STUDIA INFORMATICA 2001
Volume 22 Number 1 (43)

Bożena MAŁYSIAK
Politechnika Śląska, Instytut Informatyki

MECHANIZMY DOSTĘPU W SYSTEMACH ZARZĄDZANIA BAZAMI DANYCH

Streszczenie. W artykule został przedstawiony przegląd mechanizmów dostępu w systemach zarządzania bazami danych (SZBD), takich jak: Oracle, Gupta Centura i MS SQL Server. Opisano sposoby tworzenia użytkowników, przydzielania i odbierania im uprawnień.

DATA ACCESS IN DATABASE MANAGEMENT SYSTEMS

Summary. This article provides an overview of the data access mechanisms in database management system (DBMS) such as Oracle, Gupta Centura and MS SQL Server. It describes methods of users creating, granting and revoking privileges.

1. Wprowadzenie

We współczesnym świecie większość potrzebnych, ważnych informacji, do których powinien być szybki i prosty dostęp, przechowywana jest w wersji elektronicznej na dyskach lub innego rodzaju nośnikach. Informacje te w postaci danych gromadzone są i przechowywane w bazach danych. Bardzo ważne jest więc zarówno zapewnienie szybkiego dostępu do umieszczonych w bazie danych, jak i odpowiedniego bezpieczeństwa [9]. Zmieniające się potrzeby rynku systemów przetwarzania danych doprowadziły do konieczności kontrolowania dostępu do informacji na poziomie poszczególnych tabel bazy danych. Zwykle użytkownik bazy danych bardzo rzadko spotykał się z koniecznością tworzenia całej bazy danych, twórcy SZBD nałożyli więc ograniczenia, by operacje te mógł przeprowadzać administrator bazy danych (użytkownik posiadający odpowiednie uprawnienia).

Współczesne systemy zarządzania bazami danych posiadają więc rozbudowane mechanizmy kontroli dostępu do bazy danych, jak i ochrony samych danych zarówno z poziomu serwera bazy danych, jak i aplikacji [10]. Można zabezpieczyć dostęp do bazy danych, używając uwierzytelniania i uprawnień kont logowania (użytkowników), jak również użyć metod ochrony dostępu na poziomie aplikacji, takich jak: widoki (ang. view), procedury przechowywane (ang. stored procedure).

W niniejszej pracy przedstawiono różne sposoby implementacji mechanizmów ochrony na przykładzie systemów zarządzania bazami danych (SZBD), takich jak: Gupta Centura, MS SQL Server czy Oracle.

Kolejne rozdziały zawierają:

- sposoby tworzenia użytkowników,
- schematy uprawnień oraz sposoby ich nadawania w wymienionych SZBD.

2. Mechanizmy dostępu w serwerze SQLBase

Serwer SQLBase 7.5.1 określa (identyfikuje) nazwę bazy danych, użytkownika, który chce się podłączyć oraz jego hasło poprzez dane, które wpisane są przez program klienta *(np. sqltalk)*. Jeśli nie znajdzie żadnych danych (nazwy bazy danych, użytkownika czy hasła), to podłącza się do domyślnie zadeklarowanej w pliku sql.ini bazy danych jako zdefiniowany również w tym pliku domyślny użytkownik. Podłączenie do bazy następuje w wyniku wykonania operacji *CONNECT*:

```
CONNECT <nazwa bazy danych> <nazwa użytkownika> / <hasło>
```

Pierwsze połączenie z bazą danych rozpoczyna transakcję.

2.1. Poziomy uprawnień

W serwerze SQLBase zdefiniowane są trzy poziomy uprawnień *(CONNECT, RESOURCE, DBA)* oraz specjalne konto użytkownika systemowego *SYSADM*, mające najwyższy poziom uprawnień [1].

CONNECT – to najniższy poziom uprawnień, jaki jest początkowo nadawany użytkownikowi, utożsamiany z utworzeniem konta użytkownika. Nadanie tego poziomu uprawnień użytkownikowi umożliwia mu połączenie się z bazą danych i wykonywanie operacji na tabelach, do których zostały nadane mu odpowiednie uprawnienia. Odebranie tego poziomu równoznaczne jest z usunięciem użytkownika z bazy danych. Przed usunięciem użytkownika należy odebrać mu wszystkie nadane wcześniej przywileje do tabel czy perspektyw. Użytkownik, który ma zostać skasowany, nie może być właścicielem tabel.

RESOURCE - kolejny poziom uprawnień, nadanie którego umożliwia użytkownikowi tworzenie własnych obiektów oraz nadawanie innym uprawnień do wykonywania operacji na swoich tabelach. Użytkownik z poziomem uprawnień *RESOURCE* posiada automatycznie prawa do wykonywania wszystkich operacji na tabelach, których jest właścicielem. Odebranie użytkownikowi tego poziomu uprawnień powoduje utratę wszystkich uprawnień z nim związanych. Wszystkie tabele czy perspektywy stworzone wcześniej przez tego użytkownika pozostają w bazie danych.

DBA - poziom uprawnień administratorskich; nadanie użytkownikowi tego poziomu uprawnień pozwala mu na wykonywanie wszystkich operacji na tabelach znajdujących się w bazie danych, włączając w to prawo nadawania, modyfikowania czy odbierania przywilejów do konkretnych tabel pozostałym użytkownikom w bazie danych. Odebranie użytkownikowi tego poziomu uprawnień powoduje utratę wszystkich uprawnień związanych z tym poziomem. Wszystkie tabele czy perspektywy stworzone wcześniej przez tego użytkownika pozostają w bazie danych.

SYSADM - systemowe konto, posiadające najwyższy poziom uprawnień. Tylko użytkownik *SYSADM* może tworzyć użytkowników, podnosić im poziomy uprawnień lub zmieniać hasła. Żadnemu użytkownikowi nie można przypisać poziomu uprawnień *SYSADM*, w bazie danych może być tylko jedno konto o takich własnościach. Poziom uprawnień *SYSADM* nie może zostać odebrany.

2.2. Tworzenie użytkowników

Jak wspomniano powyżej, utworzenie konta użytkownika w serwerze SQLBase równoznaczne jest z nadaniem mu poziomu uprawnień *CONNECT*. Poziomy uprawnień nadawane są przez *użytkownika SYSADM* poleceniem *GRANT*.

W celu usunięcia konta użytkownika z bazy danych należy odebrać mu kolejno wszystkie uprawnienia. Do tego celu służy polecenie *REVOKE*.

2.3. Nadawanie uprawnień do poszczególnych tabel czy perspektyw

Oprócz poziomów uprawnień można przydzielać użytkownikom prawa do wykonywania pewnych operacji na poszczególnych tabelach lub perspektywach. Do tego celu również służy polecenie *GRANT* w nieco innej postaci [1]:

```
GRANT ALL, SELECT, INSERT, DELETE, INDEX, ALTER, UPDATE ON
<nazwa_właściciela.nazwa_tabeli> LUB <nazwa_właściciela.nazwa_perspektywy>
TO <nazwa_użytkownika> LUB <PUBLIC>
```

Prawa do wykonywania operacji na konkretnych obiektach bazy danych mogą być przyznawane użytkownikom przez właścicieli tych obiektów lub przez użytkowników z poziomem uprawnień *DBA*.

Uprawnienia do wykonywania operacji na określonych obiektach odbiera się również poleceniem *REVOKE* [1]:

```
REVOKE ALL, SELECT, INSERT, DELETE, INDEX, ALTER, UPDATE ON
<nazwa_właściciela.nazwa_tabeli> LUB <nazwa_właściciela.nazwa_perspektywy>
FROM <nazwa_użytkownika/PUBLIC>
```

Podczas tworzenia bazy danych serwer SQLBase tworzy słownik danych, który jest zbiorem tabel zawierających informacje o obiektach znajdujących się w bazie danych. Informacje o poziomach uprawnień użytkowników czy przywilejach użytkowników do poszczególnych tabel można znaleźć odpowiednio w systemowych tabelach: *SYSUSERAUTH* i *SYSTABAUTH*. Właścicielem tabel systemowych jest użytkownik *SYSADM*. Tylko użytkownicy, którym *SYSADM* lub użytkownik z poziomem *DBA* przyznał prawo wstawiania, modyfikacji i usuwania kolumn w tabelach systemowych mogą wykonywać te operacje, ale tylko w obrębie kolumn zdefiniowanych przez użytkownika (siebie). Nie mogą natomiast usuwać tabel systemowych, kolumn systemowych oraz wstawiać lub usuwać wierszy w tabelach systemowych.

2.4. Przykład

W bazie danych *MAGAZYNY*, gromadzącej dane o stanie magazynu, stworzyć użytkownika *kierownik* o haśle *k2j3*, mogącego tworzyć tabele oraz wprowadzać do nich niezbędne dane oraz użytkownika *pracownik* o haśle *ppp34*, mogącego przeglądać tabelę *towary* (właścicielem jej jest użytkownik *kierownik*) i modyfikować kolumnę *liczba*, określającą zapas danego towaru.

```
connect MAGAZYNY SYSADM/SYSADM;
grant CONNECT to kierownik identified by k2j3;
grant RESOURCE to kierownik;
grant CONNECT to pracownik identified by ppp34;
```

Przypisanie uprawnień do tabeli podłączony do bazy danych jako użytkownik *SYSADM* lub *kierownik:*

```
grant select on kierownik.towar to pracownik;
grant update (liczba) on kierownik.towar to pracownik;
```

wyświetlenie poziomów uprawnień i przywilejów do tabel dla stworzonych użytkowników:

```
select * from SYSADM.SYSUSERAUTH;
select * from SYSADM.SYSTABAUT;
```

usunięcie konta *kierownik* i *pracownik* z bazy *MAGAZYNY*:

```
connect MAGAZYNY SYSADM/SYSADM;
revoke RESOURCE from kierownik;
revoke ALL on kierownik.towar from pracownik;
connect MAGAZYNY kierownik/k2j3;
```

```
drop table towar;
connect MAGAZYNY SYSADM/SYSADM
revoke CONNECT from kierownik;
revoke CONNECT from pracownik;
```

3. Mechanizmy dostępu w serwerze ORACLE

Schemat uprawnień w SZBD Oracle jest stopniowo rozbudowywany. W serwerze Oracel6 można było wyodrębnić trzy grupy użytkowników: *CONNECT*, *RESOURCE* i *DBA*. Użytkownicy z uprawnieniami systemowymi typu *CONNECT* mieli podstawowy dostęp do bazy, jaki wyznaczały im przydzielone uprawnienia do konkretnych obiektów, nie mogli tworzyć własnych obiektów bazy danych czy wykonywać operacji administracyjnych na instancji bazy danych. Użytkownicy z uprawnieniami systemowymi typu *RESOURCE* mogli również wykonywać pewne operacje na obiektach, których nie byli właścicielami (operacje te wyznaczały przydzielone im odpowiednie uprawnienia do konkretnych istniejących już w bazie danych obiektów), a ponadto mieli już uprawnienia do tworzenia własnych obiektów. Użytkownicy z ostatniej grupy *DBA* mieli najwyższe uprawnienia, dostęp do każdego obiektu, dostęp do operacji administracyjnych na instancji bazy danych [2].

Ten schemat uprawnień okazał się jednak zbyt ogólny, np. każdy użytkownik z uprawnieniami *DBA* mógł tworzyć, otwierać czy zamykać bazę danych (zwykle wystarczy, że jeden użytkownik jest administratorem bazy danych). Z tego powodu schematy uprawnień w serwerach Oracle7 i Oracle8 zostały znacznie rozbudowane. Trzy grupy uprawnień systemowych zostały podzielone na 78 uprawnień. Im bardziej szczegółowy podział uprawnień, tym więcej pracy ma administrator. W celu usprawnienia przydziału uprawnień stworzono w systemie Oracle mechanizm ról. Role to zbiory uprawnień systemowych i obiektowych, w szczególności mogą one zawierać również inne role, nadawane użytkownikom.

Przed administratorem staje więc trudne zadanie opracowania schematu bezpieczeństwa instancji bazy danych, zaprojektowanie przejrzystego i rozsądnego mechanizmu ról i uprawnień na poziomie bazy danych.

Oprócz mechanizmów kontroli dostępu są jeszcze uprawnienia obiektowe, które umożliwiają szczegółową kontrolę operacji, jakie użytkownik może wykonywać na konkretnych, istniejących w bazie danych obiektach.

Dokładnie proces tworzenia użytkowników, przydzielania im uprawnień systemowych, obiektowych i ról oraz składnia tych poleceń zostały omówione w pracy [4]. Dlatego w tym rozdziale elementy te zostaną przedstawione w dość ogólnej postaci, natomiast będą

przedstawione pewne modyfikacje i nowe mechanizmy ochrony, które pojawiły się w wersji Oracle8.

3.1. Tworzenie użytkowników

Logiczna przestrzeń bazy danych, w której użytkownik tworzy własne obiekty, to schemat użytkownika. W serwerze Oracle stworzenie użytkownika jest tożsame z powstaniem schematu użytkownika o tej samej nazwie. Do tworzenia użytkowników służy polecenie: *CREATE USER*.

W procesie tworzenia użytkownika można określić między innymi: domyślną przestrzeń tabel (ang. *Default tablespace*) - miejsce, w którym będą zapisywane tworzone przez niego obiekty), tymczasową przestrzeń tabel (ang. *temporary tablespace*), ilość miejsca, jaką może zająć użytkownik w określonej przestrzeni tabel (ang. *quota*) oraz nadać użytkownikowi określony profil (ang. *profile*), ograniczając mu dostęp do zasobów systemowych.

Tworzenie profili wykonywane jest poleceniem: *CREATE PROFILE*, nadawanie profili - poleceniem: *ALTER USER <nazwa_użytkownika> PROFILE <nazwa_profilu>*, a uaktywnianie ich - ustawieniem zmiennej *RESOURCE_LIMIT = true*. Informacje o zdefiniowanych profilach można uzyskać poprzez wyświetlanie zawartości perspektyw słownika danych: *dba_profiles, dba_users*.

Tworzyć użytkowników lub modyfikować ich definicje może tylko użytkownik posiadający przywilej systemowy *CREATE USER, ALTER USER* [7], [8]. Użytkownik może zmienić sam sobie tylko hasło. Serwer Oracle posiada specjalne konto *INTERNAL*, użytkownik ten posiada prawa *DBA* oraz może tworzyć, podnosić i zamykać bazę danych.

Użytkowników usuwa się poleceniem *DROP USER <nazwa_użytkownika> [cascade]*, usuniecie użytkownika z opcją *cascade* powoduje usunięcie również wszystkich obiektów kasowanego użytkownika.

Słownik bazy danych zawiera perspektywy, które umożliwiają wyświetlenie nazw zdefiniowanych użytkowników: *dba_users, all_users*.

3.2. Uprawnienia systemowe, obiektowe i role

Uprawnienia systemowe, jak wspomniano wcześniej, umożliwiają kontrolę dostępu użytkownika do instancji bazy danych (*CREATE SESSION, ALTER SESSION, ALTER DATABASE*,), określają jakiego typu operacje mogą być wykonywane przez użytkowników na grupach obiektów: tabelach, perspektywach, indeksach itd. Umożliwiają tworzenie, modyfikowanie i usuwanie różnych struktur (*CREATE TABLE, CREATE INDEX, ALTER ANY TABLE, DROP ANY TABLE*) [11].

Wymagającymi dużej odpowiedzialności i najbardziej niebezpiecznymi ze względu na możliwości przywilejami systemowymi w odniesieniu do typów obiektów są przywileje zawierające słowo *ANY*. Jeśli nadano użytkownikowi przywilej CREATE TABLE, może on tworzyć własne tabele, jak również je kasować, natomiast, gdy zostaną mu nadane uprawnienia *SELECT ANY TABLE, DROP ANY TABLE*, będzie on miał dostęp do każdej istniejącej w systemie tabeli, będzie ją mógł przeglądać czy też kasować. Ze względów bezpieczeństwa uprawnienia systemowe (nawet z ANY) nie pozwalają na korzystanie ze słownika danych [2].

Z każdym obiektem w bazie danych związane są również uprawnienia, określające kto i jaki ma dostęp do tego obiektu (np. *ALTER, DELETE, EXECUTE, INDEX, INSERT, SELECT, UPDATE, REFERENCES lub wszystkie (ALL)*). Każdy użytkownik ma pełne prawa do obiektów, które utworzył. Uprawnienia obiektowe umożliwiają administratorom przyznanie uprawnień do obiektów tylko pewnym, wybranym użytkownikom, pozwalają więc na bardziej szczegółowy rozdział uprawnień i ochronę poszczególnych obiektów przed niepowołanym dostępem.

Mechanizmem ułatwiającym nadawanie użytkownikom uprawnień zarówno systemowych, jak i obiektowych są role. Zazwyczaj największy problem stanowi dokonanie właściwego podziału na grupy użytkowników, pełniących podobne lub wręcz takie same funkcje w bazie danych, korzystających z tych samych obiektów, jak również stworzenie właściwych ról, zawierających odpowiednie uprawnienia. Jeśli administrator zmodyfikuje zbiór uprawnień związanych z rolą, zmiany te automatycznie są widziane przez użytkowników mających daną rolę. Role tworzy się poleceniem:

```
CREATE ROLE <nazwa_roli>,
```

a usuwa poleceniem:

```
DROP ROLE <nazwa_roli>.
```

W systemie Oracle istnieją predefiniowane role: *CONNECT (*uprawnienia niezbędne do przyłączenia się do bazy danych i do tworzenia podstawowych obiektów), *RESOURCE* (to co CONNECT i dodatkowo jeszcze tworzenie procedur, funkcji i pakietów, wyzwalaczy), *DBA* (wszystkie uprawnienia potrzebne administratorowi), *EXP_FULL_DATABASE, IMP_FULL-_DATABASE (*dwie ostatnie, potrzebne są użytkownikowi, który będzie wykonywał pełen eksport lub import bazy danych, używając narzędzi export/import*)* [3]. System Oracle umożliwia również przekazywanie użytkownikom uprawnień: administratora instancji w postaci predefiniowanych ról: *SYSDBA* lub *SYSOPER*. Użytkownik, który otrzyma te role, będzie mógł wykonywać wszystkie czynności administracyjne na bazie danych (tworzyć, startować, zamykać). By podłączyć się do bazy jako administrator instancji należy wpisać:

```
CONNECT <nazwa_użytkownika> as SYSDBA | SYSOPER;
```

Słownik bazy danych zawiera informacje o zdefiniowanych w bazie rolach. W celu ich wyświetlenia należy wykorzystać perspektywy: *dba_roles, dba_role_privs, role_sys_privs, role_tab_privs, role_role_privs.*

3.3. Nadawanie uprawnień użytkownikom

W procesie tworzenia użytkownikowi nie są przydzielane żadne domyślne uprawnienia. Wszystkie uprawnienia, jakie powinien on posiadać, administrator bazy danych musi mu niezależnie nadać poleceniem *GRANT* [7].

```
GRANT <nazwa_przywileju> TO <nazwa_użytkownika/nazwa_roli>
[WITH ADMIN/GRANT OPTION]
```

Dodanie opcji `WITH ADMIN/GRANT OPTION` umożliwia przekazywanie nadanego uprawnienia innym użytkownikom.

Przywileje systemowe są zwykle nadawane przez administratora, obiektowe mogą być przyznawane przez właściciela obiektu. By nadać uprawnienia do konkretnego obiektu, trzeba podać zarówno nazwę przywileju, jak i nazwę obiektu.

Użytkownikom można również odbierać nadane uprawnienia. Służy do tego celu polecenie *REVOKE.*

```
REVOKE <nazwa_przywieju> FROM <nazwa_użytkownika/nazwa_roli>
```

Administrator ma możliwość sprawdzenia, jakie uprawnienia systemowe, obiektowe czy role zostały nadane użytkownikom. Do tego celu służą perspektywy: *dba_sys_privs, dba_tab_privs, dba_role_privs* [11].

3.4. Dodatkowe mechanizmy kontroli dostępu w SZBD Oracle8

Serwer Oracle8 posiada kilka nowych rozwiązań, dotyczących systemu haseł, takich jak: historia haseł, blokowanie konta użytkownika, definiowanie okresu ważności hasła, weryfikacja użytkownika oparta na regułach oraz dodatkowe możliwości definiowania profilu użytkownika. Dodano również predefiniowane role [3].

Standardowo w systemie Oracle istnieją dwa sposoby weryfikacji użytkowników: z poziomu bazy danych oraz z poziomu systemu operacyjnego (zewnętrzne uwierzytelnianie). W serwerze Oracle8 istnieje jeszcze jedna metoda sprawdzania użytkowników *(ang. enterprise authentication)* [5]. W przypadku gdy środowisko składa się z wielu baz danych, serwer *Oracle8 Security Service (OSS)* stanowi globalny punkt sprawdzania użytkowników. Nadal trzeba tworzyć konta użytkowników we wszystkich dostępnych dla nich bazach danych, dane dotyczące kontroli dostępu i hasło przechowywane są w centralnym miejscu. Pozwala to na jedno logowanie się do wszystkich baz danych, a zmiana hasła będzie uwzględniana przez wszystkie systemy baz danych korzystające z *OSS.* Jeśli stworzono rolę z parametrem *GLOBALLY,* oznacza to, że użytkownik musi

zostać uwierzytelniony poprzez *Oracle Security Service* zanim stworzona rola będzie mogła zostać włączona.

Polecenia *CREATE USER* i *ALTER USER* w serwerze Oracle8 zostały wzbogacone o dodatkowe parametry [3]:

- *password expire* – umożliwia natychmiastowe wygaśnięcie hasła,
- *account lock | unlock* - polecenie z parametrem *Account Lock* – utworzy konto użytkownika, ale nie będzie się on mógł podłączyć do momentu, gdy konto nie zostanie odblokowane.

Zdefiniowano nowe role:

- *SELECT_CATALOG_ROLE* – umożliwiająca wydawanie zapytań do tabel i perspektyw słownika bazy danych;
- *EXECUTE_CATALOG_ROLE* – umożliwiająca wykonywanie procedur i funkcji składowanych, będących częścią słownika bazy danych;
- *DELETE_CATALOG_ROLE* – umożliwiająca usuwanie rekordów z dziennika obserwacji *SYS.AUD$*.

W serwerze Oracle8 wprowadzono również możliwość weryfikacji hasła użytkownika. Administrator może wymusić na użytkownikach wybór hasła o określonym stopniu złożoności, zabronić wprowadzania haseł związanych z nazwą użytkownika itd. Do weryfikacji haseł służy składowana funkcja weryfikacji. Funkcja ta uruchamiana jest po zmianie hasła poleceniem *ALTER USER*. Przykładowa funkcja weryfikacji hasła jest zaimplementowana i dostarczana wraz z oprogramowaniem SZBD Oracle8 w skrypcie *utlpwdmg.sql*. Funkcje tę należy utworzyć będąc podłączonym do bazy jako użytkownik *SYS* i przypisać ją do konkretnych użytkowników za pomocą profilu, którego parametr *password_verify_function* określa nazwę funkcji weryfikacji.

Nowe parametry profilu związane są z kontrolą podłączania się do bazy danych i kontrolą haseł użytkowników bazy danych oraz określają [5]:

- *failed_login_attempts* – dozwoloną liczbę nieudanych prób podłączenia się, po przekroczeniu której następuje automatyczne zablokowanie konta użytkownika na czas określony parametrem *password_lock_time*;
- *password_life_time* – czas ważności hasła (w dniach); po jego przekroczeniu hasło musi zostać zmienione;
- *password_reuse_time* – okres czasu, jaki musi upłynąć, by użytkownik mógł ponownie użyć poprzednie hasło; (parametr *Password_reuse_max* przyjmuje wtedy wartość *UNLIMITED*);

– *password_reuse_max* – liczba zmian hasła, po przekroczeniu której użytkownik będzie mógł ponownie wybrać używane kiedyś hasło; (parametr *password_reuse_time* przyjmuje wtedy wartość *UNLIMITED*);
– *password_lock_time* – liczba dni, na jaką zostanie zablokowane konto użytkownika, po przekroczeniu liczby nieudanych prób podłączania się (określonej parametrem *failed_login_attempts*);
– *password_grace_time* – liczba dni, przez jakie hasło będzie jeszcze ważne, po przekroczeniu czasu jego obowiązywania (określonego parametrem *password_life_time);*
– *password_verify_function* – nazwa funkcji weryfikacji hasła.

Zarządzanie hasłami jest uaktywniane poprzez wykonania (jako użytkownik *SYS*) skryptu *utlpwdmg.sql*. Podsumowując można stwierdzić, że dla różnych kategorii użytkowników wskazane jest zastosowanie różnych zasad zarządzania hasłami poprzez stworzenie dla każdej kategorii różnych profili i przypisanie ich odpowiednim użytkownikom.

3.5. Przykład

W systemie istnieje baza danych *MAGAZYNY*, gromadząca dane o stanie magazynu. Stworzyć użytkownika *kierownik* o haśle *k2j3*, nie mającego ograniczeń limitu w przestrzeni tabel *users*. Wymusić natychmiastowe wygaśnięcie hasła. Stworzyć rolę *kier_role* zawierającą przywileje umożliwiające: podłączenie się do bazy, tworzenie tabel, przeglądanie, wstawianie, modyfikowanie danych w istniejących już tabelach. Przypisać użytkownikowi *kierownik* rolę *kier_role*. Stworzyć użytkownika *pracownik* o haśle *ppp34*, mogącego przeglądać tabelę *towary* (właścicielem jej jest użytkownik *kierownik)* i modyfikować kolumnę *liczba*, określającą zapas danego towaru. Po stworzeniu konto *pracownik* powinno być zablokowane. Stworzyć profil *ogr_has* ograniczający liczbę nieudanych prób podłączenia się do bazy do 3, po 3 nieudanej próbie konto powinno zostać zablokowane na 2 dni. Ograniczyć czas ważności hasła do 90 dni oraz ograniczyć czas ważności hasła po upłynięciu terminu wygaśnięcia do 3 dni, udostępnić weryfikację hasła za pomocą zdefiniowanej w systemie funkcji weryfikacji *wer_has*. Przypisać profil użytkownikowi pracownik i odblokować jego konto.

Tworzenie użytkownika *kierownik:*

```
create user kierownik identified by k2j3
default tablespace users
quota UNLIMITED on users
password expire;
```

Tworzenie roli *kier_role* i przypisywanie jej uprawnień:

```
create role kier_role;
grant CREATE SESSION, SELECT ANY TABLE, CREATE ANY TABLE,
```

```
INSERT ANY TABLE, UPDATE ANY TABLE to kier_role;
```

Przypisanie użytkownikowi roli *kier_role*:

```
grant kier_role to kierownik
```

Tworzenie użytkownika *pracownik:*

```
create user pracownik identified by ppp34
default tablespace users
account lock;
```

Przypisanie użytkownikowi *pracownik* uprawnień:

```
grant CREATE SESSION to pracownik;
grant SELECT, UPDATE(LICZBA) on kierownik.towar to pracownik;
```

Tworzenie profilu *ogr_has*:

```
create profile ogr_has limit
FAILED_LOGIN_ATTEMPTS 3
PASSWORD_LOCK_TIME 2
PASSWORD_LIFE_TIME 90
PASSWORD_GRACE_TIME 3
PASSWORD_VERIFY_FUNCTION wer_has;
```

Przypisanie profilu *ogr_has* użytkownikowi *pracownik* i odblokowanie jego konta:

```
alter user pracownik
profile ogr_has
account unlock;
```

4. Mechanizmy dostępu w SZBD MS SQL Server7

Użytkownik, by nawiązać połączenie z serwerem SQL, musi mieć konto logowania. Serwer SQL rozróżnia dwa rodzaje uwierzytelniania (tak jak serwer Oracle7): uwierzytelnienie serwera SQL i systemu Windows NT. W pierwszym z nich administrator systemu serwera SQL definiuje konto logowania i hasło, w drugim grupa lub konto systemu Windows NT określa dostęp użytkownika do serwera SQL i jego zasobów.

Zalety trybu uwierzytelniania systemu Windows NT to [12]:

- udostępnianie zaawansowanych funkcji zabezpieczeń, takich jak: szyfrowanie haseł, wygasanie haseł, minimalną długość haseł, blokowanie kont w przypadku wprowadzenia nieprawidłowego hasła,
- umożliwianie dodania grupy użytkowników do serwera SQL przez dodanie pojedynczego konta logowania,
- umożliwianie szybkiego dostępu do serwera SQL (bez konieczności pamiętania innego konta logowania i hasła).

Zalety trybu mieszanego (zwłaszcza uwierzytelnienia serwera SQL) to:

- umożliwianie aplikacjom klienckim przeznaczonym dla systemów innych niż system Windows NT, internetowym aplikacjom klienckim nawiązania połączenia z serwerem SQL,
- umożliwianie dodania kolejnej warstwy zabezpieczeń do systemu Windows NT.

Weryfikacja użytkowników (jak w większości SZBD) występuje tu na dwóch poziomach zabezpieczeń: uwierzytelnienie logowania, kontrola poprawności uprawnień kont użytkowników bazy danych i ról. Po poprawnym połączeniu się z serwerem SQL użytkownik może podłączyć się do bazy danych. Nastąpi to tylko wtedy, gdy użytkownik będzie miał uprawnienia dostępu do bazy danych. Użytkownicy mający dostęp do bazy danych muszą mieć w niej stworzone swoje konta. Konta użytkowników i role w bazie danych identyfikują użytkownika i określają uprawnienia do wykonywania operacji na obiektach w bazie, są specyficzne dla bazy danych. Użytkownicy, którzy nie mają uprawnień do żadnej bazy danych, mogą uzyskać dostęp do serwera i utworzyć zapytania do niektórych tabel systemowych, nie mogą natomiast uzyskać dostępu do żadnej bazy danych użytkowników.

W serwerze SQL istnieje mechanizm ról. Role umożliwiają zgrupowanie użytkowników, którzy powinni posiadać w bazie danych podobne uprawnienia. W serwerze SQL istnieją zdefiniowane już role: serwerów i baz danych. Użytkownik może tworzyć sobie również własne role bazy danych. Użytkownicy mogą należeć do wielu ról (w serwerze Oracle - użytkownikom przypisywane są role, w serwerze SQL użytkownicy przypisywani są do zdefiniowanych ról).

4.1. Tworzenie kont logowania

Konta logowania można utworzyć jako nowe konta serwera SQL, używając jednego z domyślnych kont logowania lub wykorzystując użytkowników i grupy systemu Windows NT [13]. Informacje o kontach logowania przechowywane są w tabeli systemowej *syslogins* bazy danych *Master*. Do konta logowania serwera SQL zostaje przypisana domyślna baza danych, jednakże przypisanie jej do konta nie powoduje utworzenia w niej konta użytkownika. Zostaje określony tylko domyślny kontekst akcji wykonywanych przez użytkownika. Jeśli konto użytkownika nie zostanie utworzone dla konta logowania, to można uzyskać dostęp do domyślnie przypisanej bazy danych poprzez podłączenie się jako użytkownik *gość (ang. guest)*, jeśli konto tego typu istnieje.

4.1.1. Dodanie konta logowania systemu Windows NT do serwera SQL

Aby umożliwić kontu użytkownika lub grupie Windows NT nawiązanie połączenia z serwerem SQL należy użyć programu *SQL Server Enterprise Manager* lub systemowej

procedury przechowywanej *sp_grantlogin*. Tylko administratorzy systemu lub zabezpieczeń (członkowie ról odpowiednio: *sysadmin i securityadmin*) mogą wykonywać tę procedurę.

```
Sp_grantlogin {'nazwa dodawanego użytkownika lub grupy'}
```

Serwer SQL posiada pojedyncze konto logowania dla grupy systemu Windows NT; dodatkowo usunięcie grupy lub użytkownika systemu Windows NT nie powoduje usunięcia tego użytkownika lub grupy z serwera SQL.

W SQL serwerze istnieją systemowe procedury przechowywane, które mogą zostać wykorzystane do zarządzania kontami logowania systemu Windows NT:

- *sp_revokelogin* – usuwająca wpisy dla kont logowania użytkowników lub grup systemu Windows NT z serwera SQL,
- *sp_denylogin* – uniemożliwiająca użytkownikowi lub grupie systemu Windows NT nawiązanie połączenia z serwerem SQL.

4.1.2. Dodawanie konta logowania serwera SQL

Podobnie jak w poprzednim punkcie do dodawania konta logowania serwera SQL można wykorzystać program *SQL Server Enterprise Manager* lub systemową procedurę przechowywaną *sp_addlogin*. Tylko administratorzy systemu lub zabezpieczeń (członkowie ról odpowiednio: *sysadmin i securityadmin*) mogą wykonywać tę procedurę.

```
Sp_addlogin {'nazwa_konta'}[,'hasło [,'baza_danych']],
```

Procedura ta dodaje rekordy do tabeli systemowej *syslogins* bazy *master*. Po jej wykonaniu użytkownik może podłączyć się do serwera SQL.

Użytkownicy mogą zmieniać własne hasła w dowolnej chwili, używając systemowej procedury *sp_password*.

Serwer SQL ma dwa domyślne konta logowania, posiadające wszystkie prawa dotyczące serwera SQL i wszystkich baz danych:

- *BUILTIN/Administrators* – domyślne konto logowania dla wszystkich administratorów systemu Windows NT,
- *sa* – administrator systemu.

4.1.3. Przypisywanie kont logowania do użytkowników i ról

Konta logowania serwera SQL można mapować na konta użytkowników i role w poszczególnych bazach danych. W tabeli systemowej *sysusers* w bazie danych jeden wiersz odpowiada każdemu użytkownikowi lub grupie systemu Windows NT, użytkownikowi serwera SQL lub roli w bazie danych. Natomiast uprawnienia stosowane są w odniesieniu do wpisów w tabeli *sysusers* i przechowywane w tabeli *sysprotects* bieżącej bazy danych.

Podobnie jak w poprzednim punkcie, w celu dodania konta użytkownika do bazy danych można wykorzystać program *SQL Server Enterprise Manager* lub systemową procedurę

przechowywaną *sp_grantdbaccess*. Tylko właściciele bazy danych lub administratorzy dostępu do bazy danych mogą wykonywać tę procedurę.

```
Sp_grantdbaccess ('nazwa_logowania')[,nazwa_w_bd']
```

Konta logowania *sa* i członkowie roli *sysadmin* mapowani są na specjalne konto użytkownika o nazwie *dbo*. Konto to jest kontem domyślnym i nie może być usunięte. Każdy obiekt otworzony przez administratora systemu automatycznie należy do użytkownika *dbo*.

W bazie danych serwera SQL istnieje konto użytkownika *gość (ang. guest)*, umożliwiające podłączanie się do bazy bez kont użytkowników.

Jak wspomniano wcześniej, w serwerze SQL istnieje mechanizm ról, umożliwiający grupowanie użytkowników wykonujących podobne czynności, przypisując ich do jednej roli.

4.2. Role serwerów, baz danych i użytkowników

Serwer SQL udostępnia zdefiniowane role serwerów i baz danych - często wykonywanych funkcji administracyjnych, umożliwia to proste i szybkie przyznawanie uprawnień administracyjnych określonemu użytkownikowi.

Predefiniowane role serwerów zostały stworzone w celu pogrupowania przywilejów administracyjnych na poziomie serwera [12].

Najczęściej używane role serwerów pozwalają na:

- *dbcreator* – tworzenie i modyfikację bazy danych,
- *diskadmin* – zarządzanie plikami dyskowymi,
- *proccessadmin* – zarządzanie procesami serwera SQL,
- *securityadmin* – zarządzanie podłączeniami do serwera i kontrolę podłączeń,
- *serveradmin* – konfigurowanie ustawień w całym serwerze,
- *setupadmin* – instalowanie replikacji,
- *sysadmin* – wykonywanie dowolnych działań.

Stałe role są zarządzane niezależnie od baz danych na poziomie serwera. W celu przypisania konta logowania do stałej roli można wykorzystać program *SQL Server Enterprise Manager* lub systemową procedurę przechowywaną *sp_addsrvrolemember*. Tylko członkowie stałych ról serwera mogą wykonywać tę procedurę.

```
Sp_addsrvrolemember 'konto_logowania', nazwa_roli
```

Podczas gdy konto logowania dodawane jest do roli serwera, w tabeli *syslogins* w bazie *Master* aktualizowany jest odpowiedni wiersz, zawierający informacje, że konto logowania jest członkiem roli i posiada uprawnienia takie jak dana rola serwera.

W celu usunięcia członka ze stałej roli serwera należy użyć systemowej procedury przechowywanej: *sp_dropsrvrolemember*.

Stałe role baz danych grupują przywileje administracyjne na poziomie bazy danych.

Najczęściej używane role baz danych to [12]:

- *public* – posiada wszystkie domyślne uprawnienia użytkowników w bazie danych, należy do niej każdy użytkownik bazy danych; domyślnie rola ta nie posiada żadnych uprawnień, administrator musi więc zdecydować, jakie uprawnienia powinni mieć wszyscy użytkownicy bazy danych i takie jej przyznać,
- *db_owner* – wykonywanie dowolnych działań roli bazy danych,
- *db_accessadmin* – dodawanie lub usuwanie użytkowników, grup i ról bazy danych,
- *db_ddladmin* – dodawanie, modyfikowanie lub usuwanie obiektów bazy danych,
- *db_security* – przypisywanie uprawnień dotyczących instrukcji i obiektów,
- *db_backupoperator* – wykonywanie kopii zapasowych i przywracanie baz danych,
- *db_datareader* - odczytywanie danych z dowolnej tabeli,
- *db_datawriter* – dodawanie, zmienianie lub usuwanie danych ze wszystkich tabel,
- *db_denydatareader* – brak uprawnień do odczytywanych danych z dowolnej tabeli,
- *db_denydatawiter* – brak uprawnień do zmieniania danych w dowolnej tabeli.

Stałe role baz danych są przechowywane w tabeli systemowej *sysusers* poszczególnych baz danych.

Użytkownicy mogą tworzyć własne role. Utworzenie roli bazy danych zdefiniowanej przez użytkownika umożliwia utworzenie grupy użytkowników posiadających zestaw wspólnych uprawnień. W celu dodania nowej roli do bazy danych można wykorzystać program *SQL Server Enterprise Manager* lub systemową procedurę przechowywaną *sp_addrole*. Tylko właściciele bazy danych lub administratorzy dostępu do bazy danych mogą wykonywać tę procedurę.

```
sp_addrole nazwa_roli, właściciel
```

W celu przypisania konta zabezpieczeń do stałej lub zdefiniowanej przez użytkownika roli bazy danych należy zastosować *SQL Server Enterprise Manager* lub systemową procedurę przechowywaną *sp_addrolemember*.

```
sp_addrolemember nazwa_rola, konto_zabezpieczeń
```

W serwerze SQL istnieją dodatkowe systemowe procedury, które mogą być użyte do zarządzania rolami baz danych:

sp_droprole – usuwa rolę z serwera SQL, z bieżącej bazy danych,

sp_droprolemember – usuwa konto zabezpieczeń z roli serwera SQL.

4.3. Typy uprawnień

Użytkownik, któremu nie nadano uprawnień, posiada wszystkie uprawnienia przydzielone roli *public*, może więc wykonywać instrukcje nie wymagające uprawnień, przeglądać tabele systemowe, wykonywać określone procedury systemowe w celu uzyskania

informacji z bazy danych master i baz danych użytkowników, do których ma dostęp, uzyskiwać dostęp do dowolnej bazy danych z kontem *gość (ang. guest)*.

Uprawnienia określają, jakie operacje użytkownicy mogą wykonywać na poszczególnych obiektach bazy danych. Uprawnienia użytkownika w bazie danych wyznaczają uprawnienia konta użytkownika i ról, których jest członkiem. Każda baza danych posiada własne niezależne systemy uprawnień.

Typy uprawnień:

- *instrukcji* – obejmujące tworzenie bazy danych oraz jej elementów. Uprawnienia dotyczące instrukcji, np. *CREATE DATABASE*, są przypisywane do instrukcji, a nie do elementu zdefiniowanego w bazie danych; tylko członkowie ról *sysadmin, db_owner lub db_securityadmin* mogą udzielać uprawnień dotyczących instrukcji,
- *obiektów* – obejmują operacje na danych lub wykonywanie procedur (np.: *SELECT, INSERT, UPDATE, EXEC*),
- *predefiniowane uprawnienia* – uprawnienia właścicieli obiektów (mogą wykonywać wszystkie operacje na obiektach, których są właścicielami) oraz członków stałych ról (posiadają uprawnienia zdefiniowane w roli, której są członkami).

Uprawnienia dla użytkownika lub roli mogą być:

- *udzielane* – do tego celu służy polecenie *GRANT* (zezwolenie na wykonanie operacji)

Składnia polecenia *GRANT* dla uprawnień dotyczących instrukcji:

```
GRANT {ALL | instrukcja [,..]} TO nazwa_konta_zabezpieczeń[,..]
```

Polecenie *GRANT* dla uprawnień dotyczących obiektów:

```
GRANT
{ALL [PRIVILEGES] | uprawnienie [,...]}
{[(kolumna [, ...])] ON {tabela | perspektywa}
| ON {tabela | perspektywa} [(kolumna [, ...])]
| ON {procedura_przechowywana | procedura_rozszerzona}}
TO konto_zabezpieczeń [, ...]
[AS {grupa | rola}]
```

- *odmawiane* – do tego celu służy polecenie *DENY* (odmowa wykonania jakiegoś polecenia)

Składnia polecenia *DENY* dla uprawnień dotyczących instrukcji:

```
- DENY {ALL | instrukcja [,...]} TO konto_zabezpieczeń [,..]
```

Polecenie *DENY* dla uprawnień dotyczących obiektów:

```
DENY
{{ALL [PRIVILEGES] | uprawnienie [,...]} [(kolumna [, ...])] ON
{tabela | perspektywa}
|{procedura_przechowywana | procedura_rozszerzona}}
```

```
TO konto_zabezpieczeń [, ...]
```

- *cofane* – do tego celu służy polecenie *REVOKE* (odebranie prawa wykonania pewnych operacji, może jednak być zastąpione przez uprawnienie przypisane do roli)

Składnia polecenia *REVOKE* dla uprawnień dotyczących instrukcji:

```
- REVOKE {ALL | instrukcja [,...]} FROM konto_zabezpieczeń [,..]
```

Polecenie *REVOKE* dla uprawnień dotyczących obiektów:

```
REVOKE
{ALL [PRIVILEGES] | uprawnienie [,...]}
{{{{kolumna [, ...]}}} ON {tabela | perspektywa}
|{procedura_przechowywana | procedura_rozszerzona}}
FROM konto_zabezpieczeń [, ...]
[AS {grupa | rola}]
```

Uprawnienia są przechowywane w postaci wpisów w tabeli systemowej *sysprotects*.

4.4. Przykład

W systemie istnieje baza danych *MAGAZYNY*, gromadząca dane o stanie magazynu. Użytkownik *kierownik* jest użytkownikiem systemu Windows NT, powinien posiadać wszystkie uprawnienia pozwalające na tworzenie tabel, przeglądanie, wstawianie, modyfikowanie danych w istniejących już tabelach w tej bazie. Przypisać użytkownika *kierownik* do roli *kier_role*. Stworzyć konto logowania serwera SQL *pracownik* o haśle *ppp34*. Nadać użytkownikowi uprawnienia umożliwiające przeglądanie tabeli *towary* (właścicielem jej jest użytkownik *kierownik*) i modyfikowanie kolumny *liczba*, określającej zapas danego towaru, natomiast bezwzględnie uniemożliwić mu modyfikację kolumny *suma*.

SQL Server ma ustawiony mieszany tryb uwierzytelniania.

Tworzenie konta logowania użytkownika kierownik systemu Windows NT:

```
exec sp_grantlogin 'ztint/kierownik'
```

Tworzenie konta logowania serwera SQL:

```
exec sp_addlogin 'pracownik', 'ppp34', 'MAGAZYNY'
```

Umożliwienie dostępu do bazy danych *MAGAZYNY* kontom *kierownik* i *pracownik*, baza *MAGAZYNY* to bieżąca baza danych:

```
Exec sp_grantdbaccess 'ztint/kierownik'
Exec sp_grantdbaccess 'pracownik'
```

Przypisanie użytkownikowi *kierownik* ról bazy danych w bieżącej bazie danych:

```
Exec sp_addrolemember db_ddladmin, 'ztint/kierownik'
Exec sp_addrolemember db_datawriter, 'ztint/kierownik'
Exec sp_addrolemember db_datareader. 'ztint/kierownik'
```

Przypisanie użytkownikowi pracownik uprawnień do tabeli towary:

```
Grant  select, update (liczba) on ztint/kierownik.towary to pracownik
```

Odmowa dostępu do kolumny *suma* w tabeli *towary*:

```
Deny update (suma) on ztint/kierownik.towary to pracownik.
```

5. Perspektywy we wszystkich systemach

Perspektywy w bazach danych tworzone są w celu upraszczania dostępu do danych, bardziej szczegółowego zarządzania prawami (w odniesieniu do poszczególnych kolumn czy nawet wierszy) oraz zwiększania bezpieczeństwa danych. We wszystkich SZBD perspektywy tworzone są poleceniem *CREATE VIEW*:

```
CREATE VIEW <nazwa_perspektywy> [<kolumny_perspektywy>] AS SELECT
[WITH CHECK OPTION]
```

Perspektywy są używane do wyświetlania części jednej lub więcej tabel, do wstawiania, aktualizacji i usuwania wierszy w tabelach. Dostarczają dodatkowego poziomu ochrony przez to, że umożliwiają zdefiniowanie fragmentu danych w bazie danych, zawierającego tylko informacje potrzebne danemu użytkownikowi.

Poprzez nadanie użytkownikom odpowiednich uprawnień do tak stworzonej perspektywy można w bardziej szczegółowy sposób sterować dostępem do danych, udostępniając im np. tylko dane wyodrębnione w perspektywie.

Oprócz używania perspektyw w zapytaniach można ich używać w instrukcjach *INSERT*, *UPDATE* i *DELETE*. Nałożone są tu jednak pewne ograniczenia: aby za pomocą perspektywy można było modyfikować dane, zapytanie definiujące perspektywę powinno być oparte tylko na jednej tabeli, nie może zawierać między innymi: klauzuli *GROUP BY*, *DISTINCT*, funkcji agregujących.

Opcję *WITH CHECK OPTION* stosuje się wtedy, gdy chce się być pewnym, że wykonanie wstawienia (*INSERT*) i aktualizacji (*UPDATE*) nie spowoduje utworzenia wierszy, które nie zostaną wybrane za pomocą perspektywy.

Perspektywy umożliwiają zarządzanie uprawnieniami dotyczącymi tylko perspektywy, a nie uprawnieniami dotyczącymi obiektów, do których się odnoszą.

6. Podsumowanie

W pracy przedstawiono przegląd mechanizmów metod dostępu i tworzenia schematów bezpieczeństwa bazy danych jako jednego ze sposobów zabezpieczenia danych, wybranych systemów zarządzania bazami danych. Na podstawie przedstawionych materiałów można stwierdzić, że współczesne SZBD dostarczają szereg metod dostępu do przetwarzanych i przechowywanych danych.

W zależności od potrzeb i zasobności użytkownik może zdecydować się na wybór konkretnego serwera. Serwer SQLBase ma zaimplementowane stosunkowo prosto mechanizmy ochrony, sam proces przyznawania i odbierania poziomów uprawnień nie jest tak bardzo złożony i skomplikowany (3 poziomy uprawnień + konto systemowe SYSADM) jak w serwerze Oracle czy MS SQL Serverze.

Serwer Oracle cieszy się opinią jednego z najbardziej bezpiecznych i stabilnych systemów. Pociąga to za sobą potężne i bardzo rozbudowane mechanizmy ochrony i kontroli dostępu do danych (uprawnienia systemowe, mechanizm ról, profile, zabezpieczenia haseł itd.). Praca z serwerem Oracle, a zwłaszcza administracja wymaga gruntownej wiedzy na ten temat i doświadczenia, gdyż jest znacznie bardziej skomplikowana niż w pozostałych systemach. Jednakże właściwe wykorzystanie możliwości tego serwera zapewnia stworzenie bezpiecznej, stabilnej bazy danych z pełną ochroną przechowywanych i przetwarzanych w niej danych. W przeciwieństwie do MS SQL Servera, który wymaga systemu Windows NT, może być instalowany na różnych platformach sprzętowych i systemach operacyjnych. Z tego względu często jest wykorzystywany w firmach, placówkach, które pracują w środowiskach Unixowych.

Zwykle serwer ten jest wykorzystywany przez duże, dobrze rozwinięte i bogate korporacje (z uwagi na swą cenę), firmy przetwarzające olbrzymie ilości informacji, nie jest natomiast stosowany w średnich i małych firmach, gdzie wystarczającym i znacznie prostszym narzędziem jest serwer SQLBase (np. poradnie lekarskie czy stomatologiczne).

Natomiast MS SQL Server jest wygodnym, obszernym narzędziem, dostarczającym wiele potężnych mechanizmów ochrony dla zwolenników pracy w środowisku Windows i produktów firmy Microsoft.

LITERATURA

1. SQLBase, SQL Talk 7.5.1. Centura Software Corporation, 1999.
2. Greene J.: ORACLE8 Server Księga Eksperta. Helion, Gliwice 2000.
3. Wrembel R., Jezierski J., Zakrzewicz M.: System zarządzania bazą danych Oracle7 i Oracle8. Nakom, Poznań 2000.
4. Małysiak B.: Mechanizmy ochrony danych w wielodostępnym systemie zarządzania bazami danych Oracle. Informatyka, z. 36, Gliwice 1999.
5. Austin D.: Poznaj Oracle8. Mikom, Warszawa 1999.
6. Rodgers U.: ORACLE przewodnik projektanta baz danych. WNT, Warszawa 1995.
7. ORACLE7 Server Concepts Manual. Oracle Corporation, 1992.
8. ORACLE7 Server Administrator's Guide. Oracle Corporation.

9. Beynon-Davies P.: Systemy baz danych. WNT, Warszawa 1998.
10. Ullman J. D.: Database and knowledge–base systems. Computer Science Press, 1988.
11. Gnybek J.: Oracle łatwiejszy niż przypuszczasz. Helion, Gliwice 1996.
12. Microsoft SQL Server 7.0, Administracja systemu.
13. Microsoft SQL Server 7.0 Implementowanie baz danych.

Recenzent: Dr inż. Krystian Żymełka

Wpłynęło do Redakcji 31 marca 2001 r.

Abstract

Applications using databases process much information. Necessity of protection these data in a database is one of the main reasons for using security mechanisms in database management systems. The main part of creating an information system is a suitable protection of data stored there.

The primary purpose of database security is to selectively allow and disallow users to access objects for which they are authorized.

This article provides an overview of the data security mechanisms in database management systems (DBMS's) such as Oracle, Gupta Centura and MS SQL Server. It describes methods of users creating, granting and revoking privileges.

In the chapters author explains how, using privileges and roles, an administrator can control the ability to execute system operations, and control access to different objects in database in different database management systems.