Instrukcje sterowania dostępem do danych

Każda DBMS musi zawierać mechanizm gwarantujący dostęp do baz danych tylko dla tych użytkowników, którzy mają na to zezwolenie. Mechanizm ten w języku SQL oparty jest o instrukcje GRANT oraz REVOKE oraz identyfikatory użytkowników, ról i przywilejów.

Identyfikatorem użytkownika jest nazwą, pod którą jest on pamiętany w bazie danych. Użytkowników rejestruje administrator bazy danych (DBA) wraz z hasłem, które muszą podać, aby móc korzystać z bazy danych.
Każde polecenie SQL wykonywane przez użytkowników jest sygnowane jego identyfikatorem. Co więcej, każdy obiekt bazy danych stworzony w środowisku SQL ma właściciela, który rozpoznawany jest po identyfikatorze.

Przywileje związane są z zestawem akcji, które dany użytkownik może wykonywać. SQL:2003 dostarcza kontrolowanego dostępu do 9 funkcji zarządzania bazą danych. Są to:

Polecenia DML (dotyczą tabel i widoków):
INSERT, SELECT, UPDATE oraz DELETE.

Odsyłacze (dotyczą tabel):
REFERENCES pozwala na wprowadzenie ograniczeń związanych z tabelą, która zależy od innej tabeli w bazie danych.

Użycie dziedziny:
USAGE dotyczy użycia dziedziny, zbiorów znaków, collations, oraz translations.

Definiowanie typów danych:
UNDER pozwala wprowadzać ograniczenia dotyczące typów użytkownika.

Reakcja na zdarzenia:
TRIGGER wiąże się z wykonaniem zadeklarowanych wyrażeń SQL lub bloku wyrażeń w chwili, kiedy nastąpi predefiniowane zdarzenie.

Wykonanie:
EXECUTE powoduje wykonanie procedury zapisanej.

W większości systemów zarządzania bazami danych zdefiniowani są użytkownicy, którzy mają prawa do administrowania bazą danych (*database administrator (DBA)*. Użytkownicy tacy mogą w bazie danych „zrobić wszystko”.

Oprócz administratorów mogą również być zdefiniowani użytkownicy o nieco mniejszych uprawnieniach. Takimi użytkownikami są właściciele obiektów bazy danych (*database object owners)*.

Obiektami bazy danych są np. tabele, widoki. Każdy użytkownik, który utworzył obiekt w bazie danych może zadeklarować jego właściciela. Właściciel tablicy może korzystać ze wszystkich uprawnień z nią związanych, m.in. z możliwości udzielania prawa dostępu do tabeli innym użytkownikom.

Ponieważ widoki tworzone są na bazie tabel, prawa dostępu do widoków związane są z prawami dostępu do tabel. W przypadku, gdy jakiś użytkownik utworzył widok na tabeli, której nie jest właścicielem, otrzyma on do widoku takie same prawa, jakie miał on do wyjściowej tabeli. W konsekwencji, tworzenie widoków bazujących na tabelach obcych nie jest metodą pozwalającą na przeskoczenie ograniczonych uprawnień dostępu do tabel.

Przy dostępie do bazy danych użytkownik zobowiązany jest podać swoją nazwę (login) i hasło (password). Od chwili uzyskania dostępu do bazy danych użytkownik nabiera takich praw, jakie zostały mu przypisane.

W większości przypadków ZSBD posiadają grupę (typ) PUBLIC, która służy do określania użytkowników bez specjalnych uprawnień dostępu do danych.

Jeśli użytkownik z wyższymi uprawnieniami zdefiniuje pewne prawa dostępu jako PUBLIC, wtedy każdy, kto ma dostęp do systemu będzie miał prawa dostępu do danych.

Hierarchia praw dostępu

Ustanawianie praw dostępu dla użytkowników

Administratorzy bazy danych posiadają wszystkie prawa do obiektów bazy danych (a więc i do samej bazy danych, która jest obiektem). Nikt inny nie ma przywilejów do obiektów, za wyjątkiem tych użytkowników, którym prawa te zostały przyznane.

Prawa przyznawane są poleceniem GRANT o składni:

GRANT *privilege-list* ON *object* TO *user-list* [WITH GRANT OPTION] ;

gdzie:
privilege-list:
privilege [, *privilege*] ... lub ALL PRIVILEGES
privilege:
SELECT | DELETE | INSERT [(*column-name*[, *column-name*]...)]
| UPDATE [(*column-name*[, *column-name*]...)]
| REFERENCES [(*column-name*[, *column-name*]...)]
| USAGE | UNDER | TRIGGER | EXECUTE
object:
[TABLE] <table name>| DOMAIN <domain name>
| COLLATION <collation name>
| CHARACTER SET <character set name>
| TRANSLATION <transliteration name>
| TYPE <schema-resolved user-defined type name>
| SEQUENCE <sequence generator name>
| <specific routine designator>
user-list:
login-ID [, *login-ID*]...| PUBLIC

Role

Nazwa użytkownika w ogólności służy do identyfikacji użytkownika (bądź programu), a co za tym idzie, do określenia zestawu przywilejów, jakie zostały mu przyznane. Ponieważ przyznawanie uprawnień pojedynczym użytkownikom może okazać się zbyt uciążliwe i pracochłonne, w SQL:2003 wprowadzono pojęcie roli.

Rola jest zbiorem zero lub wielu uprawnień, które mogą być przyznane wielu użytkownikom na raz. Użytkownicy, którym przyznano pewną rolę, mają te same uprawnienia do wykonywania operacji na bazie danych.

Role mają własne nazwy. Niestety, nie we wszystkich bazach danych zaimplementowano role.

Role tworzy się poleceniem CREATE, jak np.:

```
CREATE ROLE Sprzedawca;
```

Po utworzeniu, rolę przypisać można użytkownikowi poleceniem GRANT:

```
GRANT Sprzedawca TO JakiśUżytkownik;
```

Dla roli można przypisywać prawa dostępu (zezwolenia) w ten sam sposób, w jaki robi się to dla użytkowników.

Zagwarantowanie Sprzedawcy praw do wstawiania nowych wierszy w tabeli Tabela:

```
GRANT INSERT ON Tabela TO Sprzedawca
```

Zagwarantowanie wszystkim użytkownikom mającym dostęp do bazy danych praw do przeglądania tabeli Tabela:

```
GRANT SELECT ON Tabela TO PUBLIC
```

Aby ograniczyć dostep do niektórych tylko kolumn tabeli można zdefiniować widok bez tych kolumn. Następnie można udzielić zezwoleń użytkownikom do przeglądania tego właśnie widoku.

Zagwarantowanie praw do modyfikacji kolumny w tabeli:

```
GRANT UPDATE Kolumna ON Tabela TO Sprzedawca;
```

Zagwarantowanie praw do uaktualniania wszystkich kolumn w tabeli:

```
GRANT UPDATE ON Tabela TO Sprzedawca;
```

Zagwarantowanie praw do usuwania wszystkich kolumn w tabeli:

```
GRANT DELETE ON Tabela TO Sprzedawca;
```

Oprócz Sprzedawcy do tabeli mają również prawa administratorzy bazy danych oraz jej właściciel.

Zagwarantowanie praw dostępu do tabeli związanej przez klucz obcy: niech będą dwie tabele: Tabela1 (zawierająca klucz obcy wierszID, który jest kluczem głównym w Tabeli2) oraz Tabela2. Polecenie tworzące Tabelę1:

```
CREATE TABLE Tabela1( wierszID INTEGER REFERENCES Tabela2);
```

Aby użytkownicy z zagwarantowanymi prawami do Tabeli1 nie mogli odwołać się do Tabeli2, należy użyć polecenia REFERENCES, przypisujące prawa do tej referencji np. roli Zarządca (SQL:2003)

```
GRANT REFERENCES (wierszID) ON Tabela2_TO Zarządca;
```

Zagwarantowanie praw używania domeny: niech będzie domena zadeklarowana jak niżej:

```
CREATE DOMAIN PriceTypeDomain DECIMAL (10,2)
CHECK (Price >= 0 AND Price <= 10000) ;

CREATE DOMAIN ProductCodeDomain CHAR (5)
CHECK (SUBSTR (VALUE, 1,1) IN ('X', 'C', 'H') AND
SUBSTR (VALUE, 5, 1) IN (9, 0) ) ;
```

Przy takiej deklaracji można utworzyć tabelę:

```
CREATE TABLE PRODUCT
(ProductCode ProductCodeDomain,
ProductName CHAR (30),
Price PriceTypeDomain) ;
```

w której wartości w kolumnach będą musiały spełniać nałożone przez dziedzinę ograniczenia. Jeśli ograniczenia te wynikają z logiki biznesowej, warto je chronić. Bo może zdarzyć się, że jakaś niepowołana osoba będzie, używając zdefiniowanej dziedziny, metodą prób i błędów (tj. tworząc nową tabelę z kolumnami o typach jak dziedziny wyżej i wstawiając w nią kolejne wartości aż do odrzucenia) ustali wartości ustalonych ograniczeń.
Do ochrony dziedziny stosuje się:

```
GRANT USAGE ON DOMAIN PriceType TO SalesMgr ;
```

Uwaga: przy usuwaniu dziedziny (DROP) mogą pojawić się problemy z tabelami ich używającymi.
Powyższe rozważania odnoszą się również do zbiorów znaków (characters sets), (collations), (translations).

Użycie procedur wyzwalanych

Wyzwalacz określa zdarzenie wyzwalające, chwilę zadziałania wyzwalacza (tuż przed lub tuż po zdarzeniu), oraz jedną lub więcej wyzwalanych akcji. Jeśli więcej niż jedno wyrażenie SQL jest wyzwalane, wszystkie one muszą być ujęte w blok BEGIN ATOMIC ... END.
Zdarzeniem wyzwalającym może być wyrażenie INSERT, UPDATE, DELETE (na przykład przed uaktualnieniem tabeli instrukcją UPDATE wyzwalacz może sprawdzić, czy nowe dane są poprawne).
Aby utworzyć wyzwalacz, użytkownik albo rola musi posiadać zezwolenie TRIGGER.
Wtedy wyzwalacz może zostać utworzony jak niżej:

```
CREATE TRIGGER CustomerDelete BEFORE DELETE
ON CUSTOMER FOR EACH ROW
WHEN State = NY
INSERT INTO CUSTLOG VALUES ('deleted a NY customer') :
```

Prawo do przyznawania zezwoleń mogą być przekazane kolejnym użytkownikom za pomocą polecenia WITH GRANT OPTION

```
GRANT UPDATE (Kolumna)
ON Tabele
TO Sprzedawca
WITH GRANT OPTION ;
```

Usuwanie uprawnień

Odbywa się przez wywołanie:

```
REVOKE [GRANT OPTION FOR] privilege-list
ON object
FROM user-list [RESTRICT|CASCADE] ;

gdzie:
privilege-list:
        privilege [, privilege] ... lub  ALL PRIVILEGES
privilege:
        SELECT | DELETE | INSERT [(column-name[, column-name]...)]
      | UPDATE [(column-name[, column-name]...)]
      | REFERENCES [(column-name[, column-name]...)]
      | USAGE | UNDER | TRIGGER | EXECUTE
object:
       [ TABLE ] <table name>| DOMAIN <domain name>
      | COLLATION <collation name>
      | CHARACTER SET <character set name>
      | TRANSLATION <transliteration name>
      | TYPE <schema-resolved user-defined type name>
      | SEQUENCE <sequence generator name>
      | <specific routine designator>
user-list:
      login-ID [, login-ID]...| PUBLIC
```

Struktury powyższej używa się w przypadku, gdy usuwane są niektóre tylko uprawnienia. Podstawową różnicą pomiędzy REVOKE a GRANT jest obecność opcjonalnego słowa kluczowego RESTRICT lub CASCADE w wyrażeniu REVOKE. Jeśli podczas przyznawania uprawnień użyto WITH GRANT OPTION, to podczas usuwania uprawnień poprzez REVOKE opcja CASCADE powoduje, że oprócz usunięcia uprawnień danej osoby usunięte zostaną również uprawnienia, które ta osoba nadała innym użytkownikom. Z drugiej strony, użycie REVOKE z opcją RESTRICT zadziała tylko w przypadku, gdy osoba mogąca przyznawać uprawnienia nikomu nie przekazała wyspecyfikowanych uprawnień. Jeśli tak jest w istocie, usuwane są uprawnienia danej osobie. Jeśli jednak jakieś uprawnienia zostały przekazane, REVOKE z opcją RESTRICT nic nie wycofa, a na dodatek zwróci kod błędu.

Można użyć wyrażenie REVOKE z opcjonalną klauzulą GRANT OPTION FOR aby usunąć tylko przyznane przez daną osobę uprawnienia, zachowując przy tym uprawnienia przyznane samej osobie. Jeśli klauzula GRANT OPTION FOR występuje razem ze słowem CASCADE, wtedy usuwane są wszystkie uprawnienia, które nadane zostały przez daną osobę razem z możliwością nadawania uprawnień przez tą osobę – jest to tak, jakby nigdy nie przyznano uprawnień do nadawania uprawnień. Jeśli klauzula GRANT OPTION FOR jest obecna razem z klauzulą RESTRICT, jedna z dwóch rzeczy może się zdarzyć: jeśli dana osoba nie udzieliła uprawnień usuwanych żadnej innej osobie, wtedy REVOKE zadziała i usunie możliwość nadawania uprawnień danej osobie. Jeśli dana osoba już udzieliła usuwane uprawnienia przynajmniej jednej innej osobie, to REVOKE nie zadziała i zwróci kod błędu.

The fact that you can grant privileges by using WITH GRANT OPTION, combined with the fact that you can also selectively revoke privileges, makes system security much more complex

than it appears at first glance. Multiple grantors, for example, can conceivably grant a privilege to any single user. If one of those grantors then revokes the privilege, the user still retains that privilege because of the still-existing grant from another grantor. If a privilege passes from one user to another by way of the WITH GRANT OPTION, this situation creates a *chain of dependency,* in which one user's privileges depend on those of another user. If you're a DBA or object owner, always be aware that, after you grant a privilege by using the WITH GRANT OPTION clause, that privilege may show up in unexpected places. Revoking the privilege from unwanted users while letting legitimate users retain the same privilege may prove challenging. In general, the GRANT OPTION and CASCADE clauses encompass numerous subtleties. If you use these clauses, check both the SQL:2003 standard and your product documentation carefully to ensure that you understand how the clauses work.

You can use this structure to revoke specified privileges while leaving othersintact. The principal difference between the REVOKEstatement and the GRANTstatement is the presence of the optional RESTRICTor CASCADEkeyword inthe REVOKEstatement. If you used WITH GRANT OPTIONto grant the privi-leges you're revoking, using CASCADEin the REVOKEstatement revokes privi-leges for the grantee and also for anyone to whom that person granted thoseprivileges as a result of the WITH GRANT OPTIONclause. On the other hand,the REVOKEstatement with the RESTRICToption works only if the granteehasn't delegated the specified privileges. In the latter case, the REVOKEstate-ment revokes the grantee's privileges. If the grantee passed on the specifiedprivileges, the REVOKEstatement with the RESTRICToption doesn't revokeanything and instead returns an error code.You can use a REVOKEstatement with the optional GRANT OPTION FORclauseto revoke only the grant option for specified privileges while enabling thegrantee to retain those privileges for himself. If the GRANT OPTION FORclause and the CASCADEkeyword are both present, you revoke all privilegesthat the grantee granted, along with the grantee's right to bestow such privi-leges — as if you'd never granted the grant option in the first place. If theGRANT OPTION FORclause and the RESTRICTclause are both present, one oftwo things happens:_If the grantee didn't grant to anyone else any of the privileges you'rerevoking, then the REVOKEstatement executes and removes thegrantee's ability to grant privileges._If the grantee has already granted at least one of the privileges you'rerevoking, the REVOKEdoesn't execute and returns an error code instead.The fact that you can grant privileges by using WITH GRANT OPTION, com-bined with the fact that you can also selectively revoke privileges, makessystem security much more complex than it appears at first glance. Multiplegrantors, for example, can conceivably grant a privilege to any single user. Ifone of those grantors then revokes the privilege, the user still retains thatprivilege because of the still existing grant from another grantor. If a privilegepasses from one user to another by way of the WITH GRANT OPTION, this sit-uation creates a *chain of dependency,*in which one user's privileges dependon those of another user. If you're a DBA or object owner, always be awarethat, after you grant a privilege by using the WITH GRANT OPTION clause,that privilege may show up in unexpected places. Revoking the privilege fromunwanted users while letting legitimate users retain the same privilege mayprove challenging. In general, the GRANT OPTIONand CASCADEclausesencompass numerous subtleties. If you use these clauses, check both theSQL:2003 standard and your product documentation carefully to ensure thatyou understand how the clauses work