**Основные методы защиты данных.**

**Управление пользователями.**

Рассматривается система безопасности, принятая в языке SQL. Излагаются общие правила разграничения доступа. Описываются режимы аутентификации и компоненты структуры безопасности (пользователи, роли баз данных), администрирование системы безопасности (создание учетных записей и управление ими, управление пользователями и ролями). Дается определение прав пользователя на доступ к объектам базы данных. Рассматриваются неявные права, вопросы запрета доступа и неявного отклонения доступа, а также конфликты доступа.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Содержание

• Управление пользователями базы данных

• Управление пользователями в среде MS SQL Server

o Режимы аутентификации

o Администрирование системы безопасности

• Управление доступом к данным

o Определение привилегий в стандарте языка

o Предоставление привилегий пользователям

o Отмена предоставленных пользователям привилегий

• Реализация прав на доступ к объектам баз данных в среде MS SQL Server

o Категории прав в среде MS SQL Server

o Предоставление прав

o Права на выполнение команд SQL

o Неявные права

o Запрещение доступа

o Неявное отклонение доступа

o Конфликты доступа

Управление пользователями базы данных

Стабильная система управления пользователями – обязательное условие безопасности данных, хранящихся в любой реляционной СУБД. В языке SQL не существует единственной стандартной команды, предназначенной для создания пользователей базы данных – каждая реализация делает это по-своему. В одних реализациях эти специальные команды имеют определенное сходство, в то время как в других их синтаксис имеет существенные отличия. Однако независимо от конкретной реализации все основные принципы одинаковы.

Управление пользователями в среде MS SQL Server

Рассмотрим вопрос создания пользователей в среде MS SQL Server.

После проектирования логической структуры базы данных, связей между таблицами, ограничений целостности и других структур необходимо определить круг пользователей, которые будут иметь доступ к базе данных.

В системе SQL-сервер организована двухуровневая настройка ограничения доступа к данным. На первом уровне необходимо создать так называемую учетную запись пользователя (login), что позволяет ему подключиться к самому серверу, но не дает автоматического доступа к базам данных. На втором уровне для каждой базы данных SQL-сервера на основании учетной записи необходимо создать запись пользователя. На основе прав, выданных пользователю как пользователю базы данных (user), его регистрационное имя (login) получает доступ к соответствующей базе данных. В разных базах данных login одного и того же пользователя может иметь одинаковые или разные имена user с разными правами доступа. Иначе говоря, с помощью учетной записи пользователя осуществляется подключение к SQL-серверу, после чего определяются его уровни доступа для каждой базы данных в отдельности.

В системе SQL-сервер существуют дополнительные объекты – роли, которые определяют уровень доступа к объектам SQL-сервера. Они разделены на две группы: назначаемые для учетных записей пользователя сервера и используемые для ограничения доступа к объектам базы данных.

Итак, на уровне сервера система безопасности оперирует следующими понятиями:

• аутентификация;

• учетная запись;

• встроенные роли сервера.

На уровне базы данных применяются следующие понятия;

• пользователь базы данных;

• фиксированная роль базы данных;

• пользовательская роль базы данных.

Режимы аутентификации

SQL Server предлагает два режима аутентификации пользователей:

• режим аутентификации средствами Windows NT/2000;

• смешанный режим аутентификации (Windows NT Authentication and SQL Server Authentication).

Администрирование системы безопасности

Для создания пользователя в среде MS SQL Server следует предпринять следующие шаги:

1. Создать в базе данных учетную запись пользователя, указав для него пароль и принятое по умолчанию имя базы данных (процедура sp\_addlogin).

2. Добавить этого пользователя во все необходимые базы данных (процедура sp\_adduser).

3. Предоставить ему в каждой базе данных соответствующие привилегии (команда GRANT) .

Создание новой учетной записи может быть произведено с помощью системной хранимой процедуры:

sp\_addlogin

[@login=] 'учетная\_запись'

[, [@password=] 'пароль']

[, [@defdb=] 'база\_данных\_по\_умолчанию']

После завершения аутентификации и получения идентификатора учетной записи (login ID) пользователь считается зарегистрированным, и ему предоставляется доступ к серверу. Для каждой базы данных, к объектам которой он намерен получить доступ, учетная запись пользователя (login) ассоциируется с пользователем (user) конкретной базы данных, что осуществляется посредством процедуры:

sp\_adduser

[@loginame=] 'учетная\_запись'

[, [@name\_in\_db=] 'имя\_пользователя']

[, [@grpname=] 'имя\_роли']

Отобразить учетную запись Windows NT в имя пользователя позволяет хранимая процедура:

sp\_grantdbaccess

[@login=] ‘учетная\_запись’

[, [@name\_in\_db=]‘имя\_пользователя’]

Пользователь, который создает объект в базе данных (таблицу, хранимую процедуру, просмотр), становится его владельцем. Владелец объекта (database object owner dbo) имеет все права доступа к созданному им объекту. Чтобы пользователь мог создать объект, владелец базы данных (dbo) должен предоставить ему соответствующие права. Полное имя создаваемого объекта включает в себя имя создавшего его пользователя.

Владелец объекта не имеет специального пароля или особых прав доступа. Он неявно имеет полный доступ, но должен явно предоставить доступ другим пользователям.

SQL Server позволяет передавать права владения от одного пользователя другому с помощью процедуры:

sp\_changeobjectowner

[@objname=] ‘имя\_объекта’

[@newowner=] ‘имя\_владельца’

Роль позволяет объединить в одну группу пользователей, выполняющих одинаковые функции.

В SQL Server реализовано два вида стандартных ролей: на уровне сервера и на уровне баз данных. При установке SQL Server создаются фиксированные роли сервера (например, sysadmin с правом выполнения любых функций SQL-сервера) и фиксированные роли базы данных (например, db\_owner с правом полного доступа к базе данных или db\_accessadmin с правом добавления и удаления пользователей). Среди фиксированных ролей базы данных существует роль public, которая имеет специальное назначение, поскольку ее членами являются все пользователи, имеющие доступ к базе данных.

Можно включить любую учетную запись SQL Server (login) или учетную запись Windows NT в любую роль сервера.

Роли базы данных позволяют объединять пользователей в одну административную единицу и работать с ней как с обычным пользователем. Можно назначить права доступа к объектам базы данных для конкретной роли, при этом автоматически все члены этой роли наделяются одинаковыми правами.

В роль базы данных можно включить пользователей SQL Server, роли SQL Server, пользователей Windows NT.

Различные действия по отношению к роли осуществляются при помощи специальных процедур:

• создание новой роли:

• sp\_addrole

• [@rolename=] 'имя\_роли'

• [, [@ownername=] 'имя\_владельца']

• добавление пользователя к роли:

• sp\_addrolemember

• [@rolename=] 'имя\_роли',

• [@membername=] 'имя\_пользователя'

• удаление пользователя из роли:

• sp\_droprolemember

• [@rolename=] 'имя\_роли',

• [@membername=] 'имя\_пользователя'

• удаление роли:

• sp\_droprole

• [@rolename=] 'имя\_роли'•

Управление доступом к данным

Определение привилегий в стандарте языка

Каждая СУБД должна поддерживать механизм, гарантирующий, что доступ к базе данных смогут получить только те пользователи, которые имеют соответствующее разрешение. Язык SQL включает операторы GRANT и REVOKE, предназначенные для организации защиты таблиц в базе данных. Механизм защиты построен на использовании идентификаторов пользователей, предоставляемых им прав владения и привилегий.

Идентификатором пользователя называется обычный идентификатор языка SQL, применяемый для обозначения некоторого пользователя базы данных. Каждому пользователю должен быть назначен собственный идентификатор, присваиваемый администратором базы данных. Из очевидных соображений безопасности идентификатор пользователя, как правило, связывается с некоторым паролем. Каждый выполняемый СУБД SQL-оператор выполняется от имени какого-либо пользователя. Идентификатор пользователя определяет, на какие объекты базы данных пользователь может ссылаться и какие операции с этими объектами он имеет право выполнять.

Каждый созданный в среде SQL объект имеет своего владельца, который изначально является единственной персоной, знающей о существовании данного объекта и имеет право выполнять с ним любые операции.

Привилегиями, или правами, называются действия, которые пользователь имеет право выполнять в отношении данной таблицы базы данных или представления. В стандарте SQL определяется следующий набор привилегий:

• SELECT – право выбирать данные из таблицы;

• INSERT – право вставлять в таблицу новые строки;

• UPDATE – право изменять данные в таблице;

• DELETE – право удалять строки из таблицы;

• REFERENCES – право ссылаться на столбцы указанной таблицы в описаниях требований поддержки целостности данных;

• USAGE – право использовать домены, проверки и наборы символов.

Привилегии INSERT и UPDATE могут ограничиваться лишь отдельными столбцами таблицы, в этом случае пользователю разрешается модифицировать значения только указанных столбцов. Аналогичным образом привилегия REFERENCES может распространяться исключительно на отдельные столбцы таблицы, что позволит использовать их имена в формулировках требований защиты целостности данных – например, в предложениях CHECK и FOREIGN KEY, входящих в определение других таблиц, тогда как применение для подобных целей остальных столбцов будет запрещено.

Когда пользователь с помощью оператора CREATE TABLE создает новую таблицу, он автоматически становится ее владельцем и получает по отношению к ней полный набор привилегий, которых остальные пользователи исходно не имеют. Чтобы обеспечить им доступ, владелец должен явным образом предоставить необходимые права, для чего используется оператор GRANT.

Создавая представление с помощью оператора CREATE VIEW, пользователь автоматически становится владельцем этого представления и также получает полный набор прав. Для создания представления пользователю достаточно иметь привилегию SELECT для всех входящих в него таблиц и привилегию REFERENCES для всех столбцов, упоминаемых в определении этого представления. Привилегии INSERT, UPDATE и DELETE в отношении созданного представления пользователь получит только в том случае, если имеет соответствующие привилегии в отношении всех используемых в представлении таблиц.

Предоставление привилегий пользователям

Оператор GRANT применяется для предоставления привилегий в отношении поименованных объектов базы данных указанным пользователям. Обычно его использует владелец таблицы с целью предоставления доступа к ней другим пользователям. Оператор GRANT имеет следующий формат:

<предоставление\_привилегий>::=

GRANT {<привилегия>[,...n] |

ALL PRIVILEGES}

ON имя\_объекта

TO {<идентификатор\_пользователя>

[,...n]| PUBLIC}

[ WITH GRANT OPTION]

Параметр <привилегия> представляет собой:

<привилегия>::=

{SELECT | DELETE | INSERT

[(имя\_столбца[,...n])]

| UPDATE [(имя\_столбца[,...n])]}

| REFERENCES [(имя\_столбца[,...n])] |

USAGE }

Из соображений упрощения в операторе GRANT можно указать ключевое слово ALL PRIVILEGES, что позволит предоставить указанному пользователю все существующие привилегии без необходимости их перечисления. Кроме того, в этом операторе может указываться ключевое слово PUBLIC, означающее предоставление доступа указанного типа не только всем существующим пользователям, но также и всем тем, кто будет определен в базе данных впоследствии.

Параметр имя\_объекта может использоваться как имя таблицы базы данных, представления, домена, набора символов, проверки.

Благодаря параметру WITH GRANT OPTION, указанные в операторе GRANT пользователи имеют право передавать все предоставленные им в отношении указанного объекта привилегии другим пользователям, которые, в свою очередь, будут наделены точно таким же правом передачи своих полномочий. Если данный параметр не будет указан, получатель привилегии не сможет передать свои права другим пользователям. Таким образом, владелец объекта может четко контролировать, кто получил право доступа к объекту и какие полномочия ему предоставлены.

Отмена предоставленных пользователям привилегий

В языке SQL для отмены привилегий, предоставленных пользователям посредством оператора GRANT, используется оператор REVOKE. С помощью этого оператора могут быть отменены все или некоторые из привилегий, полученных указанным пользователем раньше. Оператор REVOKE имеет следующий формат:

<отмена\_привилегий>::=

REVOKE[GRANT OPTION FOR]

{<привилегия>[,...n]

| ALL PRIVILEGES}

ON имя\_объекта

FROM {<идентификатор\_пользователя>

[,...n]| PUBLIC}

[RESTRICT | CASCADE]

Ключевое слово ALL PRIVILEGES означает, что для указанного пользователя отменяются все привилегии, предоставленные ему ранее тем пользователем, который ввел данный оператор. Необязательная фраза GRANT OPTION FOR позволяет для всех привилегий, переданных в исходном операторе GRANT фразой WITH GRANT OPTION, отменять возможность их передачи независимо от самих привилегий.

Если в операторе указано ключевое слово RESTRICT, успешное выполнение команды REVOKE возможно лишь в том случае, когда перечисленные в операторе привилегии не могут послужить причиной появления у каких-либо других пользователей так называемых "оставленных" привилегий. С помощью параметра CASCADE удаляются все привилегии, которые иначе могли бы остаться у других пользователей.

"Оставленными" являются привилегии, сохранившиеся у пользователя, которому они в свое время были предоставлены с помощью параметра GRANT OPTION.

Поскольку наличие привилегии необходимо для создания определенных объектов, вместе с ее удалением можно лишиться права, за счет использования которого был образован тот или иной объект (подобные объекты называются "брошенными"). Если в результате выполнения оператора REVOKE могут появиться брошенные объекты (например, представления), право будет отменено при условии, что в нем не указывается ключевое слово CASCADE. Если ключевое слово CASCADE в операторе присутствует, то для любых брошенных объектов, возникающих при выполнении исходного оператора REVOKE, будут автоматически выданы операторы DROP.

Привилегии, которые были предоставлены указанному пользователю другими пользователями, не могут быть затронуты оператором REVOKE. Следовательно, если другой пользователь также предоставил данному пользователю удаляемую привилегию, то право доступа к соответствующей таблице у указанного пользователя сохранится. Например, пусть пользователь A и пользователь Е имели право INSERT на таблицу Товар. Пользователь А предоставил пользователю B привилегию INSERT для таблицы Товар, причем с указанием WITH GRANT OPTION (этап 1). Пользователь B передал эту привилегию пользователю C (этап 2). Затем пользователь C получил ее же от пользователя E (этап 3). Далее пользователь C предоставил упомянутую привилегию пользователю D (этап 4). Когда пользователь A отменяет привилегию INSERT для пользователя B, она не может быть отменена и для пользователя C, поскольку ранее он уже получил ее от пользователя E. Если бы пользователь E не предоставил данной привилегии пользователю C, то удаление привилегии пользователя B имело бы следствием каскадное удаление привилегий для пользователей C и D (см. таблицу 1).

Реализация прав на доступ к объектам баз данных в среде MS SQL Server

Категории прав в среде MS SQL Server

При подключении к SQL Server все возможные действия пользователей определяются правами (привилегиями, разрешениями), выданными их учетной записи, группе или роли, в которых они состоят.

Права можно разделить на три категории:

• права на доступ к объектам;

• права на выполнение команд;

• неявные права.

Таблица 1.

Пользователь A Пользователь B Пользователь C Пользователь D Пользователь E

GRANT INSERT ON Товар TO B WITH GRANT OPTION Получение права

GRANT INSERT ON Товар TO C WITH GRANT OPTION Получение права от B. Получение права от E GRANT INSERT ON Товар TO C WITH GRANT OPTION

GRANT INSERT ON Товар TO D Получение права

REVOKE INSERT ON Товар TO B CASCADE Отмена права Сохранение права Сохранение права Сохранение права

Работа с данными и выполнение хранимых процедур требуют наличия класса доступа, называемого правами на доступ к объектам баз данных. Под объектами подразумеваются таблицы, столбцы таблиц, представления, хранимые процедуры.

Для различных объектов применяются разные наборы прав доступа к ним:

• SELECT, INSERT, UPDATE, DELETE, REFERENCES – для таблицы или представления;

• SELECT, UPDATE – для конкретного столбца таблицы или представления;

• EXECUTE – для хранимых процедур и функций.

Право INSERT позволяет вставлять новые строки в таблицу или представление и выдается только на уровне таблицы или представления; оно не может быть выдано на уровне столбца.

Право UPDATE выдается либо на уровне таблицы, что позволяет изменять в ней все данные, либо на уровне отдельного столбца, что разрешает изменять данные только в его пределах.

Право DELETE позволяет удалять строки из таблицы или представления, выдается только на уровне таблицы или представления, но не может быть выдано на уровне столбца.

Право SELECT разрешает выборку данных и может выдаваться как на уровне таблицы, так и на уровне отдельного столбца.

Право REFERENCES предоставляет возможность ссылаться на указанный объект. Применительно к таблицам разрешает создавать внешние ключи, ссылающиеся на первичный ключ или уникальный столбец этой таблицы.

Предоставление прав

Для управления разрешением пользователя на доступ к объектам базы данных используется команда:

<предоставление\_привилегий>::=

GRANT { ALL [ PRIVILEGES] | <привилегия>

[,...n]}

{ [( имя\_столбца [,...n])]

ON { имя\_таблицы |

имя\_просмотра} |

ON {имя\_таблицы |

имя\_просмотра }

([имя\_столбца

[,...n])]

| ON {имя\_хранимой\_процедуры |

имя\_внешней\_процедуры}}

TO { имя\_пользователя | имя\_группы |

имя\_роли} [,...n]

[WITH GRANT OPTION ]

[AS {имя\_группы | имя\_роли }]

Параметр <привилегия> представляет собой следующую конструкцию:

<привилегия>::=

{SELECT | DELETE | INSERT |

UPDATE | EXECUTE | REFERENCES }

Параметр WITH GRANT OPTION поможет пользователю, которому вы предоставляете права, назначить права на доступ к объекту другим пользователям. Его использование требует особой осторожности, поскольку при этом владелец теряет контроль над предоставлением прав на доступ другим пользователям. Лучше всего ограничить круг пользователей, обладающих возможностью управлять назначением прав.

Необязательный параметр AS {имя\_группы | имя\_роли } позволяет указать участие пользователя в роли, обеспечивающей предоставление прав другим пользователям.

Единственное право доступа, которое может быть предоставлено для хранимой процедуры, – право на ее выполнение (EXECUTE). Естественно, кроме этого владелец хранимой процедуры может просматривать и изменять ее код.

Для функции можно выдать право на ее выполнение, а кроме того, выдать право REFERENCES, что обеспечит возможность связывания функции с объектами, на которые она ссылается. Такое связывание позволит запретить внесение изменений в структуру объектов, способных привести к нарушению работы функции.

Права на выполнение команд SQL

Этот класс прав контролирует возможность создания объектов в базе данных, самой базы данных и выполнения процедуры резервного копирования. Можно использовать следующую команду для предоставления права на выполнение команд SQL:

<предоставление\_права\_выполнения>::=

GRANT {ALL | <команда>[,...n]}

TO {имя\_пользователя | имя\_группы |

имя\_роли} [,...n]

Параметр <команда> представляет собой следующую конструкцию:

<команда>::=

{CREATE DATABASE | CREATE TABLE |

CREATE VIEW | CREATE DEFAULT |

CREATE RULE | CREATE PROCEDURE

| BACKUP DATABASE |

BACKUP LOG | ALL }

Таким образом, можно предоставить право на создание базы данных, таблицы, просмотра, умолчания, правила, хранимой процедуры, резервной копии базы данных и журнала транзакций или предоставить сразу все вышеперечисленные права.

Неявные права

Выполнение некоторых действий не требует явного разрешения и доступно по умолчанию. Эти действия могут быть выполнены только членами ролей сервера или владельцами объектов в базе данных.

Неявные права не предоставляются пользователю непосредственно, он получает их лишь при определенных обстоятельствах. Например, пользователь может стать владельцем объекта базы данных, только если сам создаст объект либо если кто-то другой передаст ему право владения своим объектом. Таким образом владелец объекта автоматически получит права на выполнение любых действий с объектом, в том числе и на предоставление доступа к объекту другим пользователям. Эти права нигде не указываются, выполнять любые действия позволяет только факт владения объектом.

Запрещение доступа

Система безопасности SQL Server имеет иерархическую структуру, и поэтому роли базы данных включают в себя учетные записи и группы Windows NT, пользователей и роли SQL Server. Пользователь же, в свою очередь, может участвовать в нескольких ролях и одновременно иметь разные права доступа для разных ролей. Когда одна из ролей, в которых состоит пользователь, имеет разрешение на доступ к данным, он автоматически имеет аналогичные права. Тем не менее, если возникает необходимость, пользователю можно запретить доступ к данным или командам, тогда аннулируются все разрешения на доступ, полученные им на любом уровне иерархии. При этом гарантируется, что доступ останется запрещенным независимо от разрешений, предоставленных на более высоком уровне.

Для запрещения доступа к объектам базы данных используется команда:

<запрещение\_доступа>::=

DENY {ALL [PRIVILEGES]| | <привилегия>

[,...n]}

{ [(имя\_столбца [,...n])]

ON { имя\_таблицы |

имя\_просмотра}

| ON {имя\_таблицы | имя\_просмотра }

[имя\_столбца [,...n])]

| ON {имя\_хранимой\_процедуры |

имя\_внешней\_процедуры}}

TO {имя\_пользователя | имя\_группы |

имя\_роли}

[,...n]

[CASCADE ]

Параметр CASCADE позволяет отзывать права не только у конкретного пользователя, но также и у всех тех, кому он предоставил аналогичные права.

Для запрещения выполнения команд SQL применяется оператор:

<запрещение\_выполнения>::=

DENY {ALL | <команда>[,...n]}

TO {имя\_пользователя | имя\_группы |

имя\_роли} [,...n]

Неявное отклонение доступа

Неявное отклонение подобно запрещению доступа с тем отличием, что оно действует только на том уровне, на котором определено. Если пользователю на определенном уровне неявно отклонен доступ, он все же может получить его на другом уровне иерархии через членство в роли, имеющей право просмотра. По умолчанию доступ пользователя к данным неявно отклонен. Для неявного отклонения доступа к объектам базы данных используется команда:

<неявное\_отклонение\_доступа>::=

REVOKE [GRANT OPTION FOR]

{ALL [ PRIVILEGES]| | <привилегия>

[,...n]}

{ [(имя\_столбца [,...n])] ON

{ имя\_таблицы | имя\_просмотра}

| ON {имя\_таблицы |

имя\_просмотра }

[имя\_столбца [,...n])]

| ON {имя\_хранимой\_процедуры |

имя\_внешней\_процедуры}}

TO | FROM {имя\_пользователя |

имя\_группы |

имя\_роли}[,...n]

[CASCADE ]

[AS {имя\_группы | имя\_роли }]

Для неявного отклонения разрешения на выполнение команд SQL используется следующая команда:

<неявное\_отклонение\_разрешения>::=

REVOKE {ALL | <команда>[,...n]}

FROM {имя\_пользователя | имя\_группы |

имя\_роли}[,...n]

Смысл параметров аналогичен параметрам команд GRANT и DENY. Параметр GRANT OPTION FOR используется, когда необходимо отозвать право, предоставленное параметром WITH GRANT OPTION команды GRANT. Пользователь сохраняет разрешение на доступ к объекту, но теряет возможность предоставлять это разрешение другим пользователям.

Конфликты доступа

Разрешения, предоставленные роли или группе, наследуются их членами. Хотя пользователю может быть предоставлен доступ через членство в одной роли, роль другого уровня может иметь запрещение на действие с объектом. В таком случае возникает конфликт доступа.

При разрешении конфликтов доступа SQL Server руководствуется следующим принципом: разрешение на предоставление доступа имеет самый низкий приоритет, а на запрещение доступа – самый высокий. Это значит, что доступ к данным может быть получен только явным его предоставлением при отсутствии запрещения доступа на любом другом уровне иерархии системы безопасности. Если доступ явно не предоставлен, пользователь не сможет работать с данными.

Пример 1. Создать новую базу данных, нового пользователя для этой базы данных, предоставив ему все права.

-- создание администратором новой

-- базы данных

CREATE DATABASE basa\_user

-- создание нового пользователя с

-- именем UserA и паролем ‘123’

-- базой данных по умолчанию для

-- пользователя UserA будет база

-- с именем basa\_user.

sp\_addlogin 'UserA','123','basa\_user'

-- переход в базу данных basa\_user

USE basa\_user

-- добавление в текущую базу данных

-- (basa\_user) пользователя с именем

-- userA

sp\_adduser 'UserA'

-- предоставление пользователю userA

-- в базе данных basa\_user всех прав

GRANT ALL TO UserA

Пример 1. Создание новой базы данных, нового пользователя для этой базы данных, с предоставлением ему всех прав. (html, txt)

Пример 2. Использование ролей.

Создадим роль stud и включим в эту роль двух пользователей user1 и user2:

sp\_addrole 'stud'

sp\_addrolemember 'stud','user1'

sp\_addrolemember 'stud','user2'

Предоставим права роли stud и непосредственно пользователю user2:

GRANT SELECT, INSERT ON Товар TO stud

GRANT SELECT, INSERT ON Товар TO user2

После выполнения этих команд пользователи user1 и user2 могут выполнять команды выборки и добавления записи в таблицу Товар.

Приостановим право на выполнение вставки в таблицу Товар для роли stud:

REVOKE INSERT ON Товар TO stud

После выполнения предыдущей команды пользователь user1 теряет право вставки записи, а user2 сохраняет это право, поскольку право вставки предоставлено ему явно.

Выполним команду

DENY INSERT ON Товар TO stud.

После выполнения этой команды оба пользователя лишаются права вставки в таблицу Товар.