Система безопасности

Database Engine

* Аутентификация
* Схемы
* Безопасность баз данных
* Роли
* Авторизация
* Отслеживание изменений
* Безопасность данных и представления

Эта глава начинается с краткого обзора наиболее важных понятий безопасности баз данных, после чего рассматриваются конкретные возможности системы безопасности компонента Database Engine. Наиболее важными концепциями безопасности баз данных являются следующие:

* аутентификация;
* шифрование;
* авторизация;
* отслеживание изменений.

Аутентификация заключается в предоставлении ответа на следующий вопрос:

"Имеет ли данный пользователь законное право на доступ к системе?" Таким образом, данная концепция безопасности определяет процесс проверки подлинностиьучетных данных пользователя, чтобы не допустить использование системы несанкционированными пользователями. Аутентификацию можно реализовать путем запроса, требуя, чтобы пользователь предоставил, например, следующее:

* нечто, что известно пользователю (обычно пароль);
* нечто, что принадлежит пользователю, например, магнитную карту или идентификационную карту;
* физические характеристики пользователя, например, подпись или отпечатки пальцев.

Наиболее применяемый способ подтверждения аутентификации реализуется посредством использования имени пользователя и пароля. Система проверяет достоверность этой информации, чтобы решить, имеет ли данный пользователь законное право на доступ к системе или нет. Этот процесс может быть усилен применением шифрования.

Шифрование данных представляет собой процесс кодирования информации таким образом, что она становится непонятной, пока не будет расшифрована пользователем. Для шифрования данных применяются несколько способов, которые рассматриваются в разд. "Шифрование" далее в этой главе.  
Авторизация — это процесс, который применяется к пользователям, уже получившим доступ к системе, пройдя через процесс аутентификации, чтобы определить их права на использование определенных ресурсов.

Отслеживание изменений означает отслеживание и документирование действий несанкционированных пользователей. Иными словами, документируются все операции вставки, обновления и удаления, применяемые к объектам базы данных. Задокументированные таким образом операции в дальнейшем могут быть проверены вторизованными пользователями. (Данный процесс полезен для защиты системы от пользователей, которые имеют повышенные права.)

Прежде чем приступить к рассмотрению этих четырех концепций безопасности, сначала ознак омимся с кратким определением модели безопасности, которая применяется в SQL Server. Данная модель безопасности состоит из трех разных категорий, которые взаимодействуют друг с другом:

* Принципалы (principals). Субъекты, которые имеют разрешение на доступ к определенной сущности. Типичными принципалами являются учетные записи Windows и SQL Server. Кроме этих принципалов, также существуют группы Windows и роли SQL Server. Группа Windows — это коллекция учетных записей и групп Windows. Присвоение учетной записи пользователя членство в группе дает этому пользователю все разрешения, предоставленные данной группе. Подобным образом роль является коллекцией учетных записей.
* Защищаемые объекты (securables). Ресурсы, доступ к которым регулируется системой авторизации базы данных. Большинство защищаемых объектов создают иерархию, что означает, что некоторые из них могут быть заключенными внутри других. Большинство защищаемых объектов имеют определенное число разрешений, применимых к ним. (Защищаемые объекты подробно рассмотрены далее в этой главе.)
* Разрешения (permissions). Каждый защищаемый объект имеет связанные с ним  
  разрешения, которые могут быть предоставлены принципалу. Разрешения обсуждаются в разд. "Авторизация" далее в этой главе. (Список всех разрешений и  
  их соответствующих защищаемых объектов приводится в табл. 12.3 далее  
  в этой главе.)

Аутентификация

Система безопасности компонента Database Engine состоит из двух разных подсистем безопасности:

* системы безопасности Windows;
* системы безопасности SQL Server.

На основе этих двух подсистем безопасности компонент Database Engine может работать в одном из следующих режимов аутентификации:

* в режиме Windows;
* в смешанном режиме.

Режим Windows требует, чтобы пользователи входили в систему баз данных исключительно посредством своих учетных записей Windows. Система принимает данные учетной записи пользователя, полагая, что они уже были проверены и одобрены на уровне операционной системы. Такой способ подключения к системе баз данных называется доверительным соединением (trusted connection), т. к. система баз данных доверяет, что операционная система уже проверила подлинность учетной записи и соответствующего пароля.

Смешанный режим позволяет пользователям подключаться к компоненту Database Engine посредством аутентификации Windows или аутентификации SQL Server. Это означает, что некоторые учетные записи пользователей можно настроить для использования подсистемы безопасности Windows, а другие, вдобавок к этому, могут использовать также и подсистему безопасности SQL Server.

Реализация режима аутентификации

Выбор одного из доступных режимов аутентификации осуществляется посредством среды SQL Server Management Studio. Чтобы установить режим аутентификации Windows, щелкните правой кнопкой сервер баз данных и в контекстном меню выберите пункт Properties. Откроется диалоговое окно Server Properties, в котором нужно выбрать страницу Security, а на ней Windows Authentication Mode. Для выбора смешанного режима в этом же диалоговом окне Server Properties вам нужно выбрать Server and Windows Authentication Mode.

Шифрование данных

Шифрование — это процесс приведения данных в запутанное непонятное состояние, вследствие чего повышается уровень их безопасности. Обычно конкретная

процедура шифрования осуществляется с использованием определенного алгоритма. Наиболее важный алгоритм шифрования называется RSA, по первым буквам

фамилий его создателей — Rivers, Shamir и Adelman.

Компонент Database Engine обеспечивает безопасность данных посредством иерархии уровней шифрования и инфраструктуры управления ключами. Каждый уровень

защищает следующий за ним уровень шифрования, используя комбинацию сертификатов, асимметричных и симметричных ключей (рис. 12.1).

На рис. 12.1 главный сервисный ключ задает ключ, который управляет всеми другими ключами и сертификатами. Главный сервисный ключ создается автоматически

при установке компонента Database Engine. Этот ключ зашифрован с помощью

API-интерфейса защиты данных Windows (DPAPI — Data Protection API).

Важным свойством главного сервисного ключа является то, что он управляется

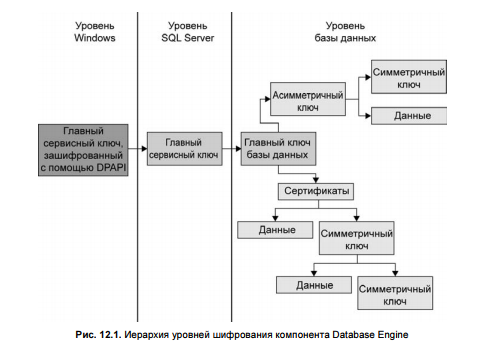
системой. Хотя системный администратор может выполнять разные задачи по обслуживанию ключа, ему следует выполнять лишь одну из них — резервное копирование главного сервисного ключа, чтобы его можно было восстановить в случае

повреждения.

Как можно видеть на рис. 12.1, главный сервисный ключ базы данных является корневым объектом шифрования на уровне базы данных для всех ключей, сертификатов и данных. Каждая база данных имеет один главный ключ базы данных, который

создается посредством инструкции CREATE MASTER KEY (см. пример 12.1). Поскольку

главный ключ базы данных защищен главным сервисным ключом системы, то система может автоматически расшифровывать главный ключ базы данных.



Существующий главный ключ базы данных можно использовать для создания пользовательских ключей. Существует три формы пользовательских ключей:

* симметричные ключи;
* асимметричные ключи;
* сертификаты.

Эти формы пользовательских ключей рассмотрены в следующих далее подразделах.

Симметричные ключи

В системе шифрования с использованием симметричного ключа оба участника обмена — отправитель и получатель сообщения — применяют один и тот же ключ. Иными словами, для шифрования информации и ее обратного расшифровывания используется этот единственный ключ.

Использование симметричных ключей имеет много преимуществ и один недостаток. Одним из преимуществ использования симметричных ключей является то обстоятельство, что с их помощью можно защитить значительно больший объем данных, чем с помощью двух других типов ключей. Кроме этого, использование ключей этого типа намного быстрее, чем использование несимметричных ключей.

Но с другой стороны, использование симметричного ключа в среде распределенной системы может сделать задачу обеспечения целостности шифрования почти невыполнимой, поскольку один и тот же ключ применяется на обоих концах обмена данными. Таким образом, основной рекомендацией является то, что симметричные ключи должны использоваться только с теми приложениями, в которых зашифрованные данные сохраняются в одном месте.

Язык Transact-SQL поддерживает несколько инструкций и системных функций применительно к симметричным ключам. В частности, для создания симметричного ключа применяется инструкция CREATE SYMMETRIC KEY, а для удаления существующего симметричного ключа — инструкция DROP SYMMETRIC KEY. Прежде чем симметричный ключ можно использовать для шифрования данных или для защиты другого ключа, его нужно открыть. Для этого используется инструкция OPEN SYMMETRIC KEY.

После того как вы откроете симметричный ключ, вам нужно для шифрования использовать системную функцию EncryptByKey. Эта функция имеет два входных параметра: идентификатор ключа и текст, который требуется зашифровать. Для расшифровки зашифрованной информации применяется системная функция  
DecryptByKey.

Асимметричные ключи

В случае если у вас имеется распределенная система или если использование симметричных ключей не обеспечивает достаточного уровня безопасности данных, тоследует использовать асимметричные ключи. Асимметричный ключ состоит из двух частей: личного закрытого ключа (private key) и соответствующего общегооткрытого ключа (public key). Каждый из этих ключей может расшифровывать данные, зашифрованные другим ключом. Благодаря наличию личного закрытого ключа асимметричное шифрование обеспечивает более высокий уровень безопасности данных, чем симметричное шифрование.

Язык Transact-SQL поддерживает несколько инструкций и системных функций применительно к асимметричным ключам. В частности, для создания нового асимметричного ключа применяется инструкция CREATE ASYMMETRIC KEY, а для изменения свойств асимметричного ключа используется инструкция ALTER ASYMMETRIC KEY. Для удаления асимметричного ключа применяется инструкция DROP ASYMMETRIC KEY.

После того как вы создали асимметричный ключ, для шифрования данных используйте системную функцию EncryptByAsymKey. Эта функция имеет два входных параметра: идентификатор ключа и текст, который требуется зашифровать. Для расшифровки информации, зашифрованной с использованием асимметричного ключа,  
применяется системная функция DecryptByAsymKey.

Сертификаты

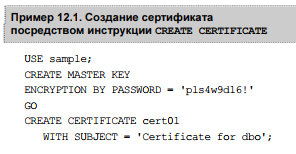
Сертификат открытого ключа, или просто сертификат, представляет собой предложение с цифровой подписью, которое привязывает значение открытого ключа к определенному лицу, устройству или службе, которая владеет соответствующим открытым ключом. Сертификаты выдаются и подписываются центром сертификации (Certification Authority (CA)). Сущность, которая получает сертификат из центра сертификации (CA), является субъектом данного сертификата (certificate subject).

Сертификаты содержат следующую информацию:

* значение открытого ключа субъекта;
* информацию, идентифицирующую субъекта;
* информацию, идентифицирующую издателя сертификата;
* цифровую подпись издателя сертификата.

Основным достоинством сертификатов является то, что они освобождают хосты от необходимости содержать набор паролей для отдельных субъектов.

Наиболее важной инструкцией применительно к сертификатам является инструкция CREATE CERTIFICATE. Использование этой инструкции показано в примере 12.1.



Чтобы создать сертификат без параметра ENCRYPTION BY, сначала нужно создать главный ключ базы данных. (Все инструкции CREATE CERTIIFCATE, которые не содержат этот параметр, защищаются главным ключом базы данных.) По этой причине первой инструкцией в примере 12.1 является инструкция CREATE MASTER KEY. После этого инструкция CREATE CERTIFICATE используется для создания нового сертификата cert01, владельцем которого является объект dbo базы данных sample, если этот объект является текущим пользователем.

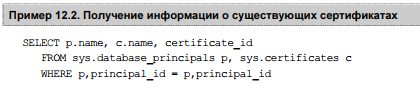
Редактирование пользовательских ключей

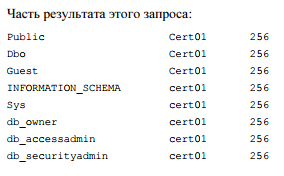
Наиболее важными представлениями каталога применительно к шифрованию являются следующие:

* sys.symmetric\_keys;
* sys.asymmetric\_keys;
* sys.certificates;
* sys.database\_principals.

Первые три представления каталога предоставляют информацию обо всех симметричных ключах, всех асимметричных ключах и всех сертификатах, установленных в текущей базе данных, соответственно. Представление каталога sys.database\_ principals предоставляет информацию обо всех принципалах в текущей базе данных. Последнее представление каталога можно соединить с любым из первых трех, чтобы получить информацию о владельце определенного ключа.

В примере 12.2 показано получение информации о существующих сертификатах. Подобным образом можно получить информацию о симметричных и асимметричных ключах.





Расширенное управление ключами SQL Server

Следующим шагом к обеспечению более высокого уровня безопасности ключей является использование расширенного управления ключами (Extensible Key Management (EKM)). Основными целями расширенного управления ключами являются следующие:

* повышение безопасности ключей посредством выбора поставщика функций шифрования;
* общее управление ключами по всему предприятию.

Способы шифрования данных

SQL Server поддерживает два способа шифрования данных:  
шифрование на уровне столбцов;  
прозрачное шифрование данных.

Шифрование на уровне столбцов позволяет шифровать конкретные столбцы данных. Для реализации этого способа шифрования используется несколько пар сопряженных функций. Далее мы не будем рассматривать этот метод шифрования, поскольку его реализация является сложным ручным процессом, требующим внесения изменений в приложение. Прозрачное шифрование данных является новой возможностью базы данных, которая зашифровывает файлы базы данных автоматически, не требуя внесения изменений в какие-либо приложения. Таким образом можно предотвратить доступ к информации базы данных несанкционированным лицам, даже если они смогут получить в свое распоряжение основные или резервные файлы базы данных. Применения прозрачного шифрования для защиты определенной базы данных реализуется в четыре этапа:

1. Используя инструкцию CREATE MASTER KEY, создается главный ключ базы данных. (Применение этой инструкции показано в примере 12.1.)

2. С помощью инструкции CREATE CERTIFICATE создается сертификат (см. пример 12.1).

3. Используя инструкцию CREATE DATABASE ENCRYPTION KEY, создается ключ шифрования.

4. Выполняется конфигурирование базы данных для использования шифрования. (Этот шаг можно реализовать, присвоив параметру ENCRYPTION инструкции ALTER DATABASE значение ON.)

Настройка безопасности компонента Database Engine

Настройку безопасности компонента Database Engine можно выполнить одним из следующих способов:

с помощью среды управления Management Studio сервера SQL Server;

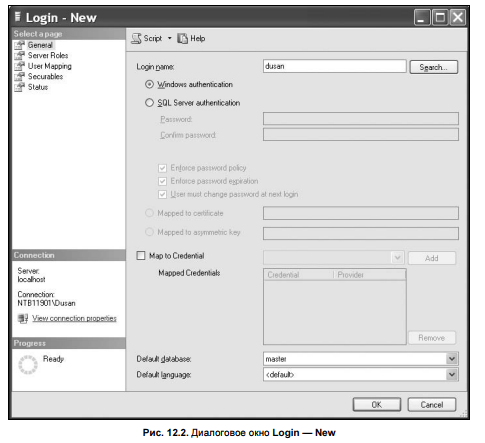
используя инструкции языка Transact-SQL.

Эти два метода рассматриваются в последующих подразделах.

Управление безопасностью с помощью среды Management Studio

Чтобы с помощью среды Management Studio создать новое регистрационное имя, разверните в обозревателе объектов узел сервера, затем разверните папку "Security", в этой папке щелкните правой кнопкой папку "Logins" и в контекстном меню выберите опцию New Login. Откроется диалоговое окно Login — New (рис. 12.2).

Первым делом нужно решить, какой способ аутентификации применять: Windows или SQL Server. В случае выбора аутентификации Windows, в качестве регистрационного имени (Login name) необходимо указать действительное имя пользователя Windows в форме domain\user\_name (домен\имя\_пользователя). А если выбрана аутентификация SQL Server, необходимо ввести новое регистрационное имя (Login name) и соответствующий пароль (Password). Факультативно можно также указать базу данных и язык по умолчанию. База данных по умолчанию — это база данных, к которой пользователь автоматически подключается сразу же после входа в компонент Database Engine. Выполнив все эти действия, пользователь может входить в систему под этой новой учетной записью.



Управление безопасностью посредством инструкций Transact-SQL

Для управления безопасностью компонента Database Engine применяются три инструкции языка Transact-SQL: CREATE LOGIN, ALTER LOGIN и DROP LOGIN.

Инструкция CREATE LOGIN создает новое регистрационное имя входа в SQL Server. Синтаксис этой инструкции следующий:

CREATE LOGIN login\_name

{WITH option\_list1 |

FROM {WINDOWS [WITH option\_list2 [,...]]

| CERTIFICATE certname | ASYMMETRIC KEY key\_name }}

В параметре login\_name указывается создаваемое регистрационное имя. Как можно видеть в синтаксисе этой инструкции, в предложении WITH можно указать один или несколько параметров для регистрационного имени или указать в предложении FROM сертификат, асимметричный ключ или учетную запись пользователя Windows, связанную с соответствующим регистрационным именем.

В списке option\_list1 указывается несколько параметров, наиболее важным из которых является параметр PASSWORD, который задает пароль для данного регистрационного имени (пример 12.3). (Другие возможные параметры — DEFAULT\_DATABASE, DEFAULT\_LANGUAGE и CHECK\_EXPIRATION.)

Как видно из синтаксиса инструкции CREATE LOGIN, предложение FROM может содержать один из следующих параметров:

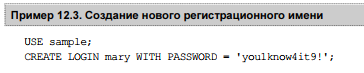
WINDOWS — указывает, что данное регистрационное имя соотносится с существующей учетной записью пользователя Window (см. пример 12.4). Этот параметр можно указать с другими подпараметрами, такими как DEFAULT\_DATABASE и DEFAULT\_LANGUAGE;

SERTIFICATE — задает имя сертификата для привязки к данному регистрационному имени;

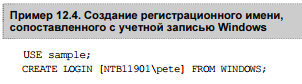
ASYMMETRIC KEY — задает имя асимметричного ключа для привязки к данному регистрационному имени. (Сертификат и асимметричный ключ уже должны присутствовать в базе данных master.)

В примерах 12.3 и 12.4 показано создание разных форм регистрационного имени.

В примере 12.3 создается регистрационное имя mary с паролем you1know4it9!.



В примере 12.4 создается регистрационное имя pete, которое сопоставляется с учетной записью пользователя Windows с таким же самым именем пользователя.



Вторая инструкция языка Transact-SQL для обеспечения безопасности — ALTER LOGIN — изменяет свойства определенного регистрационного имени. С по мощью этой инструкции можно изменить текущий пароль и его конечную дату действия, параметры доступа, базу данных по умолчанию и язык по умолчанию. Также можно задействовать или отключить определенное регистрационное имя. Наконец, инструкция DROP LOGIN применяется для удаления существующего регистрационного имени. Однако регистрационное имя, которое ссылается (владеет) на другие объекты, удалить нельзя.

Схемы

Схемы используются в модели безопасности компонента Database Engine для упрощения взаимоотношений между пользователями и объектами, и, следовательно, схемы имеют очень большое влияние на взаимодействие пользователя с компонентом Database Engine.

Схема — это коллекция объектов базы данных, имеющая одного владельца и формирующая одно пространство имен. (Две таблицы в одной и той же схеме не могут иметь одно и то же имя.)

Отделение пользователей базы данных от схем дает значительные преимущества, такие как:

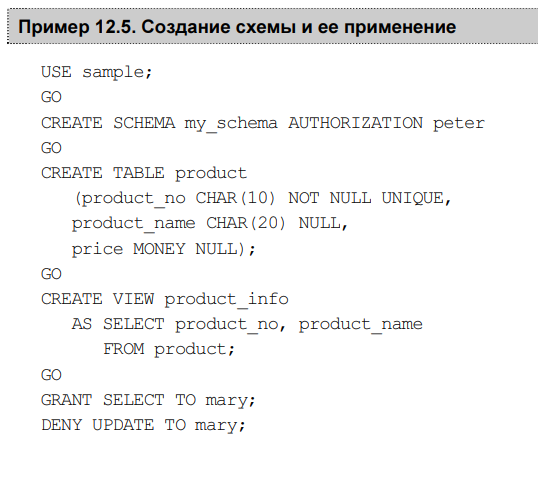
* один принципал может быть владельцем нескольких схем;
* несколько индивидуальных принципалов могут владеть одной схемой посредством членства в ролях или группах Windows;
* удаление пользователя базы данных не требует переименования объектов, содержащихся в схеме этого пользователя.

Инструкции языка DDL для работы со схемами

Для работы со схемами используются следующие три инструкции языка Transact-SQL:

* CREATE SCHEMA;
* ALTER SCHEMA;
* DROP SCHEMA.

Инструкция CREATE SCHEMA



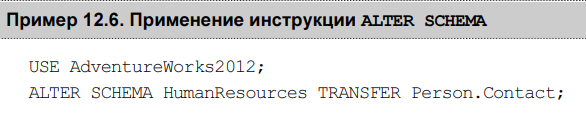
Инструкция CREATE SCHEMA является атомарной. Иными словами, если в процессе выполнения этой инструкции происходит ошибка, не выполняется ни одна из содержащихся в ней подынструкций.

Для исполнения инструкции CREATE SCHEMA пользователь должен обладать правами базы данных CREATE SCHEMA. Кроме этого, для создания объектов, указанных в инструкции CREATE SCHEMA, пользователь должен иметь соответствующие разрешения CREATE.

Инструкция ALTER SCHEMA

Инструкция ALTER SCHEMA перемещает объекты между разными схемами одной и той же базы данных. Инструкция ALTER SCHEMA имеет следующий синтаксис:

ALTER SCHEMA schema\_name TRANSFER object\_name



Инструкцию ALTER SCHEMA можно использовать для перемещения объектов между разными схемами только одной и той же базы данных. (Отдельные объекты в схеме можно изменить посредством инструкции ALTER TABLE или ALTER VIEW.)

Инструкция DROP SCHEMA

Для удаления схемы из базы данных применяется инструкция DROP SCHEMA. Схему можно удалить только при условии, что она не содержит никаких объектов. Если схема содержит объекты, попытка выполнить инструкцию DROP SCHEMA будет неуспешной. Владельца схемы можно изменить посредством инструкции ALTER AUTHORIZATION, которая изменяет владение сущностью.

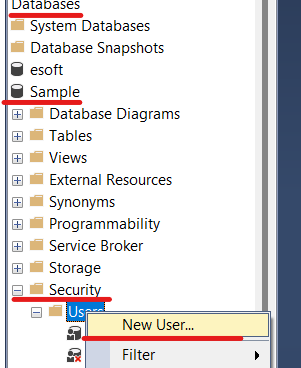
Безопасность базы данных

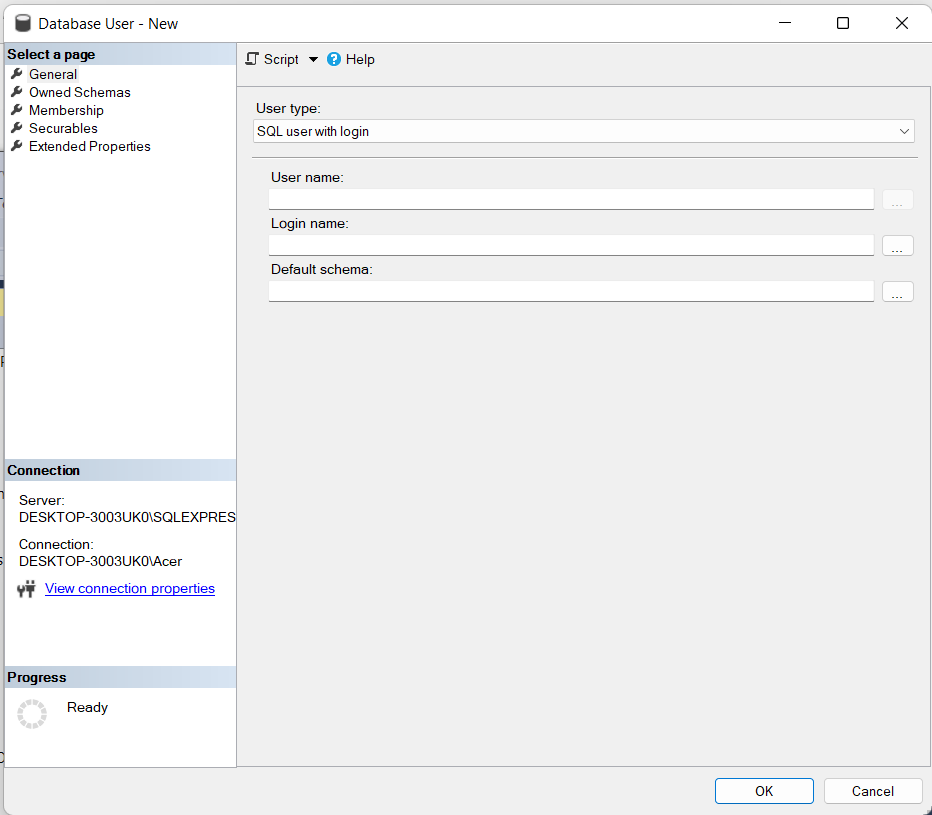
Пользователь может войти в систему баз данных, используя учетную запись пользователя Windows или регистрационное имя входа в SQL Server. Для последующего доступа и работы с определенной базой данных пользователь также должен иметь учетную запись пользователя базы данных. Для работы с каждой отдельной базой данных требуется иметь учетную запись пользователя именно для этой базы данных. Управлять безопасностью баз данных можно с помощью:

* среды Management Studio;
* инструкций языка Transact-SQL.

Управление безопасностью базы данных с помощью среды Management Studio

Чтобы добавить пользователя базы данных с помощью среды Management Studio, разверните узел сервера, папку "Databases", в этой папке разверните узел требуемой базы данных, а в ней папку "Security". Щелкните правой кнопкой мыши папку "Users" и в контекстном меню выберите пункт New User. Откроется диалоговое окно Database User — New, в котором следует ввести имя пользователя и выбрать соответствующее регистрационное имя. Факультативно можно выбрать схему по умолчанию для данного пользователя.

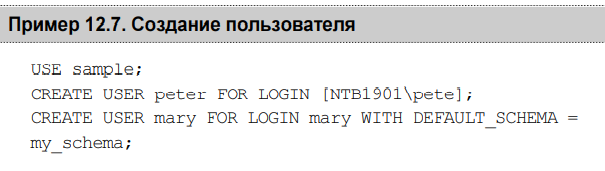




Управление безопасностью базы данных посредством инструкций языка Transact-SQL

Для добавления пользователя в текущую базу данных используется инструкция CREATE USER. Синтаксис этой инструкции выглядит таким образом:

CREATE USER user\_name [FOR {LOGIN login|CERTIFICATE cert\_name | ASYMMETRIC KEY key\_name}] [WITH DEFAULT\_SCHEMA = schema\_name]



Первая инструкция CREATE USER создает пользователя базы данных peter для пользователя pete учетной записи Windows. Схемой по умолчанию для пользователя pete будет dbo, поскольку для параметра DEFAULT SCHEMA значение не указано.

Вторая инструкция CREATE USER создает нового пользователя базы данных mary. Схемой по умолчанию для этого пользователя будет схема my\_schema.

Схемы базы данных по умолчанию

Каждая база данных в системе имеет следующие схемы по умолчанию:

* guest;
* dbo;
* INFORMATION\_SCHEMA;
* Sys. (содержит системные объекты, такие как представления каталога)

Роли

Когда нескольким пользователям требуется выполнять подобные действия в определенной базе данных и при этом они не являются членами соответствующей группы Windows, то можно воспользоваться ролью базы данных, задающей группу пользователей базы данных, которые могут иметь доступ к одним и тем же объектам базы данных.

Членами роли базы данных могут быть любые из следующих:

* группы и учетные записи Windows;
* регистрационные имена входа в SQL Server;
* другие роли.

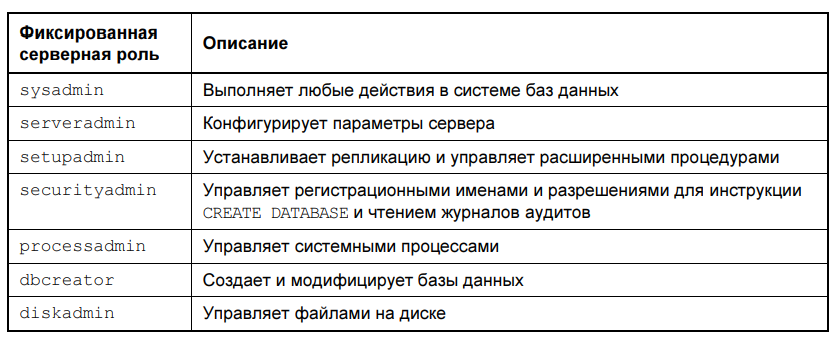
Два типа предопределенных ролей:

* фиксированные серверные роли;
* фиксированные роли базы данных.

Также рассматриваются и такие типы ролей:

* роли приложений;
* определяемые пользователем серверные роли;
* определяемые пользователем роли баз данных.

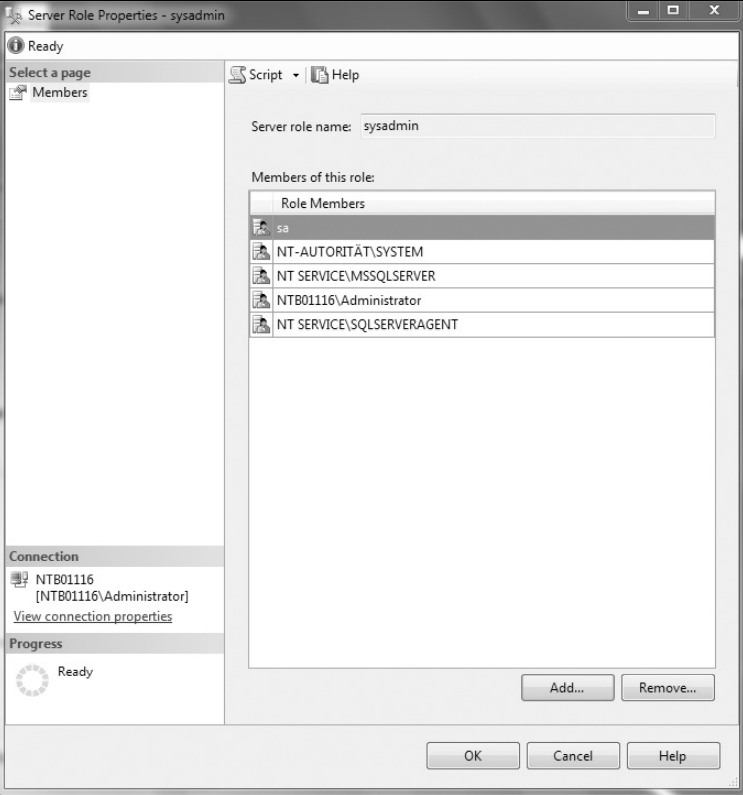
Фиксированные серверные роли



Управление фиксированными серверными ролями

Членов фиксированной серверной роли можно добавлять и удалять двумя способами:

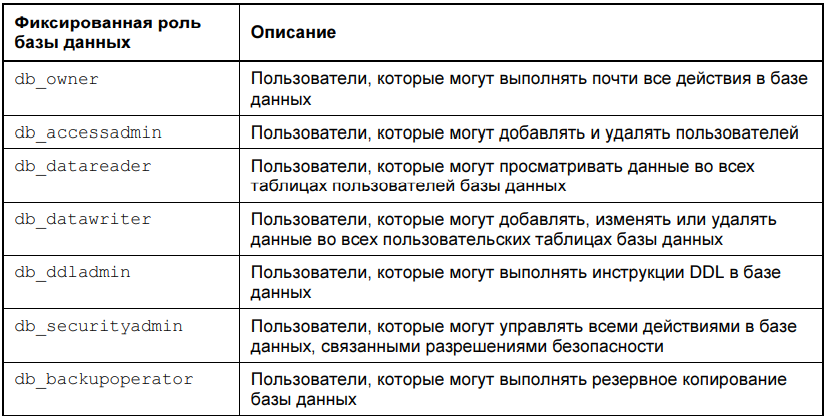
* используя среду Management Studio;
* используя инструкции языка Transact-SQL.

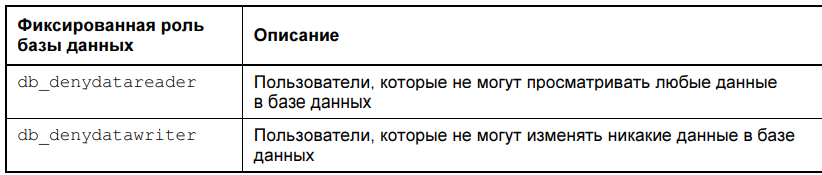


Регистрационное имя sa

Регистрационное имя sa является регистрационным именем системного администратора, используется только в тех случаях, когда нет другого способа войти в систему базы данных.

Фиксированные роли базы данных





Фиксированная роль базы данных public

Фиксированная роль базы данных public является специальной ролью, членом которой являются все законные пользователи базы данных.

По умолчанию роль public разрешает пользователям выполнять следующие действия:

* просматривать системные таблицы и отображать информацию из системной базы данных master, используя определенные системные процедуры;
* выполнять инструкции, для которых не требуются разрешения, например, PRINT.

Присвоения пользователю членства в фиксированной роли базы данных

Чтобы присвоить пользователю базы данных членство в фиксированной роли базы данных с помощью среды Management Studio, разверните сервер и папку "Databases", а в ней базу данных, затем разверните папку "Security", "Roles" и папку "Databases Roles". Щелкните правой кнопкой мыши роль, в которую требуется добавить пользователя, и в контекстном меню выберите пункт Properties. В диалоговом окне свойств роли базы данных нажмите кнопку Add и выберите пользователей, которым нужно присвоить членство в этой роли.

Роли приложений

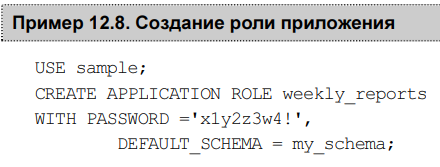
Роли приложения позволяют принудительно обеспечивать безопасность для определенного приложения. Иными словами, роли приложения позволяют приложению взять на себя ответственность за аутентификацию пользователя, вместо того, чтобы это делала система баз данных.

Управление ролями приложений посредством среды Management Studio

Чтобы создать роль приложения с помощью среды Management Studio, разверните узел сервера, папку "Databases", требуемую базу данных, папку "Security". Щелкните правой кнопкой папку "Roles", в появившемся контекстном меню выберите пункт New, а во вложенном меню выберите пункт New Application Role. В открывшемся диалоговом окне Application Role — New введите в соответствующие поля имя новой роли приложения, пароль и, необязательно, схему по умолчанию. Нажмите кнопку OK, чтобы сохранить роль.

Управление ролями приложений посредством инструкций Transact-SQL

Для создания, изменения и удаления ролей приложений применяются инструкции языка Transact-SQL CREATE APPLICATION ROLE, ALTER APPLICATION ROLE и DROP APPLICATION ROLE соответственно.



Активация ролей приложений

Эта процедура имеет следующий синтаксис: sp\_setapprole [@rolename =] 'role', [@password =] 'password' [,[@encrypt =] 'encrypt\_style']

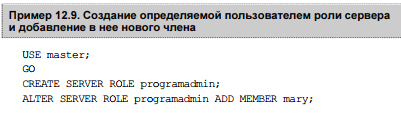
При активации роли приложения с помощью системной процедуры sp\_setupprole необходимо иметь в виду следующее:

активированную роль приложения нельзя деактивировать в текущей базе данных, пока сеанс не отсоединится от системы;

роль приложения всегда привязана к базе данных, т. е. ее область видимости ограничивается текущей базой данных. Если в течение сеанса изменить текущую базу данных, то в ней можно будет выполнять действия, зависящие от разрешений в этой базе данных.

Определяемые пользователем роли сервера

Определяемые пользователем роли сервера впервые стали применяться в SQL Server 2012. Для создания и удаления этих ролей используются инструкции языка Transact-SQL CREATE SERVER ROLE и DROP SERVER ROLE соответственно. Для добавления или удаления членов роли сервера используется инструкция ALTER SERVER ROLE. Использование инструкций CREATE SERVER ROLE и ALTER SERVER ROLE показано в примере 12.9.

****

Определяемые пользователем роли баз данных

Обычно определяемые пользователем роли базы данных применяются, когда группе пользователей базы данных требуется выполнять общий набор действий в базе данных и отсутствует применимая группа пользователей Windows. Для создания, изменения и удаления этих ролей применяется или среда Management Studio, или инструкции языка Transact-SQL CREATE ROLE, ALTER ROLE и DROP ROLE. Управление определяемыми пользователем ролями базы данных рассматривается в следующих двух подразделах.

Управление определяемыми пользователем ролями базы данных с помощью среды Management Studio

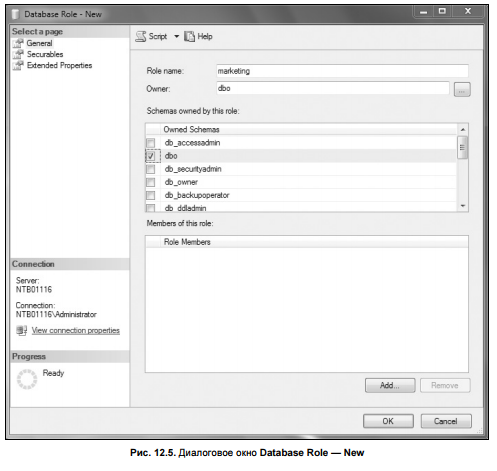
Чтобы создать определяемую пользователем роль базы данных с помощью среды Management Studio, разверните узел сервера, папку "Databases", требуемую базу данных, папку "Security". Щелкните правой кнопкой папку "Roles", в появившемся контекстном меню выберите пункт New, а во вложенном меню выберите пункт New Database Role. В открывшемся диалоговом окне Database Role — New (рис. 12.5) введите в соответствующее поле имя новой роли.

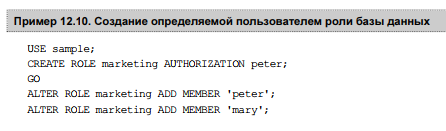
Нажмите кнопку Add, чтобы добавить членов в новую роль. Выберите требуемых членов (пользователей и/или другие роли) новой роли базы данных и нажмите кнопку OK.

Управление определяемыми пользователем ролями базы данных с помощью инструкций Transact-SQL

Для создания новой определяемой пользователем роли базы данных в текущей базе данных применяется инструкция CREATE ROLE. Синтаксис этой инструкции выглядит таким образом:

CREATE ROLE role\_name [AUTHORIZATION owner\_name]





Авторизация

Выполнять инструкции или осуществлять операции с объектами баз данных могут только авторизованные пользователи. Попытка выполнения любой из этих задач неавторизованным пользователем будет неудачной. Для выполнения задач, связанных с авторизацией, используются следующие три инструкции языка Transact-SQL:

GRANT;

DENY;

REVOKE.

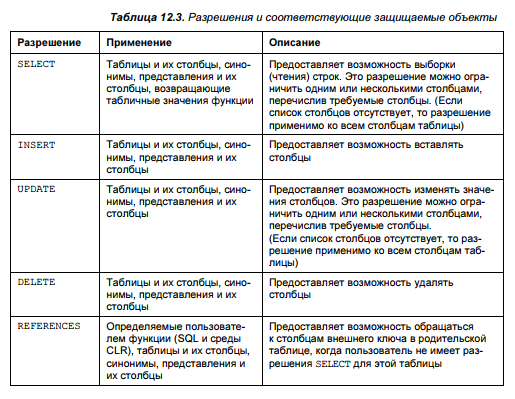
Инструкция GRANT

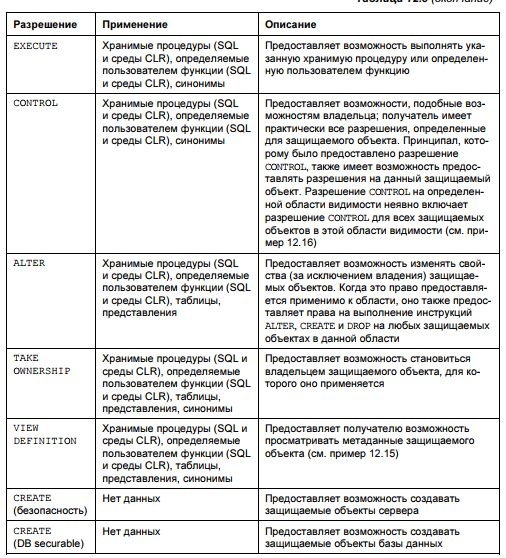
Инструкция GRANT предоставляет разрешения принципалам на защищаемые объекты. Эта инструкция имеет следующий синтаксис:

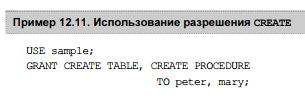
GRANT {ALL [PRIVILEGES]} permission\_list

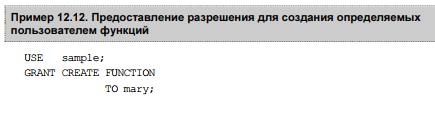
[ON [class::] securable] TO principal\_list [WITH GRANT OPTION]

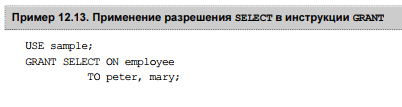
[AS principal ]

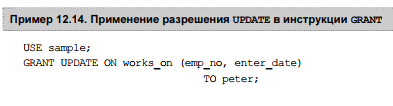


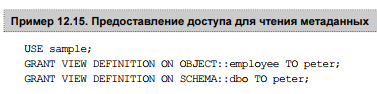


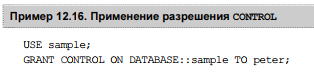


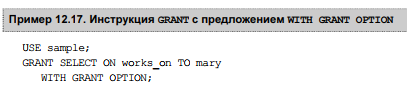












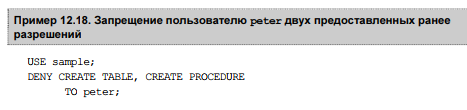
Инструкция DENY

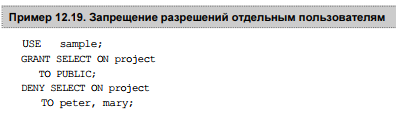
Инструкция DENY запрещает пользователю выполнять указанные действия на указанных объектах. Иными словами, эта инструкция удаляет существующие разрешения для учетной записи пользователя, а также предотвращает получение разрешений пользователем посредством его членства в группе или роли, которое он может получить в будущем. Эта инструкция имеет следующий синтаксис:

DENY {ALL [PRIVILEGES]} | permission\_list

[ON [class::] securable] TO principal\_list

[CASCADE] [AS principal]





Инструкция REVOKE

Инструкция REVOKE удаляет предоставленное или запрещенное ранее разрешение. Эта инструкция имеет следующий синтаксис:

REVOKE [GRANT OPTION FOR]

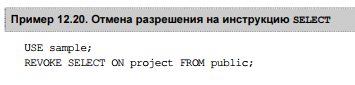
{[ALL [PRIVILEGES]] | permission\_list]}

[ON [class::] securable ]

FROM principal\_list [CASCADE] [AS principal]

Единственным новым параметром инструкции REVOKE является параметр GRANT OPTION FOR. Все другие параметры этой инструкции имеют точно такое же логическое значение, как и одноименные параметры инструкций GRANT или DENY.

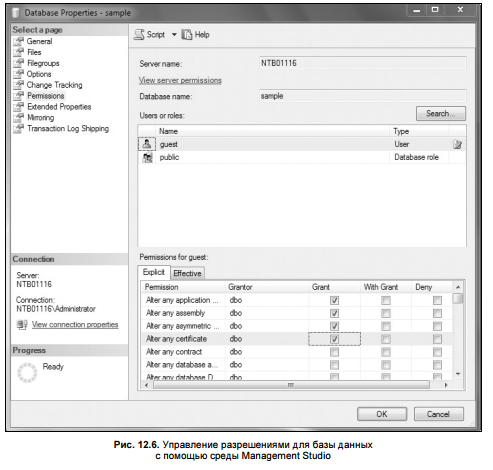
Параметр GRANT OPTION FOR используется для отмены эффекта предложения  
WITH GRANT OPTION в соответствующей инструкции GRANT. Это означает, что пользователь все еще будет иметь предоставленные ранее разрешения, но больше не сможет предоставлять их другим пользователям.



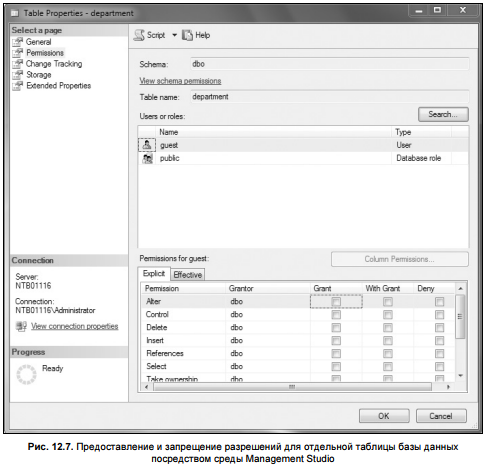
Управление разрешениями с помощью среды Management Studio

Для управления разрешениями с помощью среды Management Studio разверните сервер, а затем папку "Databases". Щелкните правой кнопкой мыши требуемую базу данных и в контекстном меню выберите пункт Properties. В открывшемся диалоговом окне свойств базы данных Database Properties — sample (рис. 12.6) выберите страницу Permissions, после чего нажмите кнопку Search, чтобы выбрать пользователей, которым предоставить или запретить разрешения. В открывшемся диалоговом окне нажмите кнопку Browse, в диалоговом окне Browse for Objects выберите требуемых пользователей или роли (например, guest и public) и нажмите кнопки ОK соответствующих окон, чтобы добавить выбранных пользователей (роли).

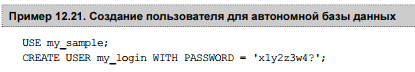
Выбранные таким образом пользователи будут отображены в окне свойств базы данных в поле Users or roles (Пользователи и роли) (см. рис. 12.6).

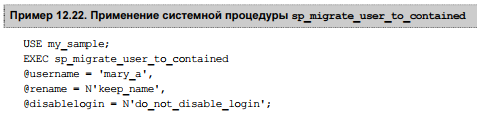


Теперь для выбранных пользователей можно разрешить или запретить выполнение определенных действий. В частности, для предоставления разрешения, для требуемого действия установите флажок в столбце Grant, а для запрещения действия установите для него флажок в столбце Deny. Установленный флажок в столбце With Grant означает, что получатель разрешения также имеет право предоставлять его другим пользователям.



Управление авторизацией и аутентификацией для автономных баз данных





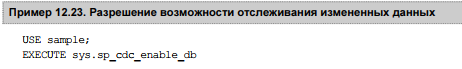
Отслеживание изменений

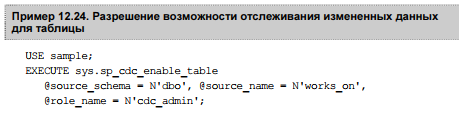
Понятие отслеживание изменений означает документирование всех операций вставок, обновлений и удалений, применяемых к таблицам базы данных. С помощью этих записей можно будет в дальнейшем определить, кто и когда обращался к данным. Отслеживание изменений можно реализовать двумя способами:

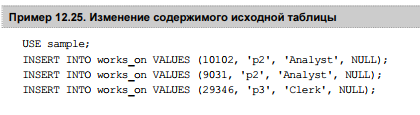
используя триггеры;

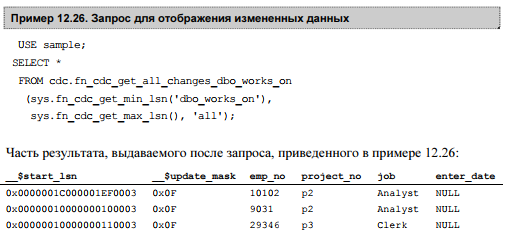
используя систему отслеживания измененных данных.

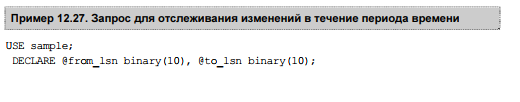
С помощью триггеров можно создать журнал аудита действий в таблицах базы данных. Применение триггеров рассматривается в разд. "Триггеры AFTER" и примере 14.1 в главе 14, поэтому в этом разделе мы будем рассматривать только второй подход.

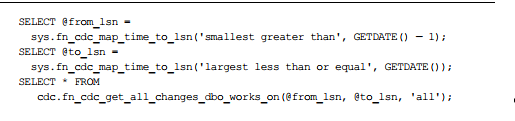








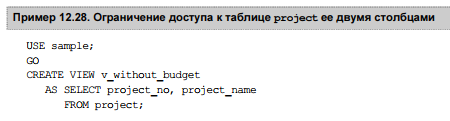


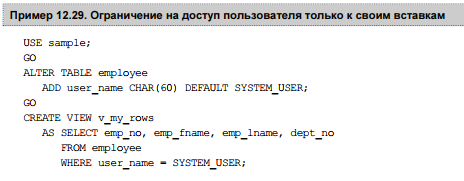


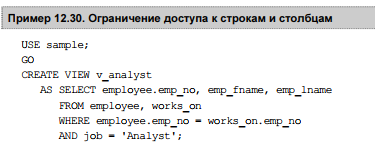
Безопасность данных и представления

Как уже упоминалось в главе 11, представления можно использовать для следующих целей:

* для ограничения использования определенных столбцов и/или строк таблиц;
* для скрытия подробностей сложных запросов;
* для ограничения вставляемых или обновляемых значений определенными диапазонами.







Резюме

Наиболее важными концепциями безопасности баз данных являются следующие:

* аутентификация;
* шифрование;
* авторизация;
* отслеживание изменений.

Аутентификация является процессом проверки подлинности учетных данных пользователя, чтобы не допустить использование системы несанкционированными пользователями. Наиболее применяемым способом аутентификации является требование, чтобы для входа в систему пользователь предоставил свое имя пользова 366 Часть II. Язык Transact-SQL теля и соответствующий пароль. При шифровании данных выполняется кодирование информации таким образом, что содержащиеся данные более не являются понимаемыми до тех пор, пока получатель данных не расшифрует их. Данные можно зашифровывать несколькими разными способами.