Система безопасности SQL Server

SQL Server обеспечивает защиту данных от неавторизированного доступа и от фальсификации. Основными функциями безопасности SQL Server являются:

- **проверка подлинности (аутентификация)** – процедура проверки соответствия некоего лица и его учетной записи в компьютерной системе. Один из способов аутентификации состоит в задании пользовательского идентификатора, в просторечии называемого «логином» (login – регистрационное имя пользователя) и пароля – некой конфиденциальной информации, знание которой обеспечивает владение определенным ресурсом. Аутентификацию не следует путать с идентификацией. Идентификация – это установление личности самого физического лица (а не его виртуальной учетной записи, коих может быть много).

- **авторизация** – это предоставление лицу прав на какие-то действия в системе.

Дополнительными функциями безопасности SQL Server являются:
- шифрование,
- контекстное переключение,
- олицетворение,
- встроенные средства управления ключами.

В основе системы безопасности SQL Server лежат три понятия:
- участники системы безопасности (Principals);
- защищаемые объекты (Securables) и
- система разрешений (Permissions).

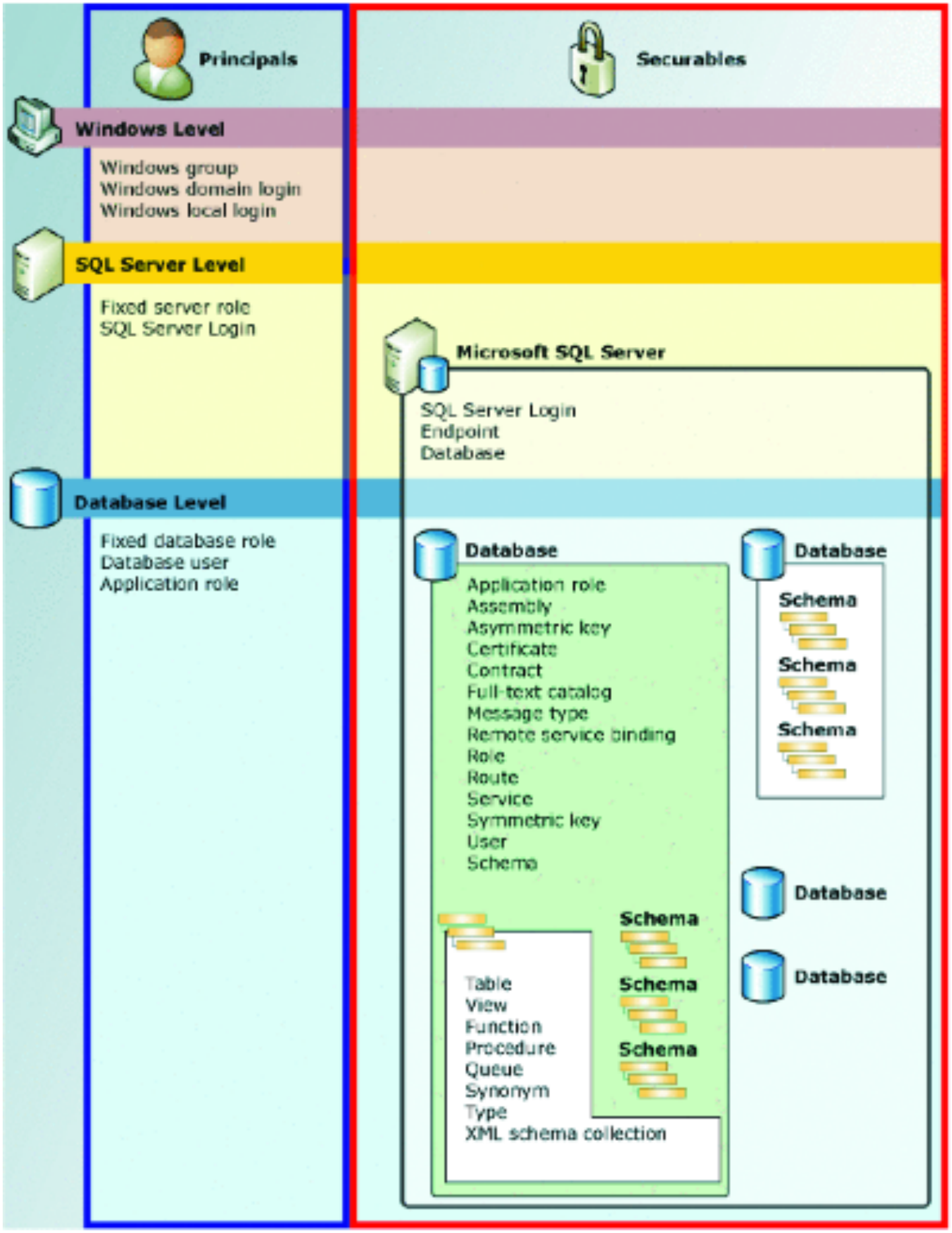
Участники системы безопасности

Участники системы безопасности или принципалы – это сущности, которые могут запрашивать ресурсы SQL Server. Принципалы могут быть иерархически упорядочены. Область влияния принципала зависит
- от области его определения: Windows, SQL Server, база данных,
- от того, коллективный это участник или индивидуальный. Имя входа Windows является примером индивидуального (неделимого) участника, а группа Windows — коллективного.

При создании учетной записи SQL Server ей назначается идентификатор и идентификатор безопасности. В представлении каталога sys.server_principals они отображаются в столбцах principal_id и SID.
Участники уровня Windows – это

A. Имя входа домена Windows,

B. Локальное имя входа Windows. Участники уровня SQL Server – это a) Имя входа SQL Server, b) Роль сервера.
   Участники уровня базы данных – это a) Пользователь базы данных, b) Роль базы данных, c) Роль приложения.


Principals
Securables
Windows Level
Windows group
Windows domain login
Windows local login
SQL Server Level
Fixed server role
SQL Server Login
Microsoft SQL Server
SQL Server Login
Endpoint
Database
Database Level
Fixed database role
Database user
Application role
Database
Application role
Assembly
Asymmetric key
Certificate
Contract
Full-text catalog
Message type
Remote service binding
Role
Route
Service
Symmetric key
User
Schema
Schema
Schema
Schema
Table
View
Function
Procedure
Queue
Synonym
Type
XML schema collection
Database
Schema
Schema
Schema
Database
Database

Замечания.

1. Имя входа «sa» SQL Server. Имя входа sa SQL Server является участником уровня сервера. Оно создается по умолчанию при установке экземпляра. В SQL Server для имени входа sa базой данных по умолчанию будет master.

2. Роль базы данных public. Каждый пользователь базы данных является членом роли базы данных public. Если пользователю не были предоставлены или запрещены особые разрешения на защищаемый объект, то он наследует на него разрешения роли public.

3. Пользователи INFORMATION_SCHEMA и sys. Каждая база данных включает в себя две сущности, которые отображены в представлениях каталога в виде пользователей: INFORMATION_SCHEMA и sys. Они необходимы для работы SQL Server; эти пользователи не являются участниками и не могут быть изменены или удалены.

Защищаемые объекты

Защищаемые объекты – это ресурсы, доступ к которым регулируется системой авторизации. Некоторые защищаемые объекты могут храниться внутри других, создавая иерархии «областей», которые сами могут защищаться. К областям защищаемых объектов относятся (a) сервер, (b) база данных и (c) схема.

Далее введем следующие ограничения:
1. для области «сервер» ограничимся рассмотрением вопросами управления учетными записями подключения (login),
2. для области «база данных» ограничимся рассмотрением вопросами управления
    1. учетными записями пользователей (user),
    2. ролями (role),
    3. ролями приложений (application role).
3. для области «схема» ограничимся рассмотрением вопросами управления разрешениями на
    1. функции (function),
    2. процедуры (stored procedure),
    3. таблицы (table),
    4. представления (view).

Разрешения

У субъекта системы есть только один путь получения доступа к объектам - иметь назначенные непосредственно или опосредовано разрешения. При непосредственном управлении разрешениями они назначаются субъекту явно, а при опосредованном разрешения назначаются через членство в группах, ролях или наследуются от объектов, лежащих выше по цепочке иерархии. Управление разрешениями производится путем выполнения инструкций языка DCL (Data Control Language): GRANT (разрешить), DENY (запретить) и REVOKE (отменить).

Управление учетными данными сервера (credential)

Поддерживаются два вида учетных записей подключения к серверу
- учетные записи сервера, создаваемые на основании учетных записей операционной системы
- учетные записи сервера, создаваемые для прямого подключения к серверу.

Проверка подлинности Windows означает, что для подключения к SQL Server проверка подлинности полностью выполняется операционной системой Microsoft Windows. В этом

случае клиент идентифицируется на основании учетной записи Windows. Проверка подлинности SQL Server означает, что для подключения к SQL Server проверка подлинности выполняется путем сравнения имени пользователя и пароля с хранящимся на сервере SQL Server списком действительных имен пользователей и паролей). Учетные данные сервера создаются при помощи инструкции CREATE CREDENTIAL. После создания учетных данных можно сопоставить их имени входа SQL Server, используя инструкцию CREATE LOGIN или ALTER LOGIN.

Управление именами входа SQL Server (login)

Сами имена входа не имеют доступа ни к одной конкретной базе данных на сервере, они позволяют только подключиться к SQL Server. Имена входа – это объекты, которым может быть дано разрешение в масштабе сервера на выполнение определенных действий. Эти действия собираются в фиксированные серверные роли: bulkadmin, dbcreator, diskadmin, processadmin, public, securityadmin, serveradmin, setupadmin, sysadmin. Все имена входа SQL Server являются обладателями роли public. Добавление имени входа в качестве члена предопределенной роли сервера выполняется при помощи хранимой процедуры sp_addsrvrolemember. Имя входа сервера может быть создано в SSMS, при помощи инструкции CREATE LOGIN или при помощи хранимой процедуры sp_addlogin.

Замечание. В SQL Server 2012 при помощи инструкции CREATE SERVER ROLE можно создать новую, определяемую пользователем роль сервера, а для изменения членства в роли сервера можно использовать инструкцию ALTER SERVER ROLE.

Примеры.

```
--Creating a new SQL login
CREATE LOGIN Carol
WITH PASSWORD = 'Th1sI$lVlyP@ssw0rd'; GO

--Creating a credential based on a Windows user
CREATE CREDENTIAL StreetCred
WITH IDENTITY = 'AughtFive\CarolStreet', SECRET = 'P@ssw0rd';
--Associating a login with a credential
ALTER LOGIN Carol WITH CREDENTIAL = StreetCred; GO

--Creating a login, and adding the user to a fixed server role
CREATE LOGIN Ted WITH PASSWORD = 'P@ssw0rd'; GO
EXEC sp_addsrvrolemember 'Ted', 'securityadmin'; GO
```

Управление учетными записями пользователей базы данных (user)

После создания имени входа в SQL Server можно предоставить этому имени доступ к конкретной базе данных. Для этого сначала надо создать пользователя базы данных для определенного ранее имени входа. Это можно сделать в SSMS, при помощи инструкции CREATE USER или при помощи хранимой процедуры sp_adduser. При входе в SQL Server под именем входа база данных запросит имя и идентификатор созданного пользователя базы данных. Если при создании пользователя базы данных не будет указано имя входа SQL Server, то новый пользователь базы данных будет сопоставлен с именем входа SQL Server, имеющим такое же имя. После создания пользователя базы данных можно (но это необязательно) предоставить ему одну из фиксированных ролей базы данных: db_accessadmin, db_backupoperator, db_datareader, db_datawriter, db_ddladmin, db_denydatareader,

db_denydatawriter, db_owner, db_securityadmin. При желании можно определить дополнительные роли базы данных при помощи инструкции CREATE ROLE. Для добавления пользователя базы данных к роли текущей базы данных используется инструкция ALTER ROLE или хранимая процедура sp_addrolemember.

Примеры

```
-- Managing Server RoleMembers
CREATE LOGIN Veronica
WITH PASSWORD = 'PalmTree1' GO
EXEC master..sp_addsrvrolemember 'Veronica', 'sysadmin' GO
USE TestDB
GO

CREATE USER Veronica
GO

-- Т. к. предложение FOR LOGIN не указано, то новый пользователь базы данных будет ----
сопоставлен с именем входа SQL Server, имеющим такое же имя.
CREATE ROLE HR_ReportSpecialist AUTHORIZATION db_owner GO
EXEC sp_addrolemember 'HR_ReportSpecialist', 'Veronica' GO
```

Замечание. В SQL Server 2012 многие системные хранимые процедуры, используемые для управления безопасностью, считаются устаревшими, но остаются доступными для обратной совместимости. Для устаревших процедур рекомендуется использовать инструкции Transact-SQL, например, ALTER LOGIN, ALTER SERVER ROLE, ALTER USER, ALTER ROLE и др.

Управление разрешениями

После добавления пользователя следует определить разрешения, управляющие действиями, которые пользователь может выполнять, с помощью инструкций GRANT, DENY и REVOKE. Необходимо, чтобы пользователь был владельцем базы данных.

А. Предоставление разрешений на объекты (инструкция GRANT)

Инструкция GRANT предоставляет разрешения на таблицу, представление, функцию, хранимую процедуру, очередь обслуживания, синоним. Синтаксис инструкции GRANT:

```
GRANT { ALL [ PRIVILEGES ] | список_разрешений } ON список_объектов
TO список_принципалов
[ WITH GRANT OPTION ]
[ AS принципал ]
```

Пояснения к инструкции GRANT

1. Ключевое слово ALL с необязательным словом RIVILEGES не включает все возможные разрешения, оно эквивалентно предоставлению всех разрешений ANSI-92, применимых к указанному объекту. Значение ALL различается для разных типов объектов. Ниже перечислены главные классы разрешений и защищаемых объектов, к которым эти разрешения могут применяться:
    1. разрешения на скалярные функции: EXECUTE, REFERENCES;

2. разрешения на функции, возвращающие табличное значение: DELETE, INSERT, REFERENCES, SELECT, UPDATE;
3. разрешения на хранимые процедуры: EXECUTE;
4. разрешения на таблицы: DELETE, INSERT, REFERENCES, SELECT, UPDATE;
5. разрешения на представления: DELETE, INSERT, REFERENCES, SELECT, UPDATE.

2. Полный список разрешений содержит 195 пунктов.
    1. Если разрешение предоставляется на таблицу, представление или функцию, возвращающую табличное значение, то справа от разрешения в круглых скобоках могут указываться имена столбцов. На столбец могут быть предоставлены только разрешения SELECT, REFERENCES и UPDATE.
    2. Объект, на который предоставляется разрешение, имеет следующее описание: [OBJECT ::] [имя_схемы]. имя_объекта. Фраза OBJECT необязательна, если указан аргумент имя_схемы. Если же она указана, указание квалификатора области (::) обязательно. Если не указан аргумент имя_схемы, подразумевается схема по умолчанию. Если указан аргумент имя_схемы, обязательно указание квалификатора области схемы (.).

3. Принципалом может быть:
    1. пользователь базы данных,
    2. роль базы данных,
    3. роль приложения.

4. Необязательная фраза WITH GRANT OPTION указывает, что принципалу также дается возможность предоставлять указанное разрешение другим принципалам.

5. Необязательная фраза AS принципал определяет принципала, у которого другой принципал, выполняющий данный запрос, наследует право предоставлять данное разрешение.

Пример. Предоставление разрешения EXECUTE на хранимую процедуру HumanResources.uspUpdateEmployeeHireInfo роли приложения Role03.

```
USE AdventureWorks;
GRANT EXECUTE ON OBJECT::HumanResources.uspUpdateEmployeeHireInfo TO Role03;
GO
```

Пример. Предоставление разрешения REFERENCES на столбец EmployeeID в представлении HumanResources.vEmployee пользователю User02 с параметром GRANT OPTION.

```
USE AdventureWorks;
GRANT REFERENCES (EmployeeID) ON OBJECT::HumanResources.vEmployee TO User02
WITH GRANT OPTION;
GO
```

B. Отмена разрешений на объекты (инструкция REVOKE)

Инструкция REVOKE отменяет разрешения, ранее предоставленные инструкцией GRANT. Синтаксис инструкции REVOKE:

```
REVOKE [ GRANT OPTION FOR ] список_разрешений ON список_объектов
{ FROM | TO } список_принципалов
[ CASCADE ]
[ AS принципал ]
```

Пояснения к инструкции REVOKE.

1. Необязательная фраза GRANT OPTION FOR показывает, что право на предоставление заданного разрешения другим принципалам будет отменено. Само разрешение отменено не будет.
2. Необязательное ключевое слово CASCADE показывает, что отменяемое разрешение также отменяется для других принципалов, для которых оно было предоставлено или запрещено этим принципалом. Каскадная отмена разрешения, предоставленного с помощью параметра WITH GRANT OPTION, приведет к отмене прав GRANT и DENY для этого разрешения.
3. Необязательная фраза AS принципал указывает принципала, от которого принципал, выполняющий данный запрос, получает право на отмену разрешения.

Пример. Отмена разрешения EXECUTE для хранимой процедуры

```
USE AdventureWorks;
REVOKE EXECUTE ON OBJECT::HumanResources.uspUpdateEmployeeHireInfo FROM Role03;
GO
```

C. Запрет разрешений на объекты (инструкция DENY)

Инструкция DENY запрещает разрешения на члены класса OBJECT защищаемых объектов. Синтаксис инструкции DENY:

```
DENY список_разрешений ON список_объектов
TO список_принципалов
[ CASCADE ]
[ AS принципал ]
```

Пример. Запрет разрешения REFERENCES на представление с CASCADE

```
USE AdventureWorks;
DENY REFERENCES (EmployeeID) ON OBJECT::HumanResources.vEmployee TO User02 CASCADE; GO
```

Цепочки владения

Если несколько объектов базы данных последовательно обращаются друг к другу, то такая последовательность известна как цепочка. Такие цепочки не могут существовать независимо, но когда SQL Server проходит по звеньям цепи, то SQL Server проверяет разрешения составляющих объектов иначе, нежели при раздельном доступе к объектам. Эти различия имеют важные последствия для обеспечения безопасности. Цепочки владения позволяют управлять доступом к нескольким объектам, таким как таблицы, назначая разрешения одному объекту, например представлению. Цепочки владения также обеспечивают небольшое

повышение производительности в случаях, когда позволено пропускать проверку наличия разрешений. Проверка разрешений в цепи выполняется так:

1. SQL Server сначала сравнивает владельца вызываемого объекта с владельцем вызывающего объекта.

2. Если оба объекта имеют одного владельца, то разрешения для ссылаемого объекта не проверяются.

Пример цепочки владения

1. Алексей выполняет инструкцию SELECT * в представлении July2003. SQL Server проверяет разрешения в представлении и подтверждает, что Алексей имеет разрешение выбирать.

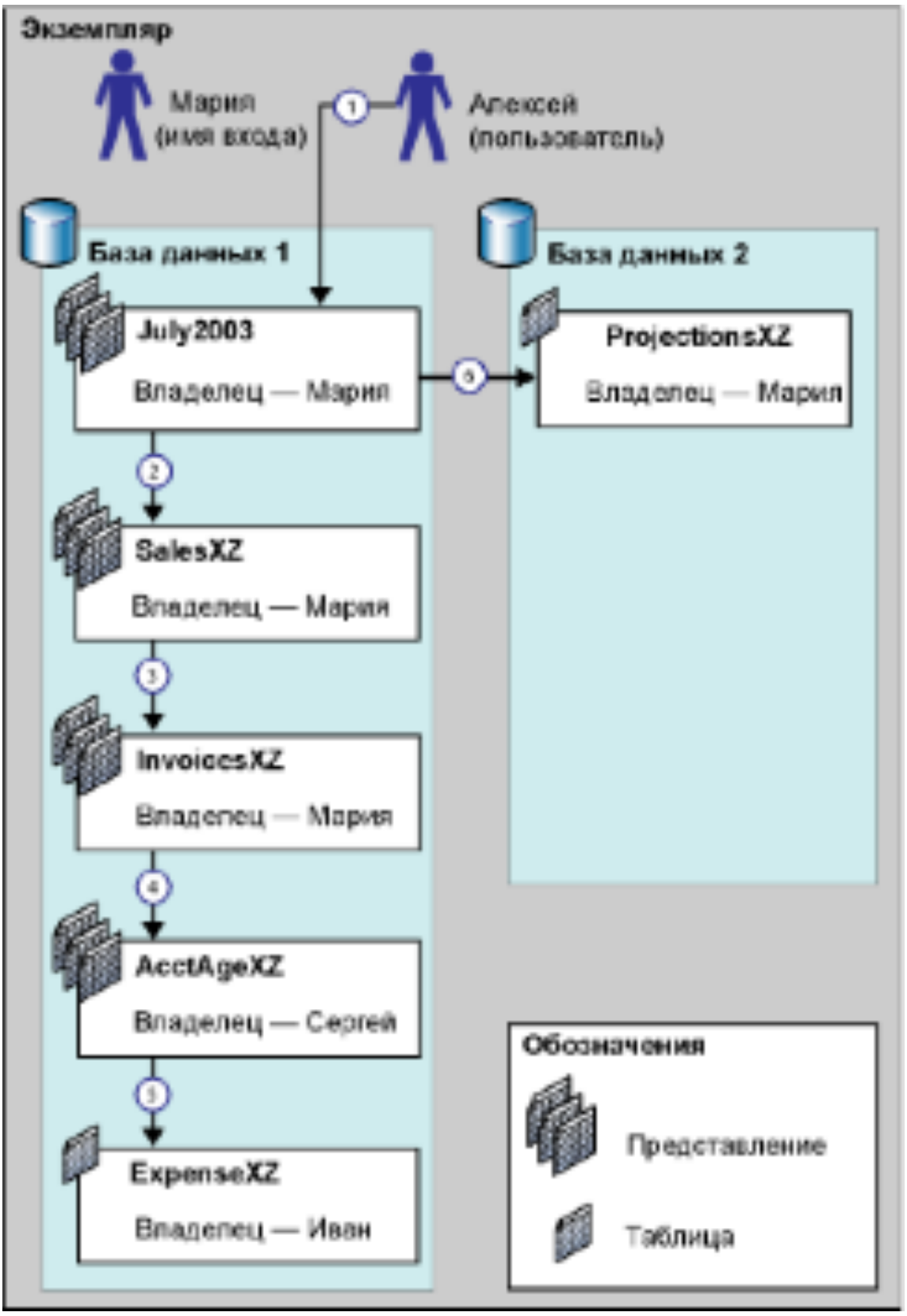
2. Представление July2003 требует данные из представления SalesXZ. SQL Server проверяет владение SalesXZ. Владелец этого представления (Мария) такой же, как у вызывающего представления, поэтому разрешения для SalesXZ не проверяются. Возвращаются необходимые данные.

3. Представление SalesXZ требует данные из представления InvoicesXZ. SQL Server проверяет владение представления SalesXZ. Владелец этого представления такой же, как у предшествующего объекта, поэтому разрешения для InvoicesXZ не проверяются. Возвращаются необходимые данные. До этого этапа все элементы последовательности имели одного владельца (Мария). Это известно как неразрывная цепочка владения.

4. Представление InvoicesXZ требует данные из представления AcctAgeXZ. SQL Server проверяет владение представления AcctAgeXZ. Владелец этого представления иной, чем у предшествующего объекта (Сергей, а не Мария), поэтому должны быть получены полные сведения о разрешениях на это представление. Если на представление AcctAgeXZ имеются разрешения, которые обеспечивают доступ со стороны пользователя Алексей, сведения будут возвращены.

5. Представление AcctAgeXZ требует данные из представления ExpenseXZ. SQL Server проверяет владение таблицы ExpenseXZ. Владелец этой таблицы иной, чем у предшествующего объекта (Иван, а не Сергей), поэтому должны быть получены полные сведения о разрешениях на эту таблицу. Если таблица ExpenseXZ имеет разрешения, которые обеспечивают доступ со стороны пользователя Алексей, сведения возвращаются.

6. Если представление July2003 пытается получить данные из таблицы ProjectionsXZ, то сервер сначала проверяет наличие цепочечных связей между базами данных Database 1 и Database 2. Если цепочечные связи между базами данных активны, то сервер проверяет владение для таблицы ProjectionsXZ. Владелец этой таблицы такой же, как у вызывающего представления (Мэрия), поэтому разрешения для этой таблицы не проверяются. Возвращаются необходимые данные.



По умолчанию межбазовые цепочки владения отключены. SQL Server можно настроить так, чтобы разрешить цепочку владения между конкретными базами данных или между всеми базами данных внутри одного экземпляра SQL Server.

Пример потенциальных опасностей при использовании цепочек владения

Включены межбазовые цепочки владения между базой данных A и базой данных B. В этом случае член предопределенной роли базы данных db_owner любой из этих баз данных может незаконно проникнуть в другую базу данных. Процедура выглядит так:

- Член предопределенной роли базы данных db_owner в базе данных A по имени X создает пользователя Y в базе данных A, который уже существует как пользователь в базе данных B.
- Затем X создает в базе данных A объект, владельцем которого является Y.
- Пользователь Y из базы данных A вызывает любой объект, принадлежащий пользователю Y в базе данных B.
- Вызывающий и вызываемый объекты имеют общего владельца, поэтому разрешения для объекта в базе данных B не будут проверяться, когда X обратится к ним через созданный ею объект. Контекст выполнения и переключение контекста

Контекст выполнения определяется подключенным к сеансу именем входа, пользователем или выполняющимся модулем. Он устанавливает идентификатор пользователя или имени входа, чьи разрешения на выполнение инструкций или совершение действий проверяются. В SQL Server контекст выполнения может переключаться на другое имя входа или пользователя при помощи выполнения инструкции EXECUTE AS или предложения EXECUTE AS в модуле. После переключения контекста SQL Server проверяет разрешения у имени входа и пользователя этой учетной записи, а не у того, кто вызвал инструкцию EXECUTE AS, или модуля. Имя входа или пользователь олицетворяется на оставшееся время выполнения сеанса или модуля либо до того момента, когда происходит явное восстановление переключения контекста (сеанс удаляется, контекст переключен на другое имя входа или на другого пользователя с помощью новой инструкции EXECUTE AS, контекст восстановлен до контекста предыдущего выполнения с помощью инструкции REVERT).

Синтаксис инструкции EXECUTE AS имеет вид:

EXECUTE AS {LOGIN | USER} = 'имя' [WITH {NO REVERT | COOKIE INTO @varbinary_переменная}] | CALLER

- LOGIN указывает, что область олицетворения — это уровень сервера.
- USER указывает, область олицетворения ограничена текущей базой данных.
- 'имя' - допустимое имя входа или имя пользователя.
- NO REVERT указывает, что переключение контекста нельзя вернуть к предыдущему контексту.
- COOKIE INTO @varbinary_переменная указывает, что контекст выполнения можно переключить к предыдущему контексту, если при вызове инструкция REVERT WITH COOKIE содержит правильное значение @varbinary_переменная. SQL Server передает куки-файл в @varbinary_переменная.
- CALLER указывает, что инструкции модуля выполняются в контексте вызывающей стороны. Вне модуля инструкция не действует.

Синтаксис предложения EXECUTE AS имеет вид:

EXECUTE AS { CALLER | SELF | OWNER | 'имя_пользователя' }

- EXECUTE AS CALLER указывает, что инструкции, содержащиеся в модуле, выполняются в контексте пользователя, вызывающего этот модуль.
- EXECUTE AS SELF указывает, что модуль выполняется от имени того пользователя, который последним модифицировал модуль.
- EXECUTE AS OWNER указывает, что инструкции, содержащиеся в модуле, выполняются в контексте текущего владельца этого модуля.
- EXECUTE AS 'имя_пользователя' указывает, что инструкции, содержащиеся в модуле, выполняются в контексте пользователя, указываемого аргументом 'имя_пользователя'.

Пример контекста выполнения

```
USE master
GO
-- Создаем имена входов
CREATE LOGIN User1 WITH PASSWORD='^*ahfn2@^(K' GO
CREATE LOGIN User2 WITH PASSWORD='*HABa7s7aas' GO
CREATE LOGIN User3 WITH PASSWORD='zxd837&^gqF' GO
CREATE DATABASE MyDB
GO
USE MyDB
GO
-- Создаем пользователей и схемы
CREATE USER User3 WITH DEFAULT_SCHEMA=User3
GO
CREATE SCHEMA User3 AUTHORIZATION User3
GO
CREATE USER User2 WITH DEFAULT_SCHEMA=User2
GO
CREATE SCHEMA User2 AUTHORIZATION User2
GO
CREATE USER User1 WITH DEFAULT_SCHEMA=User1
GO
CREATE SCHEMA User1 AUTHORIZATION User1
GO
-- Пользователь User3 имеет право создавать GRANT CREATE TABLE TO User3
GO
-- Пользователь User2 имеет право создавать GRANT CREATE PROC TO User2
GO
EXECUTE AS LOGIN='User3'
GO
CREATE TABLE User3.CustomerInformation
(

    CustomerName nvarchar(50)

)
```

```
GO
INSERT INTO CustomerInformation VALUES('Bryan''s Bowling Alley') INSERT INTO
CustomerInformation VALUES('Tammie''s Tavern') INSERT INTO CustomerInformation
VALUES('Frank''s Fresh Produce') GO
GRANT SELECT ON CustomerInformation TO User2
GO
REVERT
GO
EXECUTE AS LOGIN='User2'
GO
--create a stored proc that will return the rows in our table CREATE PROC ViewCustomerNames
AS

BEGIN
SELECT * FROM User3.CustomerInformation

END GO

GRANT EXECUTE ON ViewCustomerNames TO User1 GO
REVERT
GO

EXECUTE AS LOGIN='User1'
-- User1 cannot access table directly

SELECT * FROM User3.CustomerInformation

таблицу

процедуру

Msg 229, Level 14, State 5, Line 3
The SELECT permission was denied on the object 'CustomerInformation', database 'myDB',
schema 'User3'.

--User1 can execute the procedure but does not have permissions on the underlying table

EXEC User2.ViewCustomerNames

Msg 229, Level 14, State 5, Procedure ViewCustomerNames, Line 5
The SELECT permission was denied on the object 'CustomerInformation', database 'myDB',
schema 'User3'.

GO
REVERT
GO
EXECUTE AS LOGIN='User2'
GO
ALTER PROCEDURE ViewCustomerNames WITH EXECUTE AS OWNER AS

BEGIN
SELECT * FROM User3.CustomerInformation
```

```
END
GO
REVERT
GO
EXECUTE AS LOGIN='User1'
--User1 still cannot access table directly SELECT * from User3.CustomerInformation
```

```
Msg 229, Level 14, State 5, Line 1
The SELECT permission was denied on the object 'CustomerInformation', database 'myDB', schema 'User3'
```

```
--User1 can execute a procedure that uses the CustomerInformation table

EXEC User2.ViewCustomerNames GO
REVERT
GO
```

Аудит SQL Server

Начиная с версии SQL Server 2008 в редакции Enterprise вводится аудит SQL Server Audit – функциональность системы безопасности, которая может отслеживать практически любое действие с сервером или базой данных (выполняемое пользователями) и записывать эти действия в файловую систему или системный журнал Windows.

Система управления на основе политик

Одна из новых возможностей, появившаяся в SQL Server 2008, – система управления на основе политик (Policy-Based Management), которая позволяет создавать политики для обеспечения соответствия нормативам управления базой данных.

Система управления на основе политик позволяет администратору баз данных (АБД) создавать политики для управления объектами и экземплярами базы данных. Эти политики дают АБД возможность устанавливать правила создания и изменения объектов и их свойств. С помощью новой системы можно, например, создать политику уровня БД, запрещающую использование для БД свойства AutoShrink. Другой пример – политика, в соответствии с которой все имена табличных триггеров в таблице БД начинаются с tr_.

Система управления на основе политик предусматривает использование новых терминов и понятий. Основными из них являются:
1. Политика (Policy) – набор условий, определенных аспектами цели управления. Другими словами, политика — это набор правил для свойств БД или серверных объектов.
2. Цель управления (Target) – объект, управляемый данной системой. Сюда относятся такие объекты, как экземпляр БД, база данных, таблица, хранимая процедура, триггер, индекс.
3. Аспект (Facet)– свойство объекта (цели управления), которое используется системой управления на основе политик. Например, имя триггера или свойство базы данных AutoShrink.
4. Условие (Condition) – критерий для аспектов цели управления. Например, можно создать для факта условие, по которому все имена хранимых процедур в схеме «Banking» должны начинаться с bnk_.

Кроме того, политику можно связать с определенной категорией, что позволяет осуществлять управление набором политик, привязанных к той же самой категории. Политика может принадлежать только к одной категории.

Режим оценки политик

Существует несколько режимов оценки политик:
1. По запросу (On demand) – оценку политики запускает непосредственно администратор.
2. При изменении: запретить (On change: prevent) – для предотвращения нарушения политики используются триггеры DDL.
3. При изменении: только внесение в журнал (On change: log only) – для проверки политики при изменении используются уведомления о событии.
4. По расписанию (On schedule) – для проверки политики на нарушения используется задание агента SQL (SQL Agent).

Преимущества системы управления на основе политик

Система управления на основе политик позволяет АБД в полной мере контролировать процессы, происходящие в базе данных. Администратор получает возможность реализовать

принятые в компании политики на уровне БД. Политики, принятые только на бумаге, помогают определить основные принципы управления базой данных и могут служить прекрасным руководством к действию, но воплощать их в жизнь очень нелегко. Для обеспечения строгого соответствия БД принятым нормативом АБД приходится пристально следить за ее повседневным использованием и функционированием. Система управления на основе политик позволяет раз и навсегда выработать политики управления и быть уверенным в том, что они будут применяться постоянно и в полном объеме.

Шифрование

SQL Server поддерживает три механизма шифрования:
1. на основе сертификатов (стандарт X. 509v3),
2. на основе асимметричных ключей (алгоритм RSA),
3. на основе симметричных ключей (алгоритмы шифрования RC4, RC2, DES, AES).

Для работы с этими сущностями используются следующие операторы T-SQL:

CREATE/ALTER/DROP/BACKUP CERTIFICATE CREATE/ALTER/DROP/OPEN/CLOSE ASYMMETRIC KEY CREATE/ALTER/DROP/OPEN/CLOSE SYMMETRIC KEY

Детальное описание иерархической схемы шифрования данных в SQL Server 2005 см. в разделе «Encryption Hierarchy» MSDN.

Контекстное переключение

SQL Server позволяет задать контекст выполнения хранимых процедур и пользовательских функций с помощью выражения EXECUTE AS, помещенного в заголовок определения модуля. Данный механизм позволяет одному пользователю выполнять действия внутри модуля так, будто он аутентифицирован как другой пользователь. Параметр EXECUTE AS имеет четыре возможных значения:

- CALLER. Если программист указывает EXECUTE AS CALLER, операторы внутри модуля выполняются в контексте пользователя, вызвавшего процедуру. Поэтому пользователь, выполняющий процедуру, должен иметь соответствующие разрешения не только на запуск процедуры, но и на любые объекты базы данных, на которые она ссылается.

- USER. Если используется значение EXECUTE AS USER имя_пользователя, процедура выполняется в контексте того пользователя, чье имя указано в параметре. При выполнении процедуры SQL Server сначала проверяет, имеет ли пользователь разрешение EXECUTE на данную процедуру, затем проверяет разрешения на операторы внутри процедуры для пользователя, указанного в параметре EXECUTE AS USER. Для того чтобы иметь возможность указать AS имя_пользователя, необходимо иметь специальные разрешения (например, IMPERSONATE) или быть членом специальной роли (sysadmin или db_owner).

- SELF. EXECUTE AS SELF, аналогично EXECUTE AS USER имя_пользователя, где имя_пользователя является именем человека, создающего или изменяющего модуль. Система сохраняет идентификатор пользователя (UID) вместо самого значения SELF. Пользователь, указанный в параметре SELF, не обязательно должен быть владельцем объекта. На самом деле объекты в SQL Server 2005 не имеют владельцев, но можно думать о владельцах схем так, как будто они владеют всеми объектами в схемах.

- OWNER. Применение EXECUTE AS OWNER хорошо подходит в ситуациях, если пользователь создает хранимые процедуры и таблицы в ходе своей работы, тогда, переходя на другое место, он сможет передать их в собственность другому пользователю, будучи уверенным, что они всегда будут выполняться в контексте безопасности собственника.

Олицетворение

Можно настроить SQL Server и Windows таким образом, чтобы экземпляр SQL Server мог подключаться к другому экземпляру SQL Server в контексте пользователя Windows, прошедшего проверку подлинности. Это называется олицетворением или делегированием. При использовании делегирования экземпляр SQL Server, к которому подключается пользователь Windows с проверкой подлинности Windows, выполняет олицетворение этого пользователя при обмене данными с другим экземпляром SQL Server или поставщиком SQL Server. Этот второй экземпляр или поставщик могут располагаться на том же компьютере или на удаленном компьютере в том же домене Windows, что и первый экземпляр.

Встроенная поддержка прозрачного шифрования

Механизм прозрачного шифрования основан на использовании мастер-ключа, сертификата, защищенного мастер- ключом, и ключа для шифрования базы данных. Для работы с мастер-ключом используются следующие операторы T- SQL:

CREATE/ALTER/DROP/OPEN/CLOSE MASTER KEY

Прозрачное шифрование позволяет создавать индексы и выполнять поиск по зашифрованным даным без использования кких-либо дополнительных функций.