnetyczne lub znających kody dostępu.

Metody uwierzytelniania w bazie danych

Zanim baza danych będzie mogła udostępnić osobie lub aplikacji określone obiekty lub uprawnienia, osoba taka lub aplikacja musi najpierw zostać uwierzytelniona. Inaczej mówiąc, konieczne jest ustalenie tożsamości użytkownika, który żąda dostępu do bazy danych.

W niniejszym punkcie opisana zostanie podstawowa metoda, za pomocą której użytkownikowi daje się dostęp do bazy danych — jest to konto użytkownika, o którym mówi się również: **uwierzytelnienie w bazie danych**. Dodatkowo pokażemy sposób, w jaki można zmniejszyć liczbę haseł koniecznych do zapamiętania przez użytkownika — w tym celu należy umożliwić systemowi operacyjnemu uwierzytelnianie użytkownika, zapewniając mu tym samym automatyczne połączenie z bazą danych. Wykorzystanie trójwarstwowego systemu przy użyciu serwera aplikacji, uwierzytelniania w sieci albo mechanizmu Oracle Identity Management pozwala jeszcze bardziej ograniczyć liczbę potrzebnych haseł. Na koniec opiszemy sposób używania pliku haseł do uwierzytelniania administratora bazy danych w sytuacji, gdy baza jest wyłączona i przez to nie działają również usługi uwierzytelniania.

Uwierzytelnianie w bazie danych

W środowisku, w którym sieć jest zabezpieczona przed otoczeniem zewnętrznym systemem zapór sieciowych, a ruch w sieci przesyłany między klientem i serwerem bazodanowym jest w jakiś sposób szyfrowany, uwierzytelnianie w bazie danych jest najczęściej stosowaną i najprostszą metodą udostępniania bazy użytkownikowi. Wszystkie dane niezbędne do uwierzytelnienia użytkownika znajdują się w tabeli w przestrzeni tabel `SYSTEM`.

Wykonanie szczególnych operacji w bazie danych, takich jak jej uruchomienie albo wyłączenie, wymaga zastosowania innych, bardziej bezpiecznych form uwierzytelnienia: poprzez uwierzytelnienie w systemie operacyjnym albo przy użyciu plików haseł.

W uwierzytelnieniu sieciowym stosuje się zewnętrzne usługi uwierzytelniające, takie jak Distributed Computing Environment (DCE), Kerberos, infrastrukturę klucza publicznego (Public Key Infrastructure — PKI) oraz Remote Authentication Dial-In User Service (RADIUS). Uwierzytelnianie trójwarstwowe, choć na pierwszy rzut oka może się wydać podobne do uwierzytelnienia sieciowego, różni się od niego w ten sposób, że warstwa pośrednia — na przykład Oracle Application Server — uwierzytelnia użytkownika, jednocześnie utrzymując tożsamość klienta na serwerze. Dodatkowo warstwa pośrednia udostępnia usługi buforowania połączeń oraz implementuje logikę biznesową dla klienta.

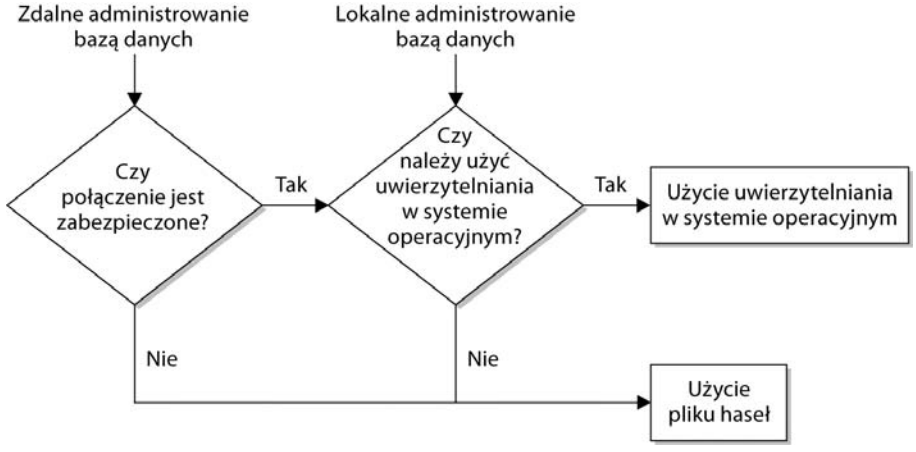
W dalszej części tego rozdziału, w punkcie pod tytułem „Konta użytkowników”, przedstawione zostaną wszystkie dostępne dla administratorów baz danych metody tworzenia kont w bazie danych dla celów uwierzytelniania.

Uwierzytelnianie administratora bazy danych

Baza danych nie zawsze jest dostępna i przez to nie zawsze może uwierzytelnić administratora bazy. Sytuacje takie mają miejsce na przykład wówczas, gdy nastąpiło nieplanowane odcięcie prądu albo baza została wyłączona w celu sporządzenia zapasowej kopii offline.

Z myślą o tego rodzaju przypadkach Oracle używa **pliku haseł** (ang. *password file*), w którym przechowywana jest lista użytkowników bazy danych uprawnionych do wykonywania czynności takich jak uruchamianie i wyłączanie bazy danych, inicjowanie procesu sporządzania kopii zapasowych itp.

Alternatywnie administrator bazy danych może skorzystać z uwierzytelniania w systemie operacyjnym, o którym więcej powiemy w następnym punkcie. Diagram zaprezentowany na rysunku 10.1 wskazuje opcje dostępne dla administratora bazy danych, z których można wybrać tę, która będzie najlepiej sprawdzać się w danym środowisku.



Rysunek 10.1. *Diagram metod uwierzytelniania*

W przypadku nawiązywania lokalnego połączenia z serwerem trzeba przede wszystkim zastanowić się, które rozwiązanie będzie wygodniejsze: użycie tego samego konta dla systemu operacyjnego i serwera Oracle czy utrzymywanie pliku haseł. W przypadku administracji prowadzonej zdalnie metodę uwierzytelniania powinno się wybrać, mając na uwadze przede wszystkim bezpieczeństwo połączenia. Bez odpowiednio zabezpieczonego połączenia potencjalny włamywacz będzie mógł bez trudu podszyć się pod użytkownika posiadającego na serwerze konto administratora i uzyskać w ten sposób pełny dostęp do bazy danych, uwierzytelniając się w systemie operacyjnym.


Uwaga

Jeżeli do uwierzytelniania będzie używany plik haseł, należy się upewnić, że plik ten będzie się znajdował w katalogu dostępnym wyłącznie dla administratorów systemu operacyjnego oraz użytkownika lub grupy będącej właścicielem instalacji Oracle.

Bardziej szczegółowe informacje na temat uprawnień systemowych zostaną przedstawione w dalszej części tego rozdziału. Teraz wystarczy wspomnieć, że istnieją dwa uprawnienia systemowe, dla których przeznaczona jest specjalna metoda uwierzytelniania; są to `SYSDBA` i `SYSOPER`. Administrator posiadający uprawnienie `SYSOPER` może uruchamiać i wyłączać bazę danych, wykonywać kopie zapasowe online i offline, archiwizować aktualne pliki dziennika powtórzeń i łączyć się z bazą danych, gdy pracuje ona w trybie `RESTRICTED SESSION`. Uprawnienie `SYSDBA` posiada wszystkie prawa `SYSOPER` i dodatkowo pozwala na tworzenie baz danych oraz nadawanie uprawnień `SYSDBA` i `SYSOPER` innym użytkownikom bazy danych.

Aby połączyć się z bazą danych z poziomu sesji SQL*Plus, do polecenia connect należy dopisać parametr AS SYSDBA lub AS SYSOPER. Oto odpowiedni przykład:

```
D:\TEMP> sqlplus /nolog

SQL*Plus: Release 10.1.0.1.0 - Production on Sun Feb 15 20:52:10 2004

Copyright (c) 1982, 2004, Oracle Corporation. All rights reserved.

SQL> connect rjb/rjb999@dw as sysdba
Connected.

SQL>
```

Oprócz odmiennego zbioru uprawnień dostępnych dla użytkowników łączących się jako SYSDBA i SYSOPER kolejną różnicą między obydwoma uprawnieniami jest domyślny schemat przypisywany w momencie łączenia się z bazą danych. Użytkownicy, którzy łączą się z bazą jako SYSDBA, są traktowani jak użytkownicy SYS, natomiast w przypadku uprawnienia SYSOPER użytkownik łączący się z bazą danych jest traktowany jako użytkownik PUBLIC:

```
SQL> show user
USER is "SYS"
```

Podobnie jak w przypadku każdego innego żądania połączenia z bazą danych, również w poleceniu sqlplus można dopisywać nazwę użytkownika i hasło, a także słowo kluczowe SYSDBA albo SYSOPER:

```
D:\TEMP> sqlplus rjb/rjb999@dw as sysdba
```

W trakcie domyślnego procesu instalacji bazy danych Oracle z wykorzystaniem narzędzia Oracle Universal Installer i szablonowej bazy danych albo z wykorzystaniem narzędzia Database Creation Assistant automatycznie utworzony zostanie plik haseł. Zdarzają się jednak sytuacje, gdy plik haseł trzeba utworzyć na nowo, jeśli zostanie on usunięty albo ulegnie uszkodzeniu. Polecenie orapwd tworzy plik haseł z pojedynczym wpisem dla użytkownika SYS oraz dodatkowymi opcjami, jeśli zostaną wskazane w poleceniu:

```
[oracle@dw10g oracle]$ orapwd
Usage: orapwd file=<fname> password=<password> entries=<users> force=<y/n>
  where
    file - name of password file (mand),
    password - password for SYS (mand),
    entries - maximum number of distinct DBAs and OPERs (opt),
    force - whether to overwrite existing file (opt)
  There are no spaces around the equal-to (=) character.
[oracle@dw10g oracle]$
```

Gdy plik haseł zostanie ponownie utworzony, konieczne będzie nadanie uprawnień SYSDBA i SYSOPER tym użytkownikom bazy danych, którzy już wcześniej posiadali wspomniane uprawnienia. Dodatkowo, jeśli hasło podane w poleceniu orapwd nie jest takie samo jak hasło użytkownika SYS w bazie danych, w momencie następnego połączenia się z bazą danych jako użytkownik SYS trzeba będzie to hasło zmienić. Operacja ta ma na celu utrzymanie synchronizacji między hasłem w bazie danych a hasłem w pliku haseł.

Systemowy parametr inicjalizacyjny REMOTE_LOGIN_PASSWORDFILE wyznacza sposób używania hasła w instancji bazy danych. Parametr może przyjmować jedną z trzech wartości: NONE, SHARED albo EXCLUSIVE.

Jeśli wartością parametru jest NONE, Oracle będzie ignorował wszelkie istniejące pliki haseł. Wszyscy uprzywilejowani użytkownicy będą musieli się uwierzytelniać w inny sposób, na przykład przez uwierzytelnianie w systemie operacyjnym, o którym więcej zostanie powiedziane w następnym punkcie.

Wartość SHARED oznacza, że kilka baz danych może korzystać z tego samego pliku haseł, lecz na podstawie pliku haseł uwierzytelniany jest wyłącznie użytkownik SYS, którego hasła nie można zmieniać. W efekcie nie jest to najbezpieczniejsze rozwiązanie, lecz dzięki niemu administrator baz danych może utrzymywać więcej niż jedną bazę z jednego konta SYS.

> Jeśli konieczne jest użycie współużytkowanego pliku haseł, należy zapewnić, że hasło użytkownika SYS posiada co najmniej osiem znaków i zawiera kombinację znaków alfabetycznych, cyfr i znaków specjalnych. W ten sposób znacznie utrudni się skuteczne przeprowadzenie ataków siłowych.

Wartość EXCLUSIVE powoduje, że plik haseł jest związany tylko z jedną bazą danych i w pliku tym mogą znajdować się dane innych kont użytkowników bazy. Od razu po utworzeniu pliku haseł warto jest przypisać tę wartość parametrowi inicjalizacyjnemu, aby zmaksymalizować bezpieczeństwo połączeń użytkowników SYSDBA i SYSOPER.

Dynamiczny widok wydajności V$PWFILE_USERS zawiera listę wszystkich użytkowników bazy danych, którzy posiadają uprawnienia SYSDBA lub SYSOPER. Na przykład:

```
SQL> select * from v$pwfile_users;

USERNAME     SYSDBA   SYSOPER
------------ -------- --------
SYS          TRUE     TRUE
RJB          TRUE     FALSE
SYSTEM       TRUE     FALSE
```

Uwierzytelnianie w systemie operacyjnym

Jeśli administrator bazy danych zdecyduje się na zaimplementowanie **uwierzytelniania w systemie operacyjnym** (ang. *operating system authentication*), wówczas użytkownik bazy danych zostanie automatycznie połączony z bazą danych, gdy zastosuje polecenie SQL o następującej składni:

```
SQL> sqlplus /
```

W podobny sposób z bazą danych łączy się administrator; brakuje tu jedynie klauzuli as sysdba albo as sysoper. Główna różnica polega na tym, że w celu uwierzytelnienia wykorzystywane są metody uwierzytelniania konta używane przez system operacyjny, a nie generowany i utrzymywany przez Oracle plik haseł.

Tak naprawdę również administratorzy mogą używać uwierzytelnienia w systemie operacyjnym, aby łączyć się z bazą jako `as sysdba` albo `as sysoper`. Jeśli login administratora w systemie operacyjnym należy do grupy `dba` Uniksa (albo `ORA_DBA` w systemie Windows), administrator będzie mógł połączyć się z bazą danych przy użyciu klauzuli `as sysdba`. Analogicznie, jeśli login w systemie operacyjnym należy do grupy `oper` w systemie Unix (albo do grupy `ORA_OPER` w systemie Windows), administrator będzie mógł się łączyć z bazą danych przy użyciu klauzuli `as sysoper` bez konieczności używania pliku haseł serwera Oracle.

Oracle Server zakłada, że jeśli użytkownik został uwierzytelniony na koncie systemu operacyjnego, to jest on uwierzytelniony również w bazie danych. W przypadku korzystania z uwierzytelniania w systemie operacyjnym Oracle nie musi utrzymywać haseł w bazie, nadal jednak musi przechowywać nazwy użytkowników. Nazwy te są nadal potrzebne do ustawienia domyślnego schematu i przestrzeni tabel, a także dla celów monitorowania.

W domyślnej instalacji bazy danych Oracle 10*g*, a także w poprzednich wersjach bazy, uwierzytelnianie w systemie operacyjnym jest dostępne wówczas, gdy użytkownika utworzy się przy użyciu klauzuli `identified externally`. Prefiks nazwy użytkownika bazy danych musi być zgodny z wartością parametru inicjalizacyjnego `OS_AUTHENT_PREFIX`. Wartością domyślną parametru jest `OPS$`. Oto odpowiedni przykład:

```
SQL> create user ops$corie identified externally;
```

Gdy użytkownik zaloguje się do systemu operacyjnego na konto o nazwie `CORIE`, zostanie również automatycznie uwierzytelniony w bazie danych Oracle, tak jakby w trakcie uwierzytelnienia w bazie danych utworzone zostało konto `OPS$CORIE`.

Jeśli parametrowi `OS_AUTHENT_PREFIX` przypisany zostanie pusty ciąg znaków, administrator bazy danych i administrator kont systemu operacyjnego będą mogli zastosować identyczne nazwy użytkowników dla celów uwierzytelniania na zewnątrz bazy.

Użycie klauzuli `identified globally` będzie miało podobny efekt jak użycie klauzuli `identified externally`, to znaczy uwierzytelnienie będzie się dokonywać poza bazą danych. Jednak uwierzytelnienie użytkownika identyfikowanego globalnie jest wykonywane przez usługę katalogową, na przykład Oracle Internet Directory (OID). OID ułatwia administratorom baz danych utrzymywanie kont oraz pozwala na stosowanie jednego loginu przez tych użytkowników, którzy potrzebują dostępu do więcej niż jednej bazy danych lub usługi.

Uwierzytelnianie sieciowe

Uwierzytelnianie przez usługę sieciową to kolejna dostępna dla administratorów baz danych opcja uwierzytelniania użytkowników w bazie. Wprawdzie szczegółowy opis tego mechanizmu wykracza poza zakres tej książki, lecz poniżej przedstawiony zostanie krótki opis każdej metody uwierzytelniania sieciowego i jej komponentów. Do komponentów tych należą Secure Sockets Layer (SSL), Distributed Computing Environment (DCE), Kerberos, PKI, RADIUS oraz usługi katalogowe.

Protokół Secure Sockets Layer

Secure Sockets Layer (SSL) to protokół, którego zręby opracowała firma Netscape Development Corporation z myślą o wykorzystaniu go w przeglądarkach internetowych. Ponieważ protokół ten jest publicznym standardem i dostęp do jego kodu źródłowego jest otwarty, cały czas jest on rozwijany przez społeczność programistów w celu zapewnienia, że nie występują w nim żadne dziury ani „tylne drzwi", które mogłyby zagrozić jego funkcjonowaniu.

W celu przeprowadzenia uwierzytelniania konieczne jest co najmniej istnienie po stronie serwera certyfikatu. SSL pozwala również na uwierzytelnianie klientów, lecz skonfigurowanie odpowiednich certyfikatów może być złożonym zadaniem administracyjnym.

Użycie SSL na TCP/IP wymaga wprowadzenia niewielkich poprawek w konfiguracji usługi nasłuchującej, a konkretnie dodania w pliku *listener.ora* kolejnego protokołu (TCPS) na porcie o innym numerze. W poniższym fragmencie stanowiącym konfigurację utworzoną przez Oracle Net Configuration Assistant (netca), usługa nasłuchująca o nazwie `LISTENER` na serwerze dw10g będzie przyjmować dane TCP na porcie numer 1521 oraz dane SSL TCP na porcie 2484:

```
# listener.ora Network Configuration File:
    /u01/app/oracle/product/10.1.0/network/admin/listener.ora
# Generated by Oracle configuration tools.
SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (SID_NAME = PLSExtProc)
      (ORACLE_HOME = /u01/app/oracle/product/10.1.0)
      (PROGRAM = extproc)
    )
    (SID_DESC =
      (GLOBAL_DBNAME = dw.world)
      (ORACLE_HOME = /u01/app/oracle/product/10.1.0)
      (SID_NAME = dw)
    )
  )

LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = TCP)(HOST = dw10g)(PORT = 1521))
      )
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = TCPS)(HOST = dw10g)(PORT = 2484))
      )
    )
  )
```

Distributed Computing Environment

Distributed Computing Environment (DCE) udostępnia oprócz usługi bezpieczeństwa szereg innych usług, takich jak zdalne wywoływanie procedur, rozproszone usługi obsługi plików oraz rozproszone usługi synchronizacji czasu. DCE obsługuje rozproszone aplikacje w heterogenicznych środowiskach na wszystkich ważniejszych platformach programowych i sprzętowych.

DCE to jeden z protokołów, który obsługuje mechanizm jednokrotnego rejestrowania się (ang. *single sign-on — SSO*). Oznacza to, że gdy użytkownik uwierzytelni się w DCE, wówczas uzyska też bezpieczny dostęp do dowolnej bazy danych Oracle skonfigurowanej w DCE bez konieczności podawania nazwy użytkownika oraz hasła.

Kerberos

Kerberos to kolejny darzony zaufaniem system uwierzytelniania od dostawcy zewnętrznego, który podobnie jak DCE zawiera mechanizmy SSO. Oracle, a konkretnie Oracle Advanced Security w wersji Enterprise Edition bazy Oracle Database 10*g*, w pełni obsługuje Kerberos w wersji 5.

Podobnie jak w przypadku innych rozwiązań uwierzytelniających warstwy pośredniej, podstawowym założeniem Kerberosa jest to, by hasła nigdy nie były przesyłane w sieci, ale by cała operacja uwierzytelniania odbywała się na serwerze Kerberos. W terminologii Kerberosa hasło to „wspólna tajemnica".

Infrastruktura klucza publicznego

Infrastruktura klucza publicznego (ang. *Public Key Infrastructure — PKI*) składa się z kilku komponentów. PKI zaimplementowano przy użyciu protokołu SSL i bazuje ona na koncepcji sekretnych kluczy prywatnych oraz powiązanych z nimi kluczy publicznych, które zabezpieczają komunikację między klientem i serwerem.

W roli usług identyfikujących i uwierzytelniających PKI wykorzystuje certyfikaty oraz instytucje certyfikujące (ang. *certificate authorities — CA*). Mówiąc w skrócie, certyfikat to klucz publiczny podmiotu, którego wiarygodność jest potwierdzona przez podmiot zewnętrzny (instytucję certyfikującą); certyfikat zawiera nazwę użytkownika, data wygaśnięcia, klucz publiczny itd.

RADIUS

Remote Authentication Dial-In User Sernice (RADIUS) to lekki protokół używany dla celów uwierzytelniania i autoryzacji. W środowisku Oracle serwer Oracle Server jest klientem serwera RADIUS, gdy z klienta Oracle wpłynie żądanie autoryzacji.

Każdą metodę obsługującą standard RADIUS — karty uwierzytelniające, karty chipowe czy też SecurID ACE — można bez trudu dodać do serwera RADIUS jako nową metodę uwierzytelniania bez konieczności wprowadzania jakichkolwiek zmian w plikach konfiguracyjnych klienta i serwera, takich jak *sqlnet.ora*.

Uwierzytelnianie trójwarstwowe

W środowisku trójwarstwowym lub wielowarstwowym serwer aplikacji może udostępniać usługi uwierzytelniające dla klienta, a także wspólny interfejs dla serwera bazy danych — nawet jeśli klienci używają różnorodnych przeglądarek albo „grubych" aplikacji klienckich. Serwer aplikacji jest natomiast uwierzytelniany w bazie danych i pozwala klientowi łączyć się z bazą, a więc utrzymuje tożsamość klienta we wszystkich warstwach.

W środowiskach wielowarstwowych zarówno użytkownikom, jak i warstwom pośrednim nadaje się jak najmniejsze uprawnienia niezbędne do wykonania przynależnych im zadań. Warstwie pośredniej nadaje się uprawnienia do wykonywania czynności w imieniu użytkownika. Służy do tego polecenie podobne do poniższego:

```
alter user kmourgos
     grant connect through oes_as
     with role all except ordmgmt;
```

W powyższym przykładzie usłudze OES_AS serwera aplikacji nadawane jest uprawnienie do wykonywania czynności w imieniu użytkownika bazy danych KMOURGOS. Użytkownikowi KMOURGOS przypisano szereg ról. Wszystkie role można włączać za pośrednictwem serwera aplikacji; nie dotyczy to jedynie roli ORDMGMT. Gdy KMOURGOS nawiąże połączenie za pośrednictwem serwera aplikacji, będzie posiadał dostęp z sieci WWW do wszystkich tabel i uprawnień nadanych mu w postaci ról oprócz funkcji zarządzania zamówieniami. Z uwagi na charakter reguł biznesowych funkcjonujących w firmie dostęp do aplikacji zarządzania zamówieniami musi być przydzielany w ramach bezpośredniego połączenia z bazą danych. Role zostaną opisane szczegółowo w punkcie pod tytułem „Tworzenie, przypisywanie i utrzymywanie ról" w dalszej części tego rozdziału.

Uwierzytelnianie po stronie klienta

Uwierzytelnianie po stronie klienta to jeden ze sposobów uwierzytelniania użytkowników w środowisku wielowarstwowym, lecz Oracle zdecydowanie odradza to rozwiązanie, chyba że wszyscy klienci znajdują się w bezpiecznej sieci za zaporami sieciowymi, do której jakikolwiek dostęp spoza zapory jest zablokowany. Dodatkowo użytkownicy nie powinni mieć żadnych praw administratora na stacjach roboczych, które mogą łączyć się z bazą danych.

Jeżeli użytkownika bazy Oracle utworzono z atrybutem IDENTIFIED EXTERNALLY i parametr inicjalizacyjny REMOTE_OS_AUTHENT ma wartość TRUE, wówczas potencjalny włamywacz bez trudu może uwierzytelnić się na stacji roboczej na koncie użytkownika lokalnego odpowiadającym kontu użytkownika Oracle, a w konsekwencji uzyskać dostęp do bazy danych.

Z tego właśnie powodu zdecydowanie zaleca się, by parametr inicjalizacyjny REMOTE_OS_AUTHENT miał wartość FALSE. Aby zmiana wartości parametru została uwzględniona, bazę danych trzeba będzie zatrzymać i ponownie uruchomić.

Oracle Identity Management

Oracle Identity Management (IM), który jest komponentem serwera Oracle Application Server 10*g*, stanowi kompletny szkielet do centralnego zarządzania kontami użytkowników, od tworzenia kont, przez autoryzację zasobów, po usuwanie kont. Dzięki IM można centralnie zarządzać kontami, a także urządzeniami, aplikacjami, usługami sieciowymi oraz wszelkimi innymi obiektami sieciowymi, które wykorzystują mechanizmy uwierzytelniania i autoryzacji.

IM pozwala zaoszczędzić czas i pieniądze. Dzięki scentralizowaniu kont użytkowników i powiązanych z nimi zasobów administrowanie odbywa się w taki sam sposób bez względu na rodzaj utrzymywanej aplikacji.

Dodatkowo IM zwiększa bezpieczeństwo firmy. Dzięki temu, że użytkownicy używają tylko jednej nazwy użytkownika i hasła do pracy z wszystkimi zasobami przedsiębiorstwa, znacznie zmniejsza się ryzyko, że któryś z nich zapisze swoje hasło w widocznym miejscu albo je zapomni. Natomiast gdy pracownik odejdzie z firmy, wówczas dostęp do aplikacji i usług będzie można mu szybko i łatwo zablokować w jednym miejscu.

Wprawdzie szczegółowy opis narzędzia Oracle Identity Management wykracza poza zakres tej książki, jednak bardzo ważne jest, by administrator bazy danych rozumiał wpływ poszczególnych komponentów IM na wydajność i bezpieczeństwo bazy danych Oracle. Informacje o koncie użytkownika oraz inne metadane muszą być gdzieś przechowywane, i to w sposób nadmiarowy, w bazie danych Oracle. Ponadto żądania uwierzytelnienia i autoryzacji muszą być przetwarzane w rozsądnym czasie, zdefiniowanym zazwyczaj w umowie o poziomie świadczenia usług (ang. *Service Level Agreement — SLA*) dla jednej lub większej liczby aplikacji.

Na przykład Oracle Internet Directory (OID), czyli jeden z głównych komponentów Oracle Identity Management, wymaga dostrojenia bazy danych w sposób podobny do tego, w jaki stroi się ją dla celów systemów OLTP, w których wykonywanych jest wiele krótkich transakcji dla znacznej liczby użytkowników i z obciążeniem rozłożonym na różne pory dnia. Na tym jednak podobieństwa się kończą! W tabeli 10.1 przedstawiono ogólne wytyczne dotyczące definiowania wartości różnych parametrów inicjalizacyjnych dla bazy danych, która utrzymuje dane katalogu Lightweight Directory Access Protocol (LDAP).

Tabela 10.1. *Wymiarowanie parametrów inicjalizacyjnych bazy danych przeznaczonej dla OID*

Parametr bazy danych	500 współbieżnych użytkowników	2000 współbieżnych użytkowników
OPEN_CURSORS	200	200
SESSIONS	445	1655
DB_BLOCK_SIZE	8 KB	8 KB
DB_CACHE_SIZE	250 MB	250 MB
SHARED_POOL_SIZE	40 MB	40 MB
PROCESSES	400	1500
SORT_AREA_SIZE	256 KB	256 KB
LOG_BUFFER	512 KB	512 KB

Zakłada się przy tym, że jedynym zadaniem bazy danych jest przechowywanie danych katalogu OID. Oprócz odpowiedniego dostrojenia podstawowych parametrów bazy danych wpływ na jej całkowitą wydajność będą też mieć inne czynniki, takie jak przepustowość sieci między serwerem i użytkownikami, zlokalizowanie współużytkowanych dysków, przepustowość dysków itd. Standardowa instalacja IM dla 500 tysięcy pozycji w katalogu będzie wymagać około 3 GB na dysku, a prędkość zapisu i odczytu pozycji katalogu na dysku szybko może stać się wąskim gardłem rozwiązania.

Konta użytkowników

Aby uzyskać dostęp do bazy danych, użytkownik musi podać **nazwę użytkownika** (ang. *username*), dzięki której uzyska dostęp do zasobów powiązanych ze wskazanym kontem. Każdy użytkownik musi posiadać hasło i jest powiązany z tylko jednym schematem w bazie danych. W schematach niektórych użytkowników może nie być żadnych obiektów, a w zamian użytkownicy tacy mogą posiadać uprawnienia umożliwiające dostęp do obiektów w innych schematach.

W tym punkcie opisana zostanie składnia oraz przedstawione zostaną przykłady tworzenia, modyfikowania i usuwania użytkowników. Dodatkowo pokażemy, w jaki sposób można stać się innym użytkownikiem bez znajomości jego hasła.

Tworzenie użytkowników

Polecenia `create user` używa się w dość prosty sposób. Ma ono parametry, które przedstawiono wraz z krótkim opisem każdego z nich w tabeli 10.2.

Tabela 10.2. *Opcje polecenia CREATE USER*

Parametr	Sposób użycia
nazwa użytkownika	Nazwa schematu, a więc i użytkownika, który ma zostać utworzony. Nazwa użytkownika może mieć do 30 znaków i nie mogą się w niej znajdować słowa zarezerwowane, chyba że nazwa zostanie podana w cudzysłowach (nie zaleca się jednak takiego postępowania).
`IDENTIFIED { BY` *hasło* `| EXTERNALLY | GLOBALLY AS` *'nazwa_zewnętrzna'* `}`	Wskazuje sposób, w jaki użytkownik będzie uwierzytelniany: przez bazę danych przy użyciu hasła, przez system operacyjny (lokalny lub zdalny) albo przez usługę (taką jak Oracle Internet Directory).
`DEFAULT TABLESPACE` *nazwa_przestrzeni_tabel*	Przestrzeń tabel, gdzie tworzone będą obiekty stałe, o ile w trakcie tworzenia tych obiektów nie zostanie jawnie wskazana inna przestrzeń tabel.
`TEMPORARY_TABLESPACE` *nazwa_przestrzeni_tabel*	Przestrzeń tabel, w której tworzone będą tymczasowe obiekty w trakcie operacji sortowania, tworzenia indeksów itd.
`QUOTA {` *rozmiar* `| UNLIMITED } ON` *nazwa_przestrzeni_tabel*	Wielkość przestrzeni dyskowej przeznaczonej dla obiektów tworzonych we wskazanej przestrzeni tabel. Rozmiar jest definiowany w kilobajtach (K) lub megabajtach (M).
`PROFILE` *profil*	Profil przypisany użytkownikowi (profile zostaną opisane w dalszej części tego rozdziału). Jeśli profil nie zostanie wskazany, użyty zostanie profil `DEFAULT`.
`PASSWORD EXPIRE`	Oznacza, że w momencie pierwszego logowania użytkownik będzie musiał zmienić hasło.
`ACCOUNT { LOCK | UNLOCK }`	Wskazuje, czy konto jest zablokowane (`LOCK`), czy odblokowane (`UNLOCK`). Domyślnie konto jest odblokowane.

W poniższym przykładzie tworzony jest użytkownik `SKING` odpowiadający pracownikowi numer 100 o nazwisku Steven King, znajdującemu się w tabeli `HR.EMPLOYEES` przykładowego schematu instalowanego wraz z bazą danych Oracle:

```
SQL> create user sking identified by sking901
  2     account unlock
  3     default tablespace users
  4     temporary tablespace temp;
User created.
```

Użytkownik SKING będzie uwierzytelniany przez bazę danych początkowym hasłem SKING901. Drugi wiersz nie jest tak naprawdę wymagany, ponieważ wszystkie tworzone konta są domyślnie tworzone jako konta odblokowane. Zarówno przestrzeń tabel dla obiektów stałych, jak i domyślna tymczasowa przestrzeń tabel są definiowane na poziomie bazy danych, dlatego dwa ostatnie wiersze przykładowego polecenia również nie są wymagane, o ile tylko użytkownikowi nie powinno się przypisać innej domyślnej przestrzeni tabel na obiekty stałe lub innej tymczasowej przestrzeni tabel.

Nawet gdy użytkownikowi SKING jawnie lub niejawnie przypisze się domyślną stałą przestrzeń tabel, użytkownik ten nie będzie mógł tworzyć żadnych obiektów w bazie danych, dopóki nie zdefiniuje się dla niego udziału dyskowego oraz praw do tworzenia obiektów w jego własnym schemacie.

Udział dyskowy (ang. *quota*) to zwyczajne ograniczenie przestrzeni dyskowej dla przestrzeni tabel, obowiązujące danego użytkownika. Dopóki użytkownikowi jawnie nie przypisze się udziału dyskowego albo nie nada się mu uprawnienia UNLIMITED TABLESPACE (o uprawnieniach więcej powiemy w dalszej części rozdziału), użytkownik nie będzie mógł tworzyć obiektów we własnym schemacie. Poniższe polecenie nadaje kontu SKING udział dyskowy o rozmiarze 250 MB w przestrzeni tabel USERS:

```
SQL> alter user sking quota 250M on users;
User altered.
```

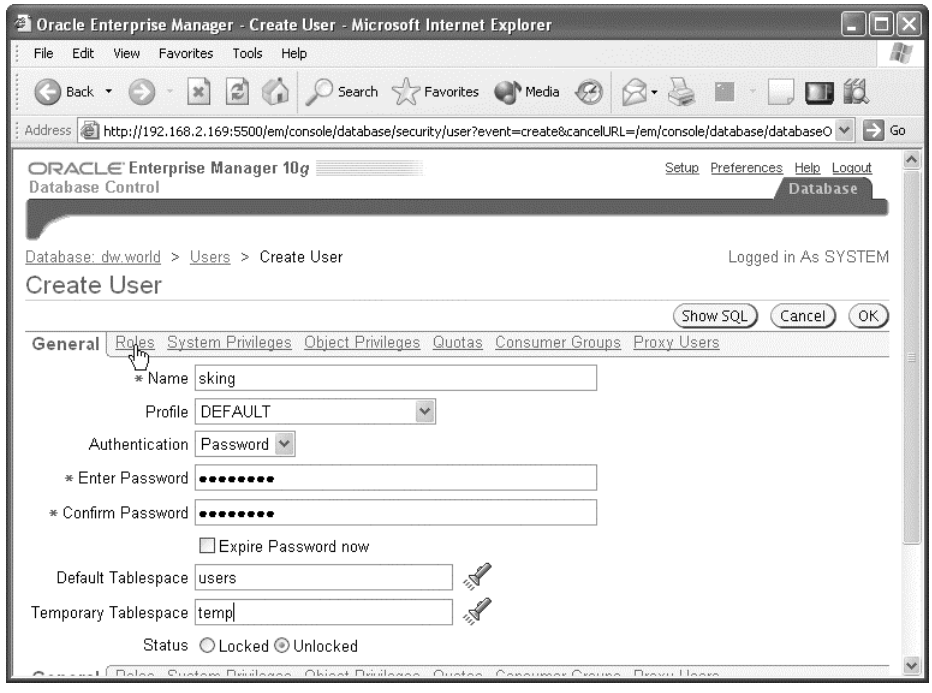
Warto zwrócić uwagę, że udział dyskowy można było przypisać użytkownikowi już w momencie jego tworzenia, podobnie zresztą jak można było użyć niemal wszystkich pozostałych opcji polecenia create user. Jedynie rolę domyślną można przypisać dopiero po utworzeniu konta użytkownika (zarządzanie rolami zostanie opisane w dalszej części rozdziału).

Dopóki nowemu użytkownikowi nie zostaną przypisane choćby podstawowe uprawnienia, użytkownik ten nie będzie mógł się nawet zalogować. Trzeba zatem nadać przynajmniej uprawnienie CREATE SESSION roli CONNECT (role zostaną opisane bardziej szczegółowo w dalszej części tego rozdziału). Rola CONNECT zawiera uprawnienie CREATE SESSION, a także szereg innych podstawowych uprawnień, takich jak CREATE TABLE i ALTER SESSION. W poniższym przykładzie użytkownikowi SKING nadane zostanie uprawnienie CONNECT:

```
SQL> grant connect to sking;
Grant succeeded.
```

Od teraz użytkownik SKING będzie posiadał udział dyskowy w przestrzeni tabel USERS oraz uprawnienia pozwalające mu na tworzenie obiektów w tej przestrzeni.

Wszystkie wspomniane opcje polecenia create user są dostępne również za pośrednictwem interfejsu WWW narzędzia Enterprise Manager, co widać na rysunku 10.2.



Rysunek 10.2. *Tworzenie użytkowników z poziomu narzędzia Enterprise Manager*

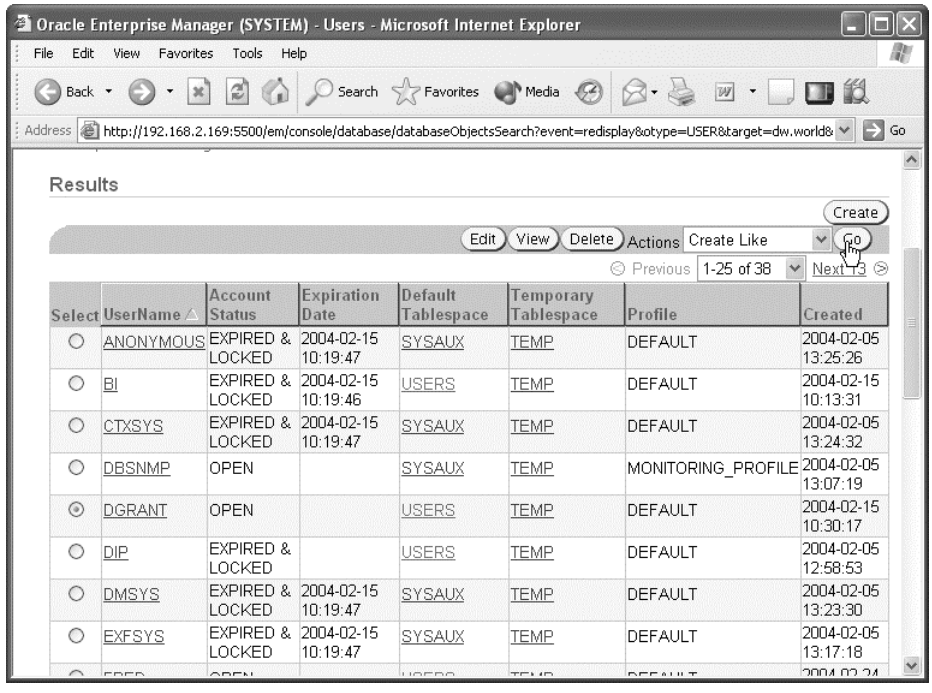
Podobnie jak w przypadku wszystkich innych operacji wykonywanych w narzędziu Enterprise Manager, kliknięcie przycisku *Show SQL* spowoduje wyświetlenie poleceń języka SQL, takich jak `create` i `grant`, które zostaną wykonane w celu utworzenia użytkownika. Jest to zatem doskonała okazja, aby skorzystać z przyjaznych funkcji interfejsu WWW, a jednocześnie odświeżyć sobie znajomość składni poleceń języka SQL!

Na rysunku 10.3 widać, że bardzo łatwo można wybrać już istniejącego użytkownika i na jego podstawie utworzyć nowego użytkownika, który będzie posiadał te same atrybuty z wyjątkiem hasła.

Pozostałe opcje dostępne w interfejsie narzędzia Enterprise Manager pozwalają na wygaśnięcie konta użytkownika, wygenerowanie kodu DDL umożliwiającego utworzenie użytkownika, a także zablokowanie i odblokowanie konta.

Modyfikowanie użytkowników

Atrybuty użytkowników zmienia się poleceniem `alter user`. Składnia polecenia `alter user` jest niemal identyczna, jak składnia polecenia `create user`. Jedyna różnica polega na tym, że `alter user` pozwala na przypisywanie ról oraz nadawanie praw aplikacjom warstwy pośredniej, aby umożliwić im wykonywanie czynności w imieniu użytkownika.



Rysunek 10.3. *Kopiowanie użytkowników z użyciem narzędzia Enterprise Manager*

W poniższym przykładzie charakterystyki użytkownika `SKING` zostaną tak zmienione, aby używał on innej domyślnej stałej przestrzeni tabel:

```
SQL> alter user sking
  2      default tablespace users2
  3      quota 500M on users2;
User altered.
```

Warto zauważyć, że użytkownik `SKING` nadal może tworzyć obiekty w przestrzeni tabel `USERS`, lecz w tym celu w każdym poleceniu `create table` i `create index` musi tę przestrzeń wskazać w sposób jawny.

Usuwanie użytkowników

Użytkowników usuwa się w bardzo łatwy sposób przy użyciu polecenia `drop user`. Jedynymi parametrami polecenia są nazwa użytkownika, którego należy usunąć, oraz opcja `cascade`. Jeżeli opcja `cascade` nie zostanie użyta, wówczas wszystkie obiekty, których użytkownik jest właścicielem, trzeba jawnie usunąć albo przenieść do innego schematu. Poniższe przykładowe polecenie usuwa użytkownika `QUEENB`, a jeżeli użytkownik ten jest również właścicielem jakichkolwiek obiektów, wówczas obiekty te również zostaną automatycznie usunięte:

```
SQL> drop user queenb cascade;
User dropped.
```

Jeżeli inne obiekty schematów, takie jak widoki albo pakiety, korzystają z obiektów usuniętych w trakcie usuwania użytkownika, wówczas obiekty schematów zostaną oznaczone jako INVALID i trzeba będzie je przypisać do innych obiektów, a następnie na nowo skompilować. Ponadto, jeżeli pierwszy użytkownik nadał drugiemu użytkownikowi uprawnienia przy użyciu klauzuli with grant option, wówczas w przypadku usunięcia pierwszego użytkownika wszystkie nadane przez niego uprawnienia zostaną odebrane drugiemu użytkownikowi.

Przełączanie się na innego użytkownika

Aby móc debugować aplikację, administrator bazy danych musi czasami łączyć się jako inny użytkownik w celu zasymulowania zidentyfikowanego problemu. Nawet bez znajomości hasła użytkownika administrator bazy może odczytać jego hasło w postaci zaszyfrowanej, zmienić hasło tego użytkownika, połączyć się z bazą, podając zmienione hasło, a następnie przywrócić poprzednie hasło. Wykorzystuje się do tego nieudokumentowaną klauzulę polecenia alter user. Zakłada się przy tym, że administrator bazy danych posiada dostęp do tabeli DBA_USERS oraz uprawnienie ALTER USER. Jeśli administrator posiada rolę DBA, wówczas obydwa wspomniane wymogi są spełnione.

W pierwszym kroku należy odczytać zaszyfrowane hasło użytkownika, przechowywane w tabeli DBA_USERS:

```
SQL> select password from dba_users
  2     where username = 'SKING';

PASSWORD
------------------------------
83C7CBD27A941428

1 row selected.
```

Odczytane hasło należy wyciąć i skopiować w środowisku GUI albo zapisać je w pliku tekstowym, aby móc je później odczytać. W kolejnym kroku trzeba tymczasowo zmienić hasło użytkownika, a następnie zalogować się przy użyciu nowego tymczasowego hasła:

```
SQL> alter user sking identified by temp_pass;
User altered.

SQL> connect sking/temp_pass@dw;
Connected.
```

Od tego momentu można zacząć debugować aplikację z perspektywy użytkownika SKING. Po zakończeniu debugowania należy przywrócić oryginalne hasło, wykorzystując do tego celu nieudokumentowaną klauzulę by values polecenia alter user:

```
SQL> alter user sking identified by values '83C7CBD27A941428';
User altered.
```

Widoki danych słownikowych dotyczących użytkownika

Szereg widoków danych słownikowych zawiera informacje na temat użytkowników oraz ich charakterystyk. Najczęściej używane tego rodzaju widoki i tabele zostały przedstawione w tabeli 10.3.

Tabela 10.3. *Widoki danych słownikowych dotyczących użytkownika*

Widok danych słownikowych	Opis
DBA_USERS	Zawiera nazwy użytkowników, zaszyfrowane hasła, statusy kont oraz domyślne przestrzenie tabel.
DBA_TS_QUOTAS	Zawiera dane na temat użycia przestrzeni dyskowej oraz limitów przestrzeni w podziale na użytkowników i przestrzenie tabel. Dane dotyczą użytkowników, dla których zdefiniowano udziały dyskowe różne od UNLIMITED.
DBA_PROFILES	Zawiera profile, które można przypisać użytkownikom oraz ograniczenia zasobów przypisane do profili.
USER_HISTORY$	Zawiera historię haseł wraz z nazwami użytkowników, zaszyfrowanymi hasłami i znacznikami czasu. Na podstawie widoku wymuszane są reguły ograniczające ponowne używanie haseł.

Metody autoryzacji w bazie danych

Gdy użytkownik uwierzytelni się w bazie danych, następny krok polega na ustaleniu typów obiektów, uprawnień i zasobów, do których ten użytkownik ma dostęp albo których może używać. W niniejszym punkcie pokażemy, w jaki sposób za pomocą profili kontroluje się proces zarządzania hasłami, a także jak przy użyciu profili można nakładać ograniczenia na różne rodzaje zasobów systemowych.

Dodatkowo opisane zostaną dwa rodzaje uprawnień obecnych w bazie danych Oracle: uprawnienia systemowe oraz uprawnienia do obiektów. Uprawnienia obydwóch rodzajów można przypisać użytkownikom bezpośrednio albo pośrednio poprzez role. Role stanowią kolejne rozwiązanie, które znacznie ułatwia administratorom baz danych przypisywanie uprawnień użytkownikom.

W końcowej części tego punktu opisane zostaną dostępne w bazie Oracle mechanizmy wirtualnych prywatnych baz danych (ang. *Virtual Private Database — VPD*) oraz sposoby ich wykorzystania do bardziej precyzyjnego kontrolowania widoczności poszczególnych części tabel zależnie od danych uwierzytelniających przypisanych użytkownikowi przez administratora. Aby bardziej rozjaśnić działanie dostępnych mechanizmów, opisany zostanie kompletny proces implementacji VPD.

Zarządzanie profilami

Dla celów wykonania zapytania użytkownika nigdy nie jest za dużo mocy procesora, przestrzeni na dysku ani przepustowości wejścia-wyjścia. Ponieważ wszystkie wymienione zasoby z natury rzeczy są ograniczone, Oracle udostępnia mechanizm, dzięki któremu można kontrolować ilość poszczególnych zasobów dostępnych do wykorzystania przez użytkowników. **Profil** w bazie Oracle to nazwany zbiór ograniczeń dla poszczególnych zasobów.

Profili można również używać jako mechanizmów autoryzacyjnych, nadzorujących proces tworzenia, ponownego wykorzystywania i walidowania haseł użytkowników. Można na przykład narzucić wymóg minimalnej długości haseł oraz dodatkowy obowiązek, by w każdym haśle znajdowała się co najmniej jedna mała i jedna wielka litera. W niniejszym punkcie opiszemy, w jaki sposób profile zarządzają hasłami i zasobami.

Polecenie CREATE PROFILE

Polecenie CREATE PROFILE pełni dwie role. Po pierwsze można utworzyć profil, który będzie ograniczał dozwolony czas połączenia użytkownika do 120 minut.

```
create profile lim_connect limit
    connect_time 120;
```

W analogiczny sposób można ograniczyć liczbę dozwolonych kolejnych logowań zakończonych niepowodzeniem, po której wyczerpaniu powinno nastąpić zablokowanie konta:

```
create profile lim_fail_login limit
    failed_login_attempts 8;
```

W jednym profilu można także połączyć obydwa powyższe ograniczenia:

```
create profile lim_connectime_faillog limit
    connect_time 120
    failed_login_attempts 8;
```

Sposób reakcji bazy Oracle na przekroczenie któregoś ze zdefiniowanych ograniczeń na wykorzystanie zasobów zależy od typu ograniczenia. Gdy przekroczone zostanie ograniczenie dozwolonego czasu połączenia albo bezczynności (na przykład CPU_PER_SESSION), wykonywana w tym momencie transakcja zostanie wycofana oraz nastąpi rozłączenie sesji. W razie przekroczenia większości innych ograniczeń (takich jak PRIVATE_SGA) następuje wycofanie bieżącej transakcji, a do użytkownika zwracany jest błąd i zyskuje on możliwość samodzielnego zatwierdzenia lub wycofania transakcji. Jeżeli dana operacja przekroczy ograniczenie zdefiniowane dla pojedynczego wywołania (na przykład LOGICAL_READS_PER_CALL), operacja zostanie anulowana, bieżąca instrukcja zostanie wycofana, a użytkownik otrzyma informację o błędzie. Pozostała część transakcji pozostanie niezmieniona, a użytkownik będzie mógł wycofać lub zatwierdzić transakcję albo podjąć próbę doprowadzenia jej do końca bez przekraczania ograniczeń nałożonych na pojedynczą instrukcję.

Oracle posiada profil DEFAULT przypisywany wszystkim nowym użytkownikom, którym nie przypisano żadnego innego profilu. Poniżej przedstawiono wyniki zapytania na widoku danych słownikowych DBA_USERS, w którym znajdują się dane na temat ograniczeń zdefiniowanych w profilu DEFAULT:

```
SQL> select * from dba_profiles
  2     where profile = 'DEFAULT';

PROFILE         RESOURCE_NAME                    RESOURCE LIMIT
--------------- -------------------------------- -------- ------------
DEFAULT         COMPOSITE_LIMIT                  KERNEL   UNLIMITED
DEFAULT         SESSIONS_PER_USER                KERNEL   UNLIMITED
DEFAULT         CPU_PER_SESSION                  KERNEL   UNLIMITED
DEFAULT         CPU_PER_CALL                     KERNEL   UNLIMITED
DEFAULT         LOGICAL_READS_PER_SESSION        KERNEL   UNLIMITED
DEFAULT         LOGICAL_READS_PER_CALL           KERNEL   UNLIMITED
DEFAULT         IDLE_TIME                        KERNEL   UNLIMITED
DEFAULT         CONNECT_TIME                     KERNEL   UNLIMITED
DEFAULT         PRIVATE_SGA                      KERNEL   UNLIMITED
DEFAULT         FAILED_LOGIN_ATTEMPTS            PASSWORD 10
DEFAULT         PASSWORD_LIFE_TIME               PASSWORD UNLIMITED
DEFAULT         PASSWORD_REUSE_TIME              PASSWORD UNLIMITED
```

```
DEFAULT         PASSWORD_REUSE_MAX              PASSWORD UNLIMITED
DEFAULT         PASSWORD_VERIFY_FUNCTION        PASSWORD NULL
DEFAULT         PASSWORD_LOCK_TIME              PASSWORD UNLIMITED
DEFAULT         PASSWORD_GRACE_TIME             PASSWORD UNLIMITED

16 rows selected.
```

Jedynym ograniczeniem z prawdziwego zdarzenia zdefiniowanym w profilu `DEFAULT` jest liczba dozwolonych kolejnych prób zalogowania się zakończonych niepowodzeniem, po których powinno nastąpić zablokowanie konta. Ponadto w profilu nie jest włączona żadna funkcja weryfikacji poprawności hasła.

Profile i kontrola haseł

W tabeli 10.4 przedstawiono dostępne w profilach parametry obsługujące hasła. Wszystkie jednostki czasu są zdefiniowane w dniach (aby zdefiniować wartość któregoś z tych parametrów na przykład w minutach, trzeba podzielić go przez 1440):

```
SQL> create profile lim_lock limit password_lock_time 5/1440;
Profile created.
```

Według przedstawionego polecenia po wykonaniu określonej liczby kolejnych prób zalogowania się zakończonych niepowodzeniem konto zostanie zablokowane jedynie na pięć minut.

Tabela 10.4. *Parametry profili dotyczące haseł*

Parametr dotyczący haseł	Opis
`FAILED_LOGIN_ATTEMPTS`	Liczba dozwolonych prób logowania zakończonych niepowodzeniem, po której osiągnięciu konto powinno zostać zablokowane.
`PASSWORD_LIFE_TIME`	Liczba dni, przez które można używać hasła, zanim konieczna będzie jego zmiana. Jeżeli hasło nie zostanie zmienione w czasie definiowanym przez parametr `PASSWORD_GRACE_TIME`, zalogowanie się bez uprzedniej zmiany hasła nie będzie możliwe.
`PASSWORD_REUSE_TIME`	Liczba dni, jaką musi odczekać użytkownik, zanim będzie mógł ponownie wykorzystać hasło. Parametru używa się wraz z parametrem `PASSWORD_REUSE_MAX`.
`PASSWORD_REUSE_MAX`	Liczba wymaganych zmian hasła, po których możliwe będzie ponowne wykorzystanie hasła. Parametru używa się wraz z parametrem `PASSWORD_REUSE_TIME`.
`PASSWORD_LOCK_TIME`	Liczba dni, przez które konto jest zablokowane po przekroczeniu liczby prób logowania zdefiniowanej parametrem `FAILED_LOGIN_ATTEMPTS`. Po upłynięciu tego okresu konto zostanie automatycznie odblokowane.
`PASSWORD_GRACE_TIME`	Liczba dni, po których trzeba zmienić hasło po wygaśnięciu jego ważności. Jeżeli hasło nie zostanie zmienione w okresie wskazywanym przez wartość parametru, ważność konta wygaśnie i użytkownik będzie mógł się do niego zalogować dopiero po zmianie hasła.
`PASSWORD_VERIFY_FUNCTION`	Skrypt PL/SQL implementujący zaawansowaną procedurę weryfikacji poprawności hasła. Jeśli parametr ma wartość `NULL` (jest to wartość domyślna), żadna weryfikacja poprawności hasła nie będzie wykonywana.

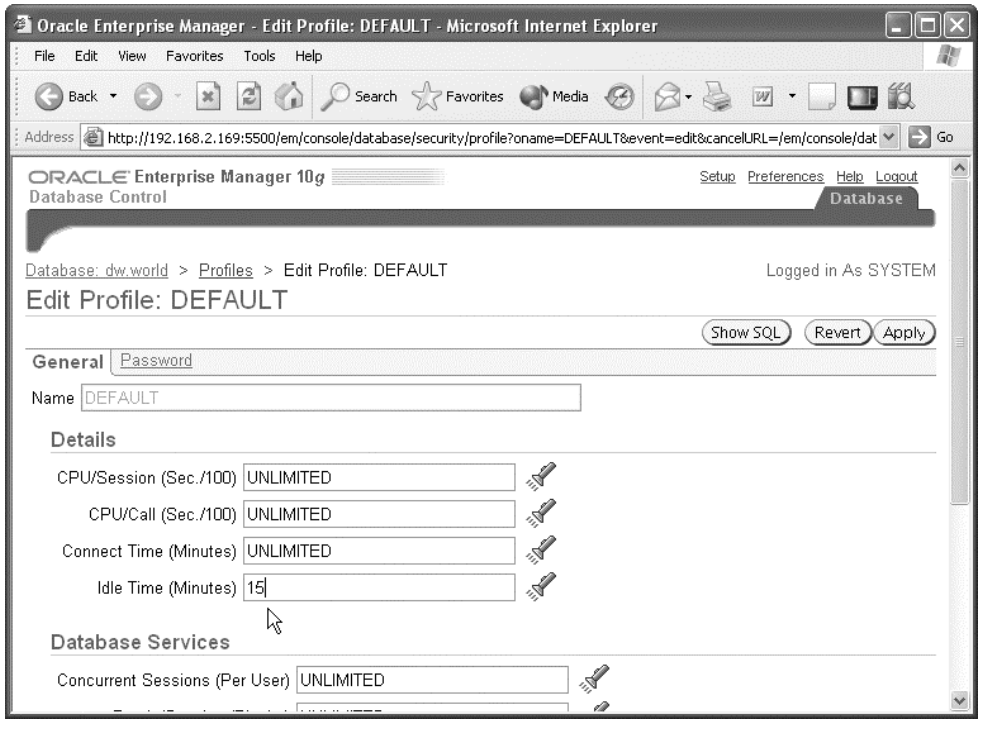
Wartość parametru `unlimited` oznacza, że ilość danego zasobu, jakiej można używać, jest nieograniczona. Wartość `default` oznacza, że jako wartość parametru dziedziczona jest wartość tego samego parametru w profilu `DEFAULT`.

Parametrów `password_reuse_time` oraz `password_reuse_max` należy używać łącznie. Zdefiniowanie jednego z tych parametrów i pominięcie drugiego z nich nie przyniesie żadnego efektu. W poniższym przykładzie tworzony jest profil, w którym parametr `password_reuse_time` ma wartość 20 dni, zaś wartością parametru `password_reuse_max` jest 5:

```
create profile lim_reuse_pass limit
    password_reuse_time 20
    password_reuse_max 5;
```

Użytkownicy, którym zostanie przypisany ten profil, będą mogli ponownie zacząć używać hasła po 20 dniach pod warunkiem, że zostało ono zmienione co najmniej pięć razy. Jeżeli któremuś z dwóch opisywanych parametrów przypisana zostanie konkretna wartość, zaś wartością drugiego parametru pozostanie `UNLIMITED`, wówczas użytkownik nie będzie w ogóle mógł ponownie używać haseł.

Podobnie jak w przypadku większości innych obiektów, również profilami można łatwo zarządzać z poziomu narzędzia Oracle Enterprise Manager. Na rysunku 10.4 przedstawiono operację zmiany profilu `DEFAULT` w taki sposób, by po 15 minutach bezczynności użytkownik był rozłączany.



Rysunek 10.4. *Zmiana ograniczeń dotyczących haseł za pomocą narzędzia Oracle Enterprise Manager*

Aby zapewnić sobie większą kontrolę nad procesem tworzenia i ponownego używania haseł (aby na przykład narzucić wymóg występowania w każdym haśle wielkich i małych liter), należy zdefiniować w każdym używanym profilu ograniczenie PASSWORD_VERIFY_FUNCTION. Oracle udostępnia szablon, który służy do określania polityki zarządzania hasłami organizacji. Szablon znajduje się w katalogu *$ORACLE_HOME/rdbms/admin/utlpwdmg.sql*. Skrócona wersja tego skryptu znajduje się w dodatku A.

Skrypt udostępnia następujące funkcje wyznaczające stopień złożoności haseł:

- Wymóg, by hasło nie było identyczne z nazwą użytkownika.
- Wymóg, by hasło miało co najmniej cztery znaki długości.
- Wymóg, by hasło nie było prostym, znanym słowem, na przykład ORACLE albo DATABASE.
- Wymóg, by hasło zawierało co najmniej jedną literę, jedną cyfrę oraz jeden znak interpunkcyjny.
- Wymóg, by nowe hasło różniło się od hasła poprzedniego co najmniej trzema znakami.

Aby móc skorzystać z polityki trzeba najpierw wprowadzić własne zmiany w skrypcie. Można na przykład zdecydować o użyciu kilku różnych funkcji weryfikacji poprawności haseł — po jednej dla każdego kraju albo oddziału firmy, aby w ten sposób móc dostosować stopień złożoności haseł w bazie danych do wymogów złożoności haseł w systemach operacyjnych używanych w danym kraju albo oddziale firmy. Można więc na przykład nadać funkcji nazwę VERIFY_FUNCTION_PL_MAZOWIECKIE. Można ponadto zmienić listę najbardziej popularnych słów i dołączyć do niej nazwy oddziałów albo budynków firmy.

Po pomyślnym skompilowaniu funkcji można zmienić istniejący profil poleceniem alter profile i nakazać mu używanie funkcji albo utworzyć nowy profil wykorzystujący funkcję. Poniższe polecenie zmieni definicję profilu DEFAULT w taki sposób, by poprawność haseł była weryfikowana funkcją VERIFY_FUNCTION_PL_MAZOWIECKIE:

```
SQL> alter profile default limit
  2      password_verify_function verify_function_pl_mazowieckie;
Profile altered.
```

Polecenie spowoduje, że poprawność haseł wszystkich istniejących użytkowników, którym przypisano profil DEFAULT, oraz każdego nowego użytkownika używającego profilu DEFAULT będzie sprawdzana przez funkcję VERIFY_FUNCTION_PL_MAZOWIECKIE. Jeśli funkcja zwróci wartość inną niż TRUE, hasło zostanie odrzucone i użytkownik będzie musiał wpisać inne hasło. Jeżeli bieżące hasło użytkownika jest niezgodne z regułami zaimplementowanymi w funkcji, będzie ono akceptowane do czasu zmiany tego hasła. W momencie zmiany hasła jego poprawność zostanie już zweryfikowana przez funkcję.

Profile i kontrola zasobów

W tabeli 10.5 przedstawiono listę opcji profilu związanych z kontrolą zasobów, które można wstawiać w poleceniu CREATE PROFILE *nazwa_profilu* LIMIT. Każdy z wymienionych parametrów może przyjmować wartość całkowitoliczbową, wartość UNLIMITED albo wartość DEFAULT.

Tabela 10.5. *Parametry profilu związane z kontrolą zasobów*

Parametr kontroli zasobów	Opis
SESSIONS_PER_USER	Dozwolona maksymalna liczba sesji współbieżnych dla jednego użytkownika.
CPU_PER_SESSION	Maksymalny czas użycia procesora w trakcie jednej sesji wyrażony w setnych częściach sekundy.
CPU_PER_CALL	Maksymalny czas użycia procesora w trakcie parsowania jednej instrukcji, jej wykonywania lub pobierania, wyrażony w setnych częściach sekundy.
CONNECT_TIME	Maksymalny czas połączenia wyrażony w sekundach.
IDLE_TIME	Wyrażony w minutach maksymalny czas bezczynności w trakcie jednej sesji, gdy nie jest wykonywane żadne zapytanie ani inna operacja.
LOGICAL_READS_PER_SESSION	Łączna liczba dozwolonych w jednej sesji operacji odczytów bloku z dysku lub z pamięci.
LOGICAL_READS_PER_CALL	Maksymalna liczba dozwolonych operacji odczytów bloku w trakcie parsowania, wykonywania lub pobierania instrukcji.
COMPOSITE_LIMIT	Łączny koszt użycia zasobów wyrażony w **jednostkach usługowych** (ang. *service units*), wyrażony jako ważona suma wartości parametrów CPU_PER_SESSION, CONNECT_TIME, LOGICAL_READS_PER_SESSION i PRIVATE_SGA.
PRIVATE_SGA	Maksymalna ilość pamięci, jaką sesja może zaalokować w obszarze dzielonym. Wartość wyrażona w bajtach, kilobajtach lub megabajtach.

Podobnie jak w przypadku parametrów związanych z zarządzaniem hasłami, również w tym przypadku wartość UNLIMITED oznacza, że nie ma ograniczenia na wielkość użycia danego zasobu. Wartość DEFAULT oznacza natomiast, że wartość parametru jest dziedziczona z profilu DEFAULT.

Dzięki parametrowi COMPOSITE_LIMIT można nakładać ograniczenia na zużycie grupy zasobów, gdy stopień użycia poszczególnych rodzajów zasobów jest znacznie zróżnicowany. Parametr może na przykład pozwalać na długotrwałe obciążanie procesora i jednocześnie oszczędne wykonywanie operacji wejścia-wyjścia w jednej sesji albo odwrotnie, bez rozłączania z powodu przekroczenia reguł polityki.

Domyślny koszt wszystkich zasobów wynosi zero:

```
SQL> select * from resource_cost;

RESOURCE_NAME                    UNIT_COST
-------------------------------- ----------
CPU_PER_SESSION                           0
LOGICAL_READS_PER_SESSION                 0
CONNECT_TIME                              0
PRIVATE_SGA                               0

4 rows selected.
```

Aby zdefiniować wagi kosztów poszczególnych zasobów, należy użyć polecenia ALTER RESOURCE COST. W poniższym przykładzie wagi są zmieniane w taki sposób, by czas użycia procesora wyznaczany przez parametr CPU_PER_SESSION był 25-krotnie istotniejszy niż czas połączenia. Inaczej mówiąc, znacznie większe jest prawdopodobieństwo, że użytkownik zostanie rozłączony z powodu zbytniego użycia procesora, a nie z powodu czasu połączenia:

```
SQL> alter resource cost
  2     cpu_per_session 50
  3     connect_time 2;
Resource cost altered.

SQL> select * from resource_cost;

RESOURCE_NAME                    UNIT_COST
-------------------------------- ----------
CPU_PER_SESSION                          50
LOGICAL_READS_PER_SESSION                 0
CONNECT_TIME                              2
PRIVATE_SGA                               0

4 rows selected.
```

W kolejnym kroku trzeba utworzyć nowy profil albo zmodyfikować już istniejący profil tak, by używał on ograniczenia złożonego:

```
SQL> create profile lim_comp_cpu_conn limit
  2     composite_limit 250;

Profile created.
```

W wyniku wykonanych czynności ograniczenie użycia zasobów dla użytkowników o profilu LIM_COMP_CPU_CONN będzie wyznaczane przez następujący wzór na łączny koszt użycia zasobów:

```
łączny_koszt = (50 * CPU_PER_SESSION) + (2 * CONNECT_TIME);
```

W tabeli 10.6 przedstawiono kilka przykładowych poziomów użycia zasobów, w których limit łącznego kosztu wynoszący 250 jest przekroczony lub nie.

Tabela 10.6. *Scenariusze użycia zasobów*

Użycie procesora (sekundy)	Czas połączenia (sekundy)	Łączny koszt	Koszt przekroczony?
0,05	100	(50·5)+(2·100) = 450	Tak
0,02	30	(50·2)+(2·30) = 160	Nie
0,01	150	(50·1)+(2·150) = 350	Tak
0,02	5	(50·2)+(2·5) = 110	Nie

W prezentowanym przykładzie parametry PRIVATE_SGA i LOGICAL_READS_PER_SESSION nie są używane, dlatego jeśli nie zostaną inaczej zdefiniowane w profilu, wartości obydwóch parametrów zostaną odziedziczone z profilu DEFAULT. Ograniczeń złożonych używa się po to, aby dać użytkownikom pewną swobodę w wykonywanych przez nich rodzajach zapytań i instrukcji DML. W niektórych dniach użytkownicy mogą wykonywać wiele zapytań, które przeprowadzają złożone obliczenia, lecz nie odczytują z tabel zbyt wielu wierszy, innym razem zaś użytkownicy mogą wykonywać liczne operacje skanowania tabel, natomiast czas ich połączenia z bazą danych jest stosunkowo krótki. W takich przypadkach lepiej jest nie ograniczać użytkowników wartością pojedynczego parametru, lecz definiować ograniczenia jako łączne użycie zasobów ważone dostępnością każdego z zasobów na serwerze.

Uprawnienia systemowe

Uprawnienie systemowe (ang. *system privilege*) to prawo wykonywania czynności na obiekcie w bazie danych, a także inne uprawnienia, które w ogóle nie dotyczą obiektów, lecz takich czynności, jak zadania wsadowe, zmiana parametrów systemowych, tworzenie ról, a nawet łączenie się z bazą danych. W wersji Oracle 10*g* Release 1 występują 173 uprawnienia systemowe. Wszystkie uprawnienia znajdują się w tabeli danych słownikowych SYSTEM_PRIVILEGE_MAP:

```
SQL> select * from system_privilege_map;

PRIVILEGE  NAME                                     PROPERTY
---------- ---------------------------------------- ----------
-3         ALTER SYSTEM                                      0
-4         AUDIT SYSTEM                                      0
-5         CREATE SESSION                                    0
-6         ALTER SESSION                                     0
-7         RESTRICTED SESSION                                0
-10        CREATE TABLESPACE                                 0
-11        ALTER TABLESPACE                                  0
-12        MANAGE TABLESPACE                                 0
-13        DROP TABLESPACE                                   0
-15        UNLIMITED TABLESPACE                              0
-20        CREATE USER                                       0
-21        BECOME USER                                       0
-22        ALTER USER                                        0
-23        DROP USER                                         0
. . .
-271       ALTER ANY SQL PROFILE                             0
-272       ADMINISTER SQL TUNING SET                         0
-273       ADMINISTER ANY SQL TUNING SET                     0
-274       CREATE ANY SQL PROFILE                            0
-275       EXEMPT IDENTITY POLICY                            0

173 rows selected.
```

W tabeli 10.7 przedstawiono najczęściej używane uprawnienia systemowe wraz z krótkim opisem każdego z nich.

Przypisywanie uprawnień systemowych

Uprawnienia przypisuje się użytkownikowi, roli lub grupie PUBLIC poleceniem grant, zaś odbiera się je poleceniem revoke. PUBLIC to specjalna grupa, do której należą wszyscy użytkownicy bazy danych; dzięki niej można w prosty sposób przyznawać uprawnienia jednocześnie wszystkim użytkownikom bazy.

Aby przyznać użytkownikowi SCOTT uprawnienie do tworzenia procedur składowanych i synonimów, można wykonać polecenie o następującej treści:

```
SQL> grant create procedure, create synonym to scott;
Grant succeeded.
```

Równie proste jest odbieranie uprawnień:

```
SQL> revoke create synonym from scott;
Revoke succeeded.
```

Tabela 10.7. *Najczęściej używane uprawnienia systemowe*

Uprawnienie systemowe	Opis
ALTER DATABASE	Zmiany w bazie danych, na przykład zmiana statusu bazy danych z MOUNT na OPEN, albo przywracanie bazy.
ALTER SYSTEM	Wykonywanie instrukcji ALTER SYSTEM: przełączenie na następną grupę dzienników powtórzeń i zmiana parametrów inicjalizacyjnych systemu w pliku SPFILE.
AUDIT SYSTEM	Wykonywanie instrukcji AUDIT.
CREATE DATABASE LINK	Tworzenie łączy bazodanowych do zdalnych baz danych.
CREATE ANY INDEX	Tworzenie indeksów w dowolnym schemacie; uprawnienie CREATE INDEX jest nadawane w schemacie użytkownika wraz z uprawnieniem CREATE TABLE.
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 SYNONYM	Tworzenie prywatnego synonimu we własnym schemacie.
CREATE ANY SYNONYM	Tworzenie prywatnego synonimu w dowolnym schemacie.
CREATE PUBLIC SYNONYM	Tworzenie publicznego synonimu.
DROP ANY SYNONYM	Usuwanie prywatnego synonimu z dowolnego schematu.
DROP PUBLIC SYNONYM	Usuwanie publicznego synonimu.
CREATE TABLE	Tworzenie tabeli we własnym schemacie.
CREATE ANY TABLE	Tworzenie tabeli w dowolnym schemacie.
CREATE TABLESPACE	Tworzenie nowej przestrzeni tabel w bazie danych.
CREATE USER	Tworzenie konta użytkownika (schematu).
ALTER USER	Wprowadzanie zmian w definicji konta użytkownika (schematu).
CREATE VIEW	Tworzenie widoku we własnym schemacie.
SYSDBA	Dopisywanie pozycji w zewnętrznym pliku haseł, jeśli taki istnieje; włączanie i wyłączanie bazy danych, tworzenie i modyfikowanie bazy danych, przywracanie bazy danych, tworzenie pliku SPFILE, łączenie się z bazą pozostającą w trybie RESTRICTED SESSION.
SYSOPER	Dopisywanie pozycji w zewnętrznym pliku haseł, jeśli taki istnieje; włączanie i wyłączanie bazy danych, modyfikowanie bazy danych, przywracanie bazy danych, tworzenie pliku SPFILE, łączenie się z bazą pozostającą w trybie RESTRICTED SESSION.

Aby pozwolić użytkownikom, którym te uprawnienia przyznano, na przyznawanie takich samych uprawnień innym użytkownikom, należy w momencie przyznawania uprawnień posłużyć się klauzulą `with admin option`. Użytkownikowi SCOTT z poprzedniego przykładu chcemy umożliwić przyznawanie uprawnienia CREATE PROCEDURE innym użytkownikom. Aby tego dokonać, trzeba ponownie przyznać użytkownikowi SCOTT uprawnienie CREATE PROCEDURE:

```
SQL> grant create procedure to scott with admin option;
Grant succeeded.
```

Od teraz SCOTT sam będzie mógł wykonywać polecenie grant create procedure. Warto pamiętać, że jeśli uprawnienie do nadawania uprawnień zostanie użytkownikowi SCOTT odebrane, to i tak użytkownicy, którym SCOTT przypisał uprawnienia, nadal będą je posiadać.

Widoki danych słownikowych uprawnień systemowych

W tabeli 10.8 przedstawiono widoki danych słownikowych dotyczących uprawnień systemowych.

Tabela 10.8. *Widoki danych słownikowych uprawnień systemowych*

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

Uprawnienia do obiektów

W odróżnieniu od uprawnień systemowych **uprawnienie do obiektu** (ang. *object privilege*) to prawo do wykonywania określonego typu operacji na konkretnym obiekcie, na przykład na tabeli lub sekwencji, który nie należy do schematu użytkownika. Podobnie jak w przypadku uprawnień systemowych, uprawnienia do obiektów nadaje się i odbiera poleceniami grant i revoke.

Podobnie jak w przypadku uprawnień systemowych, również uprawnienia do obiektów można nadawać grupie PUBLIC, a użytkownik posiadający uprawnienie do obiektu może przekazywać je kolejnym użytkownikom, przypisując uprawnienie do obiektu klauzulą with grant option.

Niektóre obiekty schematów, takie jak klastry i indeksy, kontrolują dostęp właśnie na podstawie uprawnień systemowych. W takich przypadkach użytkownik może zmieniać obiekty, jeśli jest ich właścicielem lub posiada uprawnienie systemowe ALTER ANY CLUSTER lub ALTER ANY INDEX.

Użytkownik, w którego schemacie znajdują się obiekty, automatycznie posiada wszystkie uprawnienia do tych obiektów i może przyznawać wszystkie uprawnienia do tych obiektów innym użytkownikom i rolom z opcją grant option lub bez niej.

W tabeli 10.9 znajdują się uprawnienia do obiektów w podziale na poszczególne typy obiektów, niektóre uprawnienia dotyczą bowiem wyłącznie określonych typów obiektów. Na przykład uprawnienie INSERT dotyczy wyłącznie tabel, widoków oraz widoków zmaterializowanych, natomiast uprawnienie EXECUTE dotyczy funkcji, procedur i pakietów, a nie ma nic wspólnego z tabelami.

Tabela 10.9. *Uprawnienia do obiektów*

Uprawnienie do obiektów	Opis
ALTER	Modyfikowanie definicji tabeli lub sekwencji.
DELETE	Usuwanie wierszy z tabeli, widoku lub widoku zmaterializowanego.
EXECUTE	Wykonywanie funkcji lub procedury w pakiecie lub poza nim.
DEBUG	Przeglądanie kodu PL/SQL procedur wyzwalanych, zdefiniowanych dla tabeli albo instrukcji SQL odwołujących się do tabeli. W przypadku typów obiektowych uprawnienie to pozwala na dostęp do wszystkich publicznych i prywatnych zmiennych, metod i typów zdefiniowanych na typie obiektowym.
FLASHBACK	Wykonywanie zapytań Flashback na tabelach, widokach i widokach zmaterializowanych przy użyciu utrzymywanych w bazie danych wycofania.
INDEX	Tworzenie indeksu na tabeli.
INSERT	Wstawianie wierszy do tabeli, widoku lub widoku zmaterializowanego.
ON COMMIT REFRESH	Tworzenie bazujących na tabeli zmaterializowanych widoków odświeżanych po każdym zatwierdzeniu.
QUERY REWRITE	Tworzenie bazujących na tabeli zmaterializowanych widoków dla celów przepisywania zapytań.
READ	Odczytywanie zawartości katalogu systemu operacyjnego przy użyciu definicji DIRECTORY bazy Oracle.
REFERENCES	Tworzenie ograniczeń w postaci kluczy obcych, które odwołują się do klucza głównego innej tabeli albo do klucza unikatowego.
SELECT	Odczytywanie wierszy z tabeli, widoku lub widoku zmaterializowanego wraz z odczytywaniem bieżących lub następnych wartości z sekwencji.
UNDER	Tworzenie widoku bazującego na istniejącym widoku.
UPDATE	Uaktualnianie wierszy w tabeli, widoku lub widoku zmaterializowanym.
WRITE	Zapisanie informacji w katalogu systemu operacyjnego przy użyciu definicji DIRECTORY bazy Oracle.

Warto zaznaczyć, że uprawnień DELETE, UPDATE i INSERT nie można przyznawać do widoków zmaterializowanych, jeśli widoki te nie podlegają zmianom. Niektóre z opisanych uprawnień do obiektów pokrywają się z uprawnieniami systemowymi. Jeśli na przykład nie posiada się uprawnienia FLASHBACK do tabeli, to i tak nadal można wykonywać zapytania Flashback, jeżeli jest się posiadaczem uprawnienia systemowego FLASHBACK ANY TABLE.

W poniższym przykładzie administrator bazy danych przyznaje użytkownikowi SCOTT pełny dostęp do tabeli HR.EMPLOYEES, lecz pozwala mu na przyznawanie innym użytkownikom jedynie uprawnienia SELECT:

```
SQL> grant insert, update, delete on hr.employees to scott;
Grant succeeded.

SQL> grant select on hr.employees to scott with grant option;
Grant succeeded.
```

Należy pamiętać, że jeśli administrator odbierze użytkownikowi SCOTT uprawnienie SELECT do tabeli HR.EMPLOYEES, wówczas to samo uprawnienie zostanie również odebrane innym użytkownikom, którym SCOTT je przypisał.

Uprawnienia do tabel

Typy uprawnień, które można przyznawać do tabeli, dzielą się na dwie obszerne kategorie: operacje DML oraz operacje DDL. Do operacji DML należą operacje delete, insert, select i update, natomiast do operacji DML należą operacje dodawania, usuwania i zmieniania kolumn tabeli oraz tworzenia indeksów na tabeli.

Uprawnienia do wykonywania w tabeli operacji DML można ograniczyć do konkretnych kolumn. Można na przykład pozwolić użytkownikowi SCOTT na przeglądanie i zmienianie wszystkich kolumn i wierszy tabeli HR.EMPLOYEES z wyjątkiem kolumny SALARY. Aby osiągnąć taki efekt, trzeba najpierw odebrać użytkownikowi aktualnie posiadane przez niego uprawnienie SELECT do tabeli:

```
SQL> revoke update on hr.employees from scott;
Revoke succeeded.
```

Następnie można pozwolić użytkownikowi SCOTT na zmienianie wszystkich kolumn oprócz kolumny SALARY:

```
SQL> grant update (employee_id, first_name, last_name, email,
  2                phone_number, hire_date, job_id, commission_pct,
  3                manager_id, department_id)
  4  on hr.employees to scott;

Grant succeeded.
```

Od teraz SCOTT będzie mógł zmieniać wartości we wszystkich kolumnach tabeli HR.EMPLOYEES oprócz kolumn SALARY:

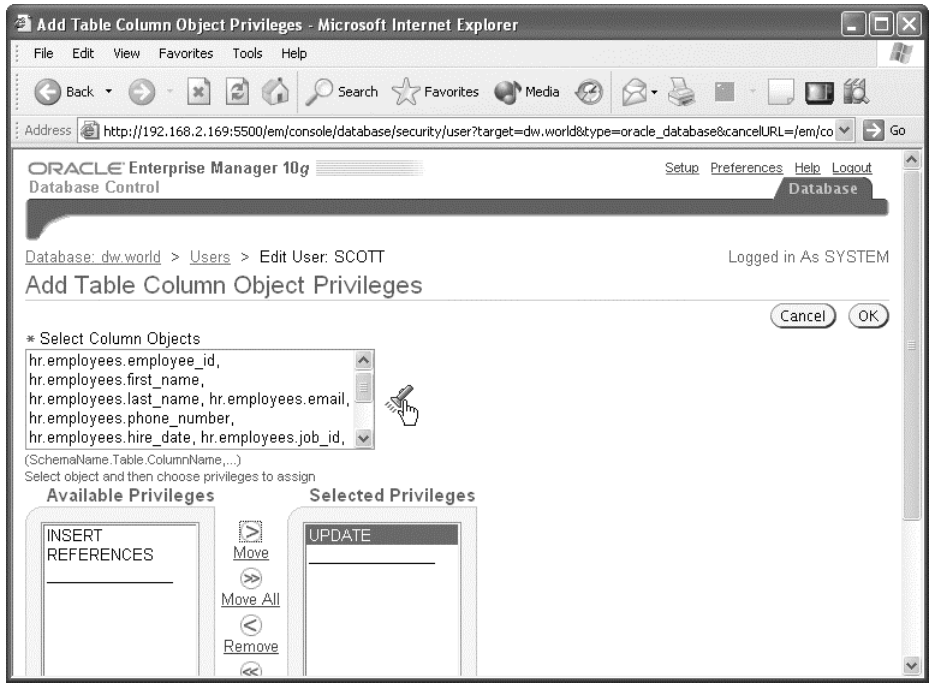
```
SQL> update hr.employees set first_name = 'Stephen' where employee_id = 100;
1 row updated.

SQL> update hr.employees set salary = 50000 where employee_id = 203;
update hr.employees set salary = 50000 where employee_id = 203
       *
ERROR at line 1:
ORA-01031: insufficient privileges
```

Analogiczną operację można bez trudu wykonać przy użyciu narzędzia OEM z interfejsem WWW, jak na rysunku 10.5.

Uprawnienia do widoków

Uprawnienia do widoków działają podobnie jak uprawnienia do tabel. Wiersze widoku można odczytywać, zmieniać, usuwać lub wstawiać przy założeniu, że widok może podlegać zmianom. Aby utworzyć widok, trzeba najpierw posiadać uprawnienie systemowe CREATE VIEW (aby móc utworzyć widok we własnym schemacie) albo uprawnienie systemowe CREATE ANY VIEW (aby móc utworzyć widok w dowolnym schemacie). Jednak w celu utworzenia widoku trzeba również posiadać co najmniej uprawnienie SELECT do tabel stanowiących podstawę



Rysunek 10.5. *Nadawanie uprawnień do kolumn za pomocą narzędzia Oracle Enterprise Manager*

widoku, a także uprawnienia `INSERT`, `UPDATE` i `DELETE`, jeżeli odpowiadające im operacje mają być wykonywane na widoku i widok może podlegać zmianom. Alternatywnie można udzielić uprawnień `SELECT ANY TABLE`, `INSERT ANY TABLE`, `UPDATE ANY TABLE` lub `DELETE ANY TABLE`, jeśli dotyczące ich obiekty nie znajdują się w schemacie użytkownika.

Aby pozwolić innym użytkownikom na korzystanie z posiadanego widoku, trzeba również posiadać uprawnienia z opcją `GRANT OPTION` do tabel stanowiących źródło widoku albo uprawnienia systemowe z opcją `ADMIN OPTION`. Jeśli chcemy utworzyć na przykład widok na tabeli `HR.EMPLOYEES`, trzeba posiadać uprawnienie `SELECT` wraz z opcją `WITH GRANT OPTION` do tabeli `HR.EMPLOYEES` albo uprawnienie systemowe `SELECT ANY TABLE` z opcją `WITH ADMIN OPTION`.

Uprawnienia do procedur

Jedynym uprawnieniem do obiektów, stosowanym względem procedur, funkcji i pakietów zawierających procedury i funkcje, jest uprawnienie `EXECUTE`. Począwszy od wersji Oracle8*i*, procedury i funkcje można wykonywać z perspektywy **użytkownika definiującego** (czyli twórcy procedury lub funkcji) albo z perspektywy **użytkownika wywołującego** (czyli użytkownika, który wykonuje procedurę lub funkcję).

Procedura z prawami użytkownika definiującego jest wykonywana tak, jakby uruchamiał ją użytkownik, który procedurę zdefiniował z obowiązującymi wszystkimi uprawnieniami użytkownika definiującego względem obiektów, do których procedura się odwołuje. Jest to wygodny sposób na narzucenie ograniczeń na prywatne obiekty bazy danych, ponieważ inni użytkownicy posiadają wówczas uprawnienie `EXECUTE` na procedurze, natomiast nie mają uprawnień do obiektów, do których procedura się odwołuje. W efekcie użytkownik, który zdefiniował procedurę, może kontrolować sposób, w jaki inni użytkownicy korzystają z tych obiektów.

Natomiast procedura z prawami użytkownika wywołującego wymaga, by użytkownik wywołujący posiadał do obiektów, do których procedura się odwołuje, prawa bezpośrednie, takie jak SELECT i UPDATE. Procedura może na przykład odwoływać się do niekwalifikowanej tabeli o nazwie ORDERS i jeśli wszyscy użytkownicy bazy danych posiadają tabelę ORDERS, wówczas każdy z nich będzie mógł użyć tej procedury. Kolejną korzyścią płynącą z użycia procedur z prawami użytkownika wywołującego jest fakt, że w procedurach tych obowiązują role. Szczegółowe informacje na temat ról znajdują się w tym rozdziale.

Domyślnie procedura jest tworzona z prawami użytkownika definiującego. Aby nakazać, by procedura używała praw użytkownika wywołującego, należy w definicji procedury użyć słów kluczowych authid *bieżący_użytkownik*, jak w poniższym przykładzie:

```
create or replace procedure process_orders (order_batch_date date)
authid current_user as
begin
    -- przetwarzanie tabeli ORDERS użytkownika z prawami użytkownika wywołującego,
    -- obowiązują wszystkie role
end;
```

Aby utworzyć procedurę, użytkownik musi posiadać uprawnienie systemowe CREATE PROCEDURE lub CREATE ANY PROCEDURE. Aby procedura została poprawnie skompilowana, użytkownik musi posiadać wszystkie bezpośrednie uprawnienia do wszystkich obiektów, do których procedura się odwołuje pomimo tego, że w fazie wykonania procedury z prawami użytkownika wywołującego obowiązują role, za pomocą których uzyskiwane są te same uprawnienia. Aby umożliwić innym użytkownikom dostęp do procedury, należy przypisać mu uprawnienia EXECUTE do procedury lub pakietu.

Widoki danych słownikowych uprawnień do obiektów

Szereg widoków danych słownikowych zawiera dane na temat uprawnień do obiektów przypisanych użytkownikowi. W tabeli 10.10 przedstawiono najważniejsze widoki zawierające dane o uprawnieniach do obiektów.

Tabela 10.10. *Widoki danych słownikowych uprawnień do obiektów*

Widok danych słownikowych	Opis
DBA_TAB_PRIVS	Uprawnienia do tabel, przyznane rolom i użytkownikom. Widok zawiera również dane na temat użytkownika, który nadał uprawnienie roli lub użytkownikowi z opcją GRANT OPTION lub bez niej.
DBA_COL_PRIVS	Uprawnienia do tabel nadane rolom i użytkownikom. Dane zawierają nazwę kolumny oraz typ uprawnienia na kolumnie.
SESSION_PRIVS	Wszystkie uprawnienia do obiektów obowiązujące dla danego użytkownika w danej sesji, przyznane bezpośrednio lub za pośrednictwem roli.
ROLE_TAB_PRIVS	Obowiązujące w bieżącej sesji uprawnienia nadane do tabeli za pośrednictwem ról.

Tworzenie, przypisywanie i utrzymywanie ról

Rola to nazwana grupa uprawnień systemowych lub uprawnień do obiektów albo grupa łącząca w sobie obydwa rodzaje uprawnień; ułatwia administrowanie uprawnieniami. Zamiast nadawać uprawnienia oddzielnie każdemu użytkownikowi, grupę uprawnień systemowych lub uprawnień do obiektów można przypisać wybranej roli, a następnie przypisać tę rolę

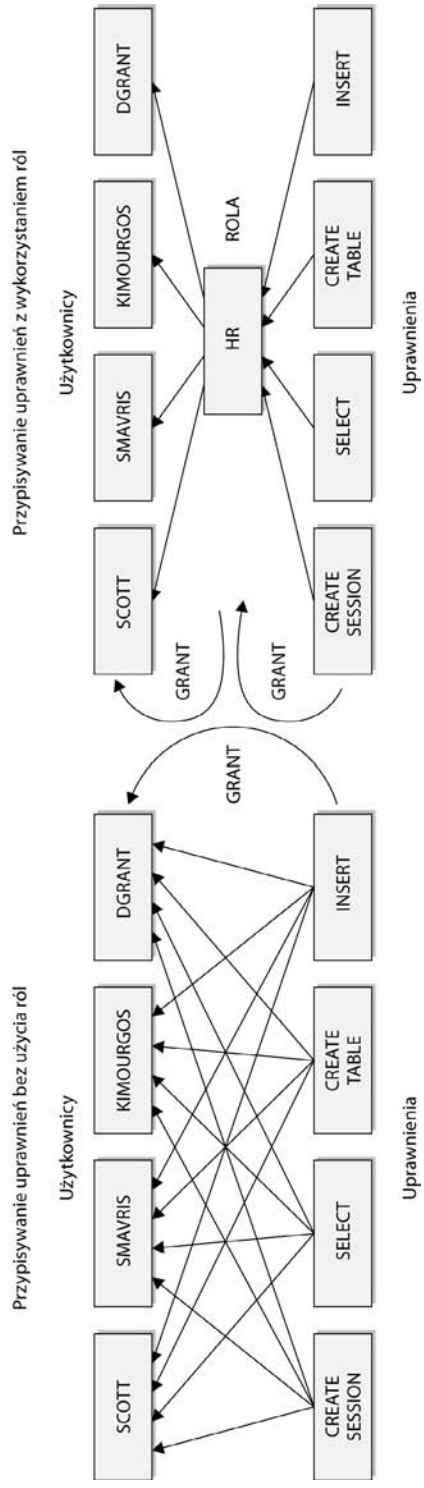
użytkownikowi. Dzięki temu można zdecydowanie zmniejszyć ilość czasu spędzanego na utrzymywaniu uprawnień użytkowników. Na rysunku 10.6 pokazano, w jaki sposób dzięki roli można zmniejszyć liczbę potrzebnych poleceń `grant` (a przez to również poleceń `revoke`), które należy wykonywać, jeśli rolom przypisze się grupy uprawnień.

Gdy zajdzie konieczność zmiany uprawnień grupy użytkowników, którym przypisano rolę, wówczas wystarczy tylko zmienić uprawnienia przypisane tej roli, a w zakresie działań dozwolonych dla posiadających ją użytkowników automatycznie zostaną uwzględnione nowe lub zmienione uprawnienia. Role można włączać poszczególnym użytkownikom; w szczególności niektóre role mogą być automatycznie włączane w momencie logowania. Ponadto role można zabezpieczać hasłami, co obok mechanizmów uwierzytelniania w bazie danych będzie stanowić kolejny poziom uprawnień.

W tabeli 10.11 zaprezentowano najczęściej używane role automatycznie udostępniane w każdej bazie danych, a także krótki opis zakresu przywilejów przypisanych każdej z ról.

Tabela 10.11. *Predefiniowane role bazy danych Oracle*

Nazwa roli	Uprawnienia
CONNECT	ALTER SESSION, CREATE CLUSTER, CREATE DATABASE LINK, CREATE SEQUENCE, CREATE SESSION, CREATE SYNONYM, CREATE TABLE, CREATE VIEW. Wymienione uprawnienia są zwykle nadawane standardowemu użytkownikowi bazy danych i umożliwiają łączenie się z bazą oraz tworzenie tabel, indeksów i widoków.
RESOURCE	CREATE CLUSTER, CREATE INDEXTYPE, CREATE OPERATOR, CREATE PROCEDURE, CREATE SEQUENCE, CREATE TABLE, CREATE TRIGGER, CREATE TYPE. Wymienione uprawnienia są zwykle przypisywane programistom aplikacji, którzy implementują procedury i funkcje PL/SQL.
DBA	Wszystkie uprawnienia systemowe z opcją WITH ADMIN OPTION. Dzięki temu użytkownik posiadający rolę DBA może nadawać innym użytkownikom uprawnienia systemowe.
DELETE_CATALOG_ROLE	Nie posiada żadnych uprawnień systemowych, a jedynie uprawnienia do obiektów (DELETE) na SYS.AUD$ oraz FGA_LOG$. Inaczej mówiąc, rola pozwala użytkownikowi na usuwanie rekordów monitorowania ze śladu monitorowania zwykłego lub szczegółowego.
EXECUTE_CATALOG_ROLE	Uprawnienie wykonywania różnych pakietów systemowych, procedur i funkcji, takich jak DBMS_FGA oraz DBMS_RLG.
SELECT_CATALOG_ROLE	Uprawnienie SELECT do 1638 tabel danych słownikowych.
EXP_FULL_DATABASE	EXECUTE_CATALOG_ROLE, SELECT_CATALOG_ROLE oraz uprawnienia systemowe takie jak BACKUP ANY TABLE oraz RESUMABLE. Użytkownik posiadający tę rolę może eksportować wszystkie obiekty znajdujące się w bazie danych.
IMP_FULL_DATABASE	Rola podobna do EXP_FULL_DATABASE, lecz posiadająca znacznie więcej uprawnień systemowych, takich jak CREATE ANY TABLE. Umożliwia importowanie wyeksportowanych wcześniej wszystkich obiektów bazy danych.
AQ_USER_ROLE	Umożliwia wykonywanie procedur niezbędnych dla mechanizmu Advanced Queuing, na przykład DBMS_AQ.
AQ_ADMINISTRATOR_ROLE	Rola menedżera kolejek mechanizmu Advanced Queuing.
SNMPAGENT	Rola używana przez agenta Enterprise Manager Intelligent Agent.
RECOVERY_CATALOG_OWNER	Służy do tworzenia użytkownika będącego właścicielem katalogu przywracania narzędzia RMAN do tworzenia kopii zapasowych i ich przywracania.
HS_ADMIN_ROLE	Umożliwia dostęp do tabel HS_* oraz pakietu DBMS_HS, aby umożliwić administrowanie usługami Oracle Heterogeneous Services.
SCHEDULER_ADMIN	Pozwala na dostęp do pakietu DBMS_SCHEDULER wraz z uprawnieniami do tworzenia zadań wsadowych.



Rysunek 10.6. *Zarządzanie uprawnieniami przy użyciu ról*

Role CONNECT, RESOURCE i DBA są dostępne głównie w celu zapewnienia zgodności z wcześniejszymi wersjami bazy Oracle i w przyszłych wersjach może ich zabraknąć. Administratorzy bazy danych powinni tworzyć własne role i przypisywać im ten sam zakres uprawnień, jaki jest obecnie przypisany rolom, które mogą zniknąć z kolejnych wersji bazy danych.

Tworzenie lub usuwanie roli

Aby utworzyć rolę, należy wykonać polecenie create role, co wymaga posiadania uprawnienia CREATE ROLE. Uprawnienie to zwykle jest przypisywane jedynie administratorom baz danych lub administratorom aplikacji. Przykładowe polecenie create role prezentuje się następująco:

```
SQL> create role hr_admin not identified;
Role created.
```

Domyślnie do włączenia lub użycia przypisanej roli nie wymaga się hasła ani uwierzytelnienia, a więc klauzula not identified ma charakter opcjonalny.

Podobnie jak w przypadku użytkowników, korzystanie z roli może wymagać autoryzacji w bazie danych (klauzula identified by *hasło*), w systemie operacyjnym (klauzula identified externally) lub w sieci albo przez usługę katalogową (klauzula identified globally).

Oprócz wspomnianych, znanych nam już metod można do autoryzowania roli użyć również pakietu. Mamy wówczas do czynienia z tak zwaną bezpieczną rolą aplikacji. Włączanie tego rodzaju roli jest wykonywane przez procedurę wchodzącą w skład pakietu. Rolę włącza się zazwyczaj tylko w określonych okolicznościach: gdy użytkownik łączy się za pośrednictwem interfejsu WWW lub z określonego adresu IP albo gdy połączenie następuje w określonej porze dnia. Rolę włączaną za pomocą procedury tworzy się w następując sposób:

```
SQL> create role hr_clerk identified using hr.clerk_verif;
Role created.
```

Procedura HR.CLERK_VERIF może nie istnieć w momencie tworzenia roli, musi natomiast być już skompilowana i zwalidowana w momencie, gdy trzeba ją włączyć użytkownikowi, któremu tę rolę przyznano. Bezpieczne role aplikacji zwykle nie są włączane użytkownikom domyślnie. Aby nakazać, by wszystkie role oprócz bezpiecznych ról aplikacji były domyślnie włączone, należy wykonać następujące polecenie:

```
SQL> alter user wgietz default role all except hr_clerk;
User altered.
```

Dzięki temu w momencie uruchomienia aplikacji HR można włączyć rolę poleceniem set role hr_clerk i wywołując tym samym procedurę HR.CLERK_VERIF. Użytkownik nie powinien znać roli ani procedury, która tę rolę włącza. Dzięki temu użytkownicy spoza aplikacji nie mają dostępu do obiektów i uprawnień udostępnianych przez włączaną rolę.

Usunięcie roli jest równie proste, jak jej utworzenie:

```
SQL> drop role keypunch_operator;
Role dropped.
```

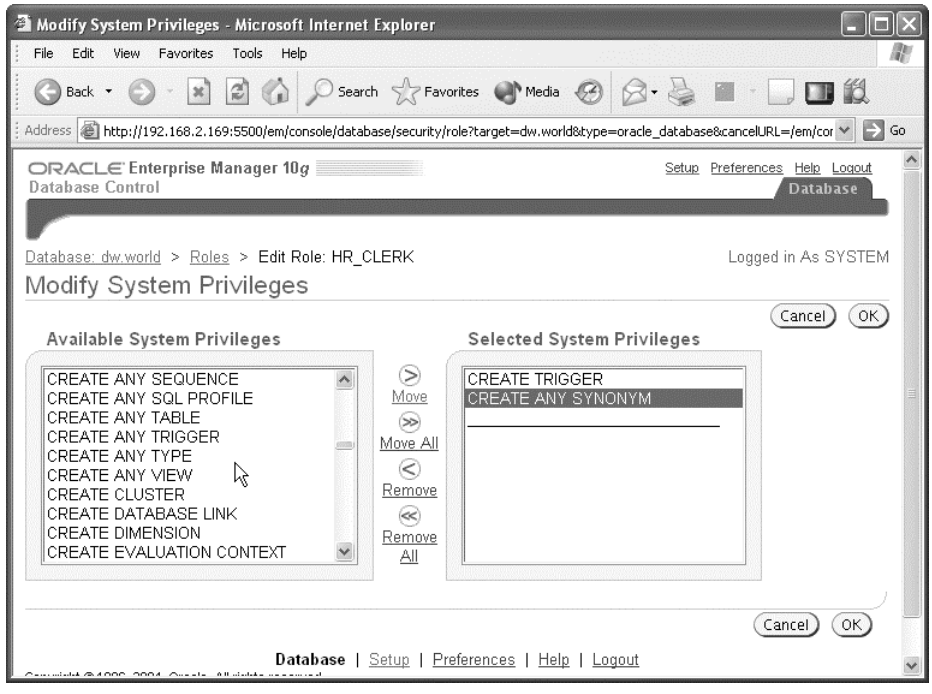
Każdy użytkownik, który połączy się z bazą danych po tym, gdy wykonano przedstawione polecenie, utraci wszystkie uprawnienia przypisane usuniętej roli. Jeżeli użytkownik będzie zalogowany w momencie usuwania roli, utrzyma posiadane uprawnienia do momentu rozłączenia się z bazą danych.

Nadawanie uprawnień roli

Przypisywanie uprawnień roli jest bardzo prostym zadaniem. Wystarczy użyć polecenia `grant` i przypisać uprawnienie roli w taki sam sposób, w jaki nadaje się uprawnienie użytkownikowi:

```
SQL> grant select on hr.employees to hr_clerk;
Grant succeeded.
SQL> grant create trigger to hr_clerk;
Grant succeeded.
```

W powyższym przykładzie roli `HR_CLERK` przypisano uprawnienie do obiektu oraz uprawnienie systemowe. Na rysunku 10.7 przedstawiono dostępny z poziomu przeglądarki internetowej interfejs narzędzia OEM, w którym można przypisać roli więcej uprawnień systemowych i uprawnień do obiektów.



Rysunek 10.7. *Nadawanie roli uprawnień w narzędziu OEM*

Przypisywanie i odbieranie roli

Gdy roli przypisano już odpowiednie uprawnienia systemowe i uprawnienia do obiektów, można tę rolę przypisać odpowiednim użytkownikom za pomocą znanej nam już składni:

```
SQL> grant hr_clerk to smavris;
Grant succeeded.
```

Wszystkie uprawnienia, które w przyszłości zostaną dodane do roli `HR_CLERK`, automatycznie staną się dostępne dla użytkownika `SMAVRIS` będącego posiadaczem roli.

Role mogą być nadawane innym rolom. W ten sposób administrator bazy danych może tworzyć hierarchie ról i ułatwiać sobie zadania administracyjne. Można na przykład utworzyć role DEPT30, DEPT50 i DEPT100, z których każda będzie posiadać uprawnienia do tabel zawierających dane na temat działów (departamentów) odpowiadających poszczególnym rolom. Pracownikowi działu (departamentu) 30. należałoby przypisać rolę DEPT30 itd. Prezes firmy powinien mieć dostęp do wszystkich tabel wszystkich departamentów, jednak w celu zapewnienia mu takiego dostępu roli ALL_DEPTS nie trzeba przypisywać wszystkich pojedynczych uprawnień do obiektów, lecz wystarczy przypisać jej wszystkie role odpowiadające pojedynczym departamentom:

```
SQL> create role all_depts;
Role created.
SQL> grant dept30, dept50, dept100 to all_depts;
Grant succeeded.
SQL> grant all_depts to sking;
Grant succeeded.
```

Rola ALL_DEPTS może również posiadać pojedyncze uprawnienia systemowe i uprawnienia do obiektów, które nie są związane z konkretnymi departamentami. Można na przykład przypisać roli ALL_DEPTS dodatkowe uprawnienia do tabel zamówień albo tabel należności.

Rolę odbiera się użytkownikowi w podobny sposób, w jaki odbiera się uprawnienia:

```
SQL> revoke all_depts from sking;
Revoke succeeded.
```

Użytkownik utraci odebrane mu przywileje w momencie, gdy po raz pierwszy po wykonaniu tej operacji zaloguje się do bazy danych. Trzeba jednak pamiętać, że jeśli inna rola posiada te same uprawnienia do tych samych obiektów co rola odebrana lub uprawnienia do tych obiektów zostały użytkownikowi przypisane w sposób bezpośredni, użytkownik utrzyma te uprawnienia do obiektów do momentu, aż nie zostaną one jawnie odebrane.

Role domyślne

Domyślnie wszystkie role przypisane użytkownikowi są włączane w momencie, gdy użytkownik nawiązuje połączenie z bazą danych. Jeżeli rola ma być używana jedynie w kontekście aplikacji, można skonfigurować ją w taki sposób, by w momencie zalogowania się użytkownika do bazy była wyłączona. Można ją następnie włączyć i wyłączyć w samej aplikacji. Jeżeli użytkownik SCOTT będzie posiadał role CONNECT, RESOURCE, HR_CLERK i DEPT30 — i role HR_CLERK oraz DEPT30 nie powinny być włączane automatycznie, należy wykonać polecenie podobne do poniższego:

```
SQL> alter user scott default role all
  2>     except hr_clerk, dept30;
User altered.
```

Gdy SCOTT połączy się z bazą danych, automatycznie uzyska wszystkie uprawnienia przypisane wszystkim powyższym rolom z wyjątkiem ról HR_CLERK i DEPT30. SCOTT może jawnie włączyć rolę we własnej sesji przy użyciu polecenia `set role`:

```
SQL> set role dept30;
Role set.
```

Gdy SCOTT zakończy pracę na danych znajdujących się w tabeli departamentu 30., może wyłączyć sobie rolę DEPT30 w bieżącej sesji:

```
SQL> set role all except dept30;
Role set.
```


Uwaga

> W wersji Oracle 10*g* parametr inicjalizacyjny MAX_ENABLED_ROLES jest parametrem przestarzałym. Parametr pozostawiono jedynie w celu zapewnienia wstecznej zgodności z poprzednimi wersjami bazy Oracle.

Role włączane hasłem

Aby zwiększyć poziom zabezpieczeń w bazie danych, administrator bazy może przypisać roli hasło. Hasło przypisuje się do roli w momencie jej tworzenia:

```
SQL> create role dept99 identified by d99secretpw;
Role created.
SQL> grant dept99 to scott;
Grant succeeded.
SQL> alter user scott default role all except hr_clerk, dept30, dept99;
User altered.
```

Gdy użytkownik SCOTT połączy się z bazą danych, używana przez niego aplikacja sama poda hasło lub będzie wymagała jego wpisania; SCOTT będzie mógł samodzielnie włączyć rolę po wpisaniu hasła:

```
SQL> set role dept99 identified by d99secretpw;
Role set.
```

Widoki danych słownikowych ról

W tabeli 10.12 znajduje się lista widoków danych słownikowych dotyczących ról.

Tabela 10.12. *Widoki danych słownikowych na temat ról*

Widok danych słownikowych	Opis
DBA_ROLES	Dane na temat wszystkich ról oraz informacje, czy wymagają one podania hasła.
DBA_ROLE_PRIVS	Role przypisane użytkownikom lub inne role.
ROLE_ROLE_PRIVS	Role przypisane innym rolom.
ROLE_SYS_PRIVS	Uprawnienia systemowe przyznane rolom.
ROLE_TAB_PRIVS	Uprawnienia do tabel i kolumn tabel przypisane rolom.
SESSION_ROLES	Role obowiązujące w danej sesji. Dotyczy wszystkich sesji użytkowników.

Widok DBA_ROLE_PRIVS stanowi doskonałe źródło informacji na temat ról przypisanych użytkownikowi, możliwości przekazywania ról innym użytkownikom (kolumna ADMIN_OPTION) oraz tego, czy rola jest domyślnie włączona (kolumna DEFAULT_ROLE):

```
SQL> select * from dba_role_privs
  2      where grantee = 'SCOTT';

GRANTEE      GRANTED_ROLE         ADMIN_OPTION DEFAULT_ROLE
------------ -------------------- ------------ ------------
SCOTT        DEPT30               NO           NO
SCOTT        DEPT50               NO           YES
SCOTT        DEPT99               NO           YES
SCOTT        CONNECT              NO           YES
SCOTT        HR_CLERK             NO           NO
SCOTT        RESOURCE             NO           YES
SCOTT        ALL_DEPTS            NO           YES
SCOTT        DELETE_CATALOG_ROLE  NO           YES

8 rows selected.
```

W podobny sposób można sprawdzić, które role przypisano roli ALL_DEPTS:

```
SQL> select * from dba_role_privs
  2>     where grantee = 'ALL_DEPTS';

GRANTEE      GRANTED_ROLE         ADMIN_OPTION DEFAULT_ROLE
------------ -------------------- ------------ ------------
ALL_DEPTS    DEPT30               NO           YES
ALL_DEPTS    DEPT50               NO           YES
ALL_DEPTS    DEPT100              NO           YES

3 rows selected.
```

Widok danych słownikowych ROLE_ROLE_PRIVS również zawiera podobne informacje, lecz dane te dotyczą jedynie ról przypisanych innym rolom. Widok nie posiada również kolumny DEFAULT_ROLE informującej, czy dana rola jest domyślnie włączona.

Aby odczytać uprawnienia przyznane użytkownikowi do tabeli lub kolumn tabeli, można wykonać dwa zapytania. Jedno będzie odczytywać uprawnienia przypisane w sposób bezpośredni, zaś drugie odczyta uprawnienia przyznane pośrednio, za pośrednictwem roli. Uprawnienia przyznane bezpośrednio można odczytać bez trudu:

```
SQL> select dtp.grantee, dtp.owner, dtp.table_name,
  2       dtp.grantor, dtp.privilege, dtp.grantable
  3  from dba_tab_privs dtp
  4  where dtp.grantee = 'SCOTT';

GRANTEE      OWNER      TABLE_NAME      GRANTOR      PRIVILEGE    GRANTABLE
------------ ---------- --------------- ------------ ------------ ----------
SCOTT        HR         EMPLOYEES       HR           SELECT       YES
SCOTT        HR         EMPLOYEES       HR           DELETE       NO
SCOTT        HR         EMPLOYEES       HR           INSERT       NO

4 rows selected.
```

Aby odczytać uprawnienia do tabel przyznane za pośrednictwem ról, należy złączyć tabele DBA_ROLE_PRIVS oraz ROLE_TAB_PRIVS. Tabela DBA_ROLE_PRIVS zawiera dane na temat ról przypisanych użytkownikowi, natomiast w tabeli ROLE_TAB_PRIVS znajdują się uprawnienia przypisane rolom:

```
SQL> select drp.grantee, rtp.owner, rtp.table_name,
  2      rtp.privilege, rtp.grantable, rtp.role
  3  from role_tab_privs rtp
  4      join dba_role_privs drp on rtp.role = drp.granted_role
  5  where drp.grantee = 'SCOTT';

GRANTEE    OWNER    TABLE_NAME      PRIVILEGE    GRANTABLE  ROLE
---------- -------- --------------- ------------ ---------- ---------------
SCOTT      HR       EMPLOYEES       SELECT       NO         HR_CLERK
SCOTT      HR       JOBS            SELECT       NO         JOB_MAINT
SCOTT      HR       JOBS            UPDATE       NO         JOB_MAINT
SCOTT      SYS      AUD$            DELETE       NO         DELETE_CATA
                                                           LOG_ROLE
SCOTT      SYS      FGA_LOG$        DELETE       NO         DELETE_CATA
                                                           LOG_ROLE

5 rows selected.
```

Z danych na temat uprawnień użytkownika `SCOTT` wynika, że posiada on uprawnienie `SELECT` na tabeli `HR.EMPLOYEES`, przyznane zarówno bezpośrednią instrukcją `grant`, jak i wynikające z przypisanej roli. Nawet w przypadku odebrania użytkownikowi któregoś z tych dwóch uprawnień `SCOTT` nadal będzie posiadał dostęp do tabeli `HR.EMPLOYEES` — do czasu, aż zostaną mu odebrane obydwa uprawnienia.

Implementowanie polityk bezpieczeństwa aplikacji przy użyciu wirtualnych prywatnych baz danych

Wirtualna prywatna baza danych (ang. *Virtual Private Database — VPD*) łączy w sobie serwerowe mechanizmy precyzyjnej kontroli dostępu z bezpiecznym kontekstem aplikacji. Funkcje kontekstowe zwracają predykat — klauzulę `where` — automatycznie dołączany do wszystkich instrukcji `SELECT` lub instrukcji DML. Inaczej mówiąc, instrukcja `select` na tabeli, widoku lub synonimie kontrolowanym przez VPD zwróci podzbiór wierszy wyznaczany przez klauzulę `where` wygenerowaną automatycznie przez funkcję polityki bezpieczeństwa obowiązującą w kontekście aplikacji. Kluczowym komponentem VPD są zabezpieczenia na poziomie wiersza (ang. *row-level security — RLS*), znane również jako precyzyjna kontrola dostępu (ang. *fine-grained access control — FGAC*).

Ponieważ VPD generuje predykaty w sposób niejawny w trakcie parsowania instrukcji, polityka bezpieczeństwa jest stosowana w sposób spójny bez względu na to, czy użytkownik wykonuje zapytanie ad hoc w celu odczytania danych z aplikacji, czy też przegląda dane na przykład z poziomu aplikacji Oracle Forms. Dzięki temu, że Oracle Server dołącza predykat do instrukcji w trakcie parsowania, aplikacja nie musi używać specjalnie przystosowanych tabel, widoków i tym podobnych obiektów, aby pozostać w zgodzie z polityką. W efekcie Oracle może optymalizować zapytania przy użyciu indeksów, widoków zmaterializowanych i z zastosowaniem operacji równoległych, co w innych przypadkach mogłoby być niemożliwe. Użycie VPD może więc zmniejszyć obciążenie bazy danych w porównaniu do zapytania, którego wyniki byłyby dodatkowo filtrowane przez aplikacje lub inne rozwiązania.

Z punktu widzenia utrzymania bazy polityki bezpieczeństwa można definiować w ramach funkcji polityki, której utworzenie jedynie przy użyciu ról i uprawnień byłoby bardzo trudne. Analogicznie usługodawcy aplikacji (ASP) wystarczy jedynie skonfigurować jedną bazę danych,

która będzie obsługiwać wielu klientów w ramach tej samej aplikacji i która będzie posiadać politykę VPD, zapewniającą, że pracownicy danego klienta będą widzieć tylko przeznaczone dla nich dane. Administrator bazy danych może utrzymywać jedną większą bazę zawierającą kilka polityk VPD zamiast oddzielnych baz danych przeznaczonych dla poszczególnych klientów.

Nowością w wersji Oracle 10*g* są operacje VPD na poziomie kolumn. Dzięki użyciu VPD na poziomie kolumn administrator może ograniczyć dostęp do określonej kolumny lub kolumn tabeli. Zapytanie będzie wówczas zwracać tę samą liczbę wierszy, lecz jeśli w kontekście użytkownika dostęp do którejś z kolumn nie będzie dozwolony, wówczas dla tak zablokowanej kolumny lub kolumn zwrócone zostaną wartości `NULL`.

Polityki VPD mogą mieć charakter statyczny, zależny od kontekstu lub dynamiczny. Polityki statyczne oraz zależne od kontekstu są nowością w wersji Oracle 10*g* i mogą istotnie zwiększyć wydajność bazy danych dzięki temu, że polityki nie trzeba wywoływać za każdym razem, gdy wykonywane jest zapytanie, ponieważ polityki są buforowane w pamięci podręcznej z myślą o ich późniejszym ponownym użyciu w danej sesji. W wersjach wcześniejszych niż Oracle 10*g* wszystkie polityki były dynamiczne, to znaczy funkcja polityki była wykonywana za każdym razem, gdy parsowana była instrukcja SQL zawierająca docelową tabelę VPD. Polityki statyczne są przetwarzane tylko raz w momencie logowania się i pozostają w pamięci podręcznej przez cały czas trwania sesji bez względu na kontekst aplikacji. W przypadku polityk zależnych od kontekstu funkcja polityki jest wywoływana w fazie parsowania w razie zmiany kontekstu aplikacji. Przykładem może być polityka implementująca regułę mówiącą, że „pracownicy mogą przeglądać jedynie historię własnych zarobków, natomiast menedżerowie mogą przeglądać historię zarobków wszystkich swoich pracowników". Jeżeli pracownik wykonujący instrukcję nie zmienił się, funkcja polityki nie musi być wywoływana ponownie. Dzięki temu zmniejsza się obciążenie wynikające z obowiązywania polityki VPD.

Konteksty aplikacji tworzy się za pomocą polecenia `create context`, a za zarządzanie politykami odpowiada pakiet `DBMS_RLS`. Funkcję, która zwraca predykaty narzucające politykę, tworzy się podobnie jak każdą inną funkcję. Jedyną różnicą jest fakt, że funkcja taka musi pobierać dwa wymagane parametry i zwracać wartość typu `VARCHAR2`. W dalszej części niniejszego rozdziału przyjrzymy się bliżej funkcjom polityki oraz przeanalizujemy działanie przykładowej VPD na podstawie tych samych schematów udostępnianych w trakcie procesu instalacji bazy danych Oracle.

Kontekst aplikacji

Poleceniem `create context` można utworzyć nazwy atrybutów definiowanych przez aplikację, które będą używane do egzekwowania polityki bezpieczeństwa, a także nazwę pakietu przeznaczonego dla funkcji i procedur używanych do definiowania kontekstu bezpieczeństwa w ramach sesji użytkownika. Oto przykład:

```
create context hr_security using vpd.emp_access;

create or replace package emp_access as
    procedure set_security_parameters;
end;
```

W powyższym kodzie nazwą kontekstu jest HR_SECURITY, zaś pakiet używany do definiowania charakterystyk lub atrybutów w trakcie sesji użytkownika nosi nazwę EMP_ACCESS. Procedura SET_SECURITY_PARAMETERS będzie wywoływana przez procedurę wyzwalaną w momencie logowania. Ponieważ kontekst HR_SECURITY jest powiązany jedynie z pakietem EMP_ACCESS, żadna inna procedura nie może zmienić atrybutów sesji. W ten sposób zapewnione jest bezpieczeństwo kontekstu aplikacji, którego po połączeniu z bazą danych nie może zmienić użytkownik ani żaden inny proces.

W standardowym pakiecie implementującym kontekst aplikacji korzysta się z wbudowanego kontekstu USERENV, który zawiera informacje na temat sesji użytkownika. W tabeli 10.13 przedstawiono najważniejsze parametry kontekstu USERENV.

Tabela 10.13. *Najważniejsze parametry kontekstu USERENV*

Parametr	Zwracana wartość
CURRENT_SCHEMA	Domyślny schemat sesji.
DB_NAME	Nazwa bazy danych zgodna z wartością parametru inicjalizacyjnego DB_NAME.
HOST	Nazwa komputera, z którego łączy się użytkownik.
IP_ADDRESS	Adres IP, z którego łączy się użytkownik.
OS_USER	Konto systemu operacyjnego, z którego zainicjowano sesję bazy danych.
SESSION_USER	Nazwa uwierzytelnionego użytkownika bazy danych.

Wywołanie na przykład SYS_CONTEXT w poniższej postaci zwróci nazwę użytkownika oraz adres IP sesji bazy danych:

```
declare
    username varchar2(30);
    ip_addr varchar2(30);
begin
    username := SYS_CONTEXT('USERENV','SESSION_USER');
    ip_addr := SYS_CONTEXT('USERENV','IP_ADDRESS');
    -- dalsze instrukcje
end;
```

Funkcji SYS_CONTEXT można użyć w podobny sposób również w instrukcji select języka SQL:

```
SQL> select SYS_CONTEXT('USERENV','SESSION_USER') username from dual;

USERNAME
-------------------------
SCOTT
```

Na podstawie pozyskanych z USERENV danych dotyczących kontekstu i autoryzacji można procedurą DBMS_SESSION.SET_CONTEXT przypisać wartości parametrom w tworzonym kontekście aplikacji:

```
dbms_session.set_context('HR_SECURITY','SEC_LEVEL','HIGH');
```

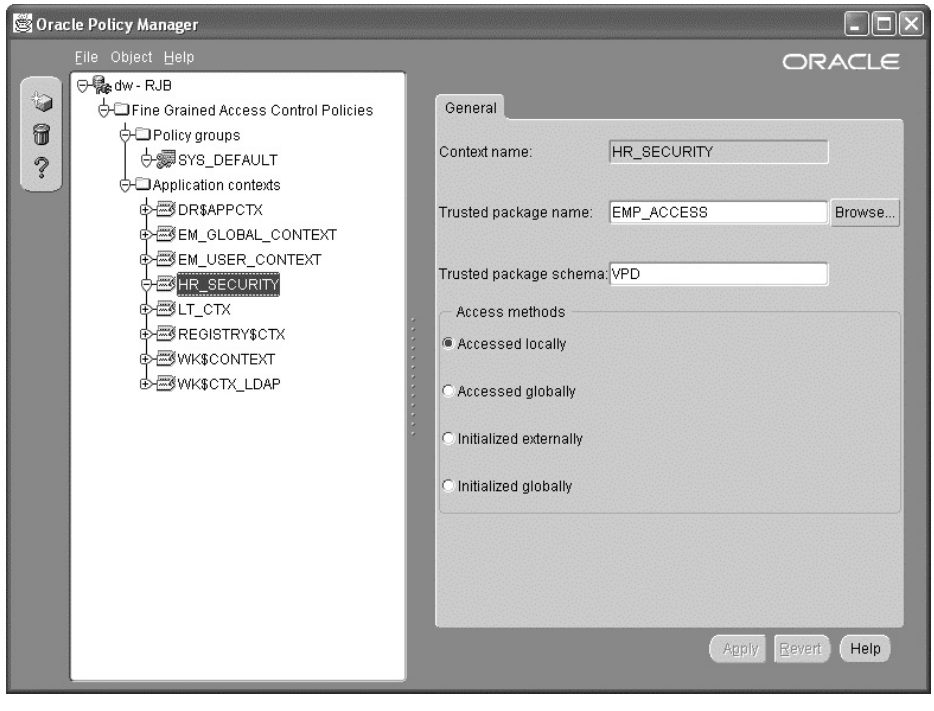
Przedstawione polecenie przypisuje zmiennej kontekstu aplikacji SEC_LEVEL funkcjonującej w kontekście HR_SECURITY wartość HIGH. Wartość można przypisywać zmiennej na podstawie różnych warunków — na przykład na podstawie danych z tabeli, w której identyfikatorom użytkowników przypisane zostały poziomy zabezpieczeń.

Aby zapewnić, że wartości zmiennych kontekstowych są ustawiane dla każdej sesji, można użyć procedury wyzwalanej w momencie logowania, która będzie wywoływać procedurę powiązaną z kontekstem. Jak wspomniano już wcześniej, wartość zmiennej funkcjonującej w kontekście może być definiowana lub zmieniana wyłącznie wewnątrz przypisanego pakietu. Poniżej przedstawiono przykładową procedurę wyzwalaną w momencie logowania, która wywołuje procedurę konfigurującą kontekst:

```
create or replace trigger vpd.set_security_parameters
    after logon on database
begin
    vpd.emp_access.set_security_parameters;
end;
```

W przykładzie tym procedura `SET_SECURITY_PARAMETERS` będzie wywoływać niezbędną procedurę `DBMS_SESSION.SET_CONTEXT`.

Oracle Enterprise Manager udostępnia narzędzie Policy Manager, za pomocą którego można konfigurować grupy kontekstów i polis w sposób zaprezentowany na rysunku 10.8.



Rysunek 10.8. *Oracle Policy Manager*

Implementacja polityki zabezpieczeń

Gdy dostępna jest już pełna infrastruktura niezbędna do skonfigurowania środowiska zabezpieczeń, kolejnym krokiem jest zdefiniowanie jednej lub więcej funkcji, która będzie generować predykat dołączany do każdej instrukcji `select` lub polecenia DML wykonywanego na chronionych tabelach. Funkcja, której zadaniem jest generowanie predykatu, przyjmuje dwa argumenty: właściciela chronionego obiektu oraz nazwę obiektu w schemacie właściciela.

Jedna funkcja może generować predykat tylko dla jednego rodzaju operacji — na przykład dla instrukcji `select` — albo dotyczyć jednocześnie wszystkich poleceń DML zależnie od sposobu, w jaki powiązano ją z chronioną tabelą. Poniżej przedstawiono treść pakietu zawierającego dwie funkcje. Jedna będzie odpowiedzialna za kontrolę dostępu w instrukcjach `select`, natomiast druga funkcja dotyczy wszystkich pozostałych instrukcji DML:

```
create or replace package body get_predicates is

    function emp_select_restrict(owner varchar2, object_name varchar2)
            return varchar2 is
        ret_predicate varchar2(1000); -- część klauzuli WHERE
    begin
        -- pozwala na przeglądanie wierszy tabeli tylko wybranym pracownikom
        -- . . . sprawdzenie zmiennych kontekstowych i utworzenie predykatu
        return ret_predicate;
    end emp_select_restrict;

function emp_dml_restrict(owner varchar2, object_name varchar2)
                return varchar2 is
        ret_predicate varchar2(1000); -- część klauzuli WHERE
    begin
        -- pozwala na wprowadzanie zmian w tabeli tylko wybranym pracownikom
        -- . . . sprawdzenie zmiennych kontekstowych i utworzenie predykatu
        return ret_predicate;
    end emp_dml_restrict;

end; -- treści pakietu
```

Każda funkcja zwraca ciąg znaków zawierający wyrażenie, które będzie dodawane do klauzuli `where` instrukcji `select` lub polecenia DML. Ani użytkownik, ani aplikacja nigdy nie mają dostępu do tak utworzonej klauzuli `WHERE`, ponieważ jest ona automatycznie dodawana do polecenia w fazie jego parsowania.

Programista odpowiada za to, by funkcje zawsze zwracały prawidłowe wyrażenie. W przeciwnym razie każda próba dostępu do zabezpieczonej tabeli zakończy się niepowodzeniem, jak w poniższym przykładzie:

```
SQL> select * from hr.employees;
select * from hr.employees
              *
ERROR at line 1:
ORA-28113: policy predicate has error
```

Komunikat o błędzie nie zawiera samego predykatu i żaden użytkownik nie uzyska dostępu do tabeli, dopóki błąd znajdujący się w funkcji generującej predykat nie zostanie usunięty. Porady na temat sposobu debugowania funkcji generujących predykaty zostaną zaprezentowane w dalszej części tego rozdziału.

Używanie pakietu DBMS_RLS

Wbudowany pakiet `DBMS_RLS` zawiera szereg podprogramów przeznaczonych dla administratora bazy danych. Podprogramy te są przeznaczone do utrzymywania polityk bezpieczeństwa powiązanych z tabelami, widokami i synonimami. W tabeli 10.14 przedstawiono podprogramy dostępne w pakiecie `DBMS_RLS`. Każdy użytkownik, który będzie chciał tworzyć albo administrować politykami, musi posiadać uprawnienia `EXECUTE` do pakietu `SYS.DBMS_RLS`.

Tabela 10.14. *Podprogramy znajdujące się w pakiecie DBMS_RLS*

Podprogram	Opis
ADD_POLICY	Dodaje do obiektu politykę precyzyjnej kontroli dostępu.
DROP_POLICY	Usuwa z obiektu politykę FGAC.
REFRESH_POLICY	Na nowo parsuje wszystkie powiązane z polityką instrukcje znajdujące się w pamięci podręcznej.
ENABLE_POLICY	Włącza lub wyłącza politykę FGAC.
CREATE_POLICY_GROUP	Tworzy grupę polityk.
ADD_GROUPED_POLICY	Dodaje politykę do grupy polityk.
ADD_POLICY_CONTEXT	Dodaje kontekst dla bieżącej aplikacji.
DELETE_POLICY_GROUP	Usuwa grupę polityk.
DROP_GROUPED_POLICY	Usuwa politykę z grupy polityk.
DROP_POLICY_CONTEXT	Usuwa kontekst aktywnej aplikacji.
ENABLE_GROUPED_POLICY	Włącza lub wyłącza grupę polityk.
DISABLE_GROUPED_POLICY	Wyłącza grupę polityk.
REFRESH_GROUPED_POLICY	Na nowo parsuje wszystkie powiązane z grupą polityk instrukcje znajdujące się w pamięci podręcznej.

W niniejszym punkcie opisane zostaną najczęściej używane podprogramy: ADD_POLICY oraz DROP_POLICY. Składnia ADD_POLICY przedstawia się następująco:

```
DBMS_RLS.ADD_POLICY
(
    object_schema          IN varchar2 null,
    object_name            IN varchar2,
    policy_name            IN varchar2,
    function_schema        IN varchar2 null,
    policy_function        IN varchar2,
    statement_types        IN varchar2 null,
    update_check           IN boolean false,
    enable                 IN boolean true,
    static_policy          IN boolean false,
    policy_type            IN binary_integer null,
    long_predicate         IN in Boolean false,
    sec_relevant_cols      IN varchar2,
    sec_relevant_cols_opt  IN binary_integer null
);
```

Warto zauważyć, że niektóre parametry mają domyślne wartości typu BOOLEAN oraz że rzadziej używane parametry znajdują się na końcowych miejscach listy argumentów. Dzięki temu w znakomitej większości przypadków składnia konkretnego wywołania procedury DBMS_RLS.ADD_POLICY staje się znacznie prostsza do zapisania i zrozumienia. Tabela 10.15 zawiera opis i sposób użycia wszystkich parametrów.

Tabela 10.15. *Parametry procedury DBMS_RLS.ADD_POLICY*

Parametr	Opis
object_schema	Schemat zawierający tabelę, widok lub synonim, który ma być chroniony przez politykę. Jeżeli parametr ma wartość NULL, używany jest schemat użytkownika wywołującego procedurę.
object_name	Nazwa tabeli, widoku lub synonimu, który ma być chroniony przez politykę.
policy_name	Nazwa polityki, którą należy dodać do obiektu. Nazwa musi być unikatowa dla każdego zabezpieczanego obiektu.
function_schema	Schemat, który jest właścicielem funkcji polityki. Jeżeli parametr ma wartość NULL, używany jest schemat użytkownika wywołującego procedurę.
policy_function	Nazwa funkcji, która będzie generować predykat dla polityki odnoszącej się do obiektu object_name. Jeżeli funkcja należy do pakietu, nazwa pakietu musi kwalifikować nazwę funkcji polityki.
statement_types	Rodzaje instrukcji, których dotyczy polityka. Wskazywać można dowolne kombinacje następujących wartości oddzielanych od siebie znakiem przecinka: SELECT, INSERT, UPDATE, DELETE i INDEX. Domyślnie uwzględniane są wszystkie wymienione wartości oprócz INDEX.
update_check	Dla typów INSERT i UPDATE parametr ten ma charakter opcjonalny, a jego wartością domyślną jest FALSE. Jeżeli parametr ma wartość TRUE, wówczas oprócz sprawdzania polityki w trakcie wykonywania instrukcji SELECT i DELETE polityka jest brana pod uwagę także w trakcie operacji INSERT i UPDATE.
enable	Domyślną wartością parametru jest TRUE. Parametr wskazuje, czy w momencie dodawania polityki jest ona włączona.
static_policy	Jeśli parametr ma wartość TRUE, polityka będzie generować taki sam ciąg znaków z predykatem dla każdego użytkownika uzyskującego dostęp do obiektu oprócz użytkownika SYS oraz użytkowników z uprawnieniem EXEMPT ACCESS POLICY. Domyślną wartością parametru jest FALSE.
policy_type	Jeżeli wartość parametru jest różna od NULL, nadpisuje ona wartość parametru static_policy. Dozwolone wartości dla parametru to STATIC, SHARED_STATIC, CONTEXT_SENSITIVE, SHARED_CONTEXT_SENSITIVE i DYNAMIC.
long_predicate	Domyślną wartością parametru jest FALSE. Jeśli parametr ma wartość TRUE, wówczas ciąg znaków predykatu może mieć nawet 32 KB długości. W przeciwnym razie maksymalna długość ciągu znaków predykatu to 4000 bajtów.
sec_relevant_cols	Parametr narzuca obowiązywanie VPD na poziomie kolumn i jest nowością w wersji Oracle 10*g*. Parametr dotyczy wyłącznie tabel i widoków. Zabezpieczane kolumny są wskazywane na liście, której elementy są oddzielane od siebie znakami przecinka lub spacji. Polityka jest stosowana jedynie wówczas, gdy wskazane kolumny z poufnymi danymi znajdują się w zapytaniu lub instrukcji DML. Domyślnie zabezpieczane są wszystkie kolumny.
sec_relevant_cols_opt	Pozwala na wyświetlanie w filtrowanych wynikach zapytania kolumn, względem których zastosowano VPD. W przypadku kolumn zawierających dane poufne wyświetlane są wartości NULL. Domyślną wartością parametru jest NULL. Jeżeli parametr ma wartość inną niż NULL, należy zdefiniować również DBMS_RLS.ALL_ROWS, aby wszystkie kolumny z danymi poufnymi były wyświetlane z wartościami NULL.

Parametr sec_relevant_cols przydaje się wówczas, gdy nie ma znaczenia, jaką część wiersza widzą użytkownicy, a jedynym wymogiem jest zablokowanie dostępu do kolumn zawierających dane poufne, takie jak numer ubezpieczenia społecznego albo wysokość wynagrodzenia. W przykładzie, który zostanie zaprezentowany w dalszej części rozdziału, utworzymy politykę bezpieczeństwa, która będzie odfiltrowywać poufne dane pracowników firmy.

W poniższym przykładzie polityka o nazwie EMP_SELECT_RESTRICT jest przypisywana do tabeli HR.EMPLOYEES. Schemat VPD jest właścicielem funkcji polityki get_predicates.emp_select_restrict. Polityka jawnie odnosi się do instrukcji SELECT wykonywanych na tabeli, natomiast w przypadku, gdy parametr UPDATE_CHECK ma wartość TRUE, wówczas polecenia update i delete również będą przez tę politykę obsługiwane w momencie modyfikowania i wstawiania wierszy do tabeli.

```
dbms_rls.add_policy (
    object_schema =>   'HR',
    object_name =>     'EMPLOYEES',
    policy_name =>     'EMP_SELECT_RESTRICT',
    function_schema => 'VPD',
    policy_function => 'get_predicates.emp_select_restrict',
    statement_types => 'SELECT',
    update_check =>    TRUE,
    enable =>          TRUE
);
```

Ponieważ nie zdefiniowano wartości parametru static_policy, przyjmuje on domyślną wartość FALSE. Oznacza to, że polityka ma charakter dynamiczny i jest sprawdzana za każdym razem, gdy parsowana jest instrukcja select. Jest to jedyny sposób postępowania obsługiwany w wersjach poprzedzających wersję Oracle 10*g*.

Za pomocą podprogramu ENABLE_POLICY łatwo można wyłączyć politykę na określony czas bez konieczności ponownego przypisywania jej do tabeli w późniejszym czasie:

```
dbms_rls.enable_policy(
    object_schema => 'HR',
    object_name =>   'EMPLOYEES',
    policy_name =>   'EMP_SELECT_RESTRICT',
    enable =>        FALSE
);
```

Jeśli dla danego obiektu wskazana zostanie więcej niż jedna polityka, wówczas wszystkie generowane predykaty zostaną połączone warunkiem AND. Jeżeli rozwiązanie takie jest niepożądane i predykaty z różnych polityk powinny zostać połączone operatorem alternatywy, politykę najprawdopodobniej trzeba będzie zredefiniować. W takiej sytuacji logika każdej z polis przypisanych do obiektu będzie musiała trafić do pojedynczej polityki, w której wszystkie fragmenty predykatu zostaną połączone ze sobą warunkiem OR.

Tworzenie VPD

W niniejszym punkcie opisany zostanie proces implementacji VPD. W przykładzie wykorzystane zostaną przykładowe schematy instalowane wraz z bazą danych Oracle 10*g*. Mówiąc bardziej konkretnie na tabeli HR.EMPLOYEES zaimplementujemy politykę FGAC, która będzie ograniczać dostęp do danych na podstawie statusu menedżera oraz numeru działu

(departamentu) pracownika. Pracownicy firmy będą mieli dostęp do dotyczącego ich wiersza tabeli HR.EMPLOYEES, natomiast menedżerowie będą mogli przeglądać dane na temat wszystkich podległych im pracowników.

> Jeżeli wraz z bazą danych nie zainstalowano przykładowych schematów, można je utworzyć przy użyciu skryptów znajdujących się w katalogu *$ORACLE_HOME/demo/schema*.

Gdy przykładowe schematy zostaną już zainstalowane, trzeba będzie utworzyć użytkowników bazy danych, którzy będą przeglądać wiersze z tabeli HR.EMPLOYEES.

```
create user smavris identified by smavris702;
grant connect, resource to smavris;

create user dgrant identified by dgrant507;
grant connect, resource to dgrant;

create user kmourgos identified by kmourgos622;
grant connect, resource to kmourgos;
```

Użytkownik KMOURGOS jest menedżerem wszystkich sprzedawców, a DGRANT jest jednym z podwładnych KMOURGOS. Z kolei SMAVRIS jest dyrektorem działu kadr firmy.

W kolejnych trzech krokach nadamy wszystkim użytkownikom bazy danych uprawnienie SELECT do tabeli HR.EMPLOYEES, a następnie utworzymy tabelę wyszukiwania odwzorowującą numery identyfikacyjne pracowników na ich konta w bazie danych. Na podstawie zawartości tabeli wyszukiwania procedura, która będzie definiować zmienne kontekstowe dla sesji użytkownika, przypisze numer identyfikacyjny użytkownika zmiennej kontekstowej. Tak zdefiniowana zmienna kontekstowa będzie następnie używana przez funkcję polityki do generowania predykatu.

```
grant select on hr.employees to public;

create table hr.emp_login_map (employee_id, login_acct)
    as select employee_id, email from hr.employees;

grant select on hr.emp_login_map to public;
```

Następnie trzeba utworzyć konto użytkownika o nazwie VPD z uprawnieniami do tworzenia kontekstów oraz utrzymywania funkcji polityk:

```
create user vpd identified by vpd439;
grant connect, resource, create any context, create public synonym to vpd;
```

Po połączeniu się ze schematem VPD utworzymy kontekst o nazwie HR_SECURITY oraz zdefiniujemy pakiet i procedurę, która będzie definiować kontekst dla aplikacji:

```
connect vpd/vpd439@dw;

create context hr_security using vpd.emp_access;

create or replace package vpd.emp_access as
    procedure set_security_parameters;
end;
```

Należy pamiętać, że procedury wchodzące w skład pakietu VPD.EMP_ACCESS to jedyne procedury, które mogą definiować zmienne kontekstowe. Kod źródłowy pakietu VPD.EMP_ACCESS przedstawia się następująco:

```
create or replace package body vpd.emp_access is

--
-- Po zalogowaniu się użytkownika należy wykonać set_security_parameters,
-- aby odczytać nazwę użytkownika, odpowiadającą wartości w kolumnie EMAIL
-- tabeli HR.EMPLOYEES.
--
-- kontekst USERENV zawiera charakterystyki użytkownika, takie jak
-- nazwę użytkownika, adres IP jego połączenia i tak dalej.
--
-- w procedurze używamy jedynie parametru SESSION_USER
-- z kontekstu USERENV.
--
    procedure set_security_parameters is
        emp_id_num number;
        emp_login varchar2(50);
    begin

        -- nazwa użytkownika bazy danych odpowiada adresowi poczty
        -- elektronicznej w tabeli HR.EMPLOYEES
        emp_login := sys_context('USERENV','SESSION_USER');

        dbms_session.set_context('HR_SECURITY','USERNAME',emp_login);

        -- odczytanie numeru identyfikacyjnego pracownika, aby ustanowić uprawnienia menedżera
        -- trzeba jednak ominąć innych użytkowników bazy danych,
        -- którzy nie występują w tabeli EMPLOYEES
        begin
            select employee_id into emp_id_num
                from hr.emp_login_map where login_acct = emp_login;

            dbms_session.set_context('HR_SECURITY','EMP_ID',emp_id_num);
        exception
            when no_data_found then
                dbms_session.set_context('HR_SECURITY','EMP_ID',0);
        end;

        -- w następnych zapytaniach zbiór zwracanych wierszy będzie ograniczony
        -- na podstawie wartości emp_id

    end; -- koniec procedury

end; -- koniec treści pakietu
```

W przedstawionej procedurze warto zwrócić uwagę na kilka elementów. Schemat użytkownika jest odczytywany z kontekstu USERENV, domyślnie dostępnego dla wszystkich użytkowników. Schemat ten zostanie przypisany zmiennej USERNAME, wchodzącej w skład nowo utworzonego kontekstu HR_SECURITY. Wartość innej zmiennej kontekstowej EMP_ID jest wyszukiwana w tabeli wyszukiwania HR.EMP_LOGIN_MAP. Jeśli danych zalogowanego użytkownika nie ma w tabeli wyszukiwania, procedura nie powinna zwracać błędu. W takim przypadku zmiennej EMP_ID należy przypisać wartość 0, co spowoduje, że predykat wygenerowany przez funkcję polityki nie pozwoli na dostęp do tabeli HR.EMPLOYEES.

W kolejnym etapie trzeba wszystkim użytkownikom bazy danych przypisać uprawnienie EXECUTE do pakietu oraz utworzyć synonim dla tego pakietu, aby przy każdym jego wywołaniu oszczędzić czasu na wpisywanie pełnej nazwy:

```
grant execute on vpd.emp_access to PUBLIC;
create public synonym emp_access for vpd.emp_access;
```

Aby zapewnić, że kontekst zostanie zdefiniowany dla każdego logującego się użytkownika, należy połączyć się z bazą jako użytkownik SYS i utworzyć procedurę wyzwalaną w momencie logowania, której zadaniem będzie definiowanie zmiennych w ramach kontekstu:

```
connect sys/sys727@dw as sysdba;

create or replace trigger vpd.set_security_parameters
    after logon on database
begin
    vpd.emp_access.set_security_parameters;
end;
```

Tak zdefiniowana procedura wyzwalana będzie wykonywana dla każdego użytkownika łączącego się z bazą danych, dlatego niezwykle istotne jest, by przetestować jej działanie dla każdej klasy użytkowników, a może nawet dla każdego użytkownika w bazie danych! Jeśli procedura wyzwalana będzie kończyć się błędem, wówczas żaden użytkownik inny niż SYSDBA nie będzie mógł się zalogować.

Jak dotąd, zdefiniowaliśmy kontekst, procedurę służącą do definiowania zmiennych kontekstowych oraz procedurę wyzwalaną, która będzie automatycznie wywoływać procedurę polityki. Jeden z trzech wcześniej zdefiniowanych użytkowników po zalogowaniu się może wykonać zapytanie dotyczące danych kontekstowych:

```
SQL> connect smavris/smavris702@dw
Connected.

SQL> select * from session_context;

NAMESPACE                ATTRIBUTE                VALUE
------------------------ ------------------------ ----------------------
HR_SECURITY              USERNAME                 SMAVRIS
HR_SECURITY              EMP_ID                   203

2 rows selected.
```

Spójrzmy, co się stanie, gdy użytkownik SMAVRIS spróbuje przyjąć tożsamość innego pracownika:

```
SQL> begin
  2     dbms_session.set_context('HR_SECURITY','EMP_ID',100);
  3  end;

begin
*
ERROR at line 1:
ORA-01031: insufficient privileges
ORA-06512: at "SYS.DBMS_SESSION", line 82
ORA-06512: at line 2
```

Jedynie pakiet VPD.EMP_ACCESS posiada uprawnienie do definiowania lub zmiany wartości zmiennych kontekstowych.

W końcowym etapie należy zdefiniować procedury, które będą generować predykat oraz jedną lub więcej spośród tych procedur przypisać tabeli HR.EMPLOYEES. Po zalogowaniu się jako użytkownik VPD, który jest już właścicielem procedur kontekstowych, zdefiniujemy pakiet odpowiedzialny za generowanie predykatów:

```
connect vpd/vpd439@dw;

create or replace package vpd.get_predicates as

    -- uwaga: funkcje zabezpieczające ZAWSZE posiadają dwa parametry:
    -- nazwę właściciela tabeli oraz nazwę tabeli

    function emp_select_restrict
        (owner varchar2, object_name varchar2) return varchar2;

    -- tutaj można dopisać kolejne funkcje dla operacji INSERT, DELETE i tak dalej

end get_predicates;


create or replace package body vpd.get_predicates is

    function emp_select_restrict
        (owner varchar2, object_name varchar2) return varchar2 is

        ret_predicate varchar2(1000); -- część klauzuli WHERE

    begin
        -- pozwala pracownikowi na dostęp do danych dotyczących jego samego
        -- lub jego bezpośrednich podwładnych
        ret_predicate := 'EMPLOYEE_ID = ' ||
                         sys_context('HR_SECURITY','EMP_ID') ||
                         ' OR MANAGER_ID = ' ||
                         sys_context('HR_SECURITY','EMP_ID');
        return ret_predicate;
    end emp_select_restrict;

end; -- koniec pakietu
```

Gdy funkcja zostanie podłączona do tabeli DBMS_RLS, będzie ona generować ciąg znaków używany w treści klauzuli WHERE w momencie każdorazowego odczytu danych z tabeli. Ciąg znaków zawsze będzie mieć postać podobną do przedstawionej poniżej:

```
EMPLOYEE_ID = 124 OR MANAGER_ID = 124
```

Podobnie jak w przypadku pakietów definiujących środowisko kontekstowe, trzeba również umożliwić użytkownikom dostęp do pakietu przedstawionego przed chwilą:

```
grant execute on vpd.get_predicates to PUBLIC;
create public synonym get_predicates for vpd.get_predicates;
```

Na koniec — co wcale nie znaczy, że jest to czynność mniej ważna — trzeba podłączyć funkcję polityki do tabeli przy użyciu procedury DBMS_RLS.ADD_POLICY:

```
dbms_rls.add_policy (
         object_schema => 'HR',
         object_name => 'EMPLOYEES',
         policy_name => 'EMP_SELECT_RESTRICT',
         function_schema => 'VPD',
         policy_function => 'get_predicates.emp_select_restrict',
         statement_types => 'SELECT',
         update_check => TRUE,
         enable => TRUE
);
```

Pracownik może odczytywać zawartość tabeli HR.EMPLOYEES w taki sam sposób jak dotychczas, lecz będzie widział jedynie wiersze z własnymi danymi oraz wiersze z danymi pracowników, którzy są jego podwładnymi — o ile tacy w ogóle występują. Po zalogowaniu się jako użytkownik KMOURGOS można spróbować odczytać wszystkie wiersze tabeli HR.EMPLOYEES, jednak uzyskamy wyłącznie dane dotyczące samego użytkownika KMOURGOS oraz osób bezpośrednio mu podlegających:

```
SQL> connect kmourgos/kmourgos622@dw;
Connected.

SQL> select employee_id, first_name, last_name,
  2         email, job_id, salary, manager_id from hr.employees;

EMPLOYEE_ID FIRST_NAME LAST_NAME   EMAIL      JOB_ID     SALARY  MANAGER_ID
----------- ---------- ----------- ---------- ---------- ------- ----------
        124 Kevin      Mourgos     KMOURGOS   ST_MAN        5800        100
        141 Trenna     Rajs        TRAJS      ST_CLERK      3500        124
        142 Curtis     Davies      CDAVIES    ST_CLERK      3100        124
        143 Randall    Matos       RMATOS     ST_CLERK      2600        124
        144 Peter      Vargas      PVARGAS    ST_CLERK      2500        124
        196 Alana      Walsh       AWALSH     SH_CLERK      3100        124
        197 Kevin      Feeney      KFEENEY    SH_CLERK      3000        124
        198 Donald     OConnell    DOCONNEL   SH_CLERK      2600        124
        199 Douglas    Grant       DGRANT     SH_CLERK      2600        124

9 rows selected.
```

W przypadku użytkownika DGRANT rzecz się przedstawia nieco inaczej:

```
SQL> connect dgrant/dgrant507@dw;
Connected.

SQL> select employee_id, first_name, last_name,
  2         email, job_id, salary, manager_id from hr.employees;

EMPLOYEE_ID FIRST_NAME LAST_NAME   EMAIL      JOB_ID     SALARY  MANAGER_ID
----------- ---------- ----------- ---------- ---------- ------- ----------
        199 Douglas    Grant       DGRANT     SH_CLERK      2600        124

1 row selected.
```

Użytkownik DGRANT widzi jedynie wiersz z własnymi danymi, ponieważ nie jest w firmie niczyim przełożonym.

W przypadku użytkownika SMAVRIS zapytanie zwróci podobne wyniki:

```
SQL> connect smavris/smavris702@dw;
Connected.

SQL> select employee_id, first_name, last_name,
  2         email, job_id, salary, manager_id from hr.employees;

EMPLOYEE_ID FIRST_NAME LAST_NAME    EMAIL      JOB_ID     SALARY  MANAGER_ID
----------- ---------- ------------ ---------- ---------- ------- ----------
        203 Susan      Mavris       SMAVRIS    HR_REP        6500        101

1 row selected.
```

Ale przecież SMAVRIS jest pracownicą działu (departamentu) HR i powinna widzieć wszystkie wiersze tabeli. Dodatkowo SMAVRIS powinna być jedyną osobą mającą dostęp do danych na temat wynagrodzeń pracowników. Oznacza to, że trzeba zmienić funkcję polityki w taki sposób, aby nadać SMAVRIS i innym pracownikom działu HR pełny dostęp do tabeli HR.EMPLOYEES. W polityce można ponadto wykorzystać ograniczenia na poziomie kolumn, aby zwracać tę samą liczbę wierszy, natomiast dane poufne zastępować wartościami NULL.

Aby ułatwić dostęp do tabeli HR.EMPLOYEES przez pracowników działu HR, trzeba najpierw odpowiednio zmienić tabelę wyszukiwania, tak aby zawierała również kolumnę JOB_ID. Jeżeli w kolumnie JOB_ID znajdować się będzie wartość HR_REP, będzie to oznaczać, że dana osoba jest pracownikiem działu HR. Najpierw trzeba zatem wyłączyć aktualnie obowiązującą politykę i utworzyć nową tabelę wyszukiwania z odpowiednimi odwzorowaniami:

```
SQL> begin
  2    dbms_rls.enable_policy(
  3          object_schema => 'HR',
  4          object_name => 'EMPLOYEES',
  5          policy_name => 'EMP_SELECT_RESTRICT',
  6          enable => FALSE
  7        );
  8  end;
PL/SQL procedure successfully completed.

SQL> drop table hr.emp_login_map;
Table dropped.

SQL> create table hr.emp_login_map (employee_id, login_acct, job_id)
  2    as select employee_id, email, job_id from hr.employees;
Table created.

SQL> grant select on hr.emp_login_map to public;
Grant succeeded.
```

Procedura VPD.EMP_ACCESS, która służy do definiowania zmiennych kontekstowych, wymaga obecności jeszcze jednej zmiennej kontekstowej wskazującej poziom bezpieczeństwa dla użytkownika odczytującego dane z tabeli. Instrukcję SELECT należy zmienić i dodać wywołanie DBMS_SESSION.SET_CONTEXT w następujący sposób:

```
. . .
     emp_job_id varchar2(50);
. . .
     select employee_id, job_id into emp_id_num, emp_job_id
```

```
        from hr.emp_login_map where login_acct = emp_login;

     dbms_session.set_context('HR_SECURITY','SEC_LEVEL',
        case emp_job_id when 'HR_REP' then 'HIGH' else 'NORMAL' end );
. . .
```

W przypadku każdego pracownika, którego stanowiskiem będzie HR_REP, zmiennej kontekstowej SEC_LEVEL przypisywana będzie wartość HIGH zamiast NORMAL. W funkcji polityki trzeba sprawdzać nowy warunek w następujący sposób:

```
create or replace package body vpd.get_predicates is

    function emp_select_restrict
        (owner varchar2, object_name varchar2) return varchar2 is

        ret_predicate varchar2(1000); -- część klauzuli WHERE

    begin
        -- pozwala pracownikowi na dostęp do danych dotyczących jego samego,
        -- chyba że posiada wysoki poziom zabezpieczeń
        if sys_context('HR_SECURITY','SEC_LEVEL') = 'HIGH' then
            ret_predicate := ''; -- brak ograniczeń w klauzuli WHERE
        else
            ret_predicate := 'EMPLOYEE_ID = ' ||
                             sys_context('HR_SECURITY','EMP_ID') ||
                             ' OR MANAGER_ID = ' ||
                             sys_context('HR_SECURITY','EMP_ID');
        end if;
        return ret_predicate;
    end emp_select_restrict;

end; -- koniec pakietu
```

Ponieważ polityka ma charakter dynamiczny, predykat jest generowany za każdym razem, gdy wykonywana jest instrukcja SELECT. Nie trzeba zatem odświeżać polityki. Gdy użytkownik SMAVRIS będący pracownikiem działu HR teraz wykona zapytanie, uzyska w odpowiedzi wszystkie wiersze z tabeli HR.EMPLOYEES:

```
SQL> connect smavris/smavris702@dw;
Connected.

SQL> select employee_id, first_name, last_name,
  2>        email, job_id, salary, manager_id from hr.employees;

EMPLOYEE_ID FIRST_NAME  LAST_NAME   EMAIL      JOB_ID     SALARY  MANAGER_ID
----------- ----------- ----------- ---------- ---------- ------- ----------
        100 Steven      King        SKING      AD_PRES      24000
        101 Neena       Kochhar     NKOCHHAR   AD_VP        17000        100
. . .
        204 Hermann     Baer        HBAER      PR_REP       10000        101
        205 Shelley     Higgins     SHIGGINS   AC_MGR       12000        101
        206 William     Gietz       WGIETZ     AC_ACCOUNT    8300        205

107 rows selected.
```

Natomiast użytkownik DGRANT nadal może przeglądać wyłącznie własne dane:

```
SQL> connect dgrant/dgrant507@dw;
Connected.

SQL> select employee_id, first_name, last_name,
  2         email, job_id, salary, manager_id from hr.employees;

EMPLOYEE_ID FIRST_NAME LAST_NAME   EMAIL      JOB_ID     SALARY  MANAGER_ID
----------- ---------- ----------- ---------- ---------- ------- ----------
        199 Douglas    Grant       DGRANT     SH_CLERK      2600        124

1 row selected.
```

Aby narzucić wymóg, by jedynie pracownicy działu HR mogli przeglądać dane na temat wynagrodzeń, należy wprowadzić niewielkie zmiany do funkcji polityki i włączyć politykę z ograniczeniami na poziomie kolumn:

```
dbms_rls.add_policy (
         object_schema => 'HR',
         object_name => 'EMPLOYEES',
         policy_name => 'EMP_SELECT_RESTRICT',
         function_schema => 'VPD',
         policy_function => 'get_predicates.emp_select_restrict',
         statement_types => 'SELECT',
         update_check => TRUE,
         enable => TRUE,
         sec_relevant_cols => 'SALARY',
         sec_relevant_cols_opt => dbms_rls.all_rows
);
```

Ostatni parametr o nazwie SEC_RELEVANT_COLS_OPT definiuje stałą DBMS_RLS.ALL_ROWS funkcjonującą w obrębie pakietu, która wskazuje, że w wynikach zapytania nadal powinny być wyświetlane wszystkie wiersze, lecz określone kolumny (w naszym przykładzie kolumna SALARY) powinny zawierać wartości NULL. W przeciwnym razie zapytania odnoszące się do kolumny SALARY nie zwracałyby żadnych wierszy.

Debugowanie polityki VPD

Nawet jeśli nie pojawia się błąd „ORA-28113: Policy predicate has error” ani „ORA-00936: missing expression”, warto jest przeanalizować predykat w takiej postaci, w jakiej jest on generowany w trakcie parsowania instrukcji. Predykaty można debugować na kilka sposobów, z których każdy ma zalety i wady.

Pierwsza metoda polega na użyciu dynamicznych widoków wydajności V$SQLAREA oraz V$VPD_POLICY. Jak wskazują nazwy widoków, V$SQL_AREA zawiera instrukcje języka SQL aktualnie znajdujące się w obszarze dzielonym, a także bieżące statystyki ich wykonania. Z kolei widok V$VPD_POLICY zawiera listę wszystkich polityk aktualnie obowiązujących w bazie danych wraz z predykatami. Po połączeniu obydwóch tabel w sposób pokazany poniżej, można uzyskać informacje przydatne do debugowania przyczyn ewentualnych problemów z wynikami zapytań:

```
SQL> select s.sql_text, v.object_name, v.policy, v.predicate
  2     from v$sqlarea s, v$vpd_policy v
  3     where s.hash_value = v.sql_hash;

SQL_TEXT             OBJECT_NAME  POLICY               PREDICATE
-------------------- ------------ -------------------- -----------------
select employee_id,  EMPLOYEES    EMP_SELECT_RESTRICT
first_name, last_nam
e, email, job_i
d, salary, manager_i
d from hr.employees

select employee_id,  EMPLOYEES    EMP_SELECT_RESTRICT  EMPLOYEE_ID = 124
first_name, last_nam                                    OR MANAGER_ID =
e, email, job_i                                        124
d, salary, manager_i
d from hr.employees

select employee_id,  EMPLOYEES    EMP_SELECT_RESTRICT  EMPLOYEE_ID = 199
first_name, last_nam                                    OR MANAGER_ID =
e, email, job_i                                        199
d, salary, manager_i
d from hr.employees

3 rows selected.
```

Jeżeli w zapytaniu uwzględnimy dodatkowo połączenie z V$SESSION, będzie można rozpoznać użytkownika, który wykonywał zapytanie SQL. Opisana metoda ma jednak pewną wadę: jeżeli baza danych będzie szczególnie obciążona, polecenia SQL mogą zostać usunięte z obszaru dzielonego i zastąpione przez inne polecenia SQL, zanim przedstawione powyżej zapytanie zostanie w ogóle wykonane.

W innej metodzie używa się polecenia alter session, aby wygenerować jednorodny tekstowy plik śledzenia zawierający większość informacji zwracanych przez poprzednie zapytanie. Polecenia definiujące proces śledzenia mają następującą postać:

```
SQL> begin
  2     dbms_rls.refresh_policy;
  3  end;
PL/SQL procedure successfully completed.

SQL> alter session set events '10730 trace name context forever, level 12';
Session altered.
```

Zdarzenie 10730 jest definiowane dla celów śledzenia predykatów polisy RLS. Innymi zdarzeniami, które można śledzić, są między innymi zdarzenia 10029 oraz 10030 dotyczące logowania i wylogowywania się sesji, 10710 związane ze śledzeniem dostępu do indeksów bitmapowych oraz 10253 związane z symulowaniem błędów zapisu do plików dziennika powtórzeń. Gdy sesja zostanie zmieniona, użytkownik DGRANT będzie mógł ponownie wykonać zapytanie:

```
SQL> select employee_id, first_name, last_name,
  2         email, job_id, salary, manager_id from hr.employees;

EMPLOYEE_ID FIRST_NAME  LAST_NAME   EMAIL      JOB_ID     SALARY  MANAGER_ID
----------- ----------- ----------- ---------- ---------- ------- ----------
        199 Douglas     Grant       DGRANT     SH_CLERK      2600        124

1 row selected.
```

Poniżej przedstawiono fragment końcowej części pliku śledzenia znajdującego się w katalogu wskazywanym przez parametr inicjalizacyjny USER_DUMP_DEST:

```
/u01/app/oracle/admin/dw/udump/dw_ora_32530.trc
. . .
*** MODULE NAME:(SQL*Plus) 2004-02-19 20:50:38.132
*** SERVICE NAME:(SYS$USERS) 2004-02-19 20:50:38.132
*** SESSION ID:(265.44) 2004-02-19 20:50:38.132
------------------------------------------------------------
Logon user : DGRANT
Table/View : HR.EMPLOYEES
Policy name : EMP_SELECT_RESTRICT
Policy function: VPD.GET_PREDICATES.EMP_SELECT_RESTRICT
RLS view :
SELECT "EMPLOYEE_ID","FIRST_NAME","LAST_NAME","EMAIL",
"PHONE_NUMBER","HIRE_DATE","JOB_ID","SALARY","COMMISSION_PCT",
"MANAGER_ID","DEPARTMENT_ID"
FROM "HR"."EMPLOYEES" "EMPLOYEES"
WHERE (EMPLOYEE_ID = 199 OR MANAGER_ID = 199)
```

W pliku śledzenia przytoczone są instrukcja SQL oraz dodany do niej predykat. Wadą przedstawionej metody jest fakt, że o ile użytkownik może posiadać dostęp do dynamicznych widoków wydajności, to programista zwykle nie posiada dostępu do katalogu zrzutu pamięci użytkownika na serwerze. W efekcie do procesu debugowania problemów z predykatami konieczne może się okazać zaangażowanie administratora bazy danych.

Po zakończeniu debugowania należy pamiętać o wyłączeniu śledzenia, aby zmniejszyć obciążenie bazy i użycie przestrzeni na dysku generowane w trakcie operacji śledzenia:

```
SQL> alter session set events '10730 trace name context off';
Session altered.
```

Monitorowanie

Oracle wykorzystuje szereg różnych metod monitorowania służących do sprawdzania, jakiego rodzaju uprawnienia są używane oraz do jakich obiektów jest udzielany dostęp. Monitorowanie nie zapobiega używaniu uprawnień, natomiast może stanowić źródło bardzo użytecznych informacji, dzięki którym będzie można odkryć ewentualne nadużycia lub przypadki nieprawidłowego korzystania z posiadanych uprawnień.

W tabeli 10.16 przestawiono rodzaje monitorowania dostępne w bazie danych Oracle.

Tabela 10.16. *Dostępne rodzaje monitorowania*

Rodzaj monitorowania	Opis
Monitorowanie instrukcji	Monitorowaniu podlegają instrukcje języka SQL w podziale na rodzaje instrukcji bez względu na to, jakie obiekty są wykorzystywane. W bazie danych można dodatkowo wskazać konkretnych użytkowników, w kontekście których należy monitorować określoną instrukcję.
Monitorowanie uprawnień	Monitorowaniu podlegają uprawnienia systemowe, takie jak `CREATE TABLE` albo `ALTER INDEX`. Podobnie jak w przypadku monitorowania instrukcji, również w monitorowaniu uprawnień można wskazać jednego lub więcej użytkowników będących przedmiotem procesu monitorowania.
Monitorowanie obiektów schematu	Monitorowaniu podlegają określone instrukcje działające na określonych obiektach schematu (na przykład instrukcje `UPDATE` wykonywane na tabeli `DEPARTMENTS`). Monitorowanie obiektów schematu zawsze obejmuje wszystkich użytkowników bazy danych.
Monitorowanie precyzyjne	Monitorowaniu podlega dostęp do tabel oraz uprawnienia uzależniony od zawartości udostępnianych obiektów. Dla celów monitorowania używany jest pakiet `DBMS_FGA`, służący do skonfigurowania polityki dla określonej tabeli.

W kolejnych kilku punktach opiszemy, w jaki sposób administrator bazy danych może zarządzać procesem monitorowania sposobu użycia uprawnień systemowych i do obiektów. Gdy wymagana jest większa precyzja, administrator może skorzystać z monitorowania precyzyjnego, aby w ten sposób monitorować dostęp do określonych wierszy i kolumn tabeli, a nie tylko przypadki udostępniania samej tabeli.

Lokalizacja danych monitorowania

Wyniki monitorowania można przesyłać do tabeli `SYS.AUD$` bazy danych albo do pliku systemu operacyjnego. Aby włączyć monitorowanie i wskazać lokalizację, w której należy zapisywać dane wynikowe procesu monitorowania, parametrowi inicjalizacyjnemu `AUDIT_TRAIL` należy przypisać jedną z czterech wartości opisanych w tabeli 10.17.

Tabela 10.17. *Dostępne wartości parametru inicjalizacyjnego AUDIT_TRAIL*

Wartość parametru	Działanie
`NONE`, `FALSE`	Wyłącza monitorowanie.
`OS`	Włącza monitorowanie. Dane wynikowe monitorowania są przesyłane do pliku systemu operacyjnego.
`DB`, `TRUE`	Włącza monitorowanie. Dane wynikowe monitorowania są przesyłane do tabeli `SYS.AUD$`.
`DB_EXTENDED`	Włącza monitorowanie. Dane wynikowe monitorowania są przesyłane do tabeli `SYS.AUD$`, a informacje dodatkowe są zapisywane w kolumnach `SQLBIND` i `SQLTEXT` typu `CLOB`.

Parametr AUDIT_TRAIL nie jest parametrem dynamicznym, dlatego aby uwzględnić zmianę wartości parametru, konieczne jest wyłączenie i ponowne włączenie bazy danych. Jeśli dane wynikowe monitorowania są zapisywane w tabeli SYS.AUD$, konieczne jest szczegółowe monitorowanie rozmiaru tej tabeli, aby jej rozmiar nie kolidował z ilością przestrzeni dyskowej wymaganą dla innych obiektów przestrzeni tabel SYS. Zaleca się, by okresowo archiwizować wiersze tabeli SYS.AUD$ oraz aby tabela była czyszczona. Oracle udostępnia rolę DELETE_CATALOG_ROLE, przeznaczoną do użycia przez specjalne konto w ramach zadania wsadowego wykonującego archiwizację i czyszczenie tabeli z danymi wynikowymi monitorowania.

Monitorowanie instrukcji

Bez względu na rodzaj monitorowania każdy proces monitorowania włącza się poleceniem audit i wyłącza poleceniem nonaudit. W przypadku monitorowania instrukcji format polecenia audit przedstawia się następująco:

```
AUDIT klauzula_instrukcji_sql BY {SESSION | ACCESS}
    WHENEVER [NOT] SUCCESSFUL;
```

Parametr *klauzula_instrukcji_sql* zawiera szereg informacji, na przykład o rodzaju instrukcji języka SQL, które mają być monitorowane, oraz dane użytkownika, który ma być monitorowany.

Dodatkowo daną czynność można rejestrować w trakcie monitorowania zawsze, gdy czynność ta występuje (by access), lub tylko jednorazowo (by session). Domyślną wartością jest by session.

Czasami trzeba monitorować instrukcje, których wykonanie zakończyło się powodzeniem, to znaczy te instrukcje, które nie zwróciły komunikatu o błędzie. W takim przypadku należy użyć klauzuli whenever successful. W pozostałych przypadkach interesują nas zazwyczaj przypadki, gdy polecenia korzystające z monitorowanych instrukcji nie powiodły się — czy to przez próbę naruszenia uprawnień, wyczerpanie się miejsca dla przestrzeni tabel, czy też z powodu błędów składniowych. Do obsługi takich przypadków używa się klauzuli whenever not successful.

Jeżeli monitorowaniem mają zostać objęte wszystkie próby dostępu do tabel albo wszystkie uprawnienia konkretnego użytkownika, wówczas dla większości kategorii metod monitorowania można użyć słowa kluczowego all, zamiast wskazywać konkretne rodzaje instrukcji albo obiekty.

W tabeli 10.18 przedstawiono rodzaje instrukcji, które można monitorować, wraz z krótkim opisem. Jeżeli użyte zostanie słowo kluczowe all, monitorowane będą wszystkie instrukcje wymienione w tabeli. Natomiast w tabeli 10.19 znajdują się instrukcje, które nie należą do kategorii wyznaczanej przez słowo kluczowe all, lecz trzeba je jawnie wskazać w każdym poleceniu audit.

Tabela 10.18. *Monitorowane instrukcje należące do kategorii ALL*

Opcja definiująca rodzaj instrukcji	Operacje języka SQL
CLUSTER	Instrukcje CREATE, ALTER, DROP lub TRUNCATE dotyczące klastra.
CONTEXT	Instrukcje CREATE lub DROP dotyczące kontekstu.
DATABASE LINK	Instrukcje CREATE lub DROP dotyczące łącza bazodanowego.
DIMENSION	Instrukcje CREATE, ALTER lub DROP dotyczące wymiaru.
DIRECTORY	Instrukcje CREATE lub DROP dotyczące obiektu DIRECTORY.
INDEX	Instrukcje CREATE, ALTER lub DROP dotyczące indeksu.
MATERIALIZED VIEW	Instrukcje CREATE, ALTER lub DROP dotyczące widoku zmaterializowanego.
NOT EXISTS	Instrukcja SQL zakończona niepowodzeniem z powodu nieistniejących obiektów, do których instrukcja się odwołuje.
PROCEDURE	Instrukcje CREATE lub DROP FUNCTION, LIBRARY, PACKAGE, PACKAGE BODY lub PROCEDURE.
PROFILE	Instrukcje CREATE, ALTER lub DROP dotyczące profilu.
PUBLIC DATABASE LINK	Instrukcje CREATE lub DROP dotyczące publicznego łącza bazodanowego.
PUBLIC SYNONYM	Instrukcje CREATE lub DROP dotyczące publicznego synonimu.
ROLE	Instrukcje CREATE, ALTER, DROP lub SET dotyczące roli.
ROLLBACK SEGMENT	Instrukcje CREATE, ALTER lub DROP dotyczące segmentu wycofania.
SEQUENCE	Instrukcje CREATE, ALTER lub DROP dotyczące sekwencji.
SESSION	Operacje logowania się i wylogowywania.
SYNONYM	Instrukcje CREATE lub DROP dotyczące synonimów.
SYSTEM AUDIT	Instrukcje AUDIT lub NONAUDIT dotyczące uprawnień systemowych.
SYSTEM GRANT	Instrukcje GRANT lub REVOKE dotyczące uprawnień systemowych i ról.
TABLE	Instrukcje CREATE, DROP lub TRUNCATE dotyczące tabeli.
TABLESPACE	Instrukcje CREATE, ALTER lub DROP dotyczące przestrzeni tabel.
TRIGGER	Instrukcje CREATE, ALTER (włączenie i wyłączenie) lub DROP dotyczące procedur wyzwalanych. Instrukcja ALTER TABLE z klauzulą ENABLE ALL TRIGGERS lub DISABLE ALL TRIGGERS.
TYPE	Instrukcje CREATE, ALTER lub DROP dotyczące typów i treści typów.
USER	Instrukcje CREATE, ALTER lub DROP dotyczące użytkownika.
VIEW	Instrukcje CREATE lub DROP dotyczące widoku.

Kilka przykładów powinno rozjaśnić znaczenie poszczególnych opcji. W naszej przykładowej bazie danych użytkownik SCOTT posiada wszystkie uprawnienia do tabel w schemacie HR oraz w innych schematach. SCOTT może tworzyć indeksy na niektórych tabelach, jednak chcemy wiedzieć, kiedy te indeksy są tworzone na wypadek wystąpienia ewentualnych problemów związanych ze zmianami w planach wykonania. Poniższe polecenie włącza monitorowanie operacji tworzenia indeksów wykonywanych przez użytkownika SCOTT:

```
SQL> audit index by scott whenever successful;
Audit succeeded.
```

Tabela 10.19. *Rodzaje instrukcji, które należy wskazywać jawnie*

Opcja definiująca rodzaj instrukcji	Operacje języka SQL
ALTER SEQUENCE	Każde polecenie ALTER SEQUENCE.
ALTER TABLE	Każde polecenie ALTER TABLE.
COMMENT TABLE	Dodawanie komentarzy do tabeli, widoku, widoku zmaterializowanego lub dowolnych kolumn tych obiektów.
DELETE TABLE	Usuwanie wierszy z tabeli lub widoku.
EXECUTE PROCEDURE	Wykonanie procedury, funkcji albo dowolnych zmiennych lub kursorów w pakiecie.
GRANT DIRECTORY	Instrukcje GRANT lub REVOKE dotyczące nadawania lub odbierania uprawnień do obiektu DIRECTORY.
GRANT PROCEDURE	Instrukcje GRANT lub REVOKE dotyczące uprawnień do procedury, funkcji lub pakietu.
GRANT SEQUENCE	Instrukcje GRANT lub REVOKE dotyczące uprawnień do sekwencji.
GRANT TABLE	Instrukcje GRANT lub REVOKE dotyczące uprawnień do tabeli, widoku lub widoku zmaterializowanego.
GRANT TYPE	Instrukcje GRANT lub REVOKE dotyczące uprawnień do obiektu TYPE.
INSERT TABLE	Instrukcja INSERT INTO dotycząca tabeli lub widoku.
LOCK TABLE	Polecenie LOCK TABLE na tabeli lub widoku.
SELECT SEQUENCE	Każde polecenie odwołujące się do wartości CURRVAL lub NEXTVAL sekwencji.
SELECT TABLE	Instrukcja SELECT FROM dotycząca tabeli, widoku lub widoku zmaterializowanego.
UPDATE TABLE	Wykonanie instrukcji UPDATE na tabeli lub widoku.

Jeszcze w tym samym dniu użytkownik SCOTT tworzy indeks na tabeli HR.JOBS:

```
SQL> create index job_title_idx on hr.jobs(job_title);
Index created.
```

Analiza śladu monitorowania w widoku danych słownikowych DBA_AUDIT_TRAIL wykaże, że SCOTT rzeczywiście utworzył indeks w dniu 24 lutego o godzinie 21.21:

```
SQL> select username, to_char(timestamp,'MM/DD/YY HH24:MI') Timestamp,
  2     obj_name, action_name, sql_text from dba_audit_trail
  3  where username = 'SCOTT';

USERNAME   TIMESTAMP      OBJ_NAME         ACTION_NAME      SQL_TEXT
---------- -------------- ---------------- ---------------- --------------
SCOTT      02/24/04 21:24 JOB_TITLE_IDX    CREATE INDEX      create index j
                                                            ob_title_idx on
                                                             hr.jobs(job_ti
                                                            tle)

1 row selected.
```

Aby wyłączyć monitorowanie instrukcji wykonywanych przez użytkownika SCOTT na tabeli HR.JOBS, należy wykonać polecenie noaudit w następujący sposób:

```
SQL> noaudit index by scott;
Noaudit succeeded.
```

Być może konieczne będzie również regularne monitorowanie operacji logowania się — zarówno tych zakończonych powodzeniem, jak i tych, które okazały się nieskuteczne. W tym celu trzeba wykonać dwa polecenia `audit`:

```
SQL> audit session whenever successful;
Audit succeeded.
SQL> audit session whenever not successful;
Audit succeeded.
```

Analiza śladu monitorowania wykaże, że zarejestrowana została jedna nieskuteczna próba zalogowania się dokonana przez użytkownika `RJB`:

```
SQL> select username, to_char(timestamp,'MM/DD/YY HH24:MI') Timestamp,
  2      obj_name, returncode, action_name, sql_text from dba_audit_trail
  3  where action_name in ('LOGON','LOGOFF')
  4  order by timestamp desc;

USERNAME   TIMESTAMP      OBJ_NAME   RETURNCODE ACTION_NAME       SQL_TEXT
---------- -------------- ---------- ---------- ---------------- ----------
DBSNMP     02/24/04 21:55                   0 LOGOFF
SYSMAN     02/24/04 21:54                   0 LOGOFF
RJB        02/24/04 21:52                1017 LOGON
RJB        02/24/04 21:52                   0 LOGON
DBSNMP     02/24/04 21:52                   0 LOGOFF
SCOTT      02/24/04 21:52                   0 LOGON
DBSNMP     02/24/04 21:51                   0 LOGOFF
RJB        02/24/04 21:51                   0 LOGOFF
SCOTT      02/24/04 21:38                   0 LOGOFF
SCOTT      02/24/04 21:24                   0 LOGOFF

10 rows selected.
```

W kolumnie `RETURNCODE` znajduje się komunikat błędu ORA. Komunikat ORA-1017 wskazuje, że podano nieprawidłowe hasło. Warto zauważyć, że gdybyśmy włączyli jedynie monitorowanie operacji logowania i wylogowywania się, wówczas do tego samego celu można by użyć widoku `DBA_AUDIT_SESSION`.

Monitorowanie instrukcji obejmuje również operacje włączania i wyłączania bazy danych. W tabeli `SYS.AUD$` można monitorować polecenie `shutdown immediate`, natomiast niemożliwe jest rejestrowanie w tej samej tabeli polecenia `startup`, ponieważ dodanie wierszy do tabeli `SYS.AUD$` jest możliwe dopiero po włączeniu bazy danych. W takich przypadkach można zajrzeć do katalogu *$ORACLE_HOME/rdbms/audit*, aby sprawdzić rejestr operacji włączania bazy danych wykonanych przez administratora systemu. Poniżej znajduje się zawartość pliku tekstowego utworzonego w momencie uruchomienia bazy danych poleceniem `startup`:

```
Audit file /u01/app/oracle/product/10.1.0/rdbms/audit/ora_2788.aud
Oracle Database 10g Enterprise Edition Release 10.1.0.2.0 - Production
With the Partitioning, OLAP and Data Mining options
ORACLE_HOME = /u01/app/oracle/product/10.1.0
System name: Linux
Node name: dw10g
Release: 2.4.21-9.0.1.EL
Version: #1 Mon Feb 9 22:26:52 EST 2004
Machine: i686
Instance name: dw
Redo thread mounted by this instance: 0 <none>
```

```
Oracle process number: 0
2788

Tue Feb 24 17:26:01 2004
ACTION : 'STARTUP'
DATABASE USER: '/'
PRIVILEGE : SYSDBA
CLIENT USER: oracle
CLIENT TERMINAL: Not Available
STATUS: 0
```

Powyższy przykład wskazuje, że baza danych została włączona przez użytkownika `oracle` systemu macierzystego, a połączenie z instancją zostało nawiązane na podstawie uwierzytelnienia w systemie operacyjnym. Inne zagadnienia dotyczące monitorowania czynności wykonywanych przez administratora systemu zostaną opisane w następnym punkcie.

Monitorowanie uprawnień

Polecenia służące do monitorowania uprawnień mają podobną składnię jak polecenia dotyczące monitorowania instrukcji. Jedyna różnica polega na tym, że w klauzuli *klauzula_instrukcji_sql* wskazuje się uprawnienia systemowe zamiast instrukcji.

Wyobraźmy sobie na przykład, że wszystkim administratorom bazy danych należy nadać uprawnienie `ALTER TABLESPACE`, lecz jednocześnie w przypadku skorzystania z tego uprawnienia proces monitorowania powinien zapisać odpowiednią informację. Polecenie, które włącza monitorowanie wspomnianego uprawnienia, ma konstrukcję podobną do polecenia włączającego monitorowanie instrukcji:

```
SQL> audit alter tablespace by access whenever successful;
Audit succeeded.
```

Od teraz za każdym razem, gdy użytkownik skorzysta z uprawnienia `ALTER TABLESPACE`, w tabeli `SYS.AUD$` pojawi się nowy wiersz z odpowiednią informacją.

Specjalny tryb monitorowania jest dostępny dla administratorów systemowych, którzy korzystają z uprawnień `SYSDBA` i `SYSOPER`. Aby włączyć specjalny poziom monitorowania, należy przypisać parametrowi inicjalizacyjnemu wartość `TRUE`. Dane monitorowania zostaną przesłane do tej samej lokalizacji, do której przesyłane są dane monitorowania w systemie operacyjnym; inaczej mówiąc, lokalizacja ta zależy od ustawień systemu operacyjnego. Wszelkie instrukcje języka SQL wykonane przy użyciu jednego ze wspomnianych uprawnień, jak również wszystkie instrukcje SQL wykonane przez użytkownika `SYS`, zostają zarejestrowane w odpowiedniej lokalizacji danych monitorowania w systemie operacyjnym.

Monitorowanie obiektów schematu

Monitorowanie dostępu do różnego rodzaju obiektów schematu odbywa się podobnie jak monitorowanie instrukcji i uprawnień:

```
AUDIT klauzula_obiektu_schematu BY {SESSION | ACCESS}
    WHENEVER [NOT] SUCCESSFUL;
```

Klauzula obiektu schematu wskazuje rodzaj obiektu, do którego dostęp ma być monitorowany, oraz sam obiekt. Monitorowaniu może podlegać trzynaście rodzajów operacji na konkretnych obiektach. Dostępne rodzaje operacji przedstawiono w tabeli 10.20.

Tabela 10.20. *Opcje monitorowania obiektów*

Opcja wyznaczająca rodzaj obiektu	Opis
ALTER	Zmiany tabeli, sekwencji lub widoku zmaterializowanego.
AUDIT	Polecenia związane z monitorowaniem dowolnego obiektu.
COMMENT	Dodawanie komentarzy do tabel, widoków lub widoków zmaterializowanych.
DELETE	Usuwanie wierszy z tabeli, widoku lub widoku zmaterializowanego.
FLASHBACK	Wykonanie operacji Flashback na tabeli lub widoku.
GRANT	Nadanie uprawnień do obiektu dowolnego rodzaju.
INDEX	Utworzenie indeksu na tabeli lub widoku zmaterializowanym.
INSERT	Wstawienie wierszy do tabeli, widoku lub widoku zmaterializowanego.
LOCK	Zablokowanie tabeli, widoku lub widoku zmaterializowanego.
READ	Wykonanie operacji odczytu zawartości obiektu DIRECTORY.
RENAME	Zmiana nazwy tabeli, widoku lub procedury.
SELECT	Odczyt wierszy z tabeli, widoku, sekwencji lub widoku zmaterializowanego.
UPDATE	Uaktualnienie tabeli, widoku lub widoku zmaterializowanego.

Aby włączyć monitorowanie wszystkich poleceń `insert` i `update` w tabeli HR.JOBS i rejestrować wszystkie przypadki wykonania tych poleceń bez względu na to, kto je wykonuje, można użyć polecenia `audit` w następujący sposób:

```
SQL> audit insert, update on hr.jobs by access whenever successful;
Audit successful.
```

Załóżmy, że użytkownik TAMARA wstawia do tabeli HR.JOBS dwa nowe wiersze:

```
SQL> insert into hr.jobs (job_id, job_title, min_salary, max_salary)
  2  values ('IN_CFO','Internet Chief Fun Officer', 7500, 50000);

1 row created.

SQL> insert into hr.jobs (job_id, job_title, min_salary, max_salary)
  2  values ('OE_VLD','Order Entry CC Validation', 5500, 20000);

1 row created.
```

W widoku DBA_AUDIT_TRAIL pojawią się wpisy dotyczące dwóch poleceń `insert` wykonanych w sesji użytkownika TAMARA:

```
USERNAME   TIMESTAMP      OWNER    OBJ_NAME   ACTION_NAME
SQL_TEXT
---------- -------------- -------- ---------- ---------------
```

```
-------------------------------------------------------------
TAMARA     02/24/04 22:54 HR       JOBS       INSERT
insert into hr.jobs (job_id, job_title, min_salary, max_salary)
 values ('IN_CFO','Internet Chief Fun Officer', 7500, 50000);
TAMARA     02/24/04 22:53 HR       JOBS       INSERT
insert into hr.jobs (job_id, job_title, min_salary, max_salary)
 values ('OE_VLD','Order Entry CC Validation', 5500, 20000);
TAMARA     02/24/04 22:51                     LOGON
3 rows selected.
```

Monitorowanie precyzyjne

Począwszy od wersji Oracle9*i*, proces monitorowania stał się znacznie bardziej szczegółowy dzięki udostępnieniu procesu precyzyjnego monitorowania obiektów (ang. *fine-grained object auditing — FGA*). FGA jest implementowane w pakiecie PL/SQL o nazwie `DBMS_FGA`.

Dzięki monitorowaniu standardowemu łatwo można sprawdzić, które obiekty były udostępniane i komu, nie wiadomo natomiast, które kolumny i wiersze zostały udostępnione. Problem ten rozwiązuje proces monitorowania precyzyjnego, w którym oprócz wskazywania predykatu (czyli klauzuli `where`) definiującego udostępniane wiersze wskazuje się również udostępnianą kolumnę lub kolumny tabeli. Dzięki użyciu monitorowania precyzyjnego można istotnie zmniejszyć liczbę pozycji danych monitorowania, ponieważ dane monitorowania będą wówczas zapisywane jedynie w przypadku udostępniania konkretnych wierszy lub kolumn.

Pakiet `DBMS_FGA` posiada cztery procedury:

`ADD_POLICY`	— dodaje politykę monitorowania za pomocą predykatu oraz kolumny podlegającej monitorowaniu.
`DROP_POLICY`	— usuwa politykę monitorowania.
`DISABLE_POLICY`	— wyłącza politykę monitorowania, lecz zachowuje powiązanie polityki z tabelą lub widokiem.
`ENABLE_POLICY`	— włącza politykę.

Użytkownik `TAMARA` zwykle odczytuje zawartość tabeli `HR.EMPLOYEES` każdego dnia, wyszukując adresy poczty elektronicznej pracowników. Administratorzy systemu podejrzewają jednak, że `TAMARA` przegląda również dane na temat wynagrodzeń menedżerów, dlatego chcą zdefiniować politykę FGA, która będzie rejestrować wszelkie przypadki udostępnienia kolumny `SALARY` każdego pracownika, który pracuje na stanowisku menedżera:

```
begin
    dbms_fga.add_policy(
        object_schema => 'HR',
        object_name => 'EMPLOYEES',
        policy_name => 'SAL_SELECT_AUDIT',
        audit_condition => 'instr(job_id,''_MAN'') > 0',
        audit_column => 'SALARY'
    );
end;
```

Dane monitorowania generowane przez proces monitorowania precyzyjnego można przeglądać w widoku danych słownikowych DBA_FGA_AUDIT_TRAIL. Jeżeli standardowo konieczne jest przeglądanie zarówno wierszy generowanych w procesie monitorowania standardowego, jak i wierszy będących wynikiem monitorowania precyzyjnego, można skorzystać z widoku DBA_COMMON_AUDIT_TRAIL, który łączy w sobie dane generowane przez obydwa procesy monitorowania.

W dalszym ciągu naszego przykładu TAMARA wykonuje dwie następujące instrukcje języka SQL:

```
SQL> select employee_id, first_name, last_name, email from hr.employees
  2      where employee_id = 114;

EMPLOYEE_ID FIRST_NAME           LAST_NAME                 EMAIL
----------- -------------------- ------------------------- --------------
114         Den                  Raphaely                  DRAPHEAL

1 row selected.

SQL> select employee_id, first_name, last_name, salary from hr.employees
  2      where employee_id = 114;

EMPLOYEE_ID FIRST_NAME           LAST_NAME                 SALARY
----------- -------------------- ------------------------- ----------
114         Den                  Raphaely                  11000

1 row selected.
```

W pierwszym zapytaniu udostępniane są dane na temat menedżera, lecz bez kolumny SALARY. Drugie zapytanie jest podobne do pierwszego, lecz dotyczy również kolumny SALARY, a więc jego wykonanie powoduje wykonanie polityki FGA, która tworzy jeden wiersz w śladzie monitorowania:

```
SQL> select to_char(timestamp,'mm/dd/yy hh24:mi') timestamp,
  2      object_schema, object_name, policy_name, statement_type
  3  from dba_fga_audit_trail
  4  where db_user = 'TAMARA';

TIMESTAMP      OBJECT_SCHEMA  OBJECT_NAME    POLICY_NAME      STATEMENT_TYPE
-------------- -------------- -------------- ---------------- --------------
02/25/04 00:04 HR             EMPLOYEES      SAL_SELECT_AUDIT SELECT

1 row selected.
```

Ponieważ w przykładzie dotyczącym VPD przedstawionym we wcześniejszej części niniejszego rozdziału ustawiono precyzyjny tryb kontroli dostępu, aby zapobiegać nieuprawnionemu odczytywaniu danych z kolumny SALARY, trzeba szczególnie dokładnie sprawdzić funkcje polityki i upewnić się, że dane z kolumny SALARY są zabezpieczone w prawidłowy sposób. Monitorowanie precyzyjne oraz monitorowanie standardowe są użytecznymi mechanizmami, dzięki którym od razu można sprawdzić, czy polityki autoryzacji zostały odpowiednio skonfigurowane.

Widoki danych słownikowych dotyczących monitorowania

W tabeli 10.21 przedstawiono widoki danych słownikowych, które dotyczą monitorowania.

Tabela 10.21. *Widoki danych słownikowych dotyczących monitorowania*

Widok danych słownikowych	Opis
AUDIT_ACTIONS	Zawiera opisy kodów rodzajów czynności rejestrowanych w śladach monitorowania, takich jak INSERT, DROP VIEW, DELETE, LOGON i LOCK.
DBA_AUDIT_OBJECT	Zawiera rekordy śladu monitorowania dotyczące obiektów w bazie danych.
DBA_AUDIT_POLICIES	Zawiera polityki monitorowania precyzyjnego w bazie danych.
DBA_AUDIT_SESSION	Zawiera wszystkie rekordy śladu monitorowania dotyczące poleceń CONNECT i DISCONNECT.
DBA_AUDIT_STATEMENT	Zawiera wpisy śladu monitorowania związane z poleceniami GRANT, REVOKE, AUDIT, NOAUDIT i ALTER SYSTEM.
DBA_AUDIT_TRAIL	Zawiera wpisy śladu monitorowania standardowego. USER_AUDIT_TRAIL zawiera wiersze monitorowania dotyczące wyłącznie połączonego użytkownika.
DBA_FGA_AUDIT_TRAIL	Zawiera wpisy śladu monitorowania pochodzące od polityk monitorowania precyzyjnego.
DBA_COMMON_AUDIT_TRAIL	Łączy w sobie wiersze monitorowania standardowego i precyzyjnego.
DBA_OBJ_AUDIT_OPTS	Zawiera obowiązujące opcje monitorowania dotyczące obiektów bazodanowych.
DBA_PRIV_AUDIT_OPTS	Zawiera obowiązujące opcje monitorowania, dotyczące uprawnień systemowych.
DBA_STMT_AUDIT_OPTS	Zawiera obowiązujące opcje monitorowania dotyczące instrukcji.

Zabezpieczanie śladu monitorowania

Sam ślad monitorowania również musi być zabezpieczany, zwłaszcza wówczas, gdy dostęp do tabeli SYS.AUD$ muszą posiadać użytkownicy inni niż użytkownik SYSTEM. Jeden z przypadków, gdy użytkownicy inni niż użytkownicy SYS mają dostęp do śladu monitorowania, zachodzi wówczas, gdy użytkownicy tacy posiadają rolę DELETE_ANY_CATALOG (na przykład po to, by archiwizować i czyścić ślad monitorowania i zapewnić tym samym, że dane śladu monitorowania nie będą ograniczać ilości przestrzeni na dysku wymaganej dla innych obiektów znajdujących się w przestrzeni tabel SYS).

Aby włączyć monitorowanie śladu monitorowania, należy połączyć się z bazą danych jako użytkownik SYSDBA i wykonać następujące polecenie:

```
SQL> audit all on sys.aud$ by access;
Audit succeeded.
```

Po wykonaniu powyższego polecenia wszystkie czynności wykonywane na tabeli SYS.AUD$, w tym polecenia select, insert, update i delete, będą rejestrowane w tej samej tabeli SYS.AUD$. Można jednak zadać pytanie: co się stanie, gdy któryś z użytkowników usunie rekordy zawierające dane o dostępie do tabeli SYS.AUD$? Otóż w takim przypadku wiersze z tabeli rzeczywiście zostaną usunięte, lecz pojawi się w niej nowy wiersz rejestrujący fakt usunięcia wierszy. Dzięki temu zawsze będzie jakiś dowód operacji wykonanej w tabeli SYS.AUD$ — czy to przypadkowej, czy to celowej. Ponadto, jeśli parametr AUDIT_SYS_OPERATIONS będzie mieć wartość TRUE, wówczas wszystkie sesje, w których wykorzystuje się klauzule as sysdba, as sysoper lub w których użytkownik łączy się jako SYS, będą rejestrowane w pliku monitorowania zapisywanym w systemie operacyjnym, do którego prawdopodobnie nawet administratorzy bazy danych nie będą mieli dostępu. Podsumowując, istnieje wiele zabezpieczeń, dzięki którym rejestrowane są wszystkie uprawnione czynności wykonywane w bazie danych, a także wszelkie próby ukrycia tych czynności!

Techniki szyfrowania danych

Szyfrowanie danych zwiększa bezpieczeństwo zarówno wewnątrz, jak i na zewnątrz bazy danych. Użytkownik może być uprawniony do odczytywania danych z większości kolumn znajdujących się w tabeli, lecz jeśli jedna z kolumn będzie zaszyfrowana i użytkownik nie będzie znał klucza rozszyfrowującego, dane z tej kolumny będą dla niego bezużyteczne. To samo dotyczy danych, które trzeba przesyłać w bezpieczny sposób za pośrednictwem sieci.

W wersji Oracle 10*g* udostępniono nowy pakiet DBMS_CRYPTO, który zastąpił pakiet DBMS_OBFUSCATION_TOOLKIT i zawiera algorytm szyfrowania Advanced Encryption Standard (AES). Algorytm ten zastąpił dotychczas stosowany algorytm Data Encryption Standard (DES).

Procedury wchodzące w skład pakietu DBMS_CRYPTO mogą generować klucze prywatne; użytkownik może również samodzielnie wskazać i przechowywać swój klucz. W odróżnieniu od pakietu DBMS_OBFUSCATION_TOOLKIT, który mógł szyfrować jedynie dane typu RAW i VARCHAR2, pakiet DBMS_CRYPTO szyfruje również dane typu BLOB i CLOB.