Лекция 16. Защита данных в реляционных СУБД. Секретность.

16.1 Подходы к реализации системы секретности. Стандарты.

При реализации системы секретности в современных СУБД используется один из двух подходов к обеспечению секретности данных:

1. **Избирательный подход**, при котором пользователь обладает различными полномочиями при работе с разными объектами.
2. **Обязательный подход**, при котором каждому объекту присваивается некоторый классификационный уровень, а каждый пользователь обладает некоторым уровнем допуска.

Совершенно очевидно, что СУБД лишь реализует принятые на этапе проектирования стратегические решения по обеспечению секретности, разграничению прав доступа. Для этого в СУБД должны присутствовать такие компоненты:

1. Компонент **определения** полномочий на некотором языке (возможно, на SQL);
2. Компонент **регулирования доступа** на основании определенных полномочий – подсистема полномочий. В самом простом случае, если пользователь обращается с запросом на использование объекта, не имея на то соответствующих полномочий, запрос отклоняется. В некоторых системах запрос может быть сужен до такого запроса, который будет находиться в рамках действующих полномочий пользователя (например, как в СУБД INGRES).
3. Компонент **идентификации** пользователя/группы. Современные СУБД позволяют задавать полномочия сразу **группе** пользователей. Набор полномочий, приписанный группе пользователей, часто называют **ролью**.

При отслеживании попыток несанкционированного доступа используется журнал транзакций.

16.1.1 Избирательное управление доступом

При избирательном подходе пользователю назначается набор полномочий в том случае, если эти полномочия отличаются от принятых в конкретной системе баз данных по умолчанию. Т.е. все пользователи обладают некоторым минимальным набором полномочий (возможно, пустым), и только некоторым назначаются дополнительные полномочия. Причем, одному пользователю могут быть назначены разные полномочия при работе с разными объектами.
Избирательные схемы весьма гибки, и большинство СУБД придерживаются именно избирательного подхода.

16.1.2 Обязательное управление доступом

Обязательная схема жестче, по сравнению с избирательными схемами секретности, и применяется к базам данных, структура которых редко меняется. Классическим примером могут служить базы данных военных или правительственных организаций. Так, согласно требованиям Министерства обороны США, все используемые в этом ведомстве системы (в том числе базы данных, и программные системы) должны поддерживать обязательное управление доступом.
Основная идея обязательной схемы состоит в том, что каждому объекту данных приписывается некоторый уровень допуска (например, «секретно», «совершенно секретно», и т.д.), а каждому

пользователю назначается один уровень допуска, с теми же значениями, что и для объектов данных.
Тогда правила безопасности формулируются так:

1. Пользователь имеет *доступ* к объекту (просмотр), если его уровень допуска больше или равен уровню допуска объекта.
2. Пользователь может *модифицировать* объект, только если его уровень допуска равен уровню допуска объекта. Это правило предотвращает ситуации, в которых пользователь с высоким уровнем допуска не может записать данные в файлы/таблицы, имеющие более низкий уровень допуска.

Требования Минобороны США к обязательному управлению доступом изложены в двух документах, называемых "**оранжевой книгой**" и "**розовой книгой**". В "оранжевой" книге перечислен набор требований к безопасности для "идеальной" вычислительной базы (Trusted Computing Base –TCB).В "розовой" книге эти требования уточняются для баз данных.

СУБД, в которых поддерживаются методы обязательной защиты, называют **системами с многоуровневой защитой** или **надежными системами**.

16.1.3 Стандарты. Классы безопасности.

В "оранжевой" книге определяется четыре класса безопасности (security classes) – D, C, B, A.
Класс D обеспечивает *минимальную* защиту, класс C – *избирательную*, B – *обязательную*, а класс A – *проверенную* защиту.

- **Избирательная защита**: класс C делится на два подкласса – C1 и C2 (более безопасный, чем C1).
  Согласно требованиям класса **C1** необходимо разделение данных и пользователя, т.е. наряду с поддержкой концепции общего доступа к данным здесь возможна организация раздельного использования данных пользователями.
  Для систем класса **C2** необходимо дополнительно организовать учет на основе процедур входа в систему, аудита (отслеживания обращений к ресурсам) и изоляции ресурсов.
- **Обязательная защита**: класс B делится на три подкласса B1, B2 и B3 (B3 наиболее безопасный).
  Для класса **B1** необходимо обеспечить «отмеченную защиту» (т.е. каждый объект должен содержать отметку о его уровне безопасности), а также неформальное сообщение о действующей системе безопасности.
  Для класса **B2** необходимо дополнительно обеспечить *формальное* утверждение о действующей стратегии безопасности, а также обнаружить и исключить *плохо защищенные каналы передачи информации*.
  Для класса **B3** необходимо дополнительно обеспечить поддержку аудита и восстановления данных, а также назначение *администратора режима безопасности*.
- **Проверенная защита**: класс **A** является наиболее безопасным. Согласно его требованиям, необходимо **математическое доказательство того, что данный метод безопасности совместим и адекватен заданной стратегии безопасности**.

16.1.4 Шифрование данных

До сих пор предполагалось, что «нелегальный» пользователь будет пытаться проникнуть в систему баз данных с помощью средств доступа, имеющихся в системе, т.е. подбирая пароли и заменяя учетные записи пользователей.
Возможны ситуации, когда пользователь пытается проникнуть в систему, минуя стандартные средства доступа, например, физически перемещая файлы данных или подключаясь к коммуникационным каналам, по которым идет передача данных между узлами распределенной СБД.
В последнем случае в качестве метода обеспечения секретности выбирают **шифрование данных**.

Первый стандарт шифрования был введен в 1977 году в США. Он получил название DES – Data Encryption Standard. Этим стандартом предусматривается, что алгоритм расшифровки идентичен алгоритму шифрования за исключением того, что ключи шифрования применяются в обратном порядке.
Однако DES может быть взломан методом «грубой силы» перебором значений. Поэтому в последнее время используется шифрование на основе открытого ключа – RSA.

16.2 Директивы SQL для обеспечения секретности

В стандарте SQL предусмотрена поддержка только избирательного управления доступом. Она включает в себя механизм представлений (views) и подсистему привилегий.
В SQL2 существуют директивы для раздачи и отмены привилегий пользователям.

Директива `GRANT` устанавливает привилегии пользователю.

```
GRANT список_привилегий, разделенный запятыми ON объект
TO список пользователей, разделенный запятыми
[WITH GRANT OPTION];
```

Различают такие типы привилегий:

`ALL PRIVILEGES` – все привилегии
`USAGE` – для использования некоторого домена
`SELECT` – разрешена выборка
`INSERT` – разрешено добавление записей
`UPDATE` – разрешено обновление записей
`DELETE` – разрешено удаления записей
`REFERENCES` – разрешено обращение в ограничениях целостности к специальным объектам (если обращение происходит к таблице, имеющей ссылки на другие, подчиненные, таблицы, то, имея привилегию `REFERENCES`, можно обращаться и к подчиненным таблицам).

`Объект` – домен или отношение.
Опция `WITH GRANT OPTION` - присваивает пользователю полномочия предоставления полномочий другим пользователям.

Директива REVOKE отменяет полномочия, выданные пользователем А пользователю В.

```
REVOKE [GRANT OPTION FOR] список_привилегий, разделенный
запятыми
ON объект
FROM список пользователей, разделенный запятыми option;
```

Где `option` – одно из значений {`CASCADE, RESTRICT`},

`CASCADE` – отмена всех привилегий, производных от данной привилегии
`RESTRICT` – запрет на отмену привилегии, если от нее есть производные привилегии.

16.3 Диаграммы распределения привилегий

Зачастую сеть распределения привилегий очень сложна, поскольку привилегии, которые были определены от имени одного пользователя, могут пересекаться и даже вступать в противоречия с привилегиями, определенными для того же объекта от имени другого пользователя.
Для визуализации сети привилегий используют *диаграммы привилегий*.

Диаграмма привилегий – это граф, вершины которого соответствуют пользователям и их привилегиям, а ребра указывают на факт выдачи некоторой привилегии одним пользователем другому.

Ребра графа определяются таким образом: если пользователь *A* обладает привилегией *P* на объект *Z*, и выдает привилегию *P* или ее часть *Q* (*Q*<*P*) пользователю *C* на этот объект *Z*, то на диаграмме привилегий это соответствует ребру между вершинами *A/P/Z* и C/*Q/Z*.
Запись *Q*<*P* мы используем для обозначения того, что *Q* слабее P. Например привилегия *P* есть привилегия *SELECT* с возможностью передачи привилегии *WITH GRANT OPTION*, а привилегия *Q* – только на *SELECT*
На рис. 16.1 показана ситуация, когда пользователю *C* выдаются привилегии на использование объектов (например, таблиц) Products и Customers.
Как видно, на диаграмме пользователю C соответствуют две вершины.

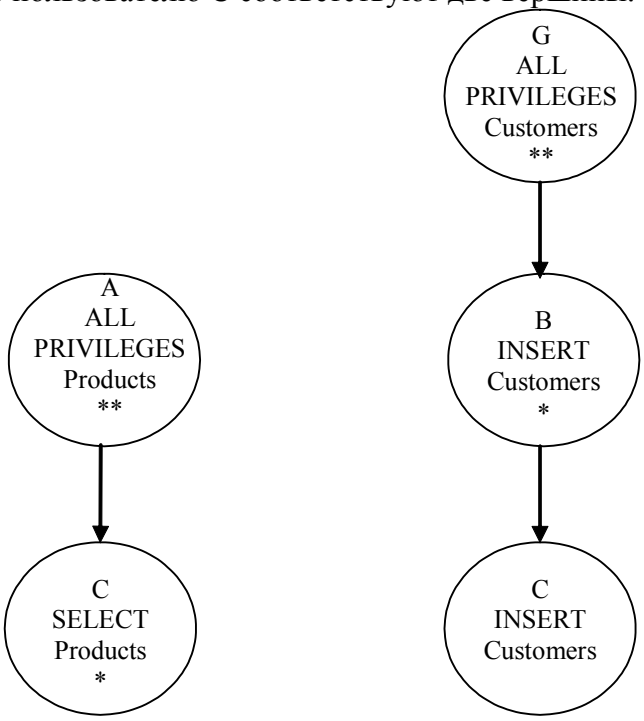


Рис. 16.1 – Пример диаграммы привилегий.

Поскольку в стандарте SQL предусмотрено, что у каждого объекта базы данных есть владелец (пользователь, создавший этот объект), то на диаграмме вершины, соответствующие привилегиям владельца объекта, помечаются двумя символами “*”.

Пример 1.
Пользователь A на рис.16.1 – владелец объекта Products, и обладает всеми привилегиями на этот объект, в том числе и привилегией SELECT. Чтобы пользователь D – не владелец объекта Products, получил равные привилегии с владельцем объекта Products, необходимо выполнить такую инструкцию SQL.

Шаг	Кем выполняется	Инструкция SQL
1	A	`GRANT ALL PRIVILEGES ON Products TO D`

В том случае, если пользователь обладает правом передачи привилегии (GRANT OPTION) на некоторый объект базы данных, на диаграмме эта вершина помечается одним символом “*”.

Пример 2.

На рис. 16.1 пользователь C обладает привилегией SELECT на объект Products, с правом передачи этой привилегии другим пользователям. Также, пользователь B обладает привилегией INSERT на объект Customers с правом передачи этой привилегии.
Для того, чтобы объявить такие привилегии, необходимо выполнить такую последовательность инструкций SQL.

Шаг	Кем выполняется	Инструкция SQL
1	A	`GRANT SELECT ON Products TO C WITH GRANT OPTION;`
2	G	`GRANT INSERT ON Customers TO B WITH GRANT OPTION;`
3	B	`GRANT INSERT ON Customers TO C`

Задавать привилегии можно не только на весь объект целиком (например, привилегия SELECT разрешает выборку всех полей из таблицы), но и на отдельные элементы этого объекта. Тогда привилегии, выданные на весь объект, и привилегии, выданные на отдельный элемент этого объекта, считаются разными.

Пример 3.
Следующая последовательность инструкций SQL соответствует диаграмме привилегий, показанной на рис. 16.2

Шаг	Кем выполняется	Инструкция SQL
1	A	`GRANT SELECT ON Products TO C;`
2	A	`GRANT SELECT(product_name) ON Products TO C;`

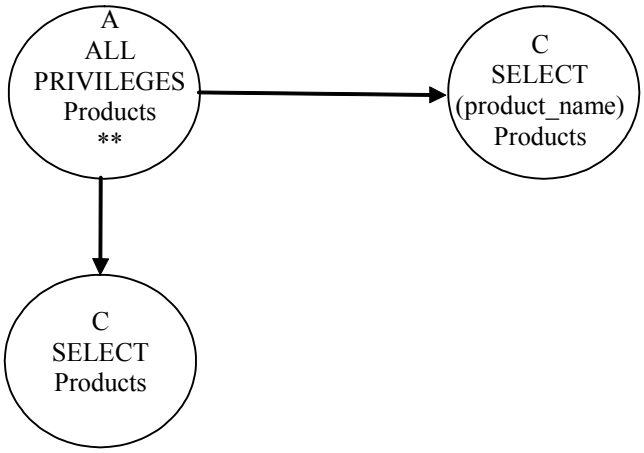


Рис. 16.2 – Диаграмма привилегий, на которой один и тот же пользователь C получает привилегию на выборку для всей таблицы и на отдельный столбец этой таблицы.

Пример 4.
Дана диаграмма привилегий как на рис. 16.2. Выполняется такая инструкция:

Шаг	Кем выполняется	Инструкция SQL
3	A	`REVOKE SELECT ON Products FROM C;`

После ее выполнения, у пользователя C все равно остается привилегия SELECT(product_name) для таблицы Products. См. рис. 16.3.

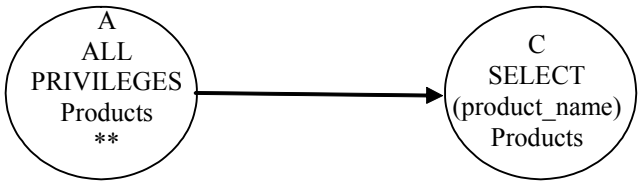


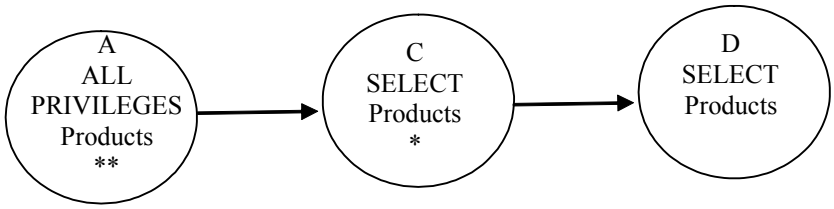
Рис. 16.3 – Диаграмма привилегий после шага 3 из примера 4.

Проиллюстрируем ситуацию выдачи и отзыва GRANT OPTION.

Пример 5.
Дана последовательность инструкций SQL:

Шаг	Кем выполняется	Инструкция SQL
1	A	`GRANT SELECT ON Products TO C WITH GRANT OPTION;`
2	C	`GRANT SELECT ON Products TO D`
3	A	`REVOKE GRANT OPTION FOR SELECT ON PRODUCTS FROM C CASCADE`

После шага 2 диаграмма привилегий будет такой:



После шага 3 диаграмма станет такой: