Защита информации в базах данных

В СУБД поддерживается один из двух наиболее общих подходов к вопросу обеспечения безопасности данных: избирательный подход и обязательный подход. В обоих подходах единицей данных или «объектом данных», может быть как вся база данных целиком, так и любой объект внутри базы данных.

Эти два подхода отличаются следующими свойствами:

- в случае избирательного управления пользователь обладает различными правами (привилегиями или полномочиями) при работе с данными объектами. Разные пользователи могут обладать разными правами доступа к одному и тому же объекту. Избирательные права характеризуются значительной гибкостью.

- избирательный принцип реализуется следующими методами. В базу данных вводится новый тип объектов БД — пользователей. Каждому пользователю в БД присваивается уникальный идентификатор. Для дополнительной защиты каждый пользователь кроме уникального идентификатора снабжается уникальным паролем, причем если идентификаторы пользователей в системе доступны системному администратору, то пароли пользователей хранятся чаще всего в специальном кодированном виде и известны только самим пользователям.

-в случае обязательного управления каждому объекту данных присваивается некоторый классификационный уровень, а каждый пользователь обладает некоторым уровнем допуска. При таком подходе доступом к определенному объекту данных обладают только пользователи с соответствующим уровнем допуска.
- пользователи могут быть объединены в специальные группы пользователей. Один пользователь может входить в несколько групп. В стандарте вводится понятие группы PUBLIC, для которой должен быть определен минимальный стандартный набор прав. По умолчанию каждый вновь создаваемый пользователь, если не указано иное, относится к группе PUBLIC.
-Привилегии или полномочия пользователей или групп — это набор действий (операций), которые они могут выполнять над объектами БД. Роль — это поименованный набор полномочий. Существует ряд стандартных ролей, которые определены в момент установки сервера баз данных. имеется возможность создавать новые роли, группируя в них произвольные полномочия. Введение ролей позволяет упростить управление привилегиями пользователей, структурировать этот процесс. Кроме того, введение ролей не связано с конкретными пользователями, поэтому роли могут быть определены и сконфигурированы до того, как определены пользователи системы.
Пользователю может быть назначена одна или несколько ролей.

- Объектами БД, которые подлежат защите, являются все объекты, хранимые в БД: таблицы, представления, хранимые процедуры и триггеры. Для каждого типа объектов есть свои действия, поэтому для каждого типа объектов могут быть определены разные права доступа.

На самом элементарном уровне концепции обеспечения безопасности баз данных необходимо поддерживать два фундаментальных принципа: проверку полномочий и проверку подлинности (аутентификацию).

Система назначения полномочий имеет иерархический характер. Самыми высокими правами и полномочиями обладает системный администратор или администратор сервера БД. Традиционно только этот тип пользователей может создавать других пользователей и наделять их определенными полномочиями.

СУБД в своих системных каталогах хранит как описание самих пользователей, так и описание их привилегий по отношению ко всем объектам.

Далее схема предоставления полномочий строится по следующему принципу. Каждый объект в БД имеет владельца — пользователя, который создал данный объект. Владелец объекта обладает всеми правами-полномочиями на данный объект, в том числе он имеет право предоставлять другим пользователям полномочия по работе с данным объектом или забирать у пользователей ранее предоставленные полномочия.

В ряде СУБД вводится следующий уровень иерархии пользователей — это администратор БД. В этих СУБД один сервер может управлять множеством СУБД (например, MS SQL Server, Sybase).

В СУБД Oracle применяется однобазовая архитектура, поэтому там вводится понятие подсхемы — части общей схемы БД и вводится пользователь, имеющий доступ к подсхеме.

В стандарте SQL не определена команда создания пользователя, но практически во всех СУБД создать пользователя можно не только в интерактивном режиме, но и программно с использованием специальных хранимых процедур. Однако для выполнения этой операции пользователь должен иметь право на запуск соответствующей системной процедуры.

В стандарте SQL определены два оператора: GRANT и REVOKE соответственно предоставления и отмены привилегий.

Оператор предоставления привилегий имеет следующий формат:

GRANT {<список действий>| ALL PRIVILEGES }

ON <имя_объекта> TO {<имя_пользователя> | PUBLIC }

[WITH GRANT OPTION]

Здесь список действий определяет набор действий из обще допустимого перечня действий над объектом данного типа.

Параметр ALL PRIVILEGES указывает, что разрешены все действия из допустимых для объектов данного тина.

<имя_объекта> — задает имя конкретного объекта: таблицы, представления, хранимой процедуры, триггера.

<имя_пользователя> или PUBLIC определяет, кому даются данные привилегии.

Параметр WITH GRANT OPTION является необязательным и определяет режим, при котором передаются не только права на указанные действия, но и право передавать эти права другим пользователям.

Для объекта типа таблица полным допустимым перечнем действий является набор из четырех операций: SELECT, INSERT, DELETE, UPDATE. При этом операция обновление может быть ограничена несколькими столбцами.

Общий формат оператора назначения привилегий для объекта типа таблица будет иметь следующий синтаксис:

GRANT {[SELECT][.INSERT][.DELETE][,UPDATE (<список столбцов»)]}

ON <имя таблицы»

TO {<имя_пользователя> | PUBLIC } [WITH GRANT OPTION]

Пусть существуют три пользователя с уникальными именами user1, user2 и user3. Все они являются пользователями одной БД.

User1 создал объект Tab1, он является владельцем этого объекта и может передать права на работу с эти объектом другим пользователям. Допустим, что пользователь user2 является оператором, который должен вводить данные в Tab1 (например, таблицу новых заказов), а пользователь user3 является большим начальником , который должен регулярно просматривать введенные данные.

Тогда резонно будет выполнить следующие назначения:

GRANT INSER1

ON Tab1

TO user2

GRANT SELECT

ON Tabl

TO user3

При назначении прав доступа на операцию модификации можно уточнить, значение каких столбцов может изменять пользователь. Допустим, что менеджер отдела имеет право изменять цену на предоставляемые услуги. Предположим, что цена задается в столбце COST таблицы Tab1. Тогда операция назначения привилегий пользователю user3 может измениться и выглядеть следующим образом:

GRANT SELECT. UPDATE (COST) ON Tabl TO user3

Если наш пользователь user1 предполагает, что пользователь user4 может его замещать в случае его отсутствия, то он может предоставить ему все права по работе с созданной таблицей Tab1.

GRANT ALL PRIVILEGES

ON Tabl

TO user4 WITH GRANT OPTION

В этом случае user4 может сам назначать привилегии по работе с таблицей Tab1 в отсутствие владельца объекта пользователя user1

GRANT INSERT

ON Tab1

TO user5

Если при передаче полномочий набор операций над объектом ограничен, то пользователь, которому переданы эти полномочия, может передать другому пользователю только те полномочия, которые есть у него, или часть этих полномочий. Кроме непосредственного назначения прав по работе с таблицами эффективным методом защиты данных может быть создание представлений, которые будут содержать только необходимые столбцы для работы конкретного пользователя и предоставление прав на работу с данным представлением пользователю.

Так как представления могут соответствовать итоговым запросам, то для этих представлений **недопустимы операции изменения**, и набор допустимых действий ограничивается операцией SELECT. Если же представления соответствуют выборке из базовой таблицы, то для такого представления допустимыми будут все 4 операции: SELECT, INSERT, UPDATE и DELETE.

Для отмены ранее назначенных привилегий в стандарте SQL определен оператор REVOKE.

Оператор отмены привилегий имеет следующий синтаксис:

REVOKE {<список операций | ALL PRIVILEGES}

ON <имя_объекта>

FROM {<список пользователей | PUBLIC } {CASCADE | RESTRICT }

Параметры CASCADE или RESTRICT определяют, каким образом должна производиться отмена привилегий. Параметр CASCADE отменяет привилегии не только пользователя, который непосредственно упоминался в операторе GRANT при предоставлении ему привилегий, но и всем пользователям, которым этот пользователь присвоил привилегии, воспользовавшись параметром WITH GRANT OPTION.

Параметр RESTRICKT ограничивает отмену привилегий только пользователю, непосредственно упомянутому в операторе REVOKE.

Но при наличии делегированных привилегий этот оператор не будет выполнен. Так, например, операция:

REVOKE ALL PRIVILEGES

ON Tabl

TO user4 RESTRICT

не будет выполнена, если пользователь user4 передал часть своих полномочий пользователю user5.

Посредством оператора REVOKE можно отобрать все или только некоторые из ранее присвоенных привилегий по работе с конкретным объектом. При этом из описания синтаксиса оператора отмены привилегий видно, что можно отобрать привилегии одним оператором сразу у нескольких пользователей или у целой группы PUBLIC.

Корректным будет следующее использование оператора REVOKE:

REVOKE INSERT

ON Tab1

TO user2,user4 CASCADE

При работе с другими объектами изменяется список операций, которые используются в операторах GRANT и REVOKE.

По умолчанию действие, соответствующее запуску (исполнению) хранимой процедуры, назначается всем членам группы PUBLIC.

Если вы хотите изменить это условие, то после создания хранимой процедуры необходимо записать оператор REVOKE.

REVOKE EXECUTE ON COUNT_EX TO PUBLIC CASCADE

Системный администратор может разрешить некоторому пользователю создавать и изменять таблицы в некоторой БД. Тогда оператор предоставления прав следующий:

GRANT CREATE TABLE, ALTER TABLE.

DROP TABLE ON DB_LIB TO userl

В этом случае пользователь user1 может создавать, изменять или удалять таблицы в DB_LIB, однако он не может разрешить создавать или изменять таблицы в этой БД другим пользователям, потому что ему дано разрешение без права делегирования своих возможностей.

В некоторых СУБД пользователь может получить права создавать БД. Например, в MS SQL Server системный администратор может предоставить пользователю main_user право на создание своей БД на данном сервере с помощью следующей команды:

```
GRANT CREATE DATABASE
ON SERVER_0
TO main user
```

По принципу иерархии пользователь main_user, создав свою БД, теперь может предоставить права на создание или изменение любых объектов в этой БД другим пользователям.

В СУБД, которые поддерживают однобазовую архитектуру, такие разрешения недопустимы (Oracle), но пользователи могут работать на уровне подсхемы (части таблиц БД и связанных с ними объектов). Поэтому там вводится понятие системных привилегий. Их очень много, 80 различных привилегий.

Для того чтобы разрешить пользователю создавать объекты внутри этой БД, используется понятие **системной привилегии**, которая может быть назначена одному или нескольким пользователям. Они выдаются только на действия и конкретный тип объекта. Поэтому, если системный администратор предоставил пользователю право создания таблиц (CREATE TABLE), то для того чтобы он мог создать триггер для таблицы, ему необходимо предоставить еще одну системную привилегию CREATE TRIGGER.

Система защиты в Oracle считается одной из самых мощных, но это имеет и обратную сторону — она весьма сложная.

Проверка полномочий

Второй задачей при работе с БД, как указывалось ранее, является **проверка полномочий пользователей**. Полномочия пользователей хранятся в специальных системных таблицах, и их проверка осуществляется ядром СУБД при выполнении каждой операции. Логически для каждого пользователя и каждого объекта в БД строится некоторая условная матрица, где по одному измерению расположены объекты, а по другому — пользователи. На пересечении каждого столбца и каждой строки расположен перечень разрешенных операций для данного пользователя над данным объектом. С первого взгляда кажется, что эта модель проверки достаточно устойчивая. Но сложность возникает тогда, когда мы используем **косвенное обращение** к объектам. Например, пользователю user_N не разрешен доступ к таблице Tab1, но этому пользователю разрешен запуск хранимой процедуры SP_N, которая делает выборку из этого объекта. По умолчанию все хранимые процедуры запускаются под именем их владельца.

Такие проблемы должны решаться организационными методами.
В любом случае проблема защиты никогда не была чисто технической задачей, это комплекс организационно-технических мероприятий, которые должны обеспечить максимальную конфиденциальность информации, хранимой в БД.
Кроме того, при работе в сети существует еще проблема проверки подлинности полномочий.
Эта проблема состоит в следующем. Допустим, процессу 1 даны полномочия по работе с БД, а процессу 2 такие полномочия не даны. Тогда напрямую процесс 2 не может обратиться к БД, но он может обратиться к процессу 1 и через него получить доступ к информации из БД. Поэтому в безопасной среде должна присутствовать модель проверки подлинности, которая обеспечивает подтверждение заявленных пользователями или процессами идентификаторов. Проверка полномочий приобрела еще большее значение в условиях массового распространения распределенных вычислений. При существующем высоком уровне связности вычислительных систем необходимо контролировать все обращения к системе.

Модель безопасности, основанная на базовых механизмах проверки полномочий и проверки подлинности, не решает таких проблем, как украденные пользовательские идентификаторы и пароли или злонамеренные действия некоторых пользователей, обладающих полномочиями, — например, когда программист, работающий над учетной системой, имеющей полный доступ к учетной базе данных, встраивает в код программы «Троянского коня» с целью хищения или намеренного изменения информации, хранимой в БД.

SQL Server 2000 поддерживает 3 режима проверки при определении прав пользователя:

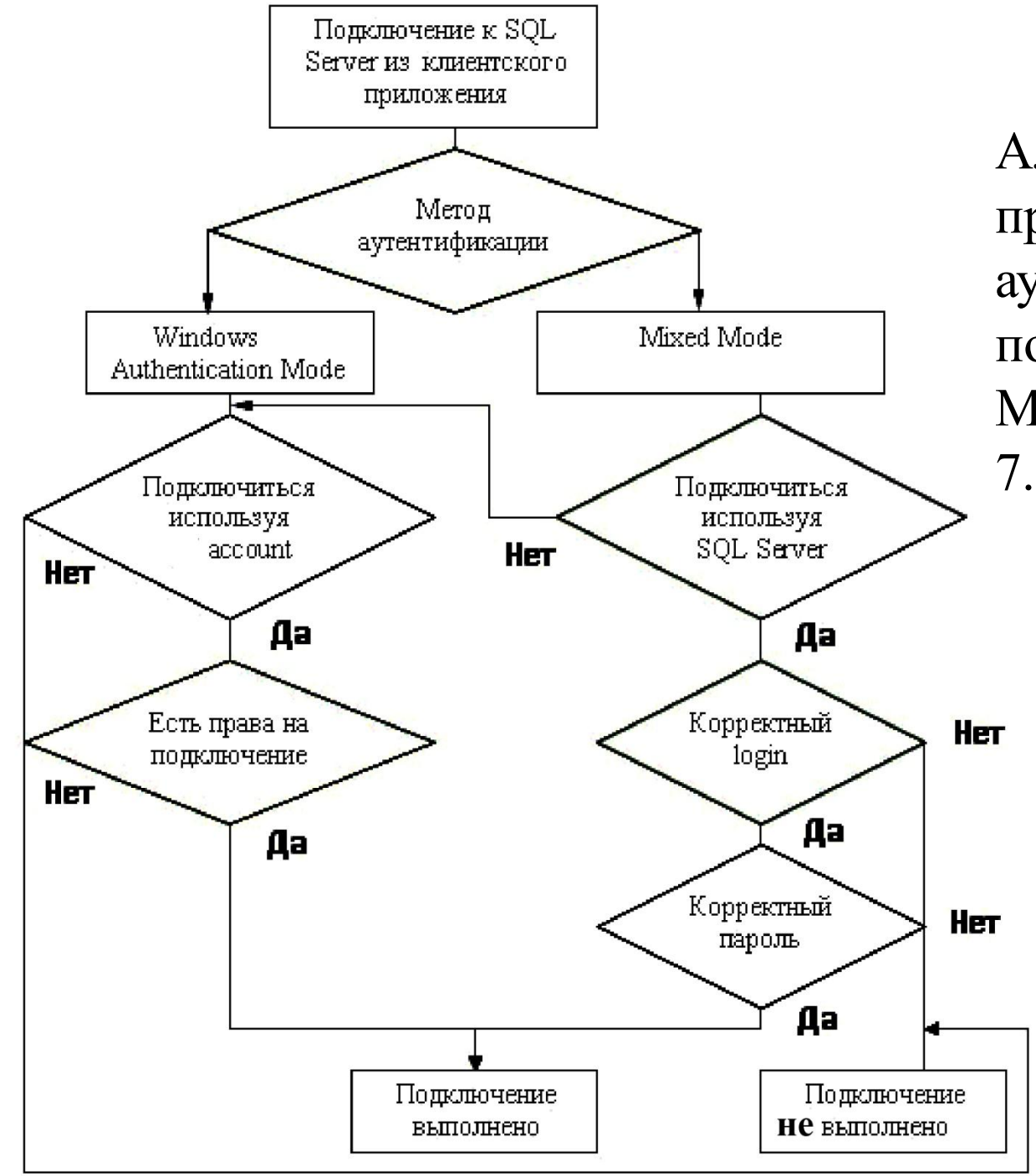
- Стандартный (standard).
- Интегрированный (integrated security).
- Смешанный (mixed).

Стандартный режим защиты предполагает, что каждый пользователь должен иметь учетную запись как пользователь домена NT Server. Учетная запись пользователя домена включает имя пользователя и его индивидуальный пароль. Пользователи доменов могут быть объединены в группы. Как пользователь домена пользователь получает доступ к определенным ресурсам домена. Для доступа к SQL Server пользователь должен иметь учетную запись пользователя MS SQL Server. Эта учетная запись должна включать уникальное имя пользователя сервера и его пароль. При подключении к операционной среде пользователь задает свое имя и пароль пользователя домена. При подключении к серверу баз данных пользователь задает свое уникальное имя пользователя SQL Server и свой пароль.

Интегрированный режим предполагает, что для пользователя задается только одна учетная запись в операционной системе, как пользователя домена, а SQL Server идентифицирует пользователя по его данным в этой учетной записи. В этом случае пользователь задает только одно свое имя и один пароль.

В случае смешанного режима часть, пользователей может быть подключена к серверу с использованием стандартного режима, а часть с использованием интегрированного режима.

Алгоритм проверки аутентификации пользователя в MS SQL Server 2000 приведен на рис



Алгоритм проверки аутентификации пользователя в MS SQL Server 7.0

При попытке подключения к серверу БД сначала проверяется, какой метод аутентификации определен для данного пользователя. Если определен Windows Authentication Mode, то далее проверяется, имеет ли данный пользователь домена доступ к ресурсу SQL Server, если он имеет доступ, то выполняется попытка подключения с использованием имени пользователя и пароля, определенных для пользователя домена; если данный пользователь имеет права подключения к SQL Server, то подключение выполняется успешно, в противном случае пользователь получает сообщение о том, что данному пользователю не разрешено подключение к SQL Server.
При использовании смешанного режима аутентификации SQL Server проводит последовательную проверку имени пользователя (login) и его пароля (password); если эти параметры заданы корректно, то подключение завершается успешно, в противном случае пользователь получает сообщение о невозможности подключиться к SQL Server.
Для СУБД Oracle всегда используется в дополнение к имени пользователя и пароля в операционной среде его имя и пароль для работы с сервером БД.