Лекция № 13

SQL и безопасность

При хранении данных в какой-либо СУБД одной из главных задач пользователя является обеспечение безопасности этих данных. Проблема безопасности особенно остро стоит в реляционных базах данных, поскольку интерактивный БОБ позволяет легко получить к ним доступ. Требования, предъявляемые к системе безопасности в типичной реляционной базе данных, довольно разнообразны.

• Доступ к данным отдельной таблицы должен быть разрешен одним пользователям и запрещен другим.

• Одним пользователям должно быть позволено изменять данные в некоторой таблице, а другим — осуществлять только выборку данных из нее.

• К ряду таблиц доступ должен производиться только к отдельным столбцам.

• Определенным пользователям должно быть запрещено обращение к некоторой таблице с помощью интерактивного БОБ, но разрешено пользоваться прикладными программами, изменяющими эту таблицу.

В настоящей главе рассматривается схема обеспечения безопасности базы данных с использованием БОБ, позволяющая реализовать все упомянутые выше виды защиты данных в реляционной СУБД.

Принципы защиты данных, применяемые в БОЬ

За реализацию системы безопасности отвечает программное обеспечение СУБД. Язык БОБ является фундаментом системы безопасности реляционной СУБД: требования, предъявляемые к системе защиты информации в базе данных, формулируются с помощью инструкций БОБ. С защитой данных в БОБ связаны три основные концепции.

• Пользователи. Действующими лицами в базе данных являются пользователи. Каждый раз, когда СУБД извлекает, вставляет, удаляет или об-

новляет данные, она делает это от имени какого-то пользователя. СУБД выполнит требуемое действие или откажется его выполнять в зависимости от того, какой пользователь запрашивает это действие.

Объекты базы данных. Эти объекты являются теми элементами, защита которых может осуществляться посредством БОЕ Обычно обеспечивается защита таблиц и представлений, но и другие объекты, такие как формы, прикладные программы и целые базы данных, также могут быть защищены. Большинству пользователей разрешается использовать одни объекты базы данных и запрещается использовать другие.

Привилегии. Это— права пользователя на проведение тех или иных действий над определенным объектом базы данных. Например, пользователю может быть разрешено извлекать строки из некоторой таблицы и добавлять их в нее, но запрещено удалять или обновлять строки этой таблицы. У другого пользователя может быть иной набор привилегий.

Для введения элементов системы безопасности применяется инструкция GRANT, с помощью которой тем или иным пользователям предоставляются определенные привилегии на использование тех или иных объектов базы данных. Вот, например, инструкция GRANT, позволяющая Сэму Кларку (Sam Clark) извлекать данные из таблицы OFFICES учебной базы данных и добавлять их в нее.

Разрешить Сэму Кларку извлекать данные из таблицы OFFICES и добавлять их в нее,

```
GRANT SELECT, INSERT
ON OFFICES
TO SAM;
```

В инструкции GRANT задается комбинация идентификатора пользователя (SAM), объекта (таблица OFFICES) и привилегий (SELECT и INSERT). Предоставленные привилегии можно позднее аннулировать посредством инструкции REVOKE.

Отменить привилегии, предоставленные ранее Сэму Кларку.

```
REVOKE SELECT, INSERT
ON OFFICES
FROM SAM;
```

Инструкции GRANT и REVOKE более подробно описываются далее в данной главе.

Идентификаторы пользователей

Каждому пользователю в реляционной базе данных присваивается идентификатор — краткое имя, однозначно определяющее пользователя для программного обеспечения СУБД. Эти идентификаторы являются основой системы безопасности. Каждая инструкция SQL выполняется в СУБД от имени конкретного пользователя. От его идентификатора зависит, будет ли разрешено или запрещено выполнение конкретной инструкции. В промышленной базе данных идентификатор пользователя назначается ее администратором. В базе данных на персональном компьютере может быть только один идентификатор пользователя, который обозначает пользователя, создавшего базу данных и являющегося ее владельцем.

На практике ограничения на имена идентификаторов зависят от реализации СУБД. Стандарт SQL1 позволял идентификатору пользователя иметь длину до 18 символов и требовал, чтобы он был допустимым SQL-именем. В некоторых СУБД для мэйнфреймов длина идентификаторов ограничивалась восемью символами. В Sybase и SQL Server идентификатор пользователя может иметь до 30 символов. Если важна переносимость, то лучше всего, чтобы у идентификатора пользователя было не более восьми символов. На рис. 15.2 показаны различные пользователи, работающие с учебной базой данных, и типичные идентификаторы, присвоенные им. Обратите внимание на то, что всем пользователям в отделе обработки заказов можно присвоить один и тот же идентификатор, так как все они имеют идентичные привилегии.

В стандарте ANSI/ISO вместо термина "идентификатор пользователя" (user-id) употребляется термин *идентификатор авторизации* (authorization-id), и он встречается иногда в документации по SQL. С технической точки зрения второй термин является более правильным, так как роль идентификатора заключается в определении прав или привилегий. Бывают ситуации (как на рис.), когда имеет смысл присвоить нескольким пользователям одинаковый идентификатор. В других ситуациях один пользователь может пользоваться двумя или тремя различными идентификаторами. В промышленной базе данных идентификатор прав доступа может относиться к программам и группам программ, а не к отдельным людям. В каждой из этих ситуаций термин "идентификатор авторизации" является более точным и ясным, чем термин "идентификатор пользователя". Однако наиболее распространена ситуация, когда каждому пользователю присваивается свой идентификатор, поэтому в документации большинства реляционных СУБД употребляется термин "идентификатор пользователя".

Аутентификация пользователей

Согласно стандарту SQL, безопасность базы данных обеспечивается с помощью идентификаторов пользователей. Однако в стандарте ничего не говорится о механизме связи идентификатора пользователя с инструкциями SQL. Например, если вы вводите инструкции SQL в интерактивном режиме, как СУБД определит, с каким идентификатором пользователя связаны эти инструкции? На сервере баз данных может существовать ежевечерне выполняющаяся программа генерации отчетов; каков в этом случае идентификатор, если нет никакого пользователя? Наконец, как обрабатываются запросы к базе данных по сети, если на локальном компьютере у пользователя может быть один идентификатор, а на удаленном компьютере — другой?

В большинстве коммерческих реляционных СУБД идентификатор пользователя создается для каждого *сеанса (сессии)* связи с базой данных. В интерактивном режиме сеанс начинается, когда пользователь запускает интерактивную программу формирования запросов, и продолжается до тех пор, пока пользователь не выйдет из программы или не введет команду смены пользователя. В приложении, использующем программный SQL, сеанс начинается, когда приложение подключается к СУБД, и заканчивается по завершении приложения. В течение сеанса все инструкции SQL ассоциируются с идентификатором пользователя, установленным для этого сеанса. Однако в современных прикладных системах возможно также, что программа устанавливает несколько соединений с базой данных и выбирает одно из них для передачи инструкций SQL.

Как правило, в начале сеанса необходимо ввести как идентификатор пользователя, так и связанный с ним пароль. Пароль служит для подтверждения того, что пользователь действительно имеет право работать под введенным идентификатором. Однако ряд СУБД поддерживает аутентификацию операционной системы, т.е. СУБД получает мандат пользователя от операционной Системы, без необходимости вводить пароль или выполнять иную аутентификацию. Хотя идентификаторы и пароли применяются в большинстве реляционных СУБД, способ, которым пользователь вводит свой идентификатор и

пароль, зависит от конкретной СУБД.

В некоторых СУБД , в особенности тех, которые доступны для разных операционных систем, реализована своя собственная система безопасности с идентификаторами пользователей и паролями. Например, когда в СУБД Oracle вы работаете с интерактивной SQL-программой SQLPLUS, можете указать имя пользователя и соответствующий пароль в командной строке.

SQLPLUS SCOTT/TIGER

Заметим, что такой ввод пароля в командной строке использовать не рекомендуется, так как в этом случае пароль не шифруется и может быть перехвачен кем угодно в системе. Гораздо лучше опустить пароль (и разделительную косую черту) и позволить SQLPLUS самому запросить у вас пароль.

Интерактивный SQL-модуль СУБД Sybase, который называется ISQL, также может принимать имя пользователя и пароль в командной строке.

ISQL -U SCOTT -P TIGER

В каждом случае, прежде чем начать интерактивный сеанс связи, СУБД проверяет достоверность идентификатора пользователя (SCOTT) и пароля (TIGER). И вновь, лучше опустить пароль и позволить ISQL самому запросить его у вас. Старые версии SQL Server также поддерживают ISQL, но более новые используют для доступа к командной строке СУБД несколько отличающуюся утилиту OSQL.

Во многих других СУБД, включая Ingres и Informix, в качестве идентификаторов пользователей используются имена пользователей, регистрируемые в операционной системе. Например, когда вы регистрируетесь в вычислительной системе на основе UNIX, то вводите имя пользователя и пароль. Чтобы запустить затем интерактивный модуль в СУБД Ingres, вы просто даете команду

ISQL SALESDB

где SALESDB — это имя базы данных Ingres, с которой вы хотите работать. Ingres автоматически получает имя пользователя UNIX и делает его вашим идентификатором пользователя в текущем сеансе работы с СУБД. Таким образом, вам не тре-

буется задавать отдельный идентификатор пользователя и пароль для базы данных. Аналогичный прием применяется и в интерактивном SQL-модуле СУБД DB2, работающем в операционной системе MVS/TSO. Учетная запись TSO автоматически становится идентификатором пользователя интерактивного SQL-сеанса.

Большинство современных СУБД имеет для доступа к базе данных приложения с графическим пользовательским интерфейсом, такие как SQL Server Management Studio, DB2 UDB Command Editor или Oracle SQL Developer. Эти инструменты при подключении к базе данных запрашивают идентификатор пользователя и пароль.

Те же средства защиты используются при программном доступе к базе данных, так что СУБД должна определять и аутентифицировать идентификатор пользователя для каждой прикладной программы, пытающейся получить доступ к базе данных. И вновь, способы и правила применения идентификаторов пользователей меняются от одной СУБД к другой. Чаще всего программа в начале сеанса выдает пользователю диалоговое окно с запросом идентификатора и пароля. Специализированные или написанные пользователями программы могут использовать идентификатор, жестко закодированный в программе.

Стандарт SQL позволяет программе использовать идентификатор авторизации, связанный с конкретным набором инструкций SQL (такой набор называется люЭу- *лем),* а не идентификатор конкретного пользователя, запустившего программу. Такой механизм обеспечивает возможность выполнения программой различных задач от имени различных пользователей, даже если при прочих условиях этим пользователям запрещен доступ к соответствующей информации. Это удобная возможность, которая находит свое применение в основных реализациях SQL. Специфика безопасности SQL в случае программного обращения к базам данных рассматривается в главе 17, "Встроенный SQL".

Группы пользователей

В больших производственных базах данных часто имеются группы пользователей со схожими задачами. Например, в учебной базе данных три человека в отделе обработки заказов образуют естественную группу пользователей; два человека в финансовом отделе образуют другую естественную группу. В пределах каждой группы все пользователи работают с одинаковыми данными и должны иметь идентичные привилегии.

Согласно стандарту ANSI/ISO, с группами пользователей можно поступить одним из трех способов.

• Каждому члену группы можно присвоить один и тот же идентификатор пользователя, как показано на рис. 15.2. Это упрощает управление системой безопасности, так как позволяет установить привилегии доступа к данным один раз (в связи с тем, что идентификатор пользователя один). Однако в этом случае людей, совместно использующих один идентификатор пользователя, нельзя будет различить ни на дисплее системного оператора, ни в отчетах СУБД.

• Всем членам группы можно присвоить разные идентификаторы пользователя. Это позволит вам дифференцировать пользователей в отчетах СУБД и устанавливать в дальнейшем различные привилегии для отделы

ных пользователей. Однако привилегии придется устанавливать для каждого пользователя индивидуально, что может быть утомительно и увеличивает вероятность возникновения ошибок.

• В тех, наиболее современных, СУБД, которые поддерживают эту возможность, можно создать *роль,* содержащую требуемые привилегии. Роль представляет собой именованную коллекцию привилегий. Можно назначить каждому пользователю его собственный идентификатор и связать с ним определенную роль. Очевидно, что это лучший выбор, поскольку можно различать пользователей, не усложняя при этом администрирование. Детальнее роли будут описаны позже в этой главе.

Выбор зависит от того, какие допускаются компромиссы в базе данных и прикладных программах.

До того, как была реализована поддержка ролей, в Sybase и SQL Server по отношению к группам пользователей был возможен иной, третий, вариант. В этих СУБД существуют идентификаторы групп, состоящих из связанных каким-либо образом пользователей. Привилегии могут быть предоставлены как пользователям, имеющим персональные идентификаторы, так и группам, также обладающим идентификаторами, и пользователи могут осуществлять действия, разрешенные и теми и другими привилегиями. Таким образом, идентификаторы групп упрощают управление системой безопасности. Однако они не соответствуют стандарту, и базы данных, в которых они применяются, могут не быть переносимы в другую СУБД.

Защищаемые объекты

Средства защиты в БОЬ применяются по отношению к отдельным об&ектол* базы данных. В стандарте БОЫ указаны два типа защищаемых объектов — таблицы и представления. Таким образом, каждая таблица и каждое представление могут быть защищены индивидуально. Доступ к ним может быть разрешен для одних пользователей и запрещен для других. Последующие версии стандарта БОЬ рас-

ширяют круг защищаемых объектов, включая в него домены и пользовательские наборы символов, и добавляют новые типы защиты таблиц и представлений.

В большинстве коммерческих СУБД дополнительно могут быть защищены и другие типы объектов. Например, в SQL Server важным объектом базы данных является *хранимая процедура, С* помощью средств защиты SQL устанавливается, какие пользователи могут создавать и удалять хранимые процедуры и какие пользователи могут их выполнять. В DB2 защищаемыми объектами являются *табличные пространства* — физические области хранения таблиц. Администратор базы данных может для одних пользователей разрешать создание новых таблиц в некоторой физической области, а для других— запрещать. Другие реляционные СУБД могут защищать иные объекты. В целом, однако, базовый механизм обеспечения безопасности в SQL — выдача и отмена привилегий на определенные объекты с помощью определенных инструкций SQL — одинаковым образом реализован практически во всех СУБД

Привилегии

Множество действий, которые пользователь имеет право выполнять над объектом базы данных, называется *привилегиями* пользователя по отношению к данному объекту. В стандарте SQL1 для таблиц и представлений определены четыре привилегии.

• Привилегия SELECT позволяет извлекать данные из таблицы или представления. Имея эту привилегию, можно задавать имя таблицы или представления в предложении FROM инструкции SELECT или подзапроса.

• Привилегия INSERT позволяет вставлять новые записи в таблицу или представление. Имея эту привилегию, можно задавать имя таблицы или представления в предложении INTO инструкции INSERT.

• Привилегия DELETE позволяет удалять записи из таблицы или представления. Имея эту привилегию, можно задавать имя таблицы или представления в предложении FROM инструкции DELETE.

• Привилегия UPDATE позволяет модифицировать записи в таблице или представлении. Имея эту привилегию, можно задавать таблицу или представление в инструкции UPDATE как целевую таблицу. Привилегия UPDATE может быть ограничена отдельными столбцами таблицы или представления, позволяя тем самым обновлять только эти столбцы и запрещая обновлять другие.

Эти четыре привилегии поддерживаются почти всеми коммерческими реализациями SQL.

Расширенные привилегии SQL

В последующих версиях стандарта SQL базовый механизм управления привилегиями стандарта SQL1 значительно расширен. Определенные в SQL1 привилегии SELECT, INSERT И UPDATE дополнены новыми возможностями. Появилась новая привилегия REFERENCES, ограничивающая право пользователя на создание

внешних ключей доступной ему таблицы для ссылок на другие таблицы. Кроме того, добавлена новая привилегия USAGE, управляющая доступом к новым структурам баз данных SQL— доменам, наборам символов, порядкам сортировки и правилам конвертирования текста.

Расширения привилегий SELECT, INSERT и UPDATE достаточно просты. Теперь они могут относиться к конкретному столбцу или столбцам таблицы, а не только ко всей таблице целиком. Давайте рассмотрим их полезность на примере нашей учебной базы данных. Предположим, вы хотите, чтобы за добавление новых служащих в таблицу SALESREPS отвечал начальник отдела кадров. Он должен вводить идентификатор принятого на работу служащего, его имя и прочую учетную информацию. Однако за назначение новому служащему планового объема продаж, т.е. за заполнение столбца QUOTA, отвечает не он, а начальник отдела сбыта. Аналогично начальник отдела кадров не должен иметь доступа к столбцу SALES имеющихся записей таблицы SALESREPS. Используя новые возможности стандарта SQL, можно реализовать эту схему, предоставив начальнику отдела кадров привилегию INSERT для соответствующих столбцов. Остальные столбцы (SALES и QUOTA) будут при создании новой записи принимать значения NULL. Начальник же отдела сбыта получает привилегию UPDATE для столбцов SALES и QUOTA, чтобы он мог назначать служащим плановые объемы продаж. При отсутствии возможности указывать привилегии для конкретных столбцов вам придется либо ослаблять упомянутые ограничения на доступ к отдельным столбцам, либо определять дополнительные представления просто для ограничения доступа.

Введение новой привилегии REFERENCES связано с более тонкой особенностью системы безопасности баз данных, касающейся определения внешних ключей и ограничений на значения столбцов. Проанализируем это на примере нашей учебной базы данных. Предположим, что служащий компании может создавать в базе данных новые таблицы (например, ему может понадобиться таблица с информацией о новом товаре), но у него нет доступа к информации о служащих в таблице SALESREPS. При такой схеме защиты может показаться, что он никак не сможет узнать используемые компанией идентификаторы служащих или выяснить, был ли принят на работу новый сотрудник.

Однако это не совсем верно. Служащий может создать новую таблицу и определить один из ее столбцов как внешний ключ для связи с таблицей SALESREPS. Как вы помните, это означает, что допустимыми значениями этого столбца будут только значения первичного ключа из таблицы SALESREPS, т.е. допустимые идентификаторы служащих. Чтобы выяснить, работает ли в компании служащий с некоторым идентификатором, нашему "хакеру" достаточно будет попытаться добавить в свою таблицу новую строку с этим значением внешнего ключа. Если это получится, значит, такой служащий в компании есть, если нет— то и служащего с таким идентификатором в компании нет.

Еще более серьезную проблему порождает возможность создания новых таблиц с ограничениями на значения столбцов. Предположим, например, что служащий пытается выполнить такую инструкцию.

```
CREATE TABLE XYZ (TRYIT DECIMAL(9,2),
CHECK ((SELECT QUOTA
FROM SALESREPS
WHERE NAME = 'VP Sales')
BETWEEN 400000 AND 500000));
```

Поскольку значение столбца TRYIT созданной служащим таблицы теперь связано со значением столбца QUOTA конкретной строки таблицы SALESREPS, то успешное выполнение приведенной инструкции означает, что квота вице- президента по сбыту находится в указанном диапазоне! В противном случае можно попробовать аналогичную инструкцию, подкорректировав границы диапазона. Заметим, впрочем, что только некоторые реализации SQL поддерживают проверку с обращениями к другим таблицам. На момент написания данного материала такой поддержкой обладала только MySQL, но не DB2, SQL Server или Oracle.

Эту лазейку для доступа к данным и закрывает введенная в стандарт SQL новая привилегия REFERENCES. Подобно привилегиям SELECT, INSERT и UPDATE, она назначается на отдельные столбцы таблицы. Если у пользователя есть привилегия REFERENCES для некоторого столбца, он может создавать новые таблицы, связанные с этим столбцом тем или иным способом (например, через внешний ключ или ограничения на значения столбцов, как в предыдущих примерах). Если же СУБД не поддерживает привилегию REFERENCES, но поддерживает внешние ключи и ограничения на значения столбцов, то иногда эту же задачу может выполнить привилегия SELECT.

Наконец, стандарт SQL определяет новую привилегию USAGE, управляющую доступом к доменам (наборам допустимых значений столбцов), пользовательским наборам символов, порядкам сортировки и правилам конвертирования текста. Привилегия USAGE просто разрешает или запрещает использование этих объектов базы данных по имени для отдельных идентификаторов пользователей. Например, имея привилегию USAGE на некоторый домен, можно определить новую таблицу и указать этот домен в качестве типа д анных какого-нибудь из ее столбцов. Без такой привилегии использовать этот домен для определения столбцов нельзя. Привилегия USAGE служит, главным образом, для упрощения администрирования больших корпоративных баз данных, которые используются и модифицируются несколькими различными командами разработчиков. Ее введение не связано с теми проблемами безопасности, с которыми связаны привилегии для доступа к таблицам и столбцам.

Привилегии владения

Когда вы создаете таблицу посредством инструкции CREATE TABLE, становитесь ее владельцем и получаете все привилегии для этой таблицы (SELECT, INSERT, DELETE, UPDATE и другие привилегии, имеющиеся в СУБД). Все прочие пользователи изначально не имеют никаких привилегий для работы со вновь созданной таблицей. Чтобы они получили доступ к таблице, вы должны явно предоставить им соответствующие привилегии с помощью инструкции GRANT.

Когда с помощью инструкции CREATE VIEW ВЫ создаете представление, то становитесь его владельцем, но не обязательно получаете по отношению к нему все

привилегии. Для успешного создания представления необходимо иметь привилегию SELECT ДЛЯ каждой исходной таблицы представления, поэтому привилегию SELECT для работы с созданным представлением вы получаете автоматически. Что

касается остальных привилегий (INSERT, DELETE и UPDATE), то вы получаете их только в том случае, если имели их для *каждой* исходной таблицы представления.

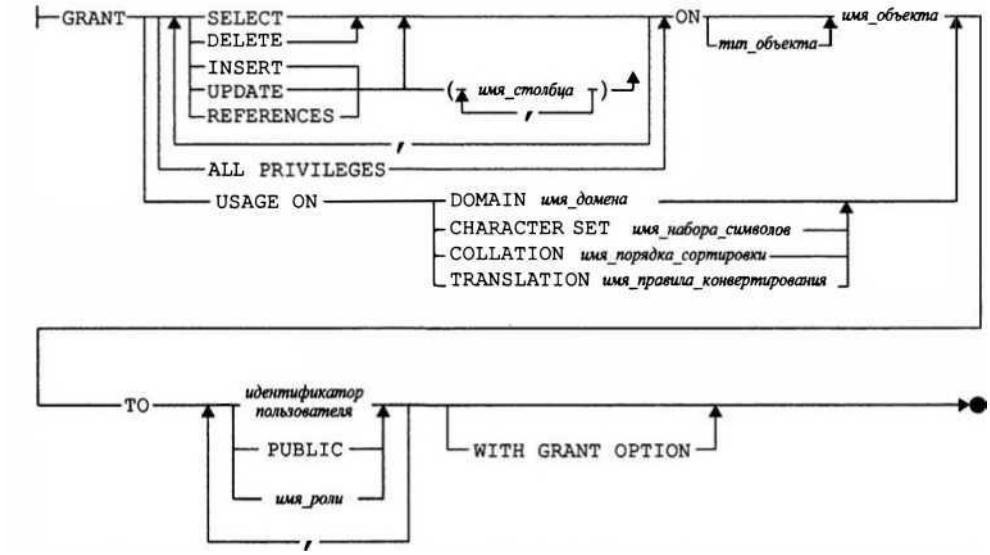
Другие привилегии

Во многих коммерческих СУБД, помимо привилегий SELECT, INSERT, DELETE и UPDATE, по отношению к таблицам и представлениям могут быть предоставлены дополнительные привилегии. Например, в Oracle и базах данных для мэйнфреймов компании IBM предусмотрены привилегии ALTER И INDEX. Имея привилегию ALTER для какой-либо таблицы, пользователь может с помощью инструкции ALTER TABLE модифицировать определение данной таблицы; имея привилегию INDEX, пользователь может посредством инструкции CREATE INDEX создать индекс для таблицы. В тех СУБД, где отсутствуют привилегии ALTER и INDEX, только владелец таблицы может пользоваться инструкциями ALTER TABLE И CREATE INDEX.

Часто в СУБД имеются дополнительные привилегии не для таблиц и представлений, а для других объектов. Например, Oracle, Sybase и SQL Server поддерживают привилегию EXECUTE для хранимых процедур, которая определяет, кто из пользователей имеет право ее выполнять. В DB2 имеется привилегия USE для табличных пространств, определяющая, кто из пользователей может создавать таблицы в определенном табличном пространстве.

Предоставление привилегий (GRANT)

Инструкция GRANT, синтаксическая диаграмма которой изображена на рис. 15.5, используется для предоставления пользователям привилегий для работы с объектами базы данных. Обычно инструкцией GRANT пользуется владелец таблицы или представления, чтобы разрешить другим пользователям доступ к этим данным. Как видно из рисунка, инструкция GRANT содержит список предоставляемых привилегий, имя таблицы или иного объекта, для которого назначаются привилегии (для всех объектов, кроме таблиц и представлений, требуется указание типа объекта), и идентификатор пользователя или роль, получающую эти привилегии. В большинстве реализаций SQL для предоставления привилегий пользователю его учетная запись должна уже иметься в базе данных.



Синтаксическая диаграмма инструкции GRANT

Синтаксическая диаграмма инструкции GRANT, изображенная на рисунке, соответствует стандарту ANSI/ISO. Многие СУБД придерживаются синтаксиса инструкции GRANT, который применяется в DB2. Он более гибок и позволяет задавать список идентификаторов пользователей и список таблиц, упрощая предоставление большого числа привилегий. Вот несколько примеров простых инструкций GRANT для учебной базы данных.

Предоставить пользователям из отдела обработки заказов полный доступ к таблице ORDERS.

```
GRANT SELECT, INSERT, DELETE, UPDATE
ON ORDERS
TO OPUSER;
```

Разрешить пользователям финансового отдела извлекать данные о заказчиках и добавлять новых заказчиков в таблицу CUSTOMERS; пользователям отдела обработки заказов разрешить только выборку из этой таблицы.

GRANT SELECT, INSERT
ON CUSTOMERS
TO ARUSER;
GRANT SELECT
ON CUSTOMERS
TO OPUSER;

Разрешить Сэму Кларку добавлять и удалять информацию об офисах.

GRANT INSERT, DELETE
ON OFFICES
TO SAM;

При предоставлении большого числа привилегий или предоставлении привилегий большому числу пользователей в инструкции GRANT ДЛЯ удобства применяются два сокращения. Вместо того чтобы перечислять все привилегии для некоторого объекта, можно использовать предложение ALL PRIVILEGES. Приведенная ниже инструкция GRANT разрешает вице-президенту по сбыту Сэму Кларку полный доступ к таблице SALESREPS.

Предоставить Сэму Кларку все привилегии по отношению к таблице SALESREPS.

GRANT ALL PRIVILEGES
ON SALESREPS
TO SAM;

Вместо того чтобы давать привилегии по отдельности каждому пользователю базы данных, можно с помощью ключевого слова PUBLIC предоставить их сразу всем пользователям, имеющим право работать с базой данных. Очевидно, что этой возможностью надо пользоваться крайне осторожно. Следующая инструкция GRANT разрешает всем пользователям извлекать данные из таблицы OFFICES.

Дать всем пользователям привилегию SELECT для таблицы OFFICES.

GRANT SELECT
ON OFFICES
TO PUBLIC;

Обратите внимание: такая инструкция GRANT дает привилегии всем нынешним и будущим пользователям базы данных, а не только тем, кто имеет право работать с базой данных в настоящее время. Это устраняет необходимость явно предоставлять привилегии новым пользователям при их появлении.

Привилегии для работы ео столбцами

Стандарт SQL1 позволял предоставлять привилегию UPDATE ДЛЯ отдельных столбцов таблицы или представления, а новейшие версии стандарта дают возможность предоставлять таким же образом привилегии SELECT, INSERT И REFERENCES. Список столбцов располагается после ключевого слова SELECT, UPDATE, INSERT

или REFERENCES и заключается в круглые скобки. Вот как выглядит инструкция GRANT, разрешающая сотрудникам отдела обработки заказов обновлять в таблице CUSTOMERS только столбцы с названием компании (COMPANY) и идентификатором закрепленного за ней служащего (CUST_REP).

Разрешить пользователям отдела обработки заказов изменять названия компаний, а также закрепленных за компаниями служащих.

```
GRANT UPDATE (COMPANY, CUST_REP)
ON CUSTOMERS
TO OPUSER;
```

Если список столбцов опущен, то привилегия действительна для всех столбцов таблицы или представления.

Разрешить пользователям финансового отдела изменять всю информацию о клиентах.

```
GRANT UPDATE
ON CUSTOMERS
TO ARUSER;
```

Стандарты SQL после SQL1 поддерживают предоставление привилегии SELECT для списка столбцов, как в приведенном далее примере.

Разрешить пользователям финансового отдела доступ только для чтения к идентификаторам, именам и офисам служащих из таблицы SALESREPS.

```
GRANT SELECT (EMPL_NUM, NAME, REP_OFFICE)
ON SALESREPS
TO ARUSER;
```

Такая инструкция GRANT устраняет необходимость в представлении REPINFO, показанном на рис. 15.3, и на практике может сделать ненужным большое количество представлений в промышленных базах данных. Однако эта возможность пока что поддерживается далеко не всеми ведущими СУБД.

Передача привилегий (GRANT OPTION)

Если вы создаете в базе данных объект и становитесь его владельцем, только вы можете предоставлять привилегии для работы с этим объектом. Когда вы предоставляете привилегии другим пользователям, они могут пользоваться объектом, но не могут передавать эти привилегии кому-то еще. Таким образом, владелец объекта сохраняет строгий контроль над тем, кому какие формы доступа к объекту разрешены.

Но бывают ситуации, когда владелец может захотеть передать другим пользователям право предоставлять привилегии для работы со своим объектом. Например, обратимся еще раз к представлениям EASTREPS И WESTREPS в учебной базе данных. Их создал и владеет ими вице-президент по сбыту Сэм Кларк. Посредством следующей инструкции GRANT OH может дать менеджеру лос-анжелесского офиса Ларри Фитчу разрешение пользоваться представлением WESTREPS.

```
GRANT SELECT
ON WESTREPS
TO LARRY;
```

Что произойдет, если Ларри захочет дать Сью Смит (идентификатор пользователя SUE) разрешение на доступ к представлению WESTREPS В СВЯЗИ С тем, что она составляет прогноз объема продаж для офиса в Лос-Анджелесе? Он не сможет этого сделать, так как предыдущая инструкция GRANT неявно запрещает подобное действие. Требуемую привилегию может предоставить только Сэм Кларк, так как именно он является владельцем представления.

Если Сэму требуется, чтобы Ларри по своему усмотрению решал, кому давать привилегию для работы с представлением WESTREPS, он может воспользоваться следующей разновидностью инструкции GRANT.

```
GRANT SELECT
ON WESTREPS
TO LARRY
WITH GRANT OPTION
```

Наличие предложения WITH GRANT OPTION означает, что инструкция GRANT наряду с привилегиями дает право на предоставление этих привилегий другим пользователям.

Теперь Ларри может выполнить инструкцию GRANT

```
GRANT SELECT
ON WESTREPS
TO SUE
```

которая позволяет Сью Смит извлекать данные из представления WESTREPS. На рис. 15.6 процедура передачи привилегий изображена графически: сначала от Сэма к Ларри, а затем от Ларри к Сью. Так как Ларри не включил в свою инструкцию GRANT предложение WITH GRANT OPTION, цепочка заканчивается на Сью; она может извлекать данные из представления WESTREPS, но не может давать право на доступ к нему другому пользователю. Однако если бы Ларри включил в свою инструкцию GRANT предложение WITH GRANT OPTION, цепочка могла бы протянуться дальше, поскольку Сью получила бы возможность предоставлять право на доступ другим пользователям.

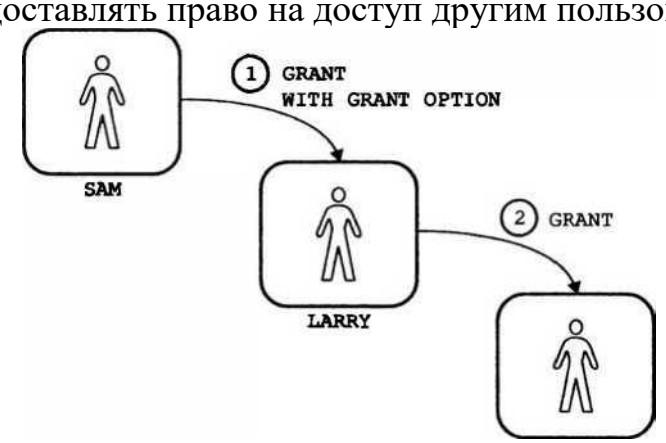
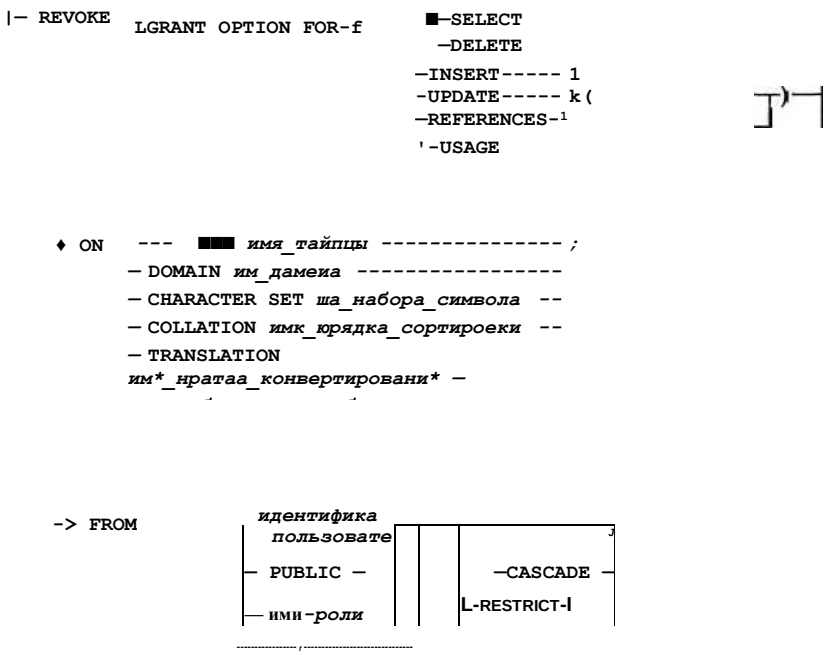


Рис. 15.6. Применение GRANT OPTION



Синтаксическая диаграмма инструкции REVOKE

Посредством инструкции REVOKE можно отобрать все или только некоторые из привилегий, предоставленных ранее пользователю. Например, рассмотрим такую последовательность инструкций.

Предоставить некоторые привилегии для работы с таблицей SALESREPS, а затем отменить часть из них.

```
GRANT SELECT, INSERT, UPDATE
ON SALESREPS
TO ARUSER, OPUSER;
REVOKE INSERT, UPDATE
ON SALESREPS
FROM OPUSER;
```

Сначала привилегии INSERT и UPDATE для работы с таблицей SALESREPS даются двум пользователям, а затем у одного из них отбираются. Однако привилегия SELECT остается у обоих пользователей. Вот еще несколько примеров инструкций REVOKE.

Отобрать все привилегии, предоставленные ранее для работы с таблицей OFFICES.

```
REVOKE ALL PRIVILEGES
ON OFFICES
FROM ARUSER;
```

Отобрать привилегии UPDATE и DELETE у двух пользователей.

```
REVOKE UPDATE, DELETE
ON OFFICES
FROM ARUSER, OPUSER;
```

Отобрать у всех пользователей все предоставленные ранее привилегии для работы с таблигцей OFFICES.

```
REVOKE ALL PRIVILEGES
ON OFFICES
FROM PUBLIC;
```

Посредством инструкции REVOKE ВЫ можете отобрать только те привилегии, которые *вы* предоставили ранее некоторому пользователю. У него могут быть также привилегии, предоставленные другими пользователями; ваша инструкция REVOKE на них не влияет. Обратите особое внимание на то, что если два разных пользователя предоставляют третьему пользователю одну и ту же привилегию на один и тот же объект, а затем один из них отменяет привилегию, то вторая привилегия остается в силе и по-прежнему разрешает пользователю доступ к объекту. Результат такого "перекрытия привилегий" иллюстрируется следующим примером.

Предположим, что вице-президент по сбыту Сэм Кларк предоставляет Ларри Фитчу привилегию SELECT для работы с таблицей SALESREPS И привилегии SELECT и UPDATE для работы с таблицей ORDERS, используя следующие инструкции.

```
GRANT SELECT
ON SALESREPS
TO LARRY;
GRANT SELECT, UPDATE
ON ORDERS
TO LARRY;
```

Через несколько дней вице-президент по маркетингу Джордж Уоткинс предоставляет Ларри привилегии SELECT и DELETE для работы с таблицей ORDERS и привилегию SELECT для работы с таблицей CUSTOMERS, используя такие инструкции.

```
GRANT SELECT, DELETE
ON ORDERS
TO LARRY;
GRANT SELECT
ON CUSTOMERS
TO LARRY;
```

Заметьте, что Ларри получил привилегии для работы с таблицей ORDERS из двух различных источников. Причем привилегию SELECT для работы с таблицей ORDERS он получил из обоих источников. Несколькими днями позже Сэм отменяет привилегии, предоставленные недавно Ларри для работы с таблицей ORDERS.

```
REVOKE SELECT, UPDATE
ON ORDERS
FROM LARRY;
```

После выполнения этой инструкции СУБД у Ларри остается привилегия SELECT для работы с таблицей SALESREPS, привилегии SELECT и DELETE для работы с таблицей ORDERS И привилегия SELECT ДЛЯ работы с таблицей CUSTOMERS; НО он теряет привилегию UPDATE ДЛЯ работы с таблицей ORDERS.

REVOKE и GRANT OPTIONS

Когда вы даете кому-нибудь привилегии с правом последующего предоставления, а затем отменяете эти привилегии, то в большинстве СУБД *автоматически* отменяются все привилегии, которые являются производными от исходных. Рассмотрим еще раз цепочку привилегий, представленных на рис. 15.6, идущую от вице-президента по сбыту Сэма Кларка к менеджеру лос-анжелесского офиса Ларри Фитчу, а затем к Сью Смит. Если теперь Сэм отменит привилегии Ларри для работы с представлением WESTREPS, ТО привилегии Сью также будут автоматически отменены.

Ситуация становится более сложной, если привилегии предоставляются двумя или более пользователями, один из которых впоследствии отменяет привилегии. Рассмотрим рис. 15.8. Он представляет собой предыдущий пример в слегка измененном виде. Здесь Ларри получает привилегию SELECT С правом передачи *как* от Сэма (вице-президента по сбыту), *так* и от Джорджа (вице-президента по маркетингу), а затем дает привилегии Сью. На этот раз, когда Сэм отменяет привилегии Ларри, остаются привилегии, предоставленные Джорджем. Привилегии Сью также остаются, поскольку они могут происходить от привилегий Джорджа.

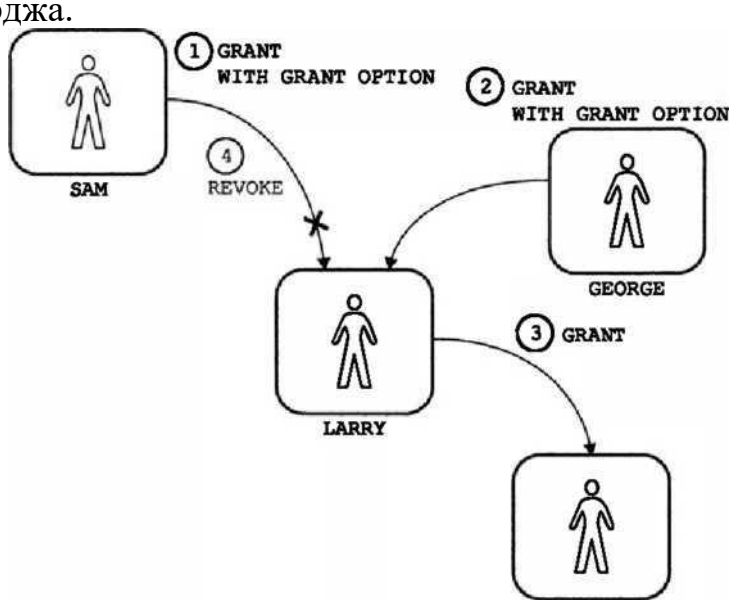


Рис. 15.8. Отмена привилегий, предоставленных двумя пользователями

Рассмотрим другой вариант этой цепочки привилегий (рис. 15.9), в котором изменен порядок событий. Здесь Ларри получает привилегию с правом передачи от Сэма, дает эту привилегию Сью и *после этого* получает привилегию с правом передачи от Джорджа. Когда на этот раз Сэм отменяет привилегии Ларри, результат будет другим и может отличаться у разных СУБД. Как и в случае, изображенном на рис. 15.8, у Ларри останется привилегия SELECT для работы с представле-

нием WESTREPS, предоставленная Джорджем. Но в СУБД DB2 и SQL/DS Сью автоматически потеряет привилегию SELECT. Почему? Потому что привилегия Сью явно происходит от привилегии, которая предоставлена Ларри Сэмом и только что отменена. Она не может быть производной от привилегии, предоставленной Ларри Джорджем, так как эта привилегия отсутствовала в тот момент, когда Ларри давал привилегию Сью.

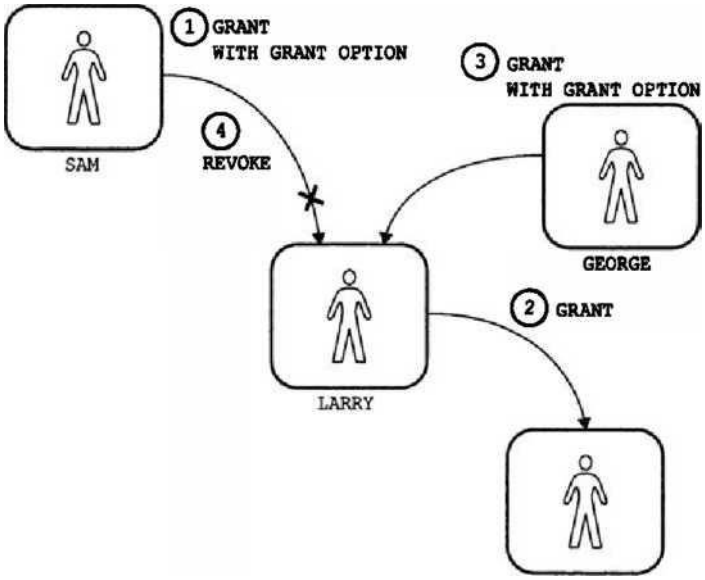


Рис. 15.9. Отмена привилегий, предоставленных в другой последовательности

В других СУБД привилегия Сью могла бы сохраниться, так как у Ларри сохранилась привилегия, предоставленная Джорджем. Таким образом, то, как далеко распространятся последствия выполнения инструкции REVOKE, зависит не только от самих привилегий, но и от хронологической последовательности инструкций GRANT и REVOKE. Для получения желаемых результатов выдача привилегий с правом передачи и аннулирование таких привилегий должны выполняться очень осторожно.

Следует не упускать из виду еще одно — дополнительные расходы на обработку каскадных отмен привилегий. При излишнем использовании GRANT OPTION отмены привилегий могут привести к серьезным проблемам с производительностью.

Безопасность на основе ролей

Управление привилегиями отдельных пользователей может оказаться весьма монотонной и утомительной работой. В связи с этим в стандарт SQL была добавлена концепция ролей. Вспомним, что *роль* — это просто именованный набор привилегий. В большинстве современных реализаций SQL роли могут предоставляться отдельным идентификаторам пользователей точно так же, как и отдельные привилегии. Более того, большинство реализаций SQL поставляется с набором предопределенных ролей. Например, привилегии, которые обычно необходимы для работы администратору базы данных, зачастую предоставляются поставщиком СУБД в виде роли.

Преимущества использования ролей следующие.

• Роли могут существовать до идентификаторов пользователей. Например, мы можем создать роль для отдела обработки заказов, вместо того чтобы предоставлять всем его сотрудникам единый идентификатор пользователя OPUSER, как показано на рис. 15.2. Когда в отдел приходит новый сотрудник, одна инструкция GRANT с указанием роли предоставляет ему все необходимые для работы в отделе привилегии.

• Роли в состоянии "пережить" удаление пользователей. Администратору не надо беспокоиться об отслеживании потерь привилегий при удалении некоторого пользователя. Например, если привилегии вице-президента по сбыту Сэма Кларка получены им при помощи роли, то список приви-