Лекция 1: Введение. Основные характеристики средств, методов и механизмов обеспечения безопасности баз данных

План:

1. Основные характеристики методов безопасности баз данных
2. Организация и поддержание логической структуры данных
3. Интерфейс ввода данных СУБД

Исследования по проблемам защиты компьютерной информации, проведенные в конце 70-хначале 80-х годов, развитые впоследствии в различных приложениях и закрепленные в соответствующих стандартах, определяют в качестве составных элементов понятия *безопасности информации* - три компонента:

- *конфиденциальность* (защита от несанкционированного доступа);
- *целостность* (защита от несанкционированного изменения информации);
- *доступность* (защита от несанкционированного удержания информации и ресурсов, от разрушения, работоспособности).

Составляющим безопасности информации противостоят соответствующие *угрозы*. Под *угрозой безопасности информации* понимается *осуществляемое или потенциально осуществимое воздействие на компьютерную систему, которое прямо или косвенно может нанести ущерб безопасности информации*. Угрозы реализуют или пытаются реализовать *нарушители* информационной безопасности.

Формализованное описание или представление *комплекса возможностей нарушителя по реализации тех или иных угроз безопасности информации* называют *моделью нарушителя (злоумышленника).*

Качественное описание комплекса организационно-техноло- гических и программнотехнических мер по обеспечению защищенности информации в компьютерной системе (КС) называют *политикой безопасности.* Формальное (математическое, алгоритмическое, схемотехническое) выражение и формулирование политики безопасности называют *моделью безопасности.*

Некоторые термины, относящиеся к обеспечению безопасности баз данных (БД) приведены ниже:

- *доступ к информации (access to information)* - ознакомление с информацией, ее обработка (в частности, копирование), модификация, уничтожение;
- *субъект доступа (access subject)* - лицо или процесс, действия которого регламентируются правилами разграничения доступа;
- *объект доступа (access object)* - единица информации автоматизированной системы, доступ к которой регламентируется правилами разграничения доступа;
- *правила разграничения доступа (security policy)* - совокупность правил, регламентирующих права субъектов доступа к объектам доступа;
- *санкционированный доступ (authorized access to information)* - доступ к информации, который не нарушает правил разграничения доступа;
- *несанкционированный доступ (unauthorized access to information)* - доступ к информации, который нарушает правила разграничения доступа с использованием штатных средств, предоставляемых средствами вычислительной техники или автоматизированными системами;
- *уровень полномочий субъекта доступа (subject privilege)* - совокупность прав доступа субъекта доступа (для краткости в дальнейшем мы будем использовать термин «привилегия»);
- *нарушитель правил разграничения доступа (security policy violator)* - субъект доступа, который осуществляет несанкционированный доступ к информации;

– *модель нарушителя правил разграничения доступа (security policy violator model)* - абстрактное (формализованное или неформализованное) описание нарушителя правил разграничения доступа;

– *целостность информации (information integrity)* — способность средства вычислительной техники (в рассматриваемом случае - информационной системы в целом) обеспечить неизменность информации в условиях случайного и (или) преднамеренного искажения (разрушения);

– *метка конфиденциальности (sensitivity label)* - элемент информации, характеризующий конфиденциальность объекта;

– *многоуровневая защита (multilevel secure)* - защита, обеспечивающая разграничение доступа субъектов с различными правами доступа к объектам различных уровней конфиденциальности.

В структуре программного обеспечения компьютера за организацию, размещение и оперирование данными во внешней (долговременной) памяти отвечает операционная система компьютера, соответствующий компонент который чаще всего называется «файловой системой». Данные во внешней памяти компьютера представлены именованными совокупностями, называемыми файлами. В большинстве случаев, операционная (файловая) система не «знает» внутренней смысловой логики организации данных в файлах и оперирует с ними как с однородной совокупностью байтов или строк символов.

С точки зрения смысла и назначения КС файлы данных имеют структуру, отражающую информационнологическую схему предметной области КС. Эта структура данных в файлах должна обязательно учитываться в операциях обработки. Вместе с тем, в силу невозможности, в большинстве случаев размещения файлов баз данных сразу целиком в оперативной памяти компьютера, структуру данных в файлах баз данных приходится учитывать при организации операций обращения к файлам во внешней памяти.

Отсюда вытекает основная особенность СУБД как вида программного обеспечения. Будучи по природе *прикладным программным обеспечением,* т.е. предназначенным для решения конкретных прикладных задач, СУБД изначально выполняли и *системные функции* — расширяли возможности файловых систем *системного программного обеспечения.* В общем плане можно выделить следующие *функции,* реализуемые *СУБД:*

– *организация и поддержание логической структуры данных (схемы базы данных);*
– *организация и поддержание физической структуры данных во внешней памяти;*
– *организация доступа к данным и их обработка в оперативной и внешней памяти.*

Организация и поддержание логической структуры данных (схемы базы данных) обеспечивается средствами *модели организации данных.* В обиходе просто «модель данных».

Модель данных определяется способом организации данных, ограничениями целостности и множеством операций, допустимых над объектами организации данных. Соответственно, модель данных разделяют на три составляющие — *структурную, целостную* и *манипуляционную.*

Известны три основные модели организации данных:

– *иерархическая;*
– *сетевая;*
– *реляционная.*

Модель организации данных, по сути, определяет *внутренний информационный язык* автоматизированного банка данных, реализующего автоматизированную информационную систему.

Модели данных, поддерживаемые СУБД, довольно часто используются в качестве критерия для классификации СУБД. Исходя из этого, различают *иерархические СУБД\ сетевые СУБД* и *реляционные СУБД.*

Другой важной функцией СУБД является *организация и поддержание физической структуры данных во внешней памяти.* Эта функция включает организацию и поддержание внутренней структуры файлов базы данных, иногда называемой *форматом файлов базы данных,* а также создание и поддержание специальных структур (индексы, страницы) для эффективного и упорядоченного доступа к данным. В этом плане эта функция тесно связана с третьей функцией СУБД - организацией доступа к данным.

Организация и поддержание физической структуры данных во внешней памяти может производиться как на основе штатных средств файловых систем, так и на уровне непосредственного управления СУБД устройствами внешней памяти.

Организация доступа к данным и их обработка в оперативной и внешней памяти осуществляется через реализацию процессов, получивших название - транзакций. *Транзакцией называют последовательную совокупность операций, имеющую отдельное смысловое значение по отношению к текущему состоянию базы данных*. Так, например, транзакция по удалению отдельной записи в базе данных последовательно включает определение страницы файла данных, содержащей указанную запись; считывание и пересылку, соответствующей страницы, в буфер оперативной памяти; собственно, удаление записи в буфере ОЗУ; проверку ограничений целостности по связям и другим параметрам после удаления и, наконец, «выталкивание» и фиксацию в файле базы данных нового состояния, соответствующей страницы данных.

Транзакции принято разделять на две разновидности — изменяющие состояние базы данных после завершения транзакции и изменяющие состояние БД лишь временно, с восстановлением исходного состояния данных после завершения транзакции. Совокупность функций СУБД по организации и управлению транзакциями называют *монитором транзакций.*

Транзакции в теории и практике СУБД по отношению к базе данных выступают внешними процессами, отождествляемыми с действиями пользователей банка данных. При этом источником, инициатором транзакций может быть, как один пользователь, так и несколько пользователей сразу. По этому критерию СУБД классифицируются на *однопользовательские* и *многопользовательские* СУБД. Как правило, в однопользовательских СУБД монитор транзакций в виде отдельного функционального элемента не реализуется. Соответственно, в многопользовательских СУБД главной функцией монитора транзакций является обеспечение эффективного совместного выполнения транзакций над общими данными сразу от нескольких пользователей.

Непосредственная обработка и доступ к данным в большинстве СУБД осуществляется через организацию в оперативной памяти штатными средствами операционной системы или собственными средствами системы *буферов оперативной памяти,* куда на время обработки и доступа помещаются отдельные компоненты файла базы данных (страницы). Поэтому другой составной частью функций СУБД по организации доступа и обработки данных является *управление буферами оперативной памяти.*

Еще одной важной функцией СУБД с точки зрения организации доступа и обработки данных является так называемая журнализация всех текущих изменений базы данных. *Журнализация* представляет собой основное средство обеспечения сохранности данных при всевозможных сбоях и разрушениях данных. Во многих СУБД для нейтрализации подобных угроз создается журнал изменений базы данных с особым режимом хранения и размещения.

Резервная копия БД и журнал изменений, как правило, размещаются на отдельных от основного файла БД носителях.

Схематично взаимодействие компонента СУБД представлено на **рис. 1.1.**

Ядром СУБД является *процессор описания и поддержания структуры базы данных.* Он реализует модель организации данных, средствами которой проектировщик строит *логическую структуру (схему) базы данных,* соответствующую инфологической схеме

предметной области КС, и обеспечивает построение и поддержание *внутренней схемы базы данных.*

Процессором описания и поддержания структуры данных в терминах используемой модели данных (иерархическая, сетевая, реляционная) обеспечиваются установки заданной логической структуры базы данных, а также трансляция (перевод) структуры базы данных во внутреннюю схему базы данных (в физические структуры данных). В КС на базе реляционных СУБД процессор описания и поддержания структуры базы данных реализуется на основе *языка базы данных,* являющегося составной частью *языка структурированных запросов SQL*

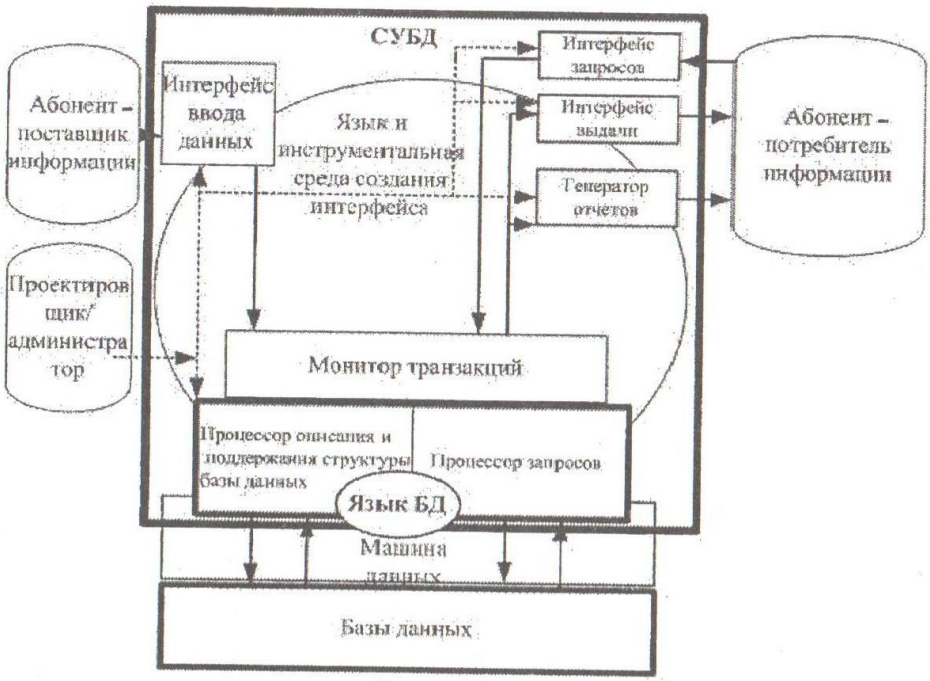


Рис.1.1. Структура и взаимодействие компонента СУБД

Интерфейс ввода данных СУБД реализует *входной информационный язык банка данных,* обеспечивая абонентам- поставщикам информации средства описания и ввода данных в информационную систему. Одной из современных тенденций развития СУБД является стремление приблизить входные информационные языки и интерфейс ввода к естественному языку общения с пользователем в целях упрощения эксплуатации информационных систем так называемых «неподготовленными» пользователями. Данная проблема решается через применение диалоговых методов организации интерфейса и использование *входных форм.* Входные формы, по сути, представляют собой электронные аналоги различного рода анкет, стандартизованных бланков и таблиц, широко используемых в делопроизводстве и интуитивно понятных большинству людей (неподготовленных пользователей). Интерфейс ввода при этом обеспечивает средства создания, хранения входных форм и их интерпретацию в терминах описания логической структуры базы данных для передачи вводимых через формы сведений процессору описания и поддержания структуры базы данных.

Интерфейс запросов совместно с процессором запросов обеспечивает концептуальную модель использования информационной системы в части стандартных типовых запросов, отражающих информационные потребности пользователей-абонентов системы. Интерфейс запросов предоставляет пользователю средства выражения своих информационных потребностей. Современной тенденцией развития СУБД является

использование диалоговонаглядных средств в виде специальных «конструкторов» или пошаговых «мастеров» формирования запросов.

Процессор запросов интерпретирует сформированные запросы в терминах *языка манипулирования данными* и совместно с процессором описания и поддержания структуры базы данных собственно и исполняет запросы. В реляционных СУБД основу процессора запросов составляет язык манипулирования данными, являющийся основной частью языка *SQL*. Тем самым на базе процессора запросов и процессора описания и поддержания структуры базы данных образуется низший уровень оперирования данными в СУБД, который иногда называют *машиной данных.* Стандартные функции и возможности машины данных используют компоненты СУБД более высокого порядка, что позволяет разделить и стандартизировать компоненты СУБД и банка данных на три уровня: логический уровень, машина данных и собственно сами данные.

Функции *монитора транзакции*, как уже отмечалось, заключаются в организации совместного выполнения транзакций от нескольких пользователей над общими данными. При этом дополнительной функцией, неразрывно связанной, в том числе, и с основной функцией, является обеспечение целостности данных и ограничений над данными, определяемыми правилами предметной области КС.

Интерфейс выдачи СУБД получает от процессора запросов результаты исполнения запросов (обращений к базе данных) и переводит эти результаты в форму, удобную для восприятия и выдачи пользователю-абоненту информационной системы. Для отображения результатов исполнения запросов в современных СУБД используются различные приемы, позволяющие «визуализировать» данные в привычной и интуитивно понятной неподготовленному пользователю форме. Обычно для этого применяются табличные способы представления структурированных данных, а также специальные ***формы выдачи*** данных.

Формы выдачи лежат также и в основе формирования так называемых ***«отчетов»,*** выдающих результаты поиска и отбора информации из БД в письменной форме для формализованного создания соответствующих текстовых документов, т.е. для документирования выводимых данных. Для подобных целей в состав современных СУБД включаются ***генераторы отчетов.***

Современные программные средства, реализующие те или иные СУБД, представляют собой совокупность ***инструментальной среды создания и использования базы данных*** в рамках определенной модели данных (реляционной, сетевой, иерархической или смешанной) и ***языка СУБД*** (язык описания данных, язык манипулирования данными, язык и средства создания интерфейса).

Пользователей СУБД можно разделить на три группы:

1. Прикладные программисты отвечают за создание программ, использующих базу данных. В смысле защиты данных программист может быть, как пользователем, имеющим привилегии создания объектов данных и манипулирования ими, так и пользователем, имеющим привилегии только манипулирования данными.

2. Конечные пользователи базы данных работают с БД непосредственно через терминал или рабочую станцию. Как правило, конечные пользователи имеют строго ограниченный набор привилегий манипулирования данными. Этот набор может определяться при конфигурировании интерфейса конечного пользователя и не изменяться. Политику безопасности в данном случае определяет администратор безопасности или администратор базы данных (если это одно и тоже должностное лицо).

3. Администраторы базы данных образуют особую категорию пользователей СУБД. Они создают сами базы данных, осуществляют технический контроль функционирования СУБД, обеспечивают необходимое быстродействие системы. В обязанности администратора, кроме того, входит обеспечение пользователям доступа к необходимым им данным, а также написание необходимых пользователю внешних представлений данных. Администратор определяет правила безопасности и целостности данных.

Модели безопасности данных. Модель безопасности включает:

– модель компьютерной (информационной) системы;

– критерии, принципы, ограничения и целевые функции защищенности информации от угроз;

– формализованные правила, ограничения, алгоритмы, схемы и механизмы безопасного функционирования системы.

В основе большинства моделей безопасности лежит субъектнообъектная модель компьютерных систем, в том числе, и баз данных как ядра автоматизированных информационных систем. База данных КС разделяется на субъекты базы данных (активные сущности), объекты базы данных (пассивные сущности) и порождаемые действиями субъектов процессы над объектами **(рис. 1.2)**.

Определяются два основополагающих принципа безопасности функционирования информационных систем:

– персонализация (идентификация) и аутентификация (подтвержение подлинности) всех субъектов и их процессов по отношению к объектам;

– разграничение полномочий субъектов по отношению к объектам и обязательная проверка полномочий любых процессов над данными.

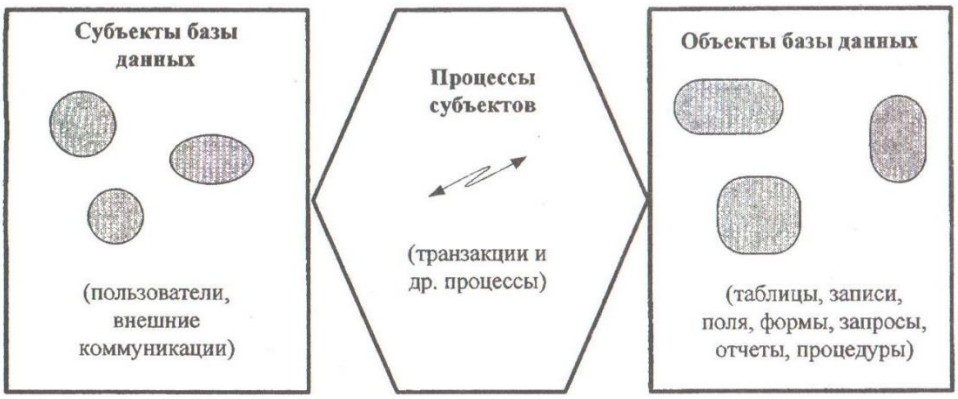


Рис. 1.2. База данных КС в моделях безопасности данных.

Соответственно, в структуре ядра СУБД выделяется дополнительный компонент, называемый монитором (сервером, менеджером, ядром) безопасности (Trusted Computing Base - TCB), который реализует определенную политику безопасности во всех процессах обработки данных. Если в схемотехническом аспекте компьютерную систему представить, как совокупность ядра, включающего компоненты представления данных и доступа (манипулирования) к данным, а также надстройки, которая реализует интерфейсные и прикладные функции, то роль и место монитора безопасности можно проиллюстрировать схемой, приведенной на **рис. 1.3**.

В узком смысле политика безопасности, реализуемая монитором безопасности компьютерной системы, собственно и определяет модель безопасности (вторая и третья компоненты).

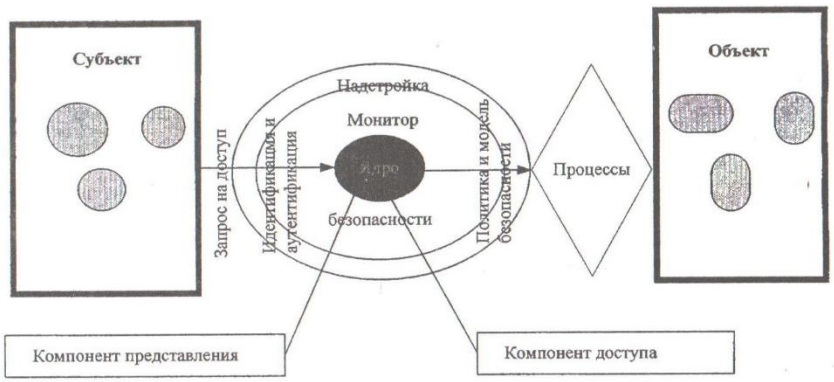


Рис. 1.3. Схематический аспект защиты информации в компьютерных системах.

Простейшая (одноуровневая) модель безопасности данных строится на основе дискреционного (избирательного) принципа разграничения доступа, при котором доступ к объектам осуществляется на основе множества разрешенных отношений доступа в виде троек: - «субъект доступа; - тип доступа; - объект доступа». Наглядным и распространенным способом формализованного представления дискреционного доступа является матрица доступа, устанавливающая перечень пользователей (субъектов) и перечень разрешенных операций (процессов) по отношению к каждому объекту базы данных (таблицы, запросы, формы, отчеты). На **рис. 1.4** приведен пример, иллюстрирующий матрицу доступа.

Важным аспектом моделей безопасности является управление доступом. Существует два подхода:

- добровольное управление доступом;
- принудительное управление доступом.

При добровольном управлении доступом вводится так называемое владение объектами. Добровольное управление доступом заключается в том, что права на доступ к объектам определяют их владельцы. Иначе говоря, соответствующие ячейки матрицы доступа заполняются теми субъектами (пользователями), которым принадлежат права владения над соответствующими объектами базы данных. В большинстве систем права владения объектами могут передаваться. В результате при добровольном управлении доступом реализуется полностью децентрализованный при нцип организации и управления процессом разграничения доступа.

		ТАБЛИЦЫ				
		Сотрудник Установленные_данные	Сотрудник Конфиденциальные данные	Операции	Командировки	Задания
Пользователи	Каримов	Ч, М				
	Салимов	Ч	Ч	Ч, С, М	Ч, С, М	Ч, С, М
	Аълоев	Ч, М, С, У	Ч, М, С, У			
	Рузиев	Ч, М, С, У	Ч, М, С, У	Ч, М, С, У	Ч, М, С, У	Ч, М, С, У

Рис. 1.4. Модель безопасности на основе матрицы доступа (дискреционный принцип разграничения доступа).

Обозначения:
Ч - чтение;

М - модификация;
С - создание;
У - удаление (записей).

Такой подход обеспечивает гибкость настраивания системы разграничения доступа в базе данных на конкретную совокупность пользователей и ресурсов, но затрудняет общий контроль и аудит состояния безопасности данных в системе.

Принудительный подход к управлению доступом предусматривает введение единого централизованного администрирования доступом. В базе данных выделяется специальный доверенный субъект (администратор), который (и только он), собственно, и определяет разрешения на доступ всех остальных субъектов к объектам базы данных. Иначе говоря, заполнять и изменять ячейки матрицы доступа может только администратор системы.

Принудительный способ обеспечивает более жесткое централизованное управление доступом. Вместе с тем он является менее гибким и менее точным в плане настройки системы разграничение доступа на потребности и полномочия пользователей, так как наиболее полное представление о содержимом и конфиденциальности объектов (ресурсов) имеют, соответственно, их владельцы.

На практике может применяться комбинированный способ управления доступом, когда определенная часть полномочий на доступ к объектам устанавливается администратором, а другая часть владельцами объектов.

Исследования различных подходов к обеспечению информационной безопасности в традиционных (некомпьютерных) сферах и технологиях показали, что одноуровневой модели безопасности данных недостаточно для адекватного отражения реальных производственных и организационных схем. В частности, традиционные подходы используют категорирование информационных ресурсов по уровню конфиденциальности (совершенно секретно - СС, секретно - С, конфиденциально - К, и т. п.). Соответственно субъекты доступа к ним (сотрудники) также категорируются по соответствующим уровням доверия, получая так называемого допуска (допуск степени 1, допуск степени 2 и т. д.). Понятие допуска определяет мандатный (полномочный) принцип, разграничения доступа к информации. В соответствии с мандатным принципом работник, обладающий допуском степени «1», имеет право работать с любой информацией уровня «СС», «С» и «К». Работник с допуском «2», соответственно, имеет право работы с любой информацией уровня «С» и «К». Работник с допуском «3» имеет право работать с любой информацией, только уровня «К».

Мандатный принцип построения системы разграничения доступа в СУБД реализует многоуровневую модель безопасности данных, называемую еще моделью Белл - ЛаПадула (по имени ее авторов - американских специалистов Д. Белл и Л. ЛаПадула), которая иллюстрируется схемой, приведенной на рис. 1.5.

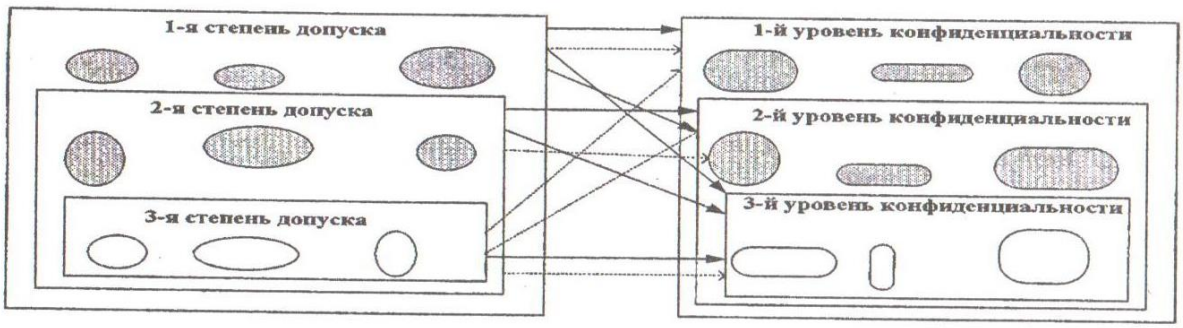


Рис. 1.5. Модель безопасности данных Белл - ЛаПадула (мандатный принцип разграничения доступа).

В модели Белл - ЛаПадула объекты и субъекты категорируются по иерархическому мандатному принципу доступа. Субъект, имеющий допуск 1-й (высшей) степени, получает доступ к объектам 1-го (высшего) уровня конфиденциальности и автоматически ко всем объектам более низких уровней конфиденциальности (т. е. к объектам 2-го и 3-го уровней). Соответственно, субъект со 2-й степенью допуска имеет доступ ко всем объектам 2-го и 3-го уровней конфиденциальности и т. д.

В модели Белл - ЛаПадула устанавливаются и поддерживаются два основных ограничения политики безопасности:

- запрет чтения вверх (no read up - NRU);
- запрет записи вниз (no write down - NWD).

Ограничение NRU является логическим следствием мандатного принципа разграничения доступа, запрещая субъектам читать данные из объектов более высокой степени конфиденциальности, чем позволяет их допуск.

Ограничение NWD предотвращает перенос (утечку) конфиденциальной информации путем ее копирования из объектов с высоким уровнем конфиденциальности в неконфиденциальные объекты или в объекты с меньшим уровнем конфиденциальности.

На практике в реальных политиках мониторов безопасности баз данных чаще всего применяется дискреционный принцип с принудительным управлением доступом, «усиливаемый» элементами мандатного принципа в сочетании с добровольным управлением доступом (допуска субъектов устанавливает и изменяет только администратор, уровень конфиденциальности объектов ус-танавли- вают и изменяют только владельцы).

Контрольные вопросы:

1. Перечислите составные элементы понятия «информационная безопасность»?
2. Какие функции реализует система управления базами данных?
3. Какие модели организации данных вы знаете?

Лекция 2: Виды систем управления базами данных и их целостность

План:

1. Виды систем управления
2. Управления базами данных и их целостность

По языкам общения СУБД делятся на открытые, замкнутые и смешанные. Открытые системы - это системы, в которых для обращения к базам данных используются универсальные языки программирования. Замкнутые системы имеют собственные языки общения с пользователями БД.

По числу уровней в архитектуре различают одноуровневые, двухуровневые, трехуровневые системы. В принципе возможно выделение и большего числа уровней. Под архитектурным уровнем СУБД понимают функциональный компонент, механизмы которого служат для поддержки некоторого уровня абстракции данных (логический и физический уровень, а также «взгляд» пользователя - внешний уровень), (рис. 1.6).

По выполняемым функциям СУБД делятся на информационные и операционные. Информационные СУБД позволяют организовать хранение информации и доступ к ней. Для выполнения более сложной обработки необходимо писать специальные программы. Операционные СУБД выполняют достаточно сложную обработку, например, автоматически позволяют получать агрегированные показатели, не хранящиеся непосредственно в базе данных, могут изменять алгоритмы обработки и т.д.

По сфере возможного применения различают универсальные и специализированные, обычно проблемно-ориентированные СУБД.

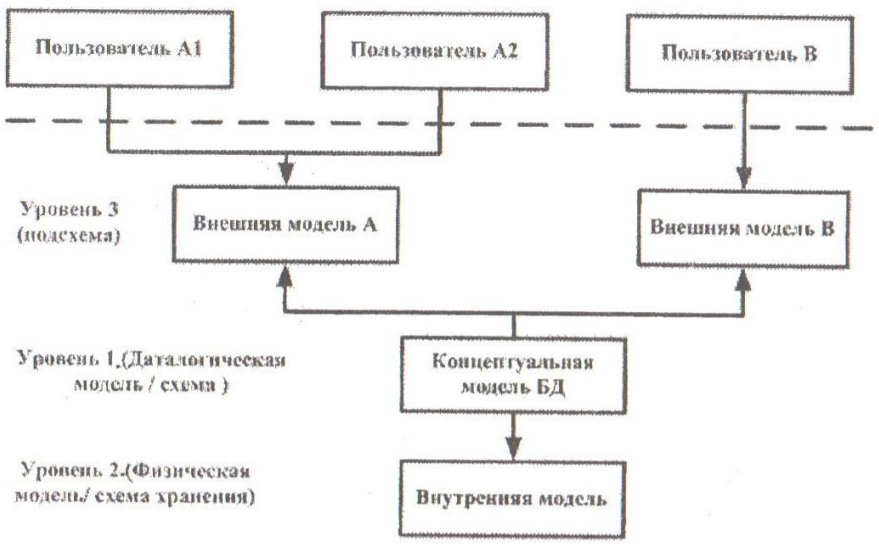


Рис. 1.6. Классификация СУБД по числу уровней в архитектуре (пример трехуровневой архитектуры).

Системы управления базами данных поддерживают разные типы данных. Набор типов данных, допустимых в разных СУБД, различен. Кроме того, ряд СУБД позволяет разработчику добавлять новые типы данных и новые операции над этими данными. Такие системы называются расширяемыми системами баз данных (РСБД).

Дальнейшим развитием концепции РСБД являются системы объектноориентированных баз данных, обладающие достаточно мощными выразительными возможностями, чтобы непосредственно моделировать сложные объекты.

По мощности СУБД делятся на настольные и корпоративные. Характерными чертами настольных СУБД являются сравнительно невысокие требования к техническим средствам, ориентация на конечного пользователя, низкая стоимость.

Корпоративные СУБД обеспечивают работу в распределенной среде, высокую производительность, поддержку коллективной работы при проектировании систем, имеют развитые средства администрирования и более широкие возможности поддержания целостности. Эти системы сложны, дороги, требуют значительных вычислительных ресурсов.

Сравнительные характеристики настольных и корпоративных СУБД приведены в табл. 1.2.

Системы обоих классов интенсивно развиваются, причем некоторые тенденции развития присущи каждому из этих классов. Прежде всего, это использование высокоуровневых средств разработки приложений, (что раньше было присуще, в основном, настольным системам), рост производительности и функциональных возможностей, работа в локальных и глобальных сетях и др.

Наиболее известными из корпоративных СУБД являются Oracle, DB2, Sybase, MS SQL Server, Progress и некоторые другие.

По ориентации на преобладающую категорию пользователей можно выделить СУБД для разработчиков и для конечных пользователей. Системы, относящиеся к первому классу, должны иметь качественные компиляторы и позволять создавать «отчуждаемые» программные продукты, обладать развитыми средствами отладки, включать средства документирования проекта и другие возможности, позволяющие строить эффективные сложные системы. Основными требованиями, предъявляемыми к системам, ориентированным на конечного пользователя, являются: удобство интерфейса, высокий

уровень языковых средств, наличие интеллектуальных модулей подсказок, повышенная защита от непреднамеренных ошибок и т.п.

Таблица.1.2

Критерий	Настольные	Корпоративные
Простота использования	+	
Стоимость программного обеспечения	+	
Стоимость эксплуатации	+	
Функциональные возможности: администрирование, работа с Интернет/интранет и др.		+
Надежность функционирования		+
Поддерживаемые объемы данных		+
Быстродействие		+
Возможности масштабирования		+
Работа в гетерогенной среде		+

Существует разделение СУБД по поколениям. К первому поколению СУБД относят системы, основанные на иерархической и сетевой моделях (60-70-е гг. XX в.), ко второму поколению - реляционные системы. СУБД третьего поколения должны поддерживать сложные структуры данных и более развитые средства обеспечения целостности данных, отвечать требованиям, предъявляемым к открытым системам.

Контрольные вопросы:

1. Какие виды СУБД различают по языкам общения?
2. Какие виды СУБД различают по числу уровней в архитектуре?

Лекция 3: Технологические аспекты информационной безопасности, языки безопасности баз данных

План:

1. Технологические аспекты защиты информации
2. Технологии надежного проектирования и администрирования

Практическая реализация политики и моделей безопасности приводящийся выше, а также аксиоматических принципов построения и функционирования, защищенных информационных систем обусловливает необходимость решения ряда программно-технологических задач, которые можно сгруппировать по следующим направлениям:

- технологии идентификации и аутентификации;
- языки безопасности баз данных;
- технологии обеспечения безопасности повторного использования объектов;
- технологии надежного проектирования и администрирования.

Атаки на хранилища и БД являются одними из самых опасных для предприятий и организаций. Согласно статистике компании infowatch, в последние годы количество утечек данных в мире неуклонно растет, при этом за последнии годы более тридцати процентов из них приходятся на внешних нарушителей и более шестидесяти выполнено с участием сотрудников организации. Даже если предположить, что в ряде случаев утечка включала данные, к которым сотрудник имеет легальный доступ, каждый третий случай приходился на внешнюю атаку. Также нужно отметить, что, согласно приведенным в

данным, на внешние атаки приходятся семь из восьми утечек объемом более десяти миллионов записей.

Злоумышленников интересуют такие виды информации, как внутренняя операционная информация, персональные данные сотрудников, финансовая информация, информация о заказчиках/клиентах, интеллектуальная собственность, исследования рынка/анализ деятельности конкурентов, платежная информация. Эти сведения в итоге хранятся в корпоративных хранилищах и БД различного объема.

Все это приводит к необходимости обеспечения защиты не только коммуникаций, операционных систем и других элементов инфраструктуры, но и хранилищ данных как еще одного барьера на пути злоумышленника. Однако на сегодняшний день работы в области обеспечения безопасности БД направлены в основном на преодоление существующих и уже известных уязвимостей, реализацию основных моделей доступа и рассмотрение вопросов, специфичных для конкретной СУБД. Целью данной работы являются комплексное рассмотрение и систематизация вопросов безопасности различных БД в свете новых угроз, общих тенденций развития информационной безопасности и возрастающей роли и разнообразия хранилищ данных.

Вопросы комплексной безопасности БД привлекают внимание исследователей, им ежегодно посвящается ряд работ как в стране, так и за рубежом. Можно отметить такие исследования, как классическая работа. В ней рассматриваются подходы к обеспечению конфиденциальности, целостности и доступности СУБД, предотвращение, определение и игнорирование атак. Предлагаются подходы к обеспечению мандатного и ролевого дискреционного доступа к реляционному серверу. Данную тему развивает работа, затрагивающая те же вопросы обеспечения разделения доступа, привилегий, аудита и шифрования данных, а также вопросы применения для обеспечения защищенного доступа встроенных механизмов, таких как триггеры, представления и хранимые процедуры. Резюмирующая их работаобобщает развитие подходов к безопасности в историческом аспекте.

Среди зарубежных работ, освещающих современные направления исследований, можно отметить. Работы российских исследователей в основном посвящены узким вопросам безопасности СУБД.

Однако эти работы, как и известные учебные пособия и материалы, в частности, также не выходят за рамки вышеозначенных тем или же, как например, отражают специфику конкретной СУБД.

Аналогичную ситуацию можно наблюдать и в работе. STIG включает известные вопросы безопасности и критерии уровневой сертификации программных средств СУБД, оценивая безопасность ПО на основании известных угроз, без учета специфики хранимых данных.

Таким образом, сегодняшние исследования в области безопасности СУБД ограничиваются развитием концепции конфиденциальности, целостности и доступности данных, что не соответствует современным требованиям к системам защиты и информационной безопасности программных решений, причем в контексте конкретных методов защиты, а не целостного рассмотрения проблемы. При этом они зачастую посвящены конкретным программным продуктам, а не всему классу соответствующего ПО.

Эволюция систем безопасности БД

Исторически развитие систем безопасности БД происходило как реакция на действия злоумышленников в соответствии с этапами эволюции самих хранилищ (БД) и изменениями типа и вида возрастающих угроз. Эти изменения были обусловлены общим развитием БД от решений на мейнфреймах до облачных хранилищ.

В архитектурном плане можно выделить следующие подходы:

- полный доступ всех пользователей к серверу БД;
- разделение пользователей на доверенных и частично доверенных средствами СУБД (системы управления БД);
- введение системы аудита (логов действий пользователей) средствами СУБД;
- введение шифрования данных; вынос средств аутентификации за пределы СУБД в операционные системы и промежуточное ПО; отказ от полностью доверенного администратора данных.

Тем не менее, введение средств защиты как реакции на угрозы не обеспечивает защиту от новых способов атак и формирует разрозненное представление о самой проблеме обеспечения безопасности. С одной стороны, крупные компании могут выделить достаточное количество средств обеспечения безопасности для своих продуктов, с другой стороны, именно по этой причине имеется большое количество разнородных решений, отсутствует понимание комплексной безопасности данных (и ее компоненты разнятся от производителя к производителю), нет общего, единого подхода к безопасности хранилищ данных и, как следствие, возможности. Усложняются прогнозирование будущих атак и перспективная разработка защитных механизмов, для многих систем сохраняется актуальность уже давно известных атак, усложняется подготовка специалистов по безопасности.

Именно разработка программных средств перспективной защиты (на опережение злоумышленника), обеспечение возможности внедрения такой технологии представляются авторам статьи наиболее актуальными задачами на текущем этапе.

Современные проблемы обеспечения безопасности БД

Список основных уязвимостей хранилищ данных, актуальный на сегодняшний день, не претерпел существенных изменений за последние более чем пять лет. Проанализировав средства обеспечения безопасности СУБД, архитектуру БД, интерфейсы, известные уязвимости и инциденты безопасности, можно выделить следующие причины возникновения такой ситуации:

- проблемами безопасности серьезно занимаются только крупные производители прежде всего в ведущих продуктах линеек для хранения данных;
- программисты БД, прикладные программисты и администраторы БД не уделяют должного внимания вопросам безопасности;
- разные масштабы и виды хранимых данных требуют разных подходов к безопасности;
- различные СУБД используют разные языковые диалекты для доступа к данным, организованным на основе одной и той же модели;
- появляются новые виды и модели хранения данных.

Рассмотрим эти положения более подробно на примере линейки продуктов от Oracle. СУБД Oracle Database Server имеет достаточно развитую систему безопасности,

включающую основные и дополнительные модули и содержащую средства гранулирования доступа до уровня записи и маскирования данных. Отметим, что продукт компании СУБД MySQL не может похвастаться таким уровнем защищенности. Это достаточно серьезная проблема, так как MySQL – широко применяемая СУБД как в электронной коммерции, так и в БД государственных структур.

Многие уязвимости, обозначенные, сохраняют актуальность за счет невнимания или незнания администраторами систем БД вопросов безопасности. Например, известные техники простой SQL-инъекции широко эксплуатируются сегодня в отношении различных web и иных приложений, в которых не уделяется внимание контролю входных данных запроса. Причинами этого являются как недостаточная информированность или внимание администраторов СУБД и прикладных программистов, так и отсутствие встроенных средств контроля известных уязвимостей в большинстве СУБД. Хорошим решением были бы автоматизация и перенос контроля таких угроз на уровень сервера, однако многообразие языковых диалектов не позволяет это сделать.

Также нужно отметить, что применение различных средств обеспечения информационной безопасности является для организации компромиссом в финансовом плане: внедрение более защищенных продуктов и подбор более квалифицированного персонала требуют больших затрат. К тому же компоненты безопасности могут отрицательно влиять на производительность систем управления БД, например уровни согласованности транзакций. Полное соответствие модели ACID – самый медленный способ обеспечения целостности при многопользовательской работе. Такие подходы, как маскирование данных или введение проверок безопасности доступа, также замедляют работу.

Еще одна причина такой ситуации – разрозненность диалектов языка запросов к СУБД. Если рассматривать только известные реляционные СУБД, несмотря на наличие развивающегося стандарта SQL (SQL-92, SQL-99, SQL-2003, SQL-2008, SQL-2012), даже крупные производители не только используют собственные расширения языка, но и не поддерживают до конца операции принятой версии стандарта. Этот факт осложняет разработку единых механизмов защиты БД уровня сервера.

Приведенные выше проблемы усугубляются с появлением и широким распространением нереляционных хранилищ данных и СУБД, оперирующих другой моделью данных, однако построенных по тем же принципам и имеющих аналогичное назначение, что и традиционные, реляционные серверы. По сути многообразие современных так называемых NoSQL (нереляционных) решений приводит к разнообразию применяемых моделей данных и размывает границу понятия БД.

Следствием этих проблем и отсутствия единых методик является нынешняя ситуация с безопасностью NoSQL-систем. Эти решения появились на рынке недавно и еще не успели пройти «путь ошибок и уязвимостей», характерный для их более зрелых реляционных аналогов. В большинстве NoSQL-систем отсутствуют не только общепринятые механизмы безопасности вроде шифрования, поддержки целостности и аудита данных, но даже развитые средства аутентификации пользователей.

Особенности систем БД как объекта защиты

В связи с появлением новых решений в области нереляционных хранилищ, размывающих границу традиционного представления о СУБД (например, система кэширования данных в памяти MemcasheDB или Hadoop HDFS), определим функции,

отличающие СУБД от файлового хранилища и других типов программных продуктов. В этом ключе в выделено несколько признаков. Переформулировав первый признак – «поддержание логически согласованного набора файлов», в силу активного развития in memory СУБД, осуществляющих хранение и все операции над данными в оперативной памяти, приведем эти критерии в следующей редакции:

- поддержание логически согласованного набора данных;
- обеспечение языка манипулирования данными;
- восстановление информации после разного рода сбоев;
- реальная параллельная работа нескольких пользователей (процессов).

Используя такой подход, можно отделить именно СУБД от файловых систем и ПО другого вида.

Отличительной особенностью систем БД от остальных видов прикладного ПО является (относительно информационной безопасности и не только) их двойственная природа. С этой точки зрения СУБД включает в себя два компонента: хранимые данные (собственно БД) и программы управления (СУБД).

Обеспечение безопасности хранимой информации, в частности, невозможно без обеспечения безопасного управления данными. Исходя из этой концепции, все уязвимости и вопросы безопасности СУБД можно разделить на две категории: зависимые от данных и независимые от данных.

Отметим, что уязвимости, независимые от данных (их структуры, организации и т.д.), являются характерными для всех прочих видов ПО. К этой группе можно отнести несвоевременное обновление ПО или наличие неиспользуемых функций.

Влияние модели данных СУБД на язык запросов

Influence of a DBMS model on a query language

Зависимыми от данных (в той или иной степени) является большое число аспектов безопасности. В частности, зависимыми напрямую можно назвать механизмы логического вывода и агрегирования данных, называемые специфичными проблемами СУБД. В то же

время многие уязвимости являются косвенно зависимыми от данных. Например, современные СУБД (считая и реляционные, и нереляционные решения) поддерживают запросы к данным с использованием некоторого языка запросов. В свою очередь, в этом качестве используются специализированные языки запросов (SQL, CQL, OQL и других), наборы доступных пользователю функций (которые, в свою очередь, тоже можно считать операторами запросного языка) или произвольные функции на языке программирования (чаще всего Java). Обобщенные интерфейсы работы с данными представлены на рисунке.

Архитектура применяемых языков, по крайней мере, то, что касается специализированных языков (запросов) и наборов функций, напрямую связана с моделью данных, применяемой для хранения информации. Таким образом, модель определяет особенности языка, а особенности языка – наличие в нем тех или иных уязвимостей. Причем уязвимости общего типа, например инъекция (под инъекцией будем понимать атаку на БД путем модификации входных запросов, заставляющую сервер СУБД выполнить нелегитимный набор действий), выполняются по-разному (SQL-инъекция, JAVA-инъекция) в зависимости от синтаксиса и семантики языка, которые, как уже сказано выше, отчасти определяются моделью данных и, следовательно, являются зависимым от данных компонентом.

Требования к безопасности БД

Таким образом, на основании разделения уязвимостей можно выделить зависимые и независимые от данных меры обеспечения безопасности хранилищ информации. Независимыми от данных можно назвать следующие требования к безопасной системе БД.

– Функционирование в доверенной среде.

Под доверенной понимается информационная среда, интегрирующая совокупность защитных механизмов, которые обеспечивают обработку информации без нарушения политики безопасности. В данном случае СУБД должна функционировать в доверенной информационной системе с соответствующими методами обмена данными.

– Организация физической безопасности файлов данных.

Данный вопрос требует более детального изучения, так как применяемые структуры данных в различных моделях данных СУБД могут иметь значение при шифровании и защите файлов данных. Однако в первом приближении вопрос физической безопасности файлов данных сходен с вопросом физической безопасности любых других файлов пользователей и приложений.

– Организация безопасной и актуальной настройки СУБД.

К данному аспекту относятся такие общие вопросы обеспечения безопасности, как своевременная установка обновлений, отключение неиспользуемых модулей или применение эффективной политики паролей.

Следующие требования можно назвать зависимыми от данных.

– Безопасность пользовательского слоя ПО.

К этой категории относятся задачи построения безопасных интерфейсов и вызовов (в том числе с учетом интерфейса СУБД и механизма доступа к данным).

– Безопасная организация данных и манипулирование ими.

Вопрос организации данных и управления ими является ключевым в системах хранения информации. Несмотря на то, что в приведенном перечне он указан последним, именно в эту область входят задачи организации данных с контролем целостности, обеспечение

защиты от логического вывода и другие, специфичные для СУБД проблемы безопасности. Фактически эта задача включает в себя основной пул зависимых от данных уязвимостей и защиты от них.

– Пути создания защищенных БД

Для преодоления названных проблем обеспечения информационной безопасности СУБД необходимо перейти от метода закрытия уязвимостей к комплексному подходу обеспечения безопасности хранилищ информации. Основными этапами этого перехода, по мнению авторов, должны стать следующие положения.

1. Разработка комплексных методик обеспечения безопасности хранилищ данных на текущем этапе.

Создание комплексных методик позволит применять их (или их соответствующие версии) при разработке хранилищ данных и пользовательского ПО. Основой для создания таких документов могут стать обобщающие проблематику работы, например или. Следование комплексной методике позволит избежать многих ошибок управления СУБД и защититься от наиболее распространенных на сегодняшний день уязвимостей.

2. Оценка и классификация угроз и уязвимостей СУБД.

Специализированная классификация угроз и уязвимостей СУБД позволит упорядочить их для последующего анализа и защиты, даст возможность установить зависимость между уязвимостями и причинами (источниками) их возникновения. В результате при введении конкретного механизма в СУБД появится возможность установить и спрогнозировать связанные с ним угрозы и заранее подготовить соответствующие средства обеспечения безопасности.

3. Разработка стандартных (применимых к различным СУБД без внесения изменений или с минимальными изменениями) механизмов обеспечения безопасности.

Стандартизация подходов и языков работы с данными позволит создать мультиплатформенные средства обеспечения безопасности, применимые к разным СУБД. С одной стороны, это методические и теоретические подходы, применимые в рамках модели данных. На сегодняшний день есть наработки таких механизмов по реляционной модели, однако они не решают всех насущных вопросов безопасности. С другой – это разработка теоретического базиса для новых СУБД, в частности, конкретизация и формализация агрегатных моделей данных. Появление готовых программных средств во многом зависит от производителей и разработчиков СУБД и их следования стандартам, а также достаточности определенных в стандарте средств для построения развитых механизмов безопасности.

4. Разработка теоретической базы информационной защиты систем хранения и манипулирования данными.

Выше отмечен ряд характерных особенностей, специфичных для хранилищ данных: двойственная природа СУБД, зависимость уязвимостей и механизмов управления от данных и описывающей их организацию модели, угрозы логического вывода, различная значимость сочетаний данных. Все это определяет специфический характер безопасности СУБД и требует новых теоретических подходов к обеспечению защиты данных в программных системах такого рода. Отдельный большой вопрос – развитие теоретического базиса в контексте формализации модели данных, а также разработка подходов обеспечения целостности информации для новых NoSQL-хранилищ.

Сформулированы проблемы информационной безопасности современных СУБД: разнообразие языковых средств, появление новых моделей данных, не подкрепленных теоретическим базисом, необходимость поиска баланса между безопасностью и ее стоимостью, развитие систем защиты как реакции на потерю средств и престижа, а также общее невнимание к вопросам безопасности.

Сформулированы критерии, выделяющие СУБД из сходных программных продуктов, с учетом новых кластерных и in memory решений, особенности этого класса ПО с точки зрения информационной безопасности и предложено базовое деление уязвимостей на зависимые и независимые от данных и их организации.

В результате сформулированы общие требования к безопасности БД, перспективные пути исследования и развития систем защиты для построения надежных и защищенных серверов по обработке информации. В них вошли как систематизация и развитие существующих подходов в виде выработки методик и стандартизации механизмов защиты, так и направления новых исследований, например, классификация уязвимостей СУБД и формализация новых моделей данных, построение прогнозирующих средств защиты.

Языки безопасности базы данных

Для определения конкретных назначений или установления правил и ограничений доступа при проектировании банков данных КС, а также в целях управления системой разграничения доступа и в более широком смысле системой коллективной обработки данных администратору системы необходим специальный инструментарий. Такой инструментарий должен основываться на определенном языке, позволяющем описывать и устанавливать те или иные назначения доступа и другие необходимые установки политики безопасности в конкретной КС.

Как следует из рассмотрения внутренней схемы баз данных, одной из основных функций систем управления базами данных является создание и поддержание собственной системы размещения и обмена данными между внешней (дисковой) и оперативной памятью. От эффективности реализации в каждой конкретной СУБД данной функции (формат файлов данных, индексирование, хэширование и буферизация) во многом зависит и эффективность функционирования СУБД в целом. Поэтому основные усилия создателей первых СУБД в конце 60-х - начале 70-х годов были сосредоточены именно в этом направлении. В результате для реализации любой функции по вводу, обработке или выводу данных требовались квалифицированные программисты для написания специальных программ на алгоритмических языках высокого уровня (в 70-х годах ФОРТРАН, КОБОЛ и др.), «знающих» особенности структуры и способы размещения данных во внешней и оперативной памяти. В итоге работа с базами данных осуществлялась через посредника в виде квалифицированного программиста, «переводящего» информационные потребности пользователя в машинный код, что схематично иллюстрируется на **рис. 1.8**.

Такое положение дел приводило к большим накладным расходам при создании и эксплуатации автоматизированных информационных систем и в определенной степени сдерживало распространение вычислительной техники в процессах информационного обеспечения деятельности предприятий и организаций.

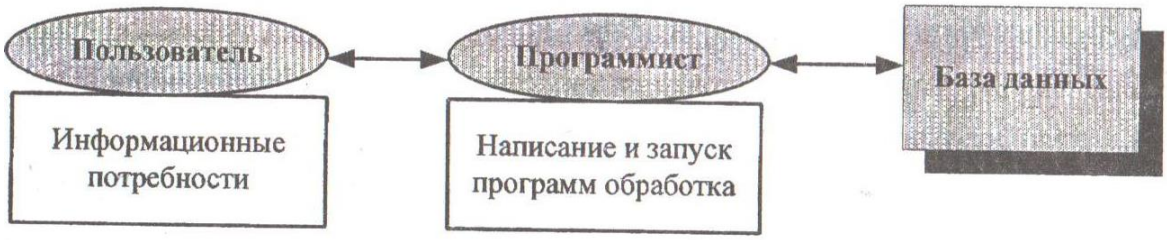


Рис. 1.8. Схема взаимодействия пользователя с базой данных в ранних СУБД.

Основателем теории реляционных СУБД Е. Коддом было выдвинуто предложение о создании специального языка для общения (взаимодействия) пользователя-непрограммиста с базами данных.

Идея такого языка сводилась к набору из нескольких фраз-прими- тивов английского языка («выбрать», «обновить», «вставить», «удалить»), через которые пользователь-непрограммист ставил бы «вопросы» к СУБД по своим информационным потребностям. В этом случае дополнительной функцией СУБД должна быть интерпретация этих «вопросов» на низкоуровневый язык машинных кодов для непосредственной обработки данных и предоставление результатов пользователю. Так родилась уже упоминавшаяся по структуре СУБД «машина данных». Иначе говоря, машина данных «понимает» язык базы данных и в результате разделяет собственно данные и задачи по их обработке. В таком подходе взаимодействие пользователя с базой данных можно проиллюстрировать схемой, приведенной на **рис. 1.9**.

Рис. 1.9. Схема взаимодействия пользователя с базой данных через язык баз данных.

В практику эти идеи шшрвые претворились в ходе реализации проекта System R (1975-1979 гг.) с участием еще одного известного специалиста по базам данных Криса Дейта. В ходе проекта System R был создан язык SEQUEL, трансформировавшийся впоследствии в язык структурированных запросов SQL (Structured Query Language). При этом дополнительно к возможностям формирования «вопросов» к базе данных пользователю также решено было предоставить и возможность описания самой структуры данных, ввода данных и их изменения. Примерно в то же время в компании IBM был создан еще один реляционный язык-QBE (Query-By-Example), т. е. язык запросов по образцу, применявшийся впоследствии во многих коммерческих системах обработки табличных данных и послуживший идеологической основой для создания визуальных «конструкторов» запросов в современных СУБД.

Быстрое и массовое распространение языка SQL в реляционных СУБД к середине 80-х годов привело фактически к принятию его в качестве стандарта по организации и обработке данных. В 1986 г. Американским национальным институтом стандартов (ANSI) и Международной организацией по стандартизации (ISO) язык был признан стандартным

языком описания и обработки данных в реляционных СУБД. В 1989 г. ANSI/ISO была принята усовершенствованная версия SQL — SQL2, а в 1992 г. третья версия — SQL3.

Язык SQL относится к так называемым декларативным (непроцедурным) языкам программирования. В отличие от процедурных языков (С, Паскаль, Фортран, Кобол, Бейсик) на лем формулируются предложения (инструкции) о том, «что сделать», но не «как сделать, как получить». Машина данных в СУБД исполняет роль интерпретатора и как раз строит машинный код, реализующий способ получения результата, задаваемого SQL-инструкциями.

Язык SQL состоит из двух частей:

- языка описания (определения) данных — DDL (Data Definition Language);
- языка манипулирования данными — DML (Data Manipulation Language).
- Синтаксис SQL-инструкций включает:
- название инструкции (команду);
- предложения, определяющие источники, условия операции;
- предикаты, определяющие способы и режимы отбора з сей, задаваемых предложениями;
- выражения, значения которых задают свойства и парам*. >? выполнения инструкции и предложения.

Структуру SQL-инструкций можно разделить на две основные части, схематично представленные на **рис. 1.10**.

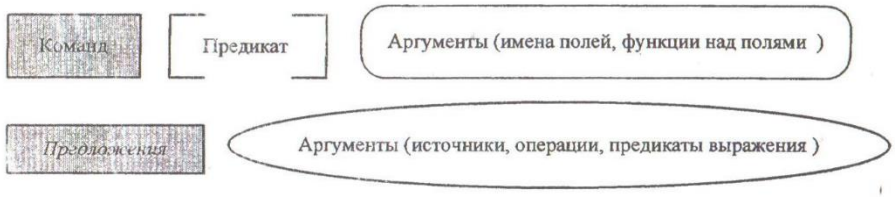


Рис. 1.10. Структура SQL-инструкций.

Первая часть включает название (команду) SQL-инструкции, предикат (необязательный элемент) и аргументы инструкции, которыми являются перечисляемые через запятую имена полей одной *mn* нескольких таблиц.

Вторая часть состоит из одного или нескольких предложений, ументы которых могут задавать источники данных (имена 5лиц, операции над таблицами), способы, условия и режимы полнения командь! (предикаты сравнения, логические и матема- (еские выражения по значениям полей таблиц). Перечень SQL- ш струкций разделяется но частям языка SQL.

В состав языка DDL входят несколько базовых инструкций, обеспечивающих основной набор функций при создании реляционных таблиц и связей между ними.

CREA.TETABLE... - создать таблицу;

CREATEJNDEX... - создать индекс;

ALTERTABLE... - изменить структуру ранее созданной таб *n* ды;

DROP... - удалить существующую таблицу и базы данных.

В структуре инструкций CREATETABLE и ALTERTABLE мажаую роль играет предложение CONSTRAINT (создать ограничения на значения данных) со следующими установками - NOT NULL (не допускаются нулевые значения по соответствующему полю), AUT01NC (поле с инкрементальным, т. е. последовательно возрастающим с каждой новой записью, характером значений) и PRIMARY KEY (определение для поля уникального, т. е. без повторов индекса, что в результате задает режим заполнения данного поля с уникальными неповторяющимися по различным строкам значениями).

В состав языка DML также входят несколько базовых инструкций, охватывающих тем не менее основные операции по вводу, обработке и выводу данных.

- SELECT... - выбрать данные из базы данных;

- INSERT... - добавить данные в базу данных;
- UPDATE... - обновить данные в базе данных;
- DELETE... - удалить данные;
- G RANT... - предоставить привилеги и пользователю;
- REVOKE... -отменить привилегии пользователю;
- COMMIT... - зафиксировать текущую транзакцию;
- ROLLBACK... - прервать текущую транзакцию.

Важное значение имеют разновидности инструкции SELECT SELECT... INTO ... (выбрать из одной или нескольких таблиц набор записей, из которого создать новую **таблицу)** и UNION SELECT, которая в дополнении с исходной инструкцией SELECT (SELECT... UNION SELECT...) реализует операцию объединения таблиц.

Помимо предложения **CONSTRAINT в** SQL-инструкциях используются следующие предложения

FROM... указывает таблицы или запросы, которые содержат поля, перечисленные в инструкции SELECT;

WHERE... определяет, какие записи из таблиц, перечисленных в предложении FROM, следует включить в результат выполнения инструкции SELECT, UPDATE или DELETE;

GROUP BY... — объединяет записи с одинаковыми значениями в указанном списке полей в одну запись;

HAVING... — определяет, какие сгруппированные записи отображаются при использовании инструкции SELECT с предложением GROUP BY;

IN... — определяет таблицы в любой внешней базе данных, с которой ядро СУБД может установить связь;

ORDERBY... — сортирует записи, полученные в результате запроса, в порядке возрастания или убывания на основе значений указанного поля или полей.

В качестве источника данных по предложению FROM, помимо таблиц и запросов, могут использоваться также результаты операций соединения таблиц в трех разновидностях—INNER JOIN... ON..., LEFT JOIN. ..ON... и RIGHT JOIN...ON... (внутреннее соеды- нение, левое и правое внешнее соединение, соответственно).

Предикаты используются для задания способов и режимов использования записей, отбираемых на основе условий в инструкции SQL. Такими предикатами являются:

ALL... — отбирает все записи, соответствующие условиям, заданным в инструкции SQL;

DISTINCT... — исключает записи, которые содержат повторяющиеся значения в выбранных полях;

DISTINCTROW... — опускает данные, основанные на целиком повторяющихся записях;

TOP... — возвращает записи, находящихся в начале или в конце диапазона, описанного с помощью предложения ORDER BY;

Выражениями в инструкциях SQL являются любые комбинации операторов, констант, значений текстовых констант, функций, имен полей, построенные по правилам математических выражений, и результатом которых является конкретное, в том числе и логическое значение.

Базовых инструкций языка SQL представлены инструкции GRANT и REVOKE, предоставляющие или отменяющие привилегии пользователям. Структура инструкции GRANT выглядит следующим образом:

С11АМТсписок_привилегий_через_запятую ON ИмяОбъекта

ТОИменаПользователей_через_запятую

[WITHGRANTOPTION];

где:

- список привилегий составляют разрешенные инструкции (операции) над объектом (таблицей) - SELECT, INSERT UPDATE DELETE;

– список пользователей представляется их именамииденти- фикаторами или может быть заменен ключевым словом PUBLIC, которое идентифицирует всех пользователей, зарегистрированных в системе;

– директива WITH GRANT OPTION наделяет перечисленных пользователей дополнительными особыми полномочиями по предоставлению указанных в списке привилегий-полномочий другим пользователям.

В большинстве случаев право подачи команд GRAND и REVOKE по конкретному объекту автоматически имеют пользователи, создавшие данный объект, т. е. их владельцы. В других подходах этим правом наделяются доверенные субъекты, т. е. администраторы.

Хотя в явном виде такой подход не предусматривает создание матрицы доступа, тем не менее, реализуется классический принцип дискреционного разграничения доступа с сочетанием как добровольного, так и принудительного управления доступом.

На самом деле в большинстве СУБД привилегии и установки доступа, как и структура базы данных, «прописываются» в системных таблицах БД, т.е. в системном каталоге БД, который можно рассматривать, в том числе, и в качестве матрицы доступа.

Как уже отмечалось, дискреционный принцип обладает большой гибкостью по настройке системы разграничения доступа на особенности предметной области базы данных и потребности пользователей, но не обеспечивает эффективной управляемости и затрудняет проведение какой - либо целенаправленной политики безопасности в системе. Преодоление этого недостатка достигается двумя путями: использованием техники «представлений» и специальными расширениями языка SQL.

«Представлением» называется глобальный авторизованный запрос на выборку данных, формирующий для пользователя «свое» представление определенного объекта (объектов), совокупность которых формирует некую виртуальную базу данных, со своей схемой (объектами) и данными (отобранными или специально преобразованными). При входе пользователя в систему в процессе его идентификации и аутентификации ядро безопасности отыскивает для пользователя соответствующие представлениязапросы и передает запрос основному ядру СУБД для выполнения. В результате выполнения запроса пользователь «видит» и имеет доступ только к тем объектам, которые соответствуют его полномочиям и функциям.

В целом создание системы разграничения доступа через технику представлений является более простым способом, чем непосредственное использование инструкций GRANT, и осуществляется в два этапа:

– Для всех зарегистрированных пользователей в системе с помощью конструкций CREATE VIEW создаются свои представления объектов базы данных.

– С помощью инструкций «GRANT SELECT ON-ИмяПредс- тавления TO - ИмяПользователя» созданные представления авторизуются со своими пользователями.

Вместе с тем такой подход является более грубым по сравнению с применением инструкции GRANT непосредственно к объектам базы данных, т. к. не обеспечивает расщепления установок доступа к объектам на уровне отдельных операций (SELECT, INSERT, UPDATE, DELETE).

Поэтому другим подходом являются специальные расширения языка SQL, основанные на событийно-процедурной идеологии с введением специальных правил (RULE) безопасности:

CREATESECURITYRULEИмяПравила

011АШ’список_привилегий_через_запятую ON ИмяОбъекта

WHERE условия

TO Имена Пользователей_через_залятую

Введение правил безопасности обеспечивает более широкие и гибкие возможности реализации различных политик безопасности с большей степенью контроля и управляемости, но, как, впрочем, и техника представлений и непосредственное

использование инструкций GRANT, не позволяет строить системы с мандатным принципом разграничения доступа.

Для решения этой задачи могут предлагаться более кардинальные расширения языка SQL с введением возможностей создания объектов базы данных с метками конфиденциальности. Следует, однако, заметить, что подобные примеры в коммерческих и сертифицированных по требованиям безопасности СУБД чрезвычайно редки.

По языкам безопасности баз отметим, что в современных СУБД для реализации установок, правил и ограничений доступа разрабатывается и используется специальный диалоговонаглядный интерфейс, автоматически формирующий соответствующие конструкции языка SQL и позволяющий в большинстве случаев обходиться без непосредственного программирования.

Контрольные вопросы:

1. Какие программно-технологические задачи решаются при обеспечении безопасности базы данных?

2. Перечислите основные и дополнительные функции системы управления базами данных.

3. Объясните схему взаимодействия пользователя с базой данных в ранних СУБД.

Лекции 4. Модели и методы разграничения доступа в базы данных

План:

1. Модели безопасности данных
2. Модель управления доступом

Одним из основных направлений информационной безопасности является создание формальных моделей информационной' безопасности, называемых также моделями разграничения доступа. Под моделью информационной безопасности понимают формально описанную политику безопасности - совокупность норм и правил, обеспечивающих эффективную защиту системы обработки информации от заданного множества угроз безопасности.

Модели безопасности играют важную роль в процессах разработки и исследования защищенных компьютерных систем, так как обеспечивают системотехнический подход, включающий решение следующих важнейших задач:

– выбор и обоснование базовых принципов архитектуры защищенных компьютерных систем, определяющих механизмы реализации средств и методов защиты информации;

– подтверждение свойств (защищенности) разрабатываемых систем путем формального доказательства соблюдения политики безопасности (требований, условий, критериев);

– составление формальной спецификации политики безопасности как важнейшей составной части организационного и документационного обеспечения разрабатываемых защищенных компьютерных систем.

По сути, модели безопасности являются исходным связующим элементом в триаде «Заказчик (Потребитель) - Разработчик (Производитель)-Эксперт (Аудитор)». На основе моделей безопасности заказчики могут формулировать те требования к защищенным КС, которые соответствуют политике безопасности, технологическим процессам обработки информации, принятым в своих организациях и предприятиях. Разработчики на основе моделей безопасности формируют технико-технологические требования и программно-технические решения по разрабатываемым системам. Эксперты, основываясь на моделях

безопасности, строят методики и спецификации оценки защищенности конкретных систем, осуществляют сертификацию разработанных систем по требованиям защиты информации.

Модель управления доступом - это структура, которая определяет порядок доступа субъектов к объектам **(рис.2.1)**.

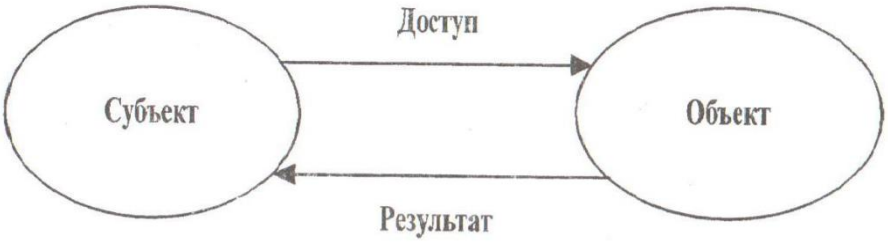


Рис. 2.1. Структура модели управления доступом**.**

Для реализации правил и целей этой модели используются технологии управления доступом и механизмы безопасности. Среди множества моделей безопасности можно выделить основные типы моделей: дискреционные модели, мандатные модели, модели с ролевым разграничением доступа. Каждая модель использует различные методы для управления доступом субъектов к объектам, каждая имеет свои преимущества и ограничения. Надо отметить, что эти методы не обязательно применяются отдельно друг от друга, а могут комбинироваться для удовлетворения различных требований к безопасности системы **рис.2.2.**

Рис.2.2. Системы контроля доступа.

Контрольные вопросы:

1. Объясните понятие «политика безопасности» и «модель безопасности».
2. Место модели безопасности при создании и исследовании защищенных компьютерных систем.
3. Структура модели управления доступом.
4. Объясните схему системы контроля доступом.

Лекции 5: Организация разграничения доступа в базы данных на основе дискреционной модели

План:

1. Мандатные модели основаны на мандатном разграничении доступа
2. Дискреционные модели безопасности

Мандатные модели основаны на мандатном разграничении доступа (Mandatory Access Control), представляющем собой совокупность правил предоставления доступа, определенных на множестве атрибутов безопасности субъектов и объектов.

Мандатное разграничение доступа - это разграничение доступа субъектов к объектам данных, основанное на характеризуемой меткой конфиденциальности информации, которая содержится в объектах, и на официальном разрешении (допуске) субъектов обращаться к информации такого уровня конфиденциальности. В отличии от дискреционного мандатный доступ накладывает ограничения на передачу информации от одного пользователя к другому. Это позволяет разрешить проблему троянских коней. Примерами мандатных моделей: модель Белл-ЛаПадула, модель MMS. Как известно, в мандатных моделях, в частности, в классической модели Белл-ЛаПадулы, в качестве классифицирующего множества используется линейная решетка уровней конфиденциальности. Отображение (классификация) субъектов доступа (пользователей) на решетку уровней конфиденциальности отражает парадигму градуированного доверия (уровни допуска), а отображение объектов доступа - парадигму рангового измерения конфиденциальности (грифы секретности), т.е. степени ущерба от неконтролируемого распространения соответствующей информации.

Метки конфиденциальности главным образом характеризуют данные: их принадлежность (группу принадлежности), важность, представительность, уровни конфиденциальности и ценности данных объекта (таблицы, столбцы, строки или поля) и пр. Метки конфиденциальности неизменны на всем протяжении существования объекта защиты (они уничтожаются только вместе с ним) и располагаются вместе с защищаемыми данными.

Реализация такого вида разграничения в ПРД СУБД не дает проигнорировать метки конфиденциальности при получении доступа к информации. Такие реализации ПРД, как правило, представляют собой комплекс средств как на машине-сервере, так и на машине-клиенте, при этом возможно использование специальной защищенной версии операционной системы.

Существует также реализация некоторой модификации мандатного разграничения доступа посредством применения сложного набора хранимых процедур. При этом метки добавляются в таблицу в качестве дополнительного атрибута, доступ к таблицам запрещается вообще и ни одно приложение не может выполнить интерактивный SQL-запрос, а только хранимую процедуру. Реализации политики разграничения доступа в этом случае достаточно сложна и предполагает определенный уровень доверия к администратору безопасности, так как он имеет право изменять структуру базы данных, а значит и хранимые процедуры, представления. Физически же администратор безопасности в данном случае не изолирован от управления конфиденциальными (секретными) данными.

Установлено, что разграничение доступа пользователей различной степени благонадежности к фрагментам мультиконфи- денциальных данных возможно реализовать посредством построения многоуровневой политики разграничения доступа, в которой управление и контроль доступа осуществляется в соответствии со степенями конфиденциальности хранимой в БД информации.

Многоуровневая политика разграничения доступа строится на основе модели Белл-ЛаПадула, которая предназначена для управления субъектами, то есть активными процессами, запрашивающими доступ к данным, и объектами, то есть файлами, таблицами, представлениями, записями, полями. Сущность моделирования по Белл-ЛаПадула заключается в классификации объектов по степени конфиденциальности, а субъектов - по уровню благонадежности. После чего формируется правило, описывающее набор полномочий уровня допуска относительно класса конфиденциальности.

Механизм защиты базы данных, основанный на модели Белл-ЛаПадула, строится на основе «обратного наследования» категорий информации классами конфиденциальности данных. Смысл обратного наследования иллюстрируется рис. 2.5 и заключается в

следующем. Задается некоторая иерархия классов конфиденциальности данных. Кроме того, по признаку значимости (важности) формируется совокупность групп данных в рамках классов, представляющая множество категорий. С повышением степени важности информации наследуется соответствующая категория информации. Это означает, что пользователь, допущенный к соответствующей категории информации, обладает правами чтения данных с меньшей важностью.

Рис. 2.5. Иерархия классов конфиденциальности данных

Доступ к данным соответствующей категории обеспечивается так называемым фильтром доступа, который выбирается в соответствии с уровнем допуска пользователя. Модель ПРД с мандатным разграничением доступа представлена на рис. 2.6.

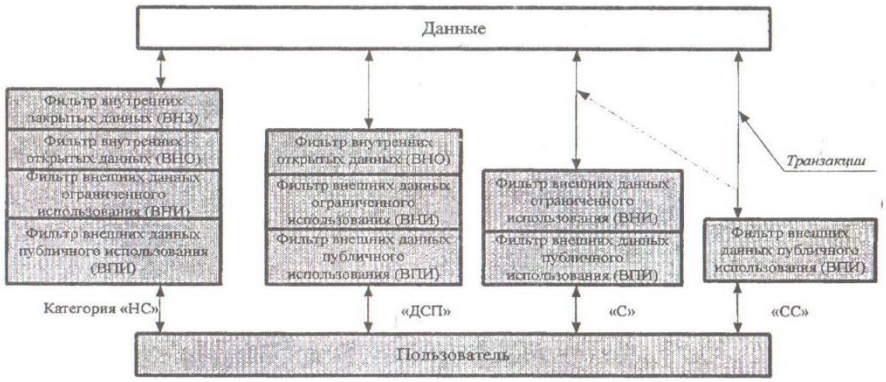


Рис. 2.6. Модель ПРД с мандатным разграничением доступа

На более высоких уровнях модели безопасности данных определяется система полномочий пользователя по обработке информации (рис. 2.7).

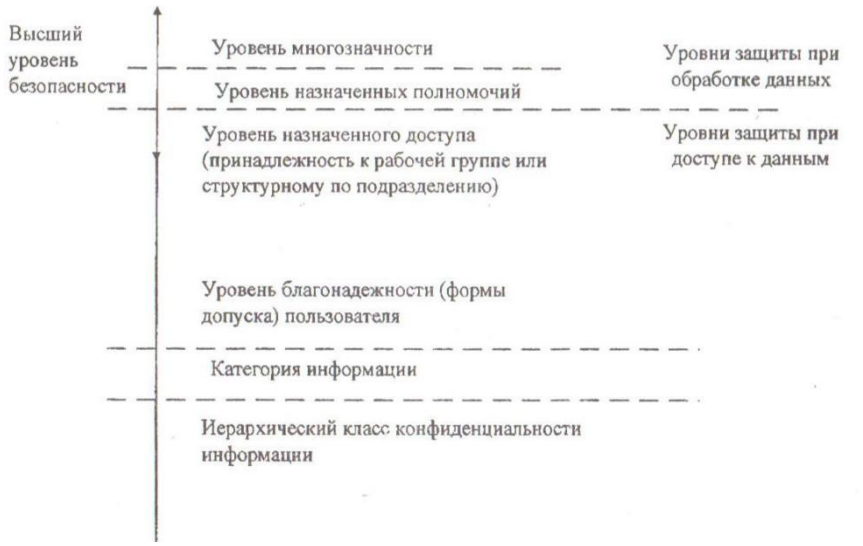


Рис. 2.7. Классификационные признаки объектов и субъектов при мандатном разграничении доступа

Свойство системы защиты, которое позволяет субъекту иметь право на запись в объект, только в том случае, если класс субъекта такой же или ниже, чем записываемого объекта, будем называть ∂ - свойством (сигма). Пример реализации освойства изображен на рис.2.8. Данные в таблице отсортированы в порядке убывания степени конфиденциальности. Процесс с допуском «С» может записать данные только в кортежи с соответствующим или более высоким классом конфиденциальности, а читать данные - только из кортежей с соответствующим или более низким классом конфиденциальности.

Реализация принципов Белл-ЛаПадула связана с требованием поддержки для одной и той же таблицы нескольких уровней защиты, что приводит к некоторому снижению устойчивости, надежности и управляемости СУБД.

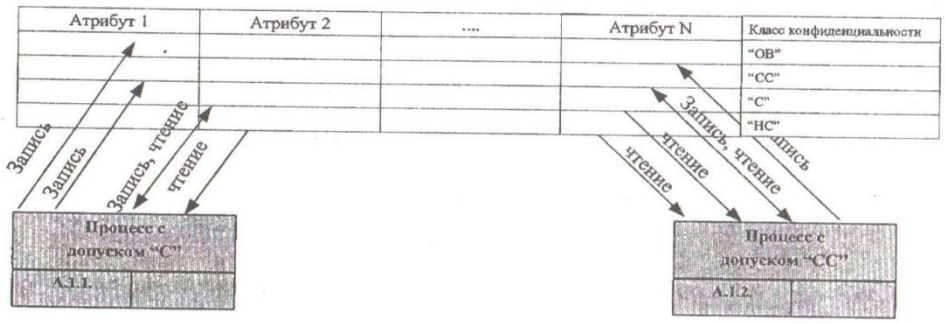


Рис. 2.8. Пример применения правил Белл-ЛаПадула.

Известны также работы по расширению мандатных моделей введением более сложных классифицирующих множеств, учитывающих, в том числе, и тематический аспект, основывающийся на использовании тематических классификаторов-рубрикаторов (дескрипторных, иерархических, фасетных) - т.н. MLS-решеток. Однако модели с MLS-решетками не изменяют сущности исходных мандатных моделей, имеющих чрезвычайно абстрактный по отношению к реальным СУБД, в частности, не регламентируют в большинстве случаев процессы присвоения и изменения классификаций сущностей (субъектов и объектов) доступа, а в более широком плане не регламентируют процессы инициализации субъектов доступа, а также множественные доступы (один субъект — одновременно к нескольким объектам, несколько субъектов - одновременно к одному объекту). Кроме того, мандатные модели ни каким образом не учитывают структуру системы объектов в СУБД,

рассматривая их как множество объектов, охваченное линейным порядком по отношению конфиденциальности (секретности). В результате применения мандатных моделей в СУБД не обеспечивает единого механизма организации данных и разграничения доступа к ним.

Дискреционные модели безопасности - модели, основанные на дискреционном управлении доступом (Discretionary Access Control). Дискреционное разграничение доступа - разграничение доступа между поименованными субъектами и поименованными объектами. Реализация дискреционного разграничения доступа осуществляется с учетом следующих положений:

– каждый пользователь должен быть аутентифицирован прежде чем он получит доступ к данным;

– в соответствии с аутентифицирующей информацией (идентификатор и пароль) пользователю назначается система его полномочий. Полномочие пользователя заключается в том, что он имеет право доступа только к фиксированному набору данных и процедур их обработки. При этом никакой запрос пользователя не должен допускать вовлечение в процесс обработки недоступных ему данных. Совокупность аутентифицирующей информации пользователя и полномочий его доступа к сегментам данным и функциям их обработки образует маркер доступа, определяющий уровень допуска.

Для задания и проверки полномочий удобно субъектобъек- тные отношения представлять в виде матрицы безопасности, которая изображена на **рис. 2.3.** В матрице безопасности перечисляются по строкам все пользователи, а по столбцам все фрагменты данных. На пересечениях отмечаются допустимые операции над данными.

Пользователь	Таблица 1		…		…	Таблица M		…	
	Поле 1	Поле 2	…	Поле N	…	Поле 1	Поле 2	…	Поле N
User 1	RWCD	RWCD		RW		RW	RWD		R
User 2	RWC	RWCD		RWC		R	RW		RWCD
…									
User X	R	RWD		RWCD		RWD	R		RWC

Рис. 2.3. Пример матрицы безопасности.

Проверка полномочий, основанная на матрице безопасности, не гарантирует защищенность системы, так как не предоставляет средств проверки подлинности пользователя (процесса), запрашивающего данные, что может привести к несанкционированному доступу (рис. 2.4).

Поэтому защищенная СУБД должна предусматривать наличие в подсистеме разграничения доступа (ПРД) проверку подлинности, которая обеспечивает подтверждение заявленных пользователем или процессом идентификаторов.

Однако, применение маркеров доступа в ПРД не позволяет организовать разграничение доступа к данным с разной степенью конфиденциальности. Действительно, если пользователь заявил санкционированные полномочия доступа к данным фрагмента Z, а в нем содержится информация различной степени конфиденциальности, то автоматически данный пользователь получает возможность работать со всеми данными фрагмента ***L.***

К достоинствам дискреционного разграничения доступа можно отнести хорошую детализацию защиты и относительно простую реализацию. Но этот вид обладает и рядом недостатков.

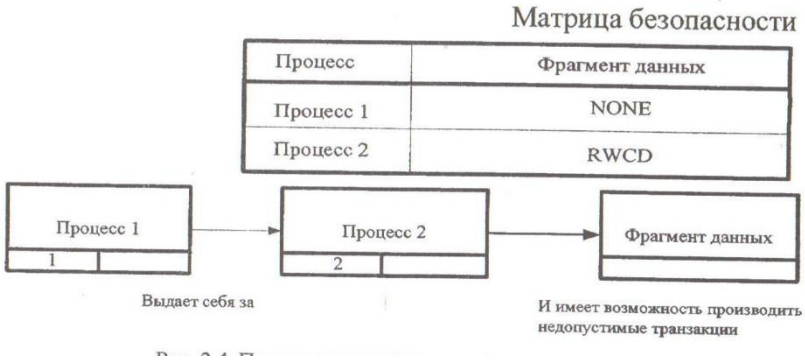


Рис. 2.4. Пример несанкционированного доступа к данным.

Доступ ограничивается только к именованным объектам, а не собственно к хранящимся данным. Так например, в случае применения реляционной СУБД объектом будет являться таблица. В этом случае нельзя в полном объеме ограничить доступ только к части информации, хранящейся в таблице. Кроме этого, существует проблема троянских программ (троянских коней). Когда пользователь вызывает какую-либо программу на компьютере, в системе инициируется некоторая последовательность операций, зачастую скрытых от пользователя. Эти операции обычно управляются операционной системой.

Применение средств дискреционного разграничения доступа не позволяет решить задачу контроля за передачей информации. Это обусловлено тем, что указанные средства не могут помешать авторизованному пользователю законным образом получить конфиденциальную информацию и затем сделать ее доступной для других неавторизованных пользователей. Это становится возможным потому, что привилегии существуют отдельно от данных (в случае реляционных СУБД - отдельно от строк реляционных таблиц), в результате чего данные оказываются «обезличенными» и ничто не мешает передать их кому угодно даже средствами самой СУБД, получив доступ к таблице или представлению.

Одна из первых моделей безопасности была модель дискреционного доступа, модель АДЕПТ-50. В модели представлено четыре типа объектов, относящихся к безопасности: пользователи (и), задания (j), терминалы (t) и файлы (f), причем каждый объект описывается четырехмерным кортежем.

Контрольные вопросы:

1. Что понимается под управлением доступа?
2. Объясните матрицу безопасности в дискреционном управлении доступом.
3. Преимущества дискреционного разграничения доступа.
4. Недостатки дискреционного разграничения доступа.

Лекции 6: Организация разграничения доступа в базы данных на основе мандатной модели

План:

1. Модели с ролевым разграничением доступа
2. Администрирование множеств авторизованных ролей пользователей
3. Администрирование иерархии ролей

Модели с ролевым разграничением доступа (Role-Based Access Control) представляют собой развитие политики дискреционного разграничения доступа. Права доступа субъектов системы к объектам группируются с учетом специфики их применения, образуя роли. При этом правила данной модели являются более гибкими, чем правила мандатной модели,

построенные на основе жестко определенной решетки ценности информации. В ролевой модели классическое понятие «субъект» заменяется понятиями «пользователь» и «роль». Пользователь - это человек, работающий с системой и выполняющий определенные служебные обязанности. Роль - это активно действующая в системе абстрактная сущность, с которой связан ограниченный, логически связанный набор полномочий, необходимый для осуществления определенной деятельности. При использовании ролевой политики управление доступом осуществляется в две стадии: во-первых, для каждой роли указывается набор полномочий, представляющих набор прав доступа к объек там, и, во-вторых, каждому пользователю назначается список доступных ему ролей.

В ПРД с ролевым разграничением доступа (РРД) пользователи не могут передавать права на доступ к информации другим пользователям, что является фундаментальным отличием РРД от дискреционного и мандатного видов разграничения доступа.

Определение членства и распределение полномочий роли при РРД (в отличие от дискреционного доступа) зависит не от администратора безопасности, а от политики безопасности, принятой в организации и конкретно СУБД. Роль можно понимать, как множество действий, которые пользователь или группа пользователей может исполнять в контексте организации. Понятие роли включает описание обязанностей, ответственности и квалификации. Функции распределяются по ролям администратором безопасности СУБД. Доступ пользователя к роли также определяется этим администратором.

Ролевой вид разграничения доступа подразумевает контроль доступа на уровне абстракции и описании сущностей, используемых в организации. Если ПРД построена на основе РРД, то добавление и удаление ролей становится несложным процессом. Таким образом, РРД позволяет администратору безопасности оперировать с абстракциями более высокого уровня, чем традиционные списки контроля доступа, используемые дискреционным доступом.

Политики безопасности, основанные на РРД, описываются в терминах пользователей, ролей, операций и защищаемых объектов (рис. 2.9).

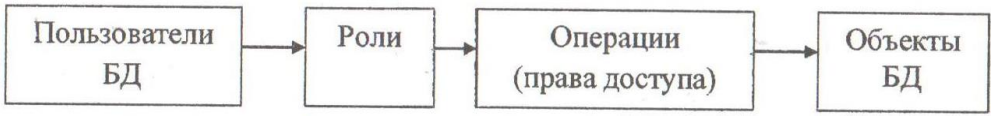


Рис. 2.9. Механизм доступа пользователей к информации, хранимой в базе данных, при РРД.

Для того, чтобы применить некоторую операцию к защищаемому РРД объекту, пользователь должен выполнять некоторую роль. До того, как пользователь может выполнять данную роль, он должен быть авторизован для данной роли администратором безопасности СУБД. Таким образом, РРД наделяет администратора способностью устанавливать ограничения на авторизацию в роли, активацию роли, выполнение операций.

Задание ролей позволяет определить более четкие и понятные' для пользователей компьютерной системы правила разграничения доступа. При этом РРД наиболее эффективно используется в компьютерных системах, для пользователей которых четко определен круг их должностных полномочий и обязанностей.

Роль является совокупностью прав доступа на объекты компьютерной системы. Однако РРД не является частным случаем дискреционного разграничения доступа, так как правила РРД определяют порядок предоставления прав доступа субъектам компьютерной системы в зависимости от сессии его работы и от имеющихся или отсутствующих у него ролей в каждый момент времени, что является характерным для систем мандатного разграничения доступа. В то же время, правила РРД являются более гибкими, чем правила мандатного разграничения доступа, построенные на основе жестко определенной решетки (шкалы) ценности информации.

Основными элементами базовой модели РРД являются:

U - множество пользователей;

R - множество ролей;

P - множество прав доступа на объекты компьютерной системы;

S - множество сессий пользователей;

– $PA: R \rightarrow 2^R$ _ функция, определяющая для каждой роли множество прав доступа; при этом для каждого $p \& p$ существует $г$ е R такая, что p е $PA(Г)$;

– $UA: U \rightarrow 2^*$ _ функция, определяющая для каждого пользователя множество ролей, на которые он может быть авторизован;

– **user**: $S \rightarrow U$. функция, определяющая для каждой сессии пользователя, от имени которого она активизирована;

– **roies** . $г \rightarrow z$ д,

– - функция, определяющая для пользователя множество ролей, на которые он авторизован в данной сессии; при

– этом в каждый момент времени для каждого seS выполняется условие $ro \wedge es(s) \subseteq UA(user(s))$

Множество ролей, на которые авторизуется пользователь в течение одной сессии, модифицируется самим пользователем. В базовой модели РРД отсутствуют механизмы, позволяющие одной сессии активизировать другую сессию. Все сессии активизируются пользователем. Важным механизмом базовой модели РРД являются ограничения, накладываемые на множества ролей, на которые может быть авторизован пользователь или на которые он авторизуется в течение одной сессии. Данный механизм также необходим для широкого использования РРД, так как обеспечивает большее соответствие, используемым в компьютерных системах, технологиям обработки информации.

В базовой модели РРД предполагается, что множества U, R, P и функции PA, UA не изменяются с течением времени или существует единственная роль - «администратор безопасности», которая предоставляет возможность изменять эти множества и функции. В реальных компьютерных системах, в которых одновременно могут работать сотни и тысячи пользователей, а структура ролей и прав доступа может быть очень сложной, проблема администрирования является чрезвычайно важной задачей. Для решения этой задачи рассматривается построенная на основе базовой модели РРД модель администрирования РРД.

Общая структура элементов базовой модели РРД имеет вид, представленный на рис.2.10.

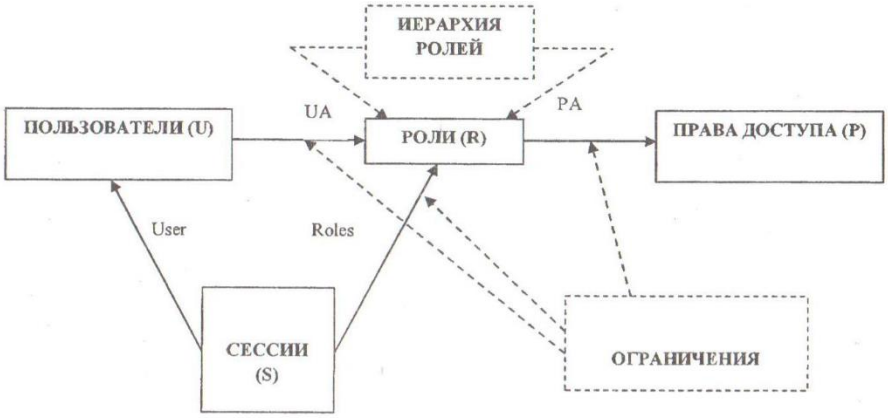
В дополнение к используемым элементам базовой модели РРД в модели администрирования РРД рассматриваются следующие элементы:

AR - множество административных ролей;

AP - множество административных прав доступа;

APA: AR -> 2^{AP} — функция, определяющая для каждой административной роли множество административных прав доступа;

AUA: U -» $2^{л/?}$ - функция, определяющая для каждого пользователя множество административных ролей, на которые он может быть авторизован.



Общая структура элементов модели администрирования РРД имеет вид, представленный на рис. 2.11.

При администрировании множеств авторизованных ролей пользователей изменяются значения функции *A UA.* Для изменения значений функции *UA()* определяются специальные административные роли из множества *AR.*

Для администрирования множеств авторизованных ролей пользователей необходимо определять:

– для каждой административной роли множество ролей, множества авторизованных пользователей, которых она позволяет изменять;

– для каждой роли предварительное условие, которому должны соответствовать пользователи, прежде чем они будут включены в множество ее авторизованных пользователей. Типовая структура иерархии ролей и иерархии административных ролей представлена на рис. 2.12. *а и б.*

Минимальная роль в иерархии - служащий *(E).* Иерархия ролей разработчиков проектов имеет максимальную роль - директор *(DIR),* минимальную роль - инженер *(ED).* В управлении выполняются работы по двум проектам. В каждом проекте определены максимальная роль - руководитель проекта *(РЫ, PL2*, соответственно), минимальная роль - инженер проекта *(El, E2*, соответственно) и не сравнимые между собой роли - инженер по производству *(PEI, PE2,* соответственно) и инженер по контролю *(OE1, QE2,* соответственно). Иерархия административных ролей состоит из четырех ролей с максимальной ролью - администратор безопасности *(SS).*

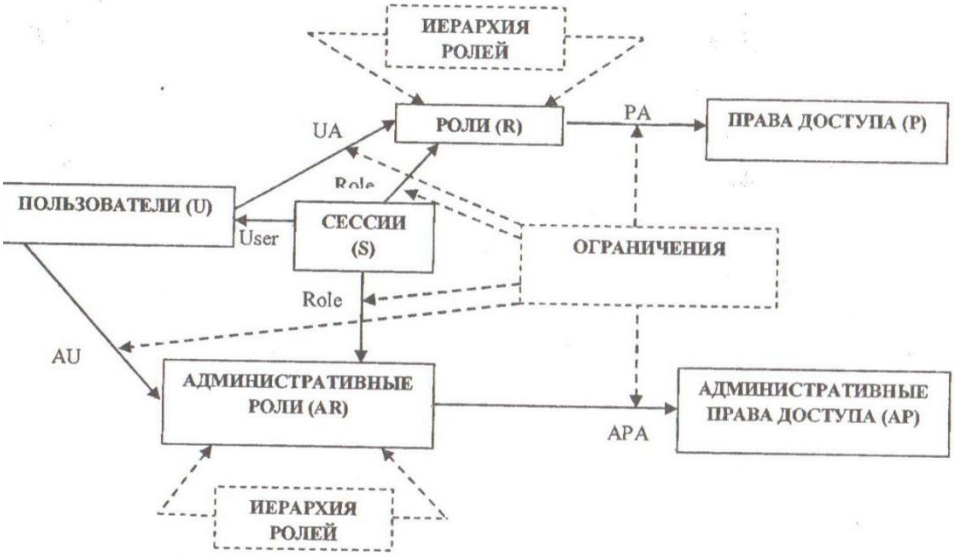


Рис.2.11. Структура модели администрирования РРД

Определение правил администрирования, позволяющих изменять иерархию ролей, является самой сложной задачей, в рассматриваемой модели администрирования РРД. Для решения данной задачи используются подходы, реализованные при определении правил администрирования множеств авторизованных ролей пользователей и прав доступа ролей. Задаются три иерархии, элементами которых являются:

- возможности - множества прав доступа и других возможностей;
- группы - множества пользователей и других групп;
- объединения - множества пользователей, прав доступа, групп, возможностей и других объединений.

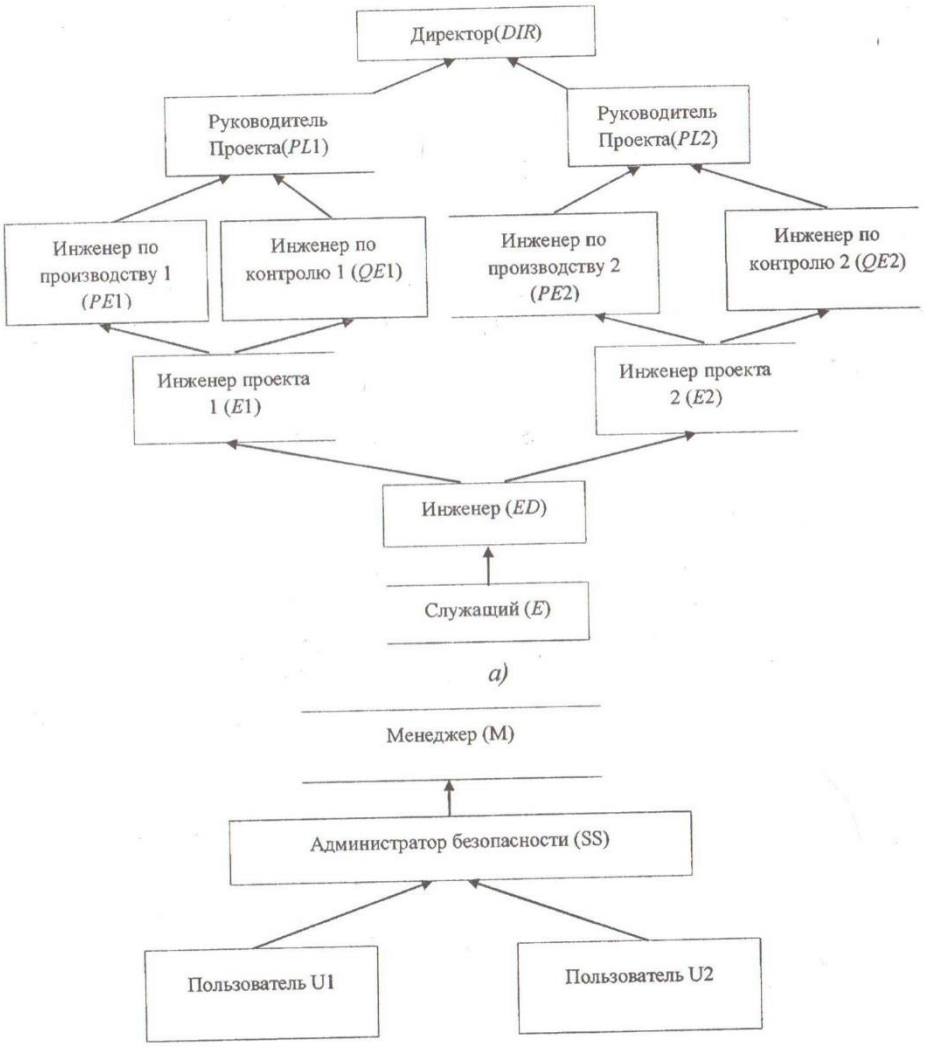


Рис.2.12. Иерархия административных ролей.

Иерархия объединений является наиболее общей и может включать в себя иерархии возможностей и групп. Определение возможностей и групп требуется для обеспечения соответствия правил администрирования ролей в модели и используемых на практике технологий обработки информации и создания административных структур организаций. Например, для выполнения своих функций пользователю может быть необходим некоторый набор прав доступа, причем отсутствие в этом наборе некоторого права доступа может сделать бессмысленным обладание имеющимися правами.

На основе иерархий возможностей, групп и объединений задается иерархия ролей, элементами которой являются роли- возможности, роли-группы, роли-объединения:

Роли-возможности - роли, которые обладают только определенными в соответствующей возможности правами доступа.

Роли-группы - роли, на которые могут быть авторизованы одновременно только все пользователи соответствующей группы.

Роли-объединения - роли, которые обладают возможностями, правами доступа и на которые могут быть авторизованы группы пользователей и отдельные пользователи.

Контрольные вопросы:

1. Объясните суть модели с ролевым разграниченем доступа.

2. Раличие модели с ролевым разграничением доступа от дискреционной и мандатной моделей.

3. Объясните механизм доступа пользователей к информации в модели с ролевым разграничением доступа.

4. Структура базовой модели с ролевым разграничением доступа.

5. Структура модели администрирования ролевого разграничения доступа.

6. Объясните иерархию административных ролей.

Лекции 7: Технология надежности проектирование и администрирования

План:

1. Технологические аспекты защиты информации
2. Надежное проектирование и администрирование

Часть угроз безопасности информации возникает изза непреднамеренных (или преднамеренных) ошибок на этапах жизненного цикла КС - при разработке программного обеспечения СУБД; при проектировании и создании на базе СУБД конкретной КС, и, в том числе, при проектировании системы разграничения доступа; при администрировании и сопровождении системы и, в том числе, при реагировании и действиях пользователей во внештатных ситуациях; при технологических операциях по резервированию, архивированию и восстановлению информации после сбоев; при выводе КС из эксплуатации. С целью нейтрализации или снижения вероятности данных угроз применяются ряд организационно - технологических и технических средств, решений, объединяемых в общую группу технологий надежного проектирования и администрирования. Их также условно можно разделить на следующие подгруппы:

– технологии надежной разработки программного обеспечения;
– технологии надежного проектирования и создания КС;
– технические средства и специальный инструментарий администрирования КС;
– протоколирование и аудит процессов безопасности.

Технологии надежной разработки программного обеспечения включают общие подходы к снижению ошибок при разработке программного кода и ряд более специфических аспектов, основанных на изначальном учете в концепции и структуре ядра системы (подсистема представления данных и подсистема доступа к данным) той или иной модели и технологий безопасности данных. Как показывает анализ, выявленных уязвимостей в системах безопасности компьютерных систем, вероятность наличия и нахождения злоумышленниками брешей существенно выше в тех случаях, когда системы защиты реализуются в виде надстройки или внешней оболочки над ядром и интерфейсом исходных незащищенных систем.

Технологии надежного проектирования и создания на базе программного обеспечения СУБД конкретных АИС направлены на предотвращение логических ошибок в

информационной инфраструктуре систем и в подсистемах разграничения доступа, строящихся на основе поддерживаемой СУБД модели и технологий безопасности данных. В этом отношении основным и широко распространенным является структурно-функциональный подход.

При наличии большого количества пользователей (субъектов) и объектов информационных систем (баз данных) схема разграничения доступа может быть очень сложной и запутанной, что создает трудности для администрирования и порождает предпосылки для логических ошибок. Для преодоления этой угрозы в рамках структурно-функционального подхода применяют технику рабочих групп.

Рабочая группа объединяет пользователей, имеющих какое- либо общее технологическое отношение к базе данных (выполняющих похожие операции) и близкие параметры конфиденциальности по отношению к общим данным.

Администратор системы может создавать рабочие группы, рассматривая их как коллективных пользователей, с определенной идентификацией и набором полномочий. Каждый пользователь обязательно должен являться членом какой-либо рабочей группы. Полномочия, определенные для рабочей группы, автоматически распространяются на всех пользователей — членов группы, что является отражением некоторых элементов зонально функционального принципа разграничения доступа. Дополнительно для каждого пользователя в его личной учетной записи могут быть уточнены и конкретизированы его полномочия.

Такой подход позволяет в большинстве случаев существенно уменьшить количество субъектов доступа в системе, сделать схему разграничения доступа более простой, «прозрачной» и управляемой, и тем самым снизить вероятность таких логических ошибок как неправильное предоставление доступа конкретного пользователя к конкретному объекту, превышение полномочий конкретного пользователя по доступу к ряду объектов, предоставление избыточных прав доступа и т.п.

Процессы в базе данных в технологиях рабочих групп помечаются как меткой пользователя, так и меткой рабочей группы, и, соответственно ядро безопасности СУБД проверяет подлинность обеих меток.

Проектирование системы доступа на основе технологии рабочих групп может проводиться «сверху» (дедуктивно) и «снизу» (индуктивно).

В первом способе сначала на основе анализа функциональной структуры и организационной иерархии пользователей (субъектов) формируются рабочие группы и осуществляются групповые назначения доступа. Далее каждый пользователь при его регистрации системе включается в состав одной или нескольких групп, отвечающих его функциям. И, наконец, в заключение для каждого пользователя анализируются особенности его функциональных потребностей и доверительных характеристик и при необходимости осуществляются индивидуальные дополнительные назначения доступа. Формирование групп, групповые и индивидуальные установки доступа при этом осуществляются администратором системы, что соответствует принудительному способу управления доступом.

Такой подход позволяет снизить вероятность ошибочных назначений доступа и обеспечивает жесткую централизованную управляемость системой доступа, но может порождать, в свою очередь, дублирование групповых и индивидуальных полномочий доступа субъектов к объектам (проблема дублирования), а также избыточность доступа

субъекта к одним и тем же объектам через участие в разных группах (проблема пересечения групп или в более широком смысле проблема оптимизации групп).

При втором (индуктивном) способе проектирования рабочих групп первоначально осуществляются индивидуальные назначения доступа субъектов (пользователей) к объектам. Назначения производится на основе опроса и анализа функциональных потребностей и доверительных характеристик пользователей, и могут осуществляться администратором системы (принудительный способ управления доступом) или через индивидуальные запрашивания субъектами доступа владельцев объектов (принцип добровольного управления доступом). Далее, уже администратором системы, производится анализ общих или схожих установок доступа у различных субъектов, на основе которого они объединяются в рабочие группы. Выделенные общие установки доступа используются в качестве групповых назначений доступа. При этом анализ схожести доступа при большом количестве субъектов и объектов представляет непростую задачу и решается администратором системы в значительной степени эвристически.

Дополнительным организационным способом повышения надежности и безопасности в процессе администрирования и сопровождения системы является разделение общего администрирования и администрирования безопасности. Общий администратор строит поддерживает и управляет информационной инфраструктурой системы информационно логическая схема, категорирование конфиденциальности объектов (ресурсов и устройств), интерфейсные и диалоговые элементы, формы, библиотеки запросов, словарно-классификационная база, резервирование и архивирование данных. Администратор безопасности организует и управляет системой разграничения доступа — доверительные характеристики (допуска) пользователей, конкретные назначения доступа, регистрация и формирование меток доступа пользователей.

Доступ к массиву учетных записей пользователей имеет только администратор безопасности. Совмещение функций общего администрирования и администрирования безопасности одновременно одним пользователем не допускается, что объективно повышает надежность системы.

Протоколирование и аудит событий безопасности являются важным средством обеспечения управляемости состоянием и процессами безопасности, создают условия для расследования фактов нарушения информационной безопасности, анализа и исключения их причин, снижения отрицательных последствий и ущерба от них.

Документированию подлежа! все события, критичные с точки зрения безопасности в системе:

- вход/выход пользователей;
- регистрация новых пользователей, смена привилегий и назначений доступа (все обращения к массивам учетных записей);
- все операции с файлами (создание, удаление, переименование, копирование, открытие, закрытие);
- обращения к/из удаленной системе(ы).
- При этом по каждому такому событию устанавливается минимально необходимый перечень регистрируемых параметров, среди которых:
- дата и время события;
- идентификатор пользователя-инициатора;
- тип события;

– источник запроса (для распределенных систем — сетевое имя терминала, рабочей станции и т. п.);
– имена затронутых объектов;
– изменения, внесенные в учеты, в системы, в том числе, в массивы учетных записей;
– метки доступа субъектов и объектов.

В СУБД такой подход хорошо вписывается в событийно-процедурную технологию с использованием техники журнализации. При этом, доступ к журналу событий имеет только администратор безопасности, который при обнаружении фактов или признаков нарушений безопасности имеет возможность восстановления хода событий, анализа и устранения источников и причин нарушения безопасности системы.

В этом отношении журнал; событий безопасности является необходимым средством аудита безопасности. Аудит безопасности заключается в контроле и отслеживании событий в системе с целью выявления, своевременного обнаружения проблем или нарушений безопасности и сигнализации об этом администратору безопасности. Ввиду того, что процессы доступа, различных процедур, операций, информационных потоков в компьютерных системах являются многоаспектными, не строго детерминированными, т. е. частично или полностью стохастическими, разработка аналитических, алгоритмических или иным образом аналитических, автоматизированных процедур обнаружения фактов и признаков нарушений информационной безопасности является чрезвычайно сложной и неопределенной задачей. Поэтому в настоящее время разрабатывается ряд эвристических и нейросетевых технологий, которые в некоторых случаях с успехом воплощаются в специальном программном инструментарии администратора безопасности, обеспечивая автоматизированный аудит безопасности системы.

В простейшем случае журнализация изменений заключается в последовательной записи во внешнюю память всех изменений, выполняемых в базе данных. Записывается следующая информация:
– порядковый номер, тип и время изменения;
– идентификатор транзакции;
– объект, подвергшийся изменению (номер хранимого файла и номер блока данных в нём, номер строки внутри блока);
– предыдущее состояние объекта и новое состояние объекта.

Формируемая таким образом информация называется журналом изменений базы данных. Журнал содержит отметки качала и завершения транзакции, и отметки принятия контрольной точка.

В СУБД с отложенной записью блоки данных внешней памяти снабжаются отметкой порядкового номера последнего изменения, которое было выполнено над этим блоком данных. В случае сбоя системы эта отметка позволяет узнать какая версия блока данных успела достичь внешней памяти.

СУБД с отложенной записью периодически выполняет контрольные точки. Во время выполнения этого процесса все незаписанные данные переносятся на внешнюю память, а в журнал пишется отметка принятия контрольной точки. После этого содержимое журнала, записанное до контрольной точки может быть удалено.

Журнал изменений может не записываться непосредственно во внешнюю память, а аккумулироваться в оперативной. В случае подтверждения транзакции СУБД дожидается записи оставшейся части журнала на внешнюю память. Таким образом, гарантируется, что

все данные, внесённые после сигнала подтверждения, *будут* перенесены во внешнюю память, не дожидаясь переписи всех измененных блоков из дискового кэша. СУБД дожидается записи оставшейся части журнала так же при выполнении контрольной точки.

В случае логического отказа или сигнала отката одной транзакции журнал сканируется в обратном направлении, и все записи отменяемой транзакции извлекаются из журнала вплоть до отметки начала транзакции. Согласно извлеченной информации выполняются действия, отменяющие действия транзакции, а в журнал записываются компенсирующие записи. Этот процесс называется откат (rollback).

В случае физического отказа, если ни журнал, ни сама база данных не повреждена, то выполняется процесс прогонки (rollforward). Журнал сканируется в прямом направлении, начиная от предыдущей контрольной точки. Все записи извлекаются из журнала вплоть до конца журнала. Извлеченная из журнала информация вносится в блоки данных внешней памяти, у которых отметка номера изменений меньше, чем записанная в журнале. Если в процессе прогонки снова возникает сбой, то сканирование журнала вновь начнется сначала, но фактически восстановление продолжиться с той точки, откуда оно прервалось.

Для увеличения отказоустойчивости СУБД может записывать одновременно несколько идентичных копий журнала изменений. Если в случае отказа одна из копий журнала окажется недоступной, СУБД восстановит базу данных, используя любую из доступных копий. Такая стратегия называется мультиплексированием журнала изменений.

Контрольные вопросы:

1. Технологии обеспечения безопасности повторного использования объектов.
2. Технологии надежного проектирования и администрирования.
3. Объясните технологии журнализации.
4. Какая информация пишется в журналы изменений?
5. Что такое мультиплексирование журналов изменений?

Лекции 8: Технологии идентификации и аутентификации

План:

1. Технологические аспекты защиты информации
2. Надежное проектирование и администрирование

Технологии идентификации и аутентификации являются обязательными элементоми защищенных систем, так как обеспечивают аксиоматический принцип персонализации субъектов и, тем самым, реализуют первый (исходный) программно-технический рубеж защиты информации в компьютерных системах.

Под идентификацией понимается различение субъектов, объектов, процессов по их образам, выражаемым именами.

Под аутентификацией понимается проверка и подтверждение подлинности образа идентифицированного субъекта, объекта, процесса.

В системотехническом плане структуру систем идентификации/аутентификации можно проиллюстрировать схемой, приведенной на рис. 1.7.

При регистрации объекта идентификации/аутентификации в системе монитором безопасности формируется его образ, информация по которому подвергается необратимому

без знания алгоритма и шифраключа, т. е. криптографическому преобразованию и сохраняется в виде ресурса, доступного в системе исключительно монитору безопасности. Таким образом, формируется информационный массив внутренних образов, объектов идентификации/аутентификации.

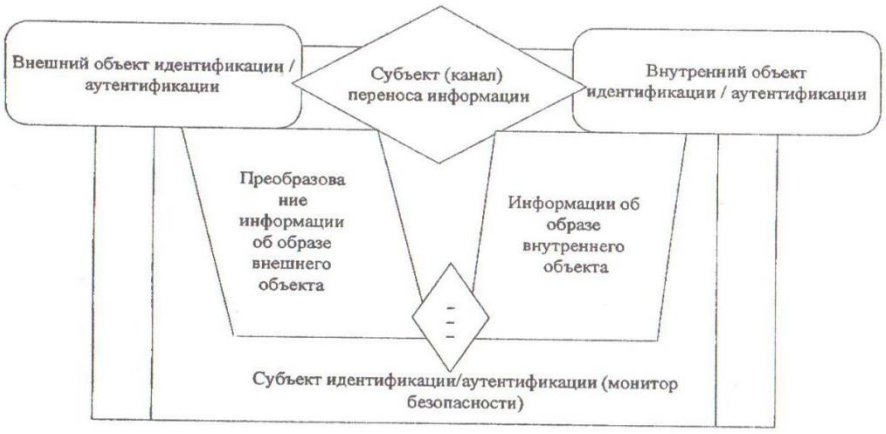


Рис.1.7. Системотехнический аспект идентификации аутентификации.

Впоследствии, при идентификации/аутентификации (очередной вход в систему пользователя, запрос процесса на доступ к объекту, проверка подлинности объекта системы, при выполнении над ним действий и т. д.) объект через канал переноса информации передает монитору безопасности информацию о своем образе, которая подвергается соответствующему преобразованию. Результат этого преобразования сравнивается с соответствующим зарегистрированным внутренним образом, и при их совпадении принимается решение о распознавании (идентификации) и подлинности (аутентификации) объекта.

Информационный массив внутренних образов объектов идентификации/аутентификации является критическим ресурсом системы, несанкционированный доступ к которому дискредитирует всю систему безопасности. Поэтому, помимо всевозможных мер по исключению угроз несанкционированного доступа к нему, сама информация о внутренних образах объектов идентификации/аутентификации находится в зашифрованном виде.

В общем плане для идентификации/аутентификации пользователей субъектов в компьютерных системах могут использоваться их биометрические параметры (отпечатки пальцев, рисунок радужной оболочки глаз, голос, почерк и т. д.), либо специальные замковоключевые устройства (смарткарты, магнитные карты и т. п.). Однако, при доступе непосредственно в КС (в базы данных), чаще всего используются парольные системы идентификации/аутентификации.

Парольные системы основаны на предъявлении пользователем в момент аутентификации специального секретного (известного только подлинному пользователю) слова или набора символов - пароля. Пароль вводится пользователем с клавиатуры, подвергается криптопреобразованию и сравнивается со своей зашифрованной соответствующим образом учетной копией в системе. При совпадении внешнего и внутреннего парольного аутентификатора осуществляется распознавание и подтверждение подлинности соответствующего субъекта.

Парольные системы являются простыми, но при условии правильной организации подбора и использования паролей, в частности, безусловного сохранения пользователями своих паролей втайне, достаточно надежным средством аутентификации, и, в силу данного обстоятельства, широко распространены.

Основной недостаток систем парольной аутентификации заключается в принципиальной оторванности, отделимости аутентификатора от субъектаносителя. В результате пароль может быть получен тем или иным способом от законного пользователя или просто подобран, подсмотрен по набору на клавиатуре, перехвачен тем или иным способом в канале ввода в систему и предъявлен системе злоумышленником.

Поэтому, в некоторых случаях парольные аутентификаторы могут усиливаться диалоговововопросными системами «коллективного вхождения». В системах коллективного вхождения парольную аутентификацию должны одновременно пройти сразу все зарегистрированные для работы в системе пользователи. Иначе говоря, поодиночке пользователи работать в системе не могут. Вероятность подбора, перехвата и т. д. злоумышленником (злоумышленниками) сразу всех паролей, как правило, существенно меньше, и, тем самым, надежность подобных систем аутентификации выше.

Аутентификации, в распределенных информационных системах, в принципе должны подвергаться и объекты (ресурсы, устройства), а также процессы (запросы, пакеты и т. д.). Аутентифицированный (подлинный) пользователь, обращаясь к объектам системы и порождая соответствующие процессы, должен, в свою очередь, убедиться в их подлинности, например, отправляя распечатать сформированный в базе данных конфиденциальный отчет на сетевой принтер, специально предназначенный для распечатки соответствующих конфиденциальных документов.

Для аутентификации процессов широкое распространение нашли технологии меток (дескрипторов) доступа.

Технология меток или дескрипторов доступа отражает сочетание одноуровневой и многоуровневой моделей безопасности данных и основывается на присвоении администратором системы всем объектам и субъектам базы данных специальных дескрипторов доступа, содержащих набор параметров уровня конфиденциальности, допустимых операций, допустимых имен объектов или субъектов доступа и других особых условий доступа. Субъект доступа, инициируя в соответствии со своим дескриптором (меткой) разрешенный процесс, передает ему свою метку доступа (помечает своей меткой). Ядро безопасности СУБД проверяет подлинность метки процесса, сравнивая ее с меткой доступа пользователя-субъекта, от имени которого выступает процесс. При положительном результате метка доступа процесса сравнивается с меткой доступа объекта, операцию с которым намеревается осуществлять процесс. Если дескрипторы доступа процесса и объекта совпадают, монитор безопасности разрешает соответствующий доступ, т. е. разрешает осуществление процесса (операции).

Проверка подлинности метки процесса предотвращает возможные угрозы нарушения безопасности данных путем формирования субъектом для инициируемого им процесса такой метки, которая не соответствует его полномочиям.

Для проверки подлинности меток в системе формируется специальный файл (массив) учетных записей. При регистрации нового пользователя в системе для него создается учетная запись, содержащая его идентификационный номер (идентификатор), парольный аутентификатор и набор дескрипторов доступа к объектам базы данных (метка доступа). При инициировании пользователем (субъектом) какоголибо процесса в базе данных и

передаче ему своей метки доступа ядро безопасности СУБД подвергает метку процесса криптопреобразованию, сравнивает ее с зашифрованной меткой соответствующего субъекта (пользователя) в массиве учетных записей и выносит решение о подлинности метки.

Массив учетных записей, в свою очередь, является объектом высшей степени конфиденциальности в системе, и доступен только администратору. Ввиду исключительной важности массива учетных записей для безопасности всей системы, помимо шифрования его содержимого, принимается ряд дополнительных мер к его защите, в том числе специальный режим его размещения, проверка его целостности. Таким образом, на сегодняшний день наработан и используется развитый набор технологий идентификации/аутентификации в защищенных компьютерных системах. Вместе с тем, основные бреши безопасности чаще всего находятся злоумышленниками именно на этом пути.

Языки безопасности базы данных

Для определения конкретных назначений или установления правил и ограничений доступа при проектировании банков данных КС, а также в целях управления системой разграничения доступа и в более широком смысле системой коллективной обработки данных администратору системы необходим специальный инструментарий. Такой инструментарий должен основываться на определенном языке, позволяющем описывать и устанавливать те или иные назначения доступа и другие необходимые установки политики безопасности в конкретной КС.

Как следует из рассмотрения внутренней схемы баз данных, одной из основных функций систем управления базами данных является создание и поддержание собственной системы размещения и обмена данными между внешней (дисковой) и оперативной памятью. От эффективности реализации в каждой конкретной СУБД данной функции (формат файлов данных, индексирование, хэширование и буферизация) во многом зависит и эффективность функционирования СУБД в целом. Поэтому основные усилия создателей первых СУБД в конце 60-х - начале 70-х годов были сосредоточены именно в этом направлении. В результате для реализации любой функции по вводу, обработке или выводу данных требовались квалифицированные программисты для написания специальных программ на алгоритмических языках высокого уровня (в 70-х годах ФОРТРАН, КОБОЛ и др.), «знающих» особенности структуры и способы размещения данных во внешней и оперативной памяти. В итоге работа с базами данных осуществлялась через посредника в виде квалифицированного программиста, «переводящего» информационные потребности пользователя в машинный код, что схематично иллюстрируется на рис. 1.8.

Такое положение дел приводило к большим накладным расходам при создании и эксплуатации автоматизированных информационных систем и в определенной степени сдерживало распространение вычислительной техники в процессах информационного обеспечения деятельности предприятий и организаций.

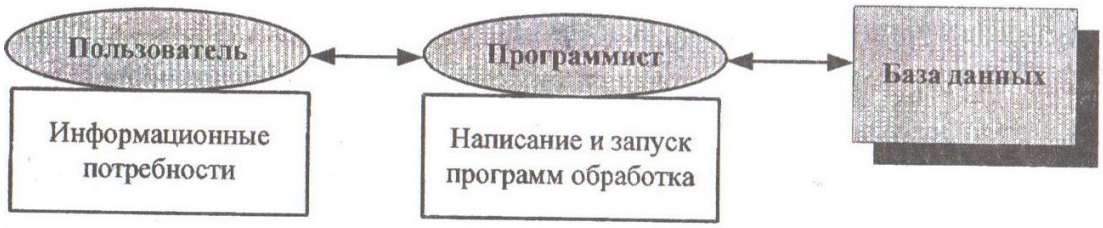


Рис. 1.8. Схема взаимодействия пользователя с базой данных в ранних СУБД.

Основателем теории реляционных СУБД Е. Коддом было выдвинуто предложение о создании специального языка для общения (взаимодействия) пользователя-непрограммиста с базами данных.

Идея такого языка сводилась к набору из нескольких фраз-прими- тивов английского языка («выбрать», «обновить», «вставить», «удалить»), через которые пользователь-непрограммист ставил бы «вопросы» к СУБД по своим информационным потребностям. В этом случае дополнительной функцией СУБД должна быть интерпретация этих «вопросов» на низкоуровневый язык машинных кодов для непосредственной обработки данных и предоставление результатов пользователю. Так родилась уже упоминавшаяся по структуре СУБД «машина данных». Иначе говоря, машина данных «понимает» язык базы данных и в результате разделяет собственно данные и задачи по их обработке. В таком подходе взаимодействие пользователя с базой данных можно проиллюстрировать схемой, приведенной на рис. 1.9.

Рис. 1.9. Схема взаимодействия пользователя с базой данных через язык баз данных.

В практику эти идеи шшрвые претворились в ходе реализации проекта System R (1975-1979 гг.) с участием еще одного известного специалиста по базам данных Криса Дейта. В ходе проекта System R был создан язык SEQUEL, трансформировавшийся впоследствии в язык структурированных запросов SQL (Structured Query Language). При этом дополнительно к возможностям формирования «вопросов» к базе данных пользователю также решено было предоставить и возможность описания самой структуры данных, ввода данных и их изменения. Примерно в то же время в компании IBM был создан еще один реляционный язык-QBE (Query-By-Example), т. е. язык запросов по образцу, применявшийся впоследствии во многих коммерческих системах обработки табличных данных и послуживший идеологической основой для создания визуальных «конструкторов» запросов в современных СУБД.

Быстрое и массовое распространение языка SQL в реляционных СУБД к середине 80-х годов привело фактически к принятию его в качестве стандарта по организации и обработке данных. В 1986 г. Американским национальным институтом стандартов (ANSI) и Международной организацией по стандартизации (ISO) язык был признан стандартным языком описания и обработки данных в реляционных СУБД. В 1989 г. ANSI/ISO была принята усовершенствованная версия SQL — SQL2, а в 1992 г. третья версия — SQL3.

Язык SQL относится к так называемым декларативным (непроцедурным) языкам программирования. В отличие от процедурных языков (С, Паскаль, Фортран, Кобол, Бейсик) на лем формулируются предложения (инструкции) о том, «что сделать», но не «как сделать, как получить». Машина данных в СУБД исполняет роль интерпретатора и как раз строит машинный код, реализующий способ получения результата, задаваемого SQL-инструкциями.

Язык SQL состоит из двух частей:

- языка описания (определения) данных — DDL (Data Definition Language);
- языка манипулирования данными — DML (Data Manipulation Language).
- Синтаксис SQL-инструкций включает:
- название инструкции (команду);
- предложения, определяющие источники, условия операции;
- предикаты, определяющие способы и режимы отбора з сей, задаваемых предложениями;
- выражения, значения которых задают свойства и парам*. > ? выполнения инструкции и предложения.

Структуру SQL-инструкций можно разделить на две основные части, схематично представленные на рис. 1.10.

Рис. 1.10. Структура SQL-инструкций.

Первая часть включает название (команду) SQL-инструкции, предикат (необязательный элемент) и аргументы инструкции, которыми являются перечисляемые через запятую имена полей одной *mn* нескольких таблиц.

Вторая часть состоит из одного или нескольких предложений, ументы которых могут задавать источники данных (имена 5лиц, операции над таблицами), способы, условия и режимы полнения команды (предикаты сравнения, логические и матема- (еские выражения по значениям полей таблиц). Перечень SQL- ш струкций разделяется но частям языка SQL.

В состав языка DDL входят несколько базовых инструкций, обеспечивающих основной набор функций при создании реляционных таблиц и связей между ними.

- CREA.TETABLE... - создать таблицу;
- CREATEJNDEX... - создать индекс;
- ALTERTABLE... - изменить структуру ранее созданной таблицы;

– DROP... - удалить существующую таблицу и базы данных.

В структуре инструкций CREATETABLE и ALTERTABLE мажаую роль играет предложение CONSTRAINT (создать ограничения на значения данных) со следующими установками - NOT NULL (не допускаются нулевые значения по соответствующему полю), AUT01NC (поле с инкрементальным, т. е. последовательно возрастающим с каждой новой записью, характером значений) и PRIMARY KEY (определение для поля уникального, т. е. без повторов индекса, что в результате задает режим заполнения данного поля с уникальными неповторяющимися по различным строкам значениями).

В состав языка DML также входят несколько базовых инструкций, охватывающих тем не менее основные операции по вводу, обработке и выводу данных.

– SELECT... - выбрать данные из базы данных;
– INSERT... - добавить данные в базу данных;
– UPDATE... - обновить данные в базе данных;
– DELETE... - удалить данные;
– G RANT... - предоставить привилеги и пользователю;
– REVOKE... -отменить привилегии пользователю;
– COMMIT... - зафиксировать текущую транзакцию;
– ROLLBACK... - прервать текущую транзакцию.

Важное значение имеют разновидности инструкции SELECT SELECT... INTO ... (выбрать из одной или нескольких таблиц набор записей, из которого создать новую **таблицу)** и UNION SELECT, которая в дополнении с исходной инструкцией SELECT (SELECT... UNION SELECT...) реализует операцию объединения таблиц.

Помимо предложения **CONSTRAINT в** SQL-инструкциях используются следующие предложения

– FROM... указывает таблицы или запросы, которые содержат поля, перечисленные в инструкции SELECT;

– WHERE... определяет, какие записи из таблиц, перечисленных в предложении FROM, следует включить в результат выполнения инструкции SELECT, UPDATE или DELETE;

– GROUP BY... — объединяет записи с одинаковыми значениями в указанном списке полей в одну запись;

– HAVING... — определяет, какие сгруппированные записи отображаются при использовании инструкции SELECT с предложением GROUP BY;

– IN... — определяет таблицы в любой внешней базе данных, с которой ядро СУБД может установить связь;

– ORDERBY... — сортирует записи, полученные в результате запроса, в порядке возрастания или убывания на основе значений указанного поля или полей.

В качестве источника данных по предложению FROM, помимо таблиц и запросов, могут использоваться также результаты операций соединения таблиц в трех разновидностях—INNER JOIN... ON..., LEFT JOIN. ..ON... и RIGHT JOIN...ON... (внутреннее соеды- нение, левое и правое внешнее соединение, соответственно).

Предикаты используются для задания способов и режимов использования записей, отбираемых на основе условий в инструкции SQL. Такими предикатами являются:

– ALL... — отбирает все записи, соответствующие условиям, заданным в инструкции SQL;

– DISTINCT... — исключает записи, которые содержат повторяющиеся значения в выбранных полях;

– DISTINCTROW... — опускает данные, основанные на целиком повторяющихся записях;

– TOP... — возвращает записи, находящихся в начале или в конце диапазона, описанного с помощью предложения ORDER BY;

Выражениями в инструкциях SQL являются любые комбинации операторов, констант, значений текстовых констант, функций, имен полей, построенные по правилам математических выражений, и результатом которых является конкретное, в том числе и логическое значение.

Базовых инструкций языка SQL представлены инструкции GRANT и REVOKE, предоставляющие или отменяющие привилегии пользователям. Структура инструкции GRANT выглядит следующим образом:

С11АМТсписок_привилегий_через_запятую ON ИмяОбъекта

ТОИменаПользователей_через_запятую

[WITHGRANTOPTION];

где:

– список привилегий составляют разрешенные инструкции (операции) над объектом (таблицей) - SELECT, INSERT UPDATE DELETE;

– список пользователей представляется их именамииденти- фикаторами или может быть заменен ключевым словом PUBLIC, которое идентифицирует всех пользователей, зарегистрированных в системе;

– директива WITH GRANT OPTION наделяет перечисленных пользователей дополнительными особыми полномочиями по предоставлению указанных в списке привилегий-полномочий другим пользователям.

В большинстве случаев право подачи команд GRAND и REVOKE по конкретному объекту автоматически имеют пользователи, создавшие данный объект, т. е. их владельцы. В других подходах этим правом наделяются доверенные субъекты, т. е. администраторы.

Хотя в явном виде такой подход не предусматривает создание матрицы доступа, тем не менее, реализуется классический принцип дискреционного разграничения доступа с сочетанием как добровольного, так и принудительного управления доступом.

На самом деле в большинстве СУБД привилегии и установки доступа, как и структура базы данных, «прописываются» в системных таблицах БД, т.е. в системном каталоге БД, который можно рассматривать, в том числе, и в качестве матрицы доступа.

Как уже отмечалось, дискреционный принцип обладает большой гибкостью по настройке системы разграничения доступа на особенности предметной области базы данных и потребности пользователей, но не обеспечивает эффективной управляемости и затрудняет проведение какой - либо целенаправленной политики безопасности в системе. Преодоление этого недостатка достигается двумя путями: использованием техники «представлений» и специальными расширениями языка SQL.

«Представлением» называется глобальный авторизованный запрос на выборку данных, формирующий для пользователя «свое» представление определенного объекта (объектов), совокупность которых формирует некую виртуальную базу данных, со своей схемой (объектами) и данными (отобранными или специально преобразованными). При входе пользователя в систему в процессе его идентификации и аутентификации ядро

безопасности отыскивает для пользователя соответствующие представлениязапросы и передает запрос основному ядру СУБД для выполнения. В результате выполнения запроса пользователь «видит» и имеет доступ только к тем объектам, которые соответствуют его полномочиям и функциям.

В целом создание системы разграничения доступа через технику представлений является более простым способом, чем непосредственное использование инструкций GRANT, и осуществляется в два этапа:

– Для всех зарегистрированных пользователей в системе с помощью конструкций CREATE VIEW создаются свои представления объектов базы данных.

– С помощью инструкций «GRANT SELECT ON-ИмяПредс- тавления ТО - ИмяПользователя» созданные представления авторизуются со своими пользователями.

Вместе с тем такой подход является более грубым по сравнению с применением инструкции GRANT непосредственно к объектам базы данных, т. к. не обеспечивает расщепления установок доступа к объектам на уровне отдельных операций (SELECT, INSERT, UPDATE, DELETE).

Поэтому другим подходом являются специальные расширения языка SQL, основанные на событийно-процедурной идеологии с введением специальных правил (RULE) безопасности:

– CREATESECURITYRULEИмяПравила
– 011АШ’список_привилегий_через_запятую ON ИмяОбъекта
– WHERE условия
– ТО Имена Пользователей_через_залятую

Введение правил безопасности обеспечивает более широкие и гибкие возможности реализации различных политик безопасности с большей степенью контроля и управляемости, но, как, впрочем, и техника представлений и непосредственное использование инструкций GRANT, не позволяет строить системы с мандатным принципом разграничения доступа.

Для решения этой задачи могут предлагаться более кардинальные расширения языка SQL с введением возможностей создания объектов базы данных с метками конфиденциальности. Следует, однако, заметить, что подобные примеры в коммерческих и сертифицированных по требованиям безопасности СУБД чрезвычайно редки.

По языкам безопасности баз отметим, что в современных СУБД для реализации установок, правил и ограничений доступа разрабатывается и используется специальный диалоговонаглядный интерфейс, автоматически формирующий соответствующие конструкции языка SQL и позволяющий в большинстве случаев обходиться без непосредственного программирования.

Технологии обеспечения безопасности повторного использования объектов

Компьютерная система в целом, устройства оперативной и внешней (дисковой) памяти в частности, являются классическим примером среды многократного повторного информационного использования. Технологии обеспечения безопасности повторного использования объектов направлены на предотвращение угроз безопасности от случайного или преднамеренного извлечения интересующей злоумышленника информации по следам предшествующей деятельности или из технологического «мусора».

Часть этих технологий реализуются на уровне операционных систем, а часть являются специфическими функциями, осуществляемыми в автоматизированных информационных системах СУБД. Данные технологии условно можно разделить на три группы:

- изоляция процессов;
- очистка памяти после завершения процессов;
- перекрытие косвенных каналов утечки информации.

Изоляция процессов является стандартным принципом и приемом обеспечения надежности многопользовательских (многопроцессных) систем и предусматривает выделение каждому процессу своих непересекающихся с другими вычислительных ресурсов, прежде всего областей оперативной памяти. В СУБД данные задачи решаются мониторами транзакций.

Очистка памяти после завершения процессов направлена непосредственно на предотвращение несанкционированного доступа к конфиденциальной информации после завершения работы процессов с конфиденциальными данными уполномоченными пользователями. Так же как и изоляция процессов, чаще всего данная функция выполняется операционными системами. Кроме того, следует отметить, что очистке подлежат не только собственно участки оперативной памяти, где во время выполнения процессов размещались конфиденциальные данные, но и участки дисковой памяти, используемые в системах виртуальной памяти.

Как уже отмечалось, при реализации систем разграничения доступа возможны косвенные каналы утечки информации. Источниками косвенных каналов утечки информации являются еще и ряд технологических аспектов, связанных с характеристиками процессов обработки данных. В этом плане различают косвенные каналы - «временные» и «по памяти».

В первом случае, некоторые элементы конфиденциальной информации неуполномоченный пользователь получает на основе анализа времени выполнения отдельных процессов уполномоченными пользователями (скажем, по неправдоподобно большому времени для очевидно простой или какой-либо типовой операции).

Во втором случае, соответственно, «подозрение» вызывает занимаемый объем некоторых объектов (файлов, таблиц и т. п.), объем содержимого которых, с точки зрения того, что «видит» пользователь, явно не соответствует их фактическому объему.

Иначе говоря, все операции обработки данных должны выполняться строго над той выборкой данных, которая соответствует «представлению» пользователя.

В наиболее критичных, с точки зрения безопасности информации, случаях применяют еще один, наиболее жесткий вариант - технологию разрешенных процедур. В этом случае пользователям системы разрешается работать с базой данных исключительно через запуск разрешенных процедур.

Данный подход основывается также на уже рассмотренной технике хранимых (stored) процедур. Администратором системы для каждого пользователя формируется набор процедур обработки данных в соответствии с полномочиями и функциями пользователя. Для безопасности хранимые процедуры в файле данных шифруются.

Ядро СУБД после идентификации и аутентификации пользователя при его входе в систему предъявляет ему разрешенный набор процедур. Непосредственного доступа к самим данным пользователь не имеет и работает только с результатами их обработки по соответствующим процедурам.

Данные КС размещаются в файлах данных, структура которых может быть известна нарушителю. Поэтому еще одной угрозой безопасности данных КС является возможность несанкционированного доступа к файлам данных вне программного обеспечения КС (СУБД) средствами операционной системы или дисковых редакторов. Для нейтрализации этой угрозы применяются методы и средства криптозащиты. В большинстве случаев криптографические средства защиты встраиваются непосредственно в программное обеспечение СУБД. Файл (файлы) базы данных при размещении на устройствах дисковой памяти шифруется. Соответственно, криптографическая подсистема при открытии файла данных его дешифрует. Специфической особенностью СУБД является особый порядок работы с файлами базы данных через организацию специальной буферизации страниц в оперативной памяти.

В развитых СУБД имеется возможность выборочного шифрования объектов базы данных исходя из уровня их конфиденциальности, вплоть до шифрования отдельных полей записей.

Контрольные вопросы:

1. Какие программно-технологические задачи решаются при обеспечении безопасности базы данных?

2. Место идентификации/аутентификации в защищенной системе.

3. Какие средства идентификации/аутентификации вы знаете?

4. Какой вид аутентификации, в основном, используется в системах управления базами данных?

5. Перечислите основные недостатки парольной аутентификации**.**

Лекция 9: Концепция информационной безопасности в распределенных системах базы данных

План:

1. Понятие распределенных информационных систем, принципы их создания и функционирования

2. Технологии и модели «клиент-сервер»

Системы распределенных вычислений появляются, прежде всего, по той причине, что в крупных автоматизированных информационных системах, построенных на основе корпоративных сетей, не всегда удается организовать централизованное размещение всех базы данных и СУБД на одном узле сети. Поэтому системы распределенных вычислений тесно связаны с системами управления распределенны баз данных.

Распределенная база данных - это совокупность логически взаимосвязанных баз данных, распределенных в компьютерной сети.

Система управления распределенной базой данных - это программная система, которая обеспечивает управление распределенной базой данных и прозрачность ее распределенности для пользователей.

Распределенная база данных может объединять базы данных, поддерживающие любые модели (иерархические, сетевые, реляционные и объектно-ориентированные базы данных) в рамках единой глобальной схемы. Подобная конфигурация должна обеспечивать для всех приложений прозрачный доступ к любым данным независимо от их местоположения и формата.

Основные принципы создания и функционирования распределенных баз данных:

– прозрачность расположения данных для пользователя (иначе говоря, для пользователя распределенная база данных должна представляться и выглядеть точно так же, как и нераспределенная);

– изолированность пользователей друг от друга (пользователь должен «не чувствовать», «не видеть» работу других пользователей в тот момент, когда он изменяет, обновляет, удаляет данные);

– синхронизация и согласованность (непротиворечивость) состояния данных в любой момент времени.

Из основных вытекает ряд дополнительных принципов:

– локальная автономия (ни одна вычислительная установка для своего успешного функционирования не должна зависеть от любой другой установки);

– отсутствие центральной установки (следствие предыдущею пункта);

– независимость от местоположения (пользователю все равно, где физически находятся данные, он работает так, как будто они находятся на его локальной установке);

– непрерывность функционирования (отсутствие плановых отключений системы в целом, например, для подключения новой установки или обновления версии СУБД);

– независимость от фрагментации данных (как от горизонтальной фрагментации, когда различные группы записей одной таблицы размещены на различных установках или в различных локальных базах, так и от вертикальной фрагментации, когда различные поля-столбцы одной таблицы размещены на разных установках);

– независимость от реплицирования (дублирования) данных (когда какая-либо таблица базы данных (или ее часть) физически может быть представлена несколькими копиями, расположенными на различных установках);

– распределенная обработка запросов (оптимизация запросов должна носить распределенный характер - сначала глобальная оптимизация, а далее локальная оптимизация на каждой из задействованных установок);

– распределенное управление транзакциями (в распределенной системе отдельная транзакция может требовать выполнения действий на разных установках, транзакция считается завершенной, если она успешно завершена на всех вовлеченных установках);

– независимость от аппаратуры (желательно, чтобы система могла функционировать на установках, включающих компьютеры разных типов);

– независимость от типа операционной системы (система должна функционировать вне зависимости от возможного различия ОС на различных вычислительных установках);

– независимость от коммуникационной сети (возможность функционирования в разных коммуникационных средах);

– независимость от СУБД (на разных установках могут функционировать СУБД различного типа, на практике ограничиваемые кругом СУБД, поддерживающих SQL).

В обиходе СУБД, на основе которых создаются распределенные информационные системы, также характеризуют термином «распределенные СУБД», и, соответственно, используют термин «распределенные базы данных».

Практическая реализация распределенных вычислений осуществляется через отступление от некоторых рассмотренных выше принципов создания и функционирования распределенных систем. В зависимости от того, какой принцип приносится в «жертву» (отсутствие центральной установки, непрерывность функционирования, согласованного состояния данных и др.) выделились несколько самостоятельных направлений в технологиях распределенных систем - технологии «Клиент-сервер», технологии реплицирования, технологии объектного связывания.

Реальные распределенные информационные системы, как правило, построены на основе сочетания всех трех технологий, но в методическом плане их целесообразно рассмотреть отдельно.

В *технологиях «Клиент-сервер»* отступают от одного из главных принципов создания и функционирования распределенных систем - отсутствия центральной установки. Поэтому можно выделить две основные идеи, лежащие в основе “клиент-серверных” технологий:

– общие для всех пользователей данные на одном или нескольких серверах;

– много пользователей (клиентов), на различных вычислительных установках, совместно (параллельно и одновременно) обрабатывающих общие данные.

Иначе говоря, системы, основанные на технологиях «Клиент- сервер», распределены только в отношении пользователей, поэтому часто их не относят к «настоящим» распределенным системам а считают отдельным классом многопользовательских систем.

Важное значение в технологиях «Клиент-сервер» имеют понятия сервера и клиента.

Под сервером в широком смысле понимается любая система процесс компьютер, владеющие каким-либо вычислительным ресурсом (памятью, временем, производительностью процессора и т.д.)

Клиентом называется также любая система, процесс, компьютер, пользователь, запрашивающие у сервера какой-либо ресурс пользующиеся каким-либо ресурсом или обслуживаемые сервером иным способом.

В своем развитии системы «Клиент-сервер» прошли несколько этапов, в ходе которых сформировались различные модели систем «Клиент-сервер». Их реализация и, следовательно, правильное понимание основаны на разделении структуры СУБД на три компонента:

– компонент представления, реализующий функции ввода и отображения данных, называемый иногда еще просто как интерфейс пользователя;

– прикладной компонент, включающий набор запросов событии, правил, процедур и других вычислительных функций реализующий предназначение автоматизированной информационной системы в конкретной предметной области;

– компонент доступа к данным, реализующий функции хранения-извлечения, физического обновления и изменения даннных

Исходя из особенностей реализации и распределения в системе этих трех компонентов различают четыре модели технологий «Клиент-сервер»:

– модель файлового сервера (File Server - FS);
– модель удаленного доступа к данным (Remote Data Access - RDA);
– модель сервера базы данных (DataBase Server - DBS);
– модель сервера приложений (Application Server - AS).

Модель файлового сервера является наиболее простой и характеризует не столько способ образования информационной системы, сколько общий способ взаимодействия компьютеров в локальной сети. Один из компьютеров сети выделяется и определяется файловым сервером, т. е. общим хранилищем любых данных. Суть FS- модели иллюстрируется схемой, приведенной на рис. 3.1.

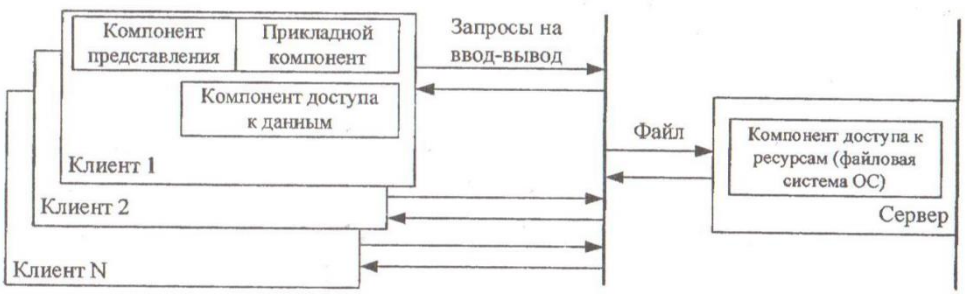


Рис. 3.1. Модель файлового сервера.

В FS-модели все основные компоненты размещаются на клиентской установке. При обращении к данным ядро СУБД, в свою очередь, обращается с запросами на ввод-вывод данных за сервисом к файловой системе. С помощью функций операционной системы в оперативную память клиентской установки полностью или частично на время сеанса работы копируется файл базы данных. Таким образом, сервер в данном случае выполняет чисто пассивную функцию.

Достоинством данной модели являются ее простота, отсутствие высоких требований к производительности сервера (главное, требуемый объем дискового пространства). Следует также отметить, что программные компоненты СУБД в данном случае не распределены, т.е. никакая часть СУБД на сервере не инсталлируется и не размещается.

Недостатки данной модели - высокий сетевой трафик, достигающий пиковых значений особенно в момент массового вхождения в систему пользователей, например в начале рабочего дня. Однако, более существенным недостатком, с точки зрения работы с общей базой данных, является отсутствие специальных механизмов безопасности файла (файлов) базы данных со стороны СУБД. Иначе говоря, разделение данных между пользователями (параллельная работа с одним файлом данных) осуществляется только средствами файловой системы ОС для одновременной работы нескольких прикладных программ с одним файлом.

Несмотря на очевидные недостатки, модель файлового сервера является естественным средством расширения возможностей персональных (настольных) СУБД в направлении поддержки многопользовательского режима и, очевидно, в этом плане еще будет сохранять свое значение.

Модель удаленного доступа к данным основана на учете специфики размещения и физического манипулирования данных во внешней памяти для реляционных СУБД. В RDA-модели компонент доступа к данным в СУБД полностью отделен от двух других компонентов (компонента представления и прикладного компонента) и размещается на сервере системы.

Компонент доступа к данным реализуется в виде самостоятельной программной части СУБД, называемой SQL-сервером, и инсталлируется на вычислительной установке сервера системы. Функции SQL-сервера ограничиваются низкоуровневыми операциями по организации, размещению, хранению и манипулированию данными в дисковой памяти сервера. Иначе говоря, SQL-сервер играет роль машины данных. Схема RDA-модели приведена на рис 3.2.

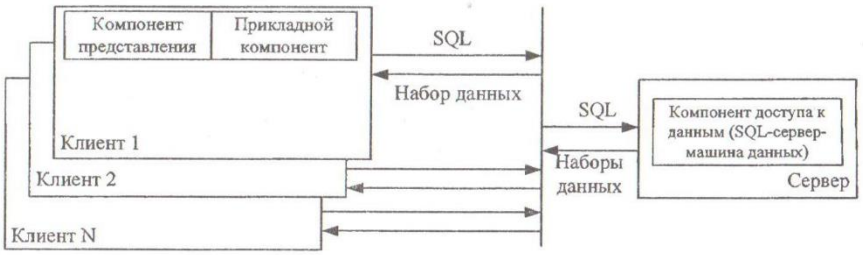


Рис 3.2. Модель удаленного доступа к данным (RDA-модель).

В файле (файлах) базы данных, размещаемом на сервере системы, находится также и системный каталог базы данных, в который помещаются в том числе и сведения о зарегистрированных клиентах, их полномочиях и т. п.

На клиентских установках инсталлируются программные части СУБД, реализующие интерфейсные и прикладные функции. Пользователь, входя в клиентскую часть системы, регистрируется через нее на сервере системы и начинает обработку данных.

Прикладной компонент системы (библиотеки запросов, процедуры обработки данных) полностью размещается и выполняется на клиентской установке. При реализации

своих функций прикладной компонент формирует необходимые SQL-инструкции, направляемые SQL-серверу. SQL-сервер, представляющий специальный программный компонент, ориентированный на интерпретацию SQL-инструкций и высокоскоростное выполнение низкоуровневых операций с данными, принимает и координирует SQL-инструкции от различных клиентов, выполняет их, проверяет и обеспечивает выполнение ограничений целостности данных и направляет клиентам результаты обработки SQL-инструкций, представляющие, как известно, наборы (таблицы) данных.

Таким образом, общение клиента с сервером происходит через SQL-инструкции, а с сервера на клиентские установки передаются только результаты обработки, т. е. наборы данных, которые могут быть существенно меньше по объему всей базы данных. В результате резко уменьшается загрузка сети, а сервер приобретает активную центральную функцию. Кроме того, ядро СУБД в виде SQL- сервера обеспечивает также традиционные и важные функции по обеспечению ограничений целостности и безопасности данных при совместной работе нескольких пользователей.

Другим, может быть неявным, достоинством RDA-модели является унификация интерфейса взаимодействия прикладных компонентов информационных систем с общими данными. Такое взаимодействие стандартизовано в рамках языка SQL специальным протоколом ODBC (Open Database Connectivity - открытый доступ к базам данных), играющим важную роль в обеспечении интероперабельности (многопротокольное™), т.е. независимости от типа СУБД на клиентских установках в распределенных системах.

Интероперабельность СУБД - способность СУБД обслуживать прикладные программы, первоначально ориентированные на разные типы СУБД. Иначе говоря, специальный компонент ядра СУБД на сервере (так называемый драйвер ODBC) способен воспринимать, обрабатывать запросы и направлять результаты их обработки на клиентские установки, функционирующие под управлением реляционных СУБД других, не «родных» типов.

Такая возможность существенно повышает гибкость в создании распределенных информационных систем на базе интеграции уже существующих в какой-либо организации локальных баз данных под управлением настольных или другого типа реляционных СУБД.

К недостаткам RDA-модели можно отнести высокие требования к клиентским вычислительным установкам, так как прикладные программы обработки данных, определяемые спецификой предметной области информационной системы, выполняются на них.

Другим недостатком является все же существенный трафик сети, обусловленный тем, что с сервера базы данных клиентам направляются наборы (таблицы) данных, которые в определенных случаях могут занимать достаточно существенный объем.

Модель сервера базы данных. Развитием PDA-модели стала модель сервера базы данных. Ее сердцевиной является механизм хранимых процедур. В отличие от PDA-модели, определенные для конкретной предметной области информационной системы события, правила и процедуры, описанные средствами языка SQL, хранятся вместе с данными на сервере системы и на нем же выполняются. Иначе говоря, прикладной компонент полностью размещается и выполняется на сервере системы. Схематично DBS-модель приведена на рис. 3.3.

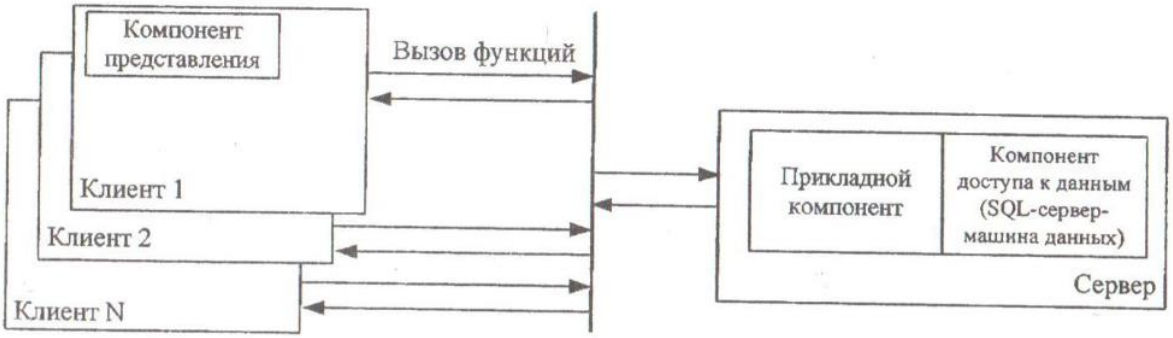


Рис. 3.3. Модель сервера базы данных.

На клиентских установках в DBS-модели размещается только интерфейсный компонент (компонент представления), что существенно снижает требования к вычислительной установке клиента. Пользователь через интерфейс системы на клиентской установке направляет на сервер базы данных только лишь вызовы необходимых процедур, запросов и других функций по обработке данных. Все затратные операции по доступу и обработке данных выполняются на сервере и клиенту направляются лишь результаты обработки, а не наборы данных, как в RDA-модели. Этим обеспечивается существенное снижение трафика сети в DBS-модели по сравнению с RDA -моделью.

Следует заметить, что на сервере системы выполняются процедуры прикладных задач одновременно всех пользователей системы. В результате резко возрастают требования к вычислительной установке сервера, причем как к объему дискового пространства и оперативной памяти, так и к быстродействию. Это основной недостаток DBS-модели.

К достоинствам же DBS-модели, помимо разгрузки сети, относится и более активная роль сервера сети, размещение, хранение и выполнение на нем механизма событий, правил и процедур, возможность более адекватно и эффективно «настраивать» распределенную информационную систему на все нюансы предметной области.

Также более надежно обеспечивается согласованность состояния и изменения данных и, вследствие этого, повышается надежность хранения и обработки данных, эффективно координируется коллективная работа пользователей с общими данными.

Чтобы разнести требования к вычислительным ресурсам сервера в отношении быстродействия и памяти по разным вычислительным установкам, используется *модель сервера приложений.*

Суть AS-модели заключается в переносе прикладного компонента информационной системы на специализированный, в отношении повышенных ресурсов по быстродействию, дополнительный сервер системы. Схема AS-модели приведена на рис. 3.4.

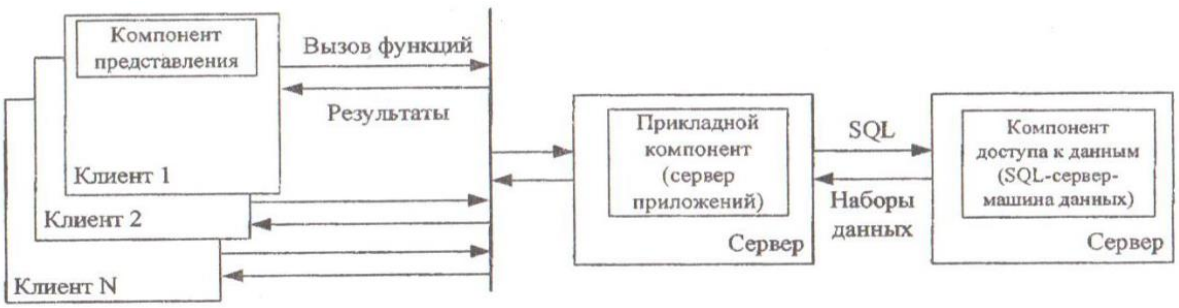


Рис. 3.4. Модель сервера приложений (AS-модель).

Как и в DBS-модели, на клиентских установках располагается только интерфейсная часть системы, т. е. компонент представления. Однако вызовы функций обработки данных

направляются на сервер приложений, где эти функции совместно выполняются для всех пользователей системы. За выполнением низкоуровневых операций по доступу и изменению данных сервер приложений, как в RDA-модели, обращается к SQL-серверу, направляя ему вызовы SQL-процедур, и получая, соответственно, от него наборы данных.

Как известно, последовательная совокупность операций над данными (SQL-инструкций), имеющая отдельное смысловое значение, называется транзакцией.

В этом отношении сервер приложений управляет формированием транзакций, которые выполняет SQL-сервер. Поэтому прог- ' раммный компонент СУБД, инсталлируемый на сервере приложений, еще называют монитором обработки транзакций (Transaction Processing Monitors - TRM), или просто монитором транзакций.

AS-модель, сохраняя сильные стороны DBS-модели, позволяет оптимально построить вычислительную схему информационной системы, однако, как и в случае RDA-модели, повышает трафик сети.

В практических случаях используются смешанные модели, когда простейшие прикладные функции и обеспечение ограничений целостности данных поддерживаются хранимыми на сервере процедурами (DBS-модель), а более сложные функции предметной области (так называемые правила бизнеса) реализуются прикладными программами на клиентских установках (RDA-модель) или на сервере приложений (AS-модель).

Контрольные вопросы:

1. Основные принципы создания и функционирования распределенных баз данных.
2. Проблемы обеспечения целостности распределенной базы данных.
3. Модель файлового сервера.
4. Модель удаленного доступа к данным.
5. Модель сервера базы данных.
6. Модель сервера приложений.

Лекция 10: Безопасность базы данных в централизованных многопользовательских информационных системах

План:

1. Типы управляемой базы данных СУБД
2. Обеспечение целостности базы данных

Существует ряд проблем обеспечения целостности распределенной БД, помимо тех аспектов, которые присущи любым БД:

– возможность одновременного доступа нескольких пользова- возможность одновременного доступа нескольких пользователей к одной и той же информации (особенно, если эти обращения к БД - корректирующие);

– физический разброс отдельных частей БД по разным компьютерам;

– разнотипность источников информации.

Первая проблема имеет место в любых распределенных БД, вторая - если база данных является распределенной, третья - если система является гетерогенной.

Первая группа проблем в области обеспечения целостности в распределенных БД обусловлена в основном возникновением опасности искажения данных при их одновременной корректировке разными пользователями.

Возможны разные схемы обеспечения целостности данных при выполнении корректирующих обращений в многопользовательском режиме:

– запрещение корректировки информации, если ее корректирует другой пользователь (блокировка);

– корректировка разных копий информационных единиц и последующее устранение возникающих коллизий.

Если СУБД предоставляет возможность выбора способа обеспечения целостности при многопользовательских обращениях, то на результат этого выбора будут влиять многие факторы, в том числе:

– степень конкуренции при выполнении корректирующих обращений - насколько часто возникает ситуация одновременной корректировки одной и той же информационной единицы;

– ограничения на время реакции системы;

– требования к актуальности и непротиворечивости данных в каждый момент времени;

– характеристика технических средств.

Вторая группа проблем обеспечения целостности в распределенных системах вызвана распределением данных и, как следствие, распределением процедур их обработки, т.е. это проблемы, обусловленные именно разнесением данных на разные узлы системы.

Как известно, существует два подхода к обеспечению целостности в распределенных информационных системах - строгая целостность (tight consistency) и нестрогая целостность (loose consistency). Первый вариант гарантирует целостность данных в любой момент времени, например, с помощью двухфазногб протокола фиксаций (2PC). Обеспечение строгой целостности требует высокого качества коммуникаций, поскольку все узлы должны быть постоянно доступны. Второй подход допускает наличие временной задержки между внесением изменений в публикуемую базу и их отражением на узлах подписчиков.

Протокол двухфазной фиксации транзакции состоит в последовательном прохождении базы данных в процессе выполнения транзакции через два этапа. Первый этап (первая фаза) - выполнение синхронизированного захвата всех объектов данных, к которым имело место обращение от имени транзакции. Объекты данных захватываются на всех серверах. На втором этапе (во второй фазе) либо происходят все изменения на всех серверах, либо, в случае хотя бы одной ошибки, происходит откат к состоянию, в котором находилась база данных до выполнения первого этапа.

Механизм двухфазной фиксации транзакции имеет ряд недостатков:

– захват всех необходимых данных на всех серверах может надолго заблокировать доступ к данным;

– велика вероятность отказа от обновления из-за какой-нибудь, пусть единичной, ошибки;

– если какой-либо сервер или выход в глобальную сеть окажется недоступным, произойдет потеря транзакции;

– использование в структуре сети координирующего узла связано с дополнительной опасностью, поскольку выход его из строя приведет к блокировке данных, затронутых транзакцией, до тех пор, пока он не будет восстановлен;

– сложность обработки транзакции, при использовании этого протокола сама служит источником дополнительного трафика, что увеличивает время реакции системы.

Кроме того, недостаточная пропускная способность сети и малая скорость передачи данных могут увеличить время реакции до недопустимого уровня.

Разные СУБД поддерживают разные технологии обеспечения целостности.

Способы защиты данных. При работе в многопользовательском режиме особую актуальность приобретает защита данных от несанкционированного доступа. Существуют различные приемы управления доступом к базе данных. Эти приемы обеспечивают разный уровень безопасности. Некоторые из них присущи любым информационным системам, другие - конкретным СУБД.

Шифрование. Шифрование базы данных - эго простейший способ защиты, при котором ее файл видоизменяется и становится недоступным Для чтения с помощью стандартных служебных программ или текстовых редакторов. Шифрование обычно применяется при электронной передаче базы данных или сохранении ее на внешний носитель.

Дешифрование базы данных - это операция, обратная шифрованию.

Скрытие объектов. Другим способом защиты объектов в базе данных от посторонних пользователей является скрытие всей базы данных при просмотре каталогов средствами операционной системы или скрытие отдельных объектов БД при работе с базой данных средствами конкретной СУБД. Этот способ защиты не является достаточно надежным, поскольку скрытые объекты относительно просто можно отобразить.

Использование параметров запуска. Эти параметры позволяют задать стартовую форму, которая автоматически открывается при открытии базы данных. При определении формы можно скрыть окно базы данных, предоставляемое СУБД, и установить собственную кнопочную форму. При этом пользователь может выполнять с базой данных только те действия, которые допускает интерфейс, предоставляемый данным приложением.

Использование пароля. Другим простейшим способом защиты является установка пароля для открытия базы данных. После установки пароля при каждом открытии базы данных будет появляться диалоговое окно, в которое требуется ввести пароль. Только те пользователи, которые введут правильный пароль, смогут открыть базу данных.

В принципе может быть установлен единый пароль для всех пользователей БД. Но наиболее гибким и распространенным способом защиты базы данных является защита на уровне пользователя, при котором каждому пользователю присваивается пароль. При запуске СУБД каждый пользователь должен быть идентифицирован; система проверяет соответствие идентификатора пользователя и его пароля.

Для каждого пользователя могут быть определены не только уникальный код, но и уровень доступа, и объекты, доступ к которым получает пользователь.

Многие СУБД позволяют кроме единичных пользователей создавать еще и их группы.

Запрещение репликации базы данных. Как указывалось выше, репликация позволяет пользователям создавать копию общей базы данных. Это может быть в принципе использовано и для дальнейшего нелегального распространения реплицированных данных. Поэтому в некоторых случаях может потребоваться запретить репликацию базы данных.

Запрещение установки паролей и настройки параметров запуска пользователями. При многопользовательском доступе к данным важно обеспечить не только сохранность и конфиденциальность данных, но и возможность доступа к данным всем тем пользователям данных, для которых это необходимо. Поэтому СУБД должна иметь механизмы, не позволяющие любым пользователям устанавливать пароль на БД.

Также желательно иметь механизм установки запрета на изменение параметров запуска, которые определяют такие свойства, как настраиваемые меню, настраиваемые панели инструментов и стартовую форму.

Создание и удаление пользователей. При работе в многопользовательской среде большое значение приобретает понятие пользователь базы данных - владелец определенного набора объектов базы данных.

Пользователи системы могут быть разделены на классы. В системе любого размера всегда имеются некоторые типы суперпользователей - пользователей, которые автоматически имеют большинство (или все) привилегий и могут передать свой статус суперпользователя кому-нибудь с помощью привилегии или группы привилегий. Администратор базы данных *(DBA)* является термином, наиболее часто используемым для такого суперпользователя и для привилегий, которыми он обладает.

Других пользователей создают администраторы баз данных; они же дают им начальные привилегии. Создавать пользователей могут только администраторы. Давать права пользователям и отбирать их могут не только администраторы, но и другие пользователи, обладающие соответствующими правами.

Пользователи могут объединяться в группы. Группа пользователей - это пользователи, наделенные одинаковым набором привилегий. Один и тот же пользователь в принципе может входить в разные группы. Каждый пользователь имеет специальное идентификационное имя или номер (Authorization Ю).

Поскольку большинство промышленно эксплуатируемых корпоративных СУБД являются SQL-серверами, рассмотрим вопросы управления пользователями на примере SQL-систем.

Конкретные формы процесса управления пользователями в различных СУБД могут значительно отличаться друг от друга. Процесс управления пользователями в большой мере зависит от используемой операционной системы, архитектуры БД. В связи с этим в соответствующей части SQL наблюдается большая зависимость от платформы и производителя.

Процесс управления пользователями можно разбить на три главных этапа. Сначала необходимо создать учетную запись пользователя в базе данных. Далее пользователя необходимо наделить привилегиями сообразно тем задачам, которые пользователь предположительно будет решать в рамках базы данных. Наконец, после того, как доступ к данным пользователю будет уже не нужен, необходимо либо удалить из базы данных его учетную запись, либо отменить ранее предоставленные ему привилегии.

Перед началом работы с БД пользователь должен быть идентифицирован с помощью процедуры входа, обычно включающей запрос имени и пароля пользователя. После входа запускается сеанс (sessions) работы с СУБД.

Определение и отмена привилегий. Распределенные БД предполагают работу с базой данных многих пользователей. Однако не всем пользователям следует разрешать выполнять любые действия с базой данных. Поэтому пользователям предоставляются привилегии.

Привилегии в базе данных делятся на две категории: системные привилегии (system privileges) и объектные привилегии (object privileges). Системные привилегии контролируют общий доступ к оазе данных. К ним относятся право создавать таблицы и другие объекты, а также право администрировать базу данных.

Объектные привилегии связаны с конкретным объектом базы данных. Объектная привилегия состоит из трех частей:

- объекта, к которому применяется привилегия;
- операции, которые она разрешает;
- пользователя, которому даются эти привилегии.

Одна из первых привилегий, которая должна быть определена, - это привилегия создателей таблиц. Если все пользователи будут иметь возможность создавать в системе базовые таблицы, это может привести к избыточности данных, их несогласованности и, как следствие, к неэффективности системы.

Пользователь, создавший таблицу, является ее владельцем. Это означает, что пользователь имеет все привилегии в созданной им таблице и может передавать привилегии другим пользователям.

Каждый пользователь в среде SQL имеет специальное идентификационное имя (или номер).

Привилегии даются оператором GRANT (ПРЕДОСТАВИТЬ) и отменяются оператором REVOKE (ОТМЕНИТЬ).

Оператор GRANT имеет следующий синтаксис:

```
GRANT привилегия....ON имя объекта
TO {пользователь, которому предоставляется привилегия.,,.}|PUBLIK
[WITH GRANT OPTION];
привилегиям
{ALL PRIVILEGES}
|{SELECT |DELETE
| {INSERT [(имя столбца.,..)]}
| {UPDATE [(имя столбца.,..)]}
I {REFERENCES [(имя столбца.,..)]}
I USAGE}.
```

Когда SQL получает оператор GRANT, он проверяет привилегии пользователя, подавшего эту команду, чтобы определить, допустим ли оператор GRANT.

- Для пользователя таблицы могут быть назначены следующие типы привилегий:
- SELECT - разрешение выполнять запросы в таблице.
- INSERT - разрешение выполнять, оператор INSERT (вставка новой строки) в таблице.
- UPDATE - разрешение выполнять, оператор UPDATE (обновление значений полей) в таблице. Можно ограничить эту привилегию для определенных столбцов таблицы.
- DELETE - разрешение выполнять, оператор DELETE (удаление записей) в таблице.
- REFERENCES - разрешение определить внешний ключ.

В одном операторе GRANT можно назначить несколько привилегий, перечислив их через запятую, или использовать аргумент ALL, означающий, что пользователю передаются все привилегии для данной таблицы.

В одном операторе GRANT можно назначить привилегии нескольким пользователям одновременно, перечислив их через запятую, или использовать аргумент PUBLIC,

означающий, что все привилегии передаются пользователям. Однако последней возможностью нужно пользоваться с осторожностью, так как PUBLIC означает не только текущих пользователей, но и всех пользователей, которые могут быть введены в систему в дальнейшем.

Предположим, что пользователь Mansurov является владельцем таблицы «Sotrudnik» и хочет позволить пользователю Karimov выполнять запросы к ней. В этом случае пользователь Mansurov должен ввести команду:

GRANT SELECT ON Sotrudnik TO Karimov;

Предложение WITH GRANT OPTION позволяет передать пользователю возможность назначать привилегии для этой таблицы. Если, например, команда выглядит следующим образом:

GRANT SELECT ON Sotrudnik TO Karimov WITH GRANT OPTION; то пользователь Karimov получает возможность, в свою очередь, передавать право назначать привилегии другим пользователям, т. е. пользователь Karimov может задать следующую команду:

GRANT SELECT ON Mansurov. Sotrudnik TO Salimov WITH GRANT OPTION.

Обратите внимание на то, что когда на таблицу ссылается пользователь, не являющийся владельцем схемы, то перед именем таблицы указывается имя схемы.

Большинство привилегий объекта использует один и тот же синтаксис. Из перечисленных выше привилегий исключение составляют UPDATE и REFERENCES.

При задании привилегии UPDATE можно использовать тот же синтаксис, который применялся выше. Это будет означать, что пользователю дается право обновлять содержимое всех столбцов таблицы. Можно также после названия привилегии в скобках указать имена столбцов (если их несколько, то имена указываются в любой последовательности через запятую), на которые распространяется данная привилегия. Например, привилегия UPDATE может выглядеть следующим образом:

GRANT UPDATE (dolgnost, oklad) ON Sotrudnik TO Karimov.

При задании привилегии REFERENCES также задаются имена столбцов.

Чтобы ограничить возможность просмотра таблицы только отдельными столбцами, следует воспользоваться механизмом создания представлений и назначать привилегии не для реальной таблицы, а для представления. Представления могут использоваться также и для обеспечения возможности ограничить просмотр только определенными строками.

Отмена привилегий осуществляется с помощью оператора REVOKE. Эта команда имеет синтаксис, схожий с синтаксисом оператора GRANT. Например, отмена привилегии на просмотр таблицы «Sotrudnik» для пользователя Karimov будет выглядеть следующим образом:

REVOKE SELECT ON Sotrudnik TO Karimov.

В конкретных СУБД могут поддерживаться привилегии, отличающиеся от привилегий, приведенных выше. Так, в некоторых СУБД имеется возможность задавать привилегию INDEX, которая позволяет пользователям создавать индексы. Но объект INDEX в стандарте SQL не определен, и синтаксис команды задания этой привилегии может отличаться от системы к системе.

Стандартом SQL не определено, кто имеет право отменять привилегии. Однако, обычно действует подход, при котором привилегии отменяются тем пользователем, который их предоставил.

Контрольные вопросы:

1. Объясните протокол двухфазной фиксации транзакции.
2. Способы шифрования и дешифрования базы данных.
3. Как происходит отображение и скрытие объектов?

Лекция 11: Аудит безопасности и резервное копирование базы данных

План:

1. Обзор аудита баз данных SQL Azure
2. Определение политики аудита уровня сервера и базы данных

Аудит экземпляра SQL Server или базы данных SQL Server включает в себя отслеживание и протоколирование событий, происходящих в системе. Объект *Подсистема аудита SQL Server* объединяет отдельные экземпляры действий или групп действий уровня сервера или базы данных, за которыми нужно проводить наблюдение. Аудит работает на уровне экземпляра SQL Server. На одном экземпляре SQL Server может существовать несколько аудитов. Объект *Спецификация аудита на уровне базы данных* также принадлежит подсистеме аудита. Для аудита вы можете создать одну спецификацию аудита базы данных для каждой базы данных SQL Server. Дополнительные сведения см. в разделе Подсистема аудита SQL Server (Database Engine).

Некоторые задачи, связанные с аудитом, можно выполнять с помощью обозревателя объектов в среде Среда SQL Server Management Studio. Например, запуск или остановку сбора данных и просмотр журналов сбора данных.

Узлы, связанные с аудитом, находятся в иерархии обозревателя объектов в среде Management Studio, как показано ниже.

Экземпляр SQL Server
---- Безопасность
-------- Аудиты
-------- Спецификации аудита сервера
Экземпляр SQL Server
---- Базы данных
-------- Имя базы данных
---------- Безопасность
---------- Спецификации аудита базы данных

Аудит базы данных — это комплекс мероприятий, результатом которых становится оценка двух определяющих эффективность информационного массива параметров: рациональность и безопасность.

С точки зрения оптимизации процессов, прямо или косвенно влияющих на КПД базы данных, аудит информационного массива должен являться отправной точкой. В зависимости от периодичности применения, аудит систем баз данных можно разделить на:

– первичный, который проводится в таких случаях, как, например, начало нового проекта или построение и настройка ИТ-процессов и связей в компании;

– регулярный, цель которого — периодическая проверка структуры базы данных, для поиска ошибок и путей оптимизации в постоянно меняющейся информационной среде.

Аудит БД имеет целью следующие пункты:

– организация быстрой и слаженной работы всех элементов базы данных;
– обновление базы до актуального на момент оценки состояния;
– устранение ошибок, влияющих на работу базы данных;
– определение и устранение причин, вызывающих ошибки в базе данных;
– оптимизация затрат временных ресурсов в работе с базой данных.

Важно также понимать, что выборочный аудит определенных секторов информационной базы данных вторичен относительно комплексного метода, который дает полную картину массива.

Для создания полноценной системы безопасности и защиты базы данных процедура аудита имеет не менее важное значение, чем грамотно настроенный брандмауэр или своевременно обновленное антивирусное обеспечение.

В результате аудита безопасности базы данных становятся понятны:

– общая оценка защищенности информационной базы;
– слабые места в системе защиты БД;
– пути исправления ситуации по устранению брешей в защите БД.

Рекомендованный специалистами в области безопасности общий список событий, подлежащих аудиту, выглядит так:

– предоставление прав доступа к базе данных;
– любые вмешательства в файловую структуру базы данных;
– все инциденты, связанные с отказом в доступе к базе данных;
– любые потенциально опасные действия в системе базы данных;
– все действия или попытки действий в системе без соответствующих прав доступа;
– несанкционированное изменение настроек сервера или программного обеспечения базы данных.

Класс событий	Идентификатор события	Описание
Аудит входа в систему	1	Записывает все новые события соединения с момента запуска трассировки, например запросы клиентов на соединение с сервером, на котором запущен экземпляр служб Microsoft SQL Server Службы Analysis Services.
Аудит выхода из системы	2	Записывает все новые события отключения с момента запуска трассировки, например выполнение клиентом команды отключения.

Аудит запусков и остановок сервера	4	Записывает операции выключения, запуска и приостановки служб.
Событие аудита разрешений на объекты	18	Записывает все изменения разрешений на объекты.
Событие аудита операций администрирования	19	Записывает серверные операции резервного копирования, восстановления, синхронизации, присоединения, отсоединения, загрузки образа и сохранения образа.

Дополнительные сведения о столбцах, связанных с каждым из классов событий «Аудит безопасности», см. в разделе Столбцы данных аудита безопасности.

Аудит базы данных SQL Azure позволяет отслеживать события базы данных и записывать их в журнал аудита в учетной записи хранения Azure. Аудит также дает следующие возможности:

– Аудит может помочь вам соблюсти требования нормативов, проанализировать работу с базой данных и получить представление о расхождениях и аномалиях, которые могут указывать на бизнес-проблемы или предполагаемые нарушения безопасности.

– Средства аудита способствуют соблюдению стандартов соответствия, но не гарантируют их выполнение. Дополнительную информацию о программах Azure, поддерживающих проверку соблюдения стандартов, см. в Центре управления безопасностью Azure.

Обзор аудита баз данных SQL Azure

Используйте аудит базы данных SQL, чтобы выполнять следующие действия:

– **Сохранить** журнал аудита выбранных событий. Вы можете указать, какие категории действий базы данных должны проходить аудит.

– **Создавать отчеты** по действиям, производимым с базой данных. Вы можете использовать предварительно настроенные отчеты и панель мониторинга для быстрого начала работы с отчетностью по действиям и событиям.

– **Анализировать** отчеты. Вы можете искать подозрительные события, необычную деятельность и тенденции.

Аудит можно настроить для различных типов категорий событий (см. раздел Настройка аудита базы данных).

Журналы аудита записываются в хранилище BLOB-объектов Azure в подписке Azure.

Определение политики аудита уровня сервера и базы данных

Политику аудита можно определить для конкретной базы данных или как политику сервера по умолчанию.

– Политика сервера применяется ко всем существующим и новым базам данных на сервере.

– Если *включен аудит больших двоичных объектов для сервера*, он *обязательно применяется к базе данных*. Аудит базы данных будет выполнен независимо от его параметров.

– Включение аудита больших двоичных объектов для базы данных (кроме его включения для сервера), *не* приводит к переопределению или изменению настроек аудита больших двоичных объектов для сервера. Оба аудита будут выполняться параллельно. Таким образом, аудит базы данных выполняется дважды: один раз в соответствии с политикой сервера и еще раз в соответствии с политикой базы данных.

Примечание

Не устанавливайте политику аудита больших двоичных объектов одновременно для сервера и базы данных, за исключением следующих случаев.

Если для определенной базы данных нужно использовать другую учетную запись хранения или другой срок хранения.

Нужно провести аудит типов событий или категорий для конкретной базы данных, которая отличается от остальных баз данных на сервере. Например, может потребоваться выполнить аудит вставки данных в таблицу только для конкретной базы данных.

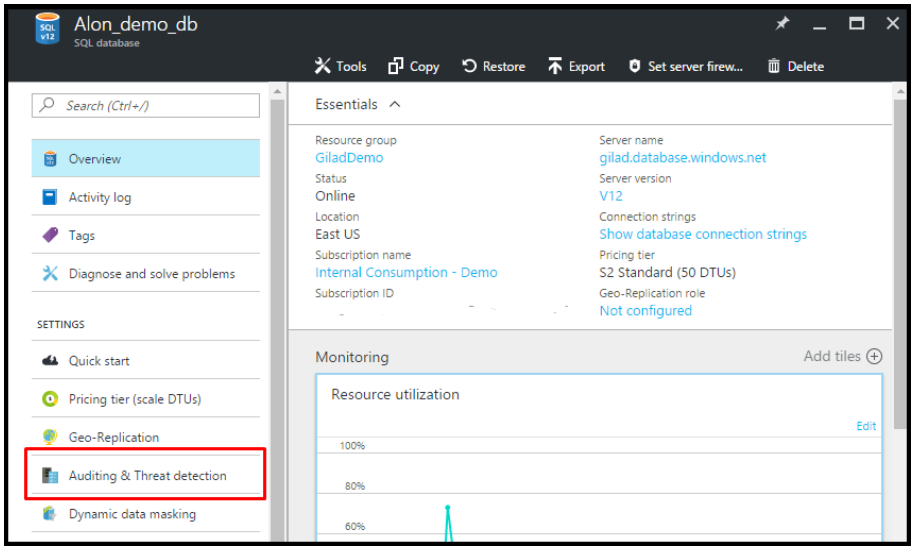
Во всех остальных случаях мы рекомендуем включать аудит больших двоичных объектов только на уровне сервера, а на уровне баз данных отключить все параметры аудита для всех баз данных.

Настройка аудита базы данных

В следующем разделе описывается настройка аудита на портале Azure.

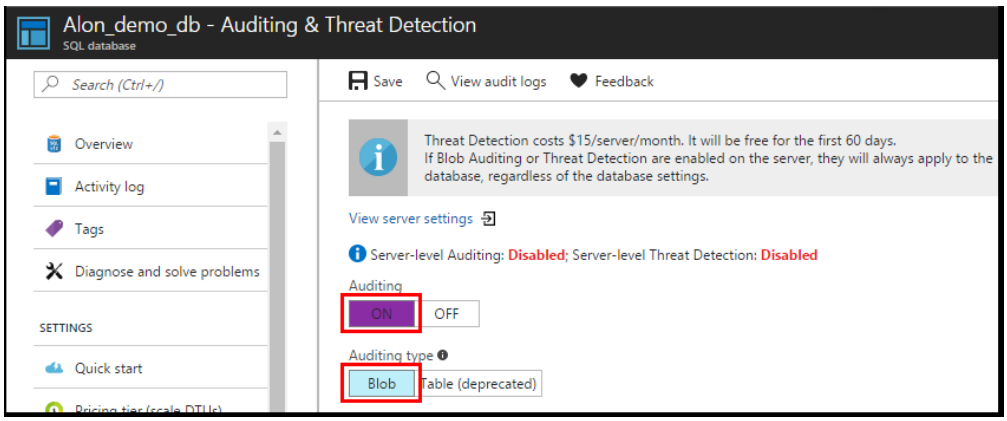
–Перейдите на [портал Azure](#).

–Перейдите в колонку **параметров** базы данных SQL или SQL Server, где нужно выполнить аудит. В колонке **Параметры** выберите **Аудит и обнаружение угроз.**

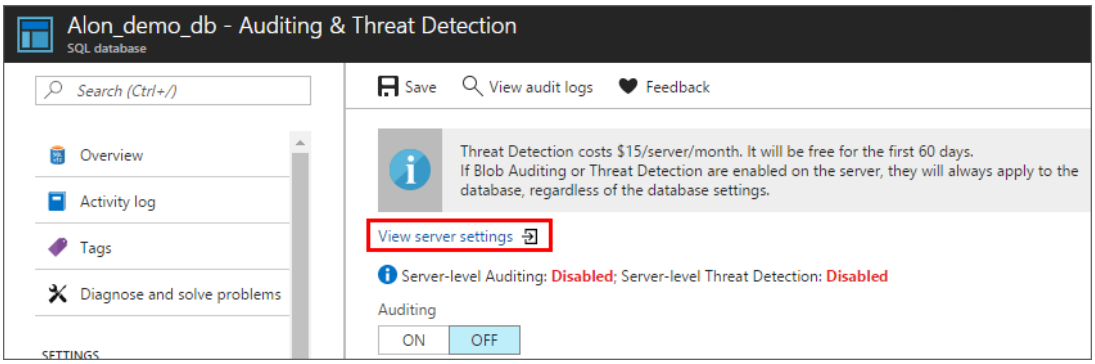


Если для вы хотите настроить политику аудита сервера, выберите ссылку **Просмотреть параметры сервера** в колонке аудита базы данных. Вы можете затем просмотреть или изменить параметры аудита сервера. Политики аудита сервера применяются ко всем существующим и создаваемым базам данных на этом сервере.

Если вы предпочитаете включить аудит больших двоичных объектов на уровне базы данных, выберите для параметра **Аудит** значение **Вкл.**, а для параметра **Тип аудита — Большой двоичный объект**.

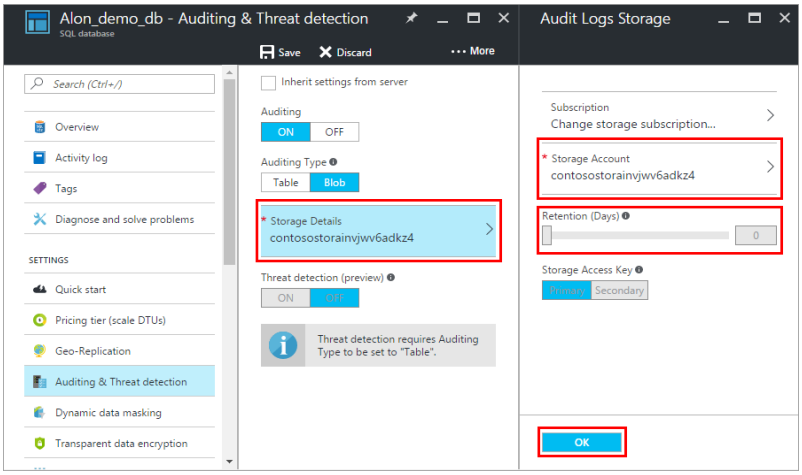


Если включен аудит больших двоичных объектов сервера, настроенный аудит для базы данных будет существовать параллельно с ним.



–Выберите **Сведения о хранилище**, чтобы открыть колонку **Хранилище журналов аудита**. Выберите учетную запись хранения Azure, в которой будут храниться журналы, и укажите срок хранения. Старые журналы будут удалены. Нажмите кнопку **ОК**.

Совет



Чтобы в полной мере воспользоваться возможностями шаблонов отчетов об аудите, используйте одну и ту же учетную запись хранения для всех баз данных.

–Настроить события аудита можно с помощью PowerShell или REST API.

–После настройки параметров аудита можно включить новую функцию обнаружения угроз и настроить адреса электронной почты для получения предупреждений системы безопасности. Использование функции обнаружения угроз позволяет настроить упреждающие оповещения об аномальной активности в базах данных, которая может указывать на потенциальные угрозы безопасности. Дополнительные сведения см. в статье Обнаружение угроз для базы данных SQL.

–Щелкните **Сохранить**.

Анализ журналов и отчетов аудита

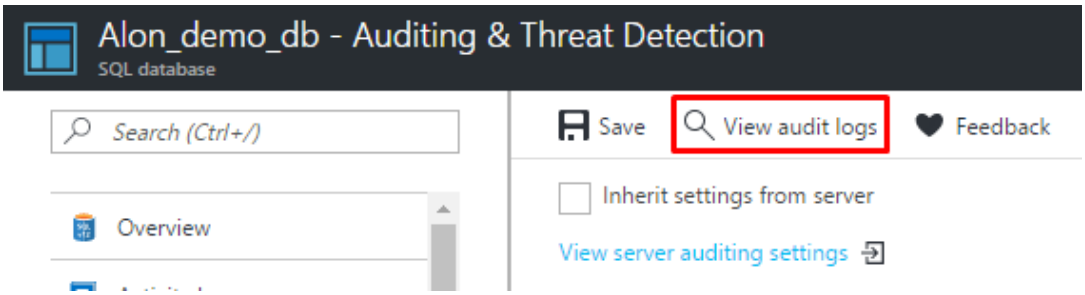
Журналы аудита объединяются в учетной записи хранения Azure, выбранной на этапе настройки. Журналы аудита можно просматривать с помощью таких инструментов, как обозреватель хранилищ Azure.

Журналы аудита больших двоичных объектов сохраняются в виде коллекции файлов больших двоичных объектов в контейнере **sqldbauditlogs**.

Дополнительные сведения об иерархии папки для хранения, соглашении об именовании и формате журнала см. в документации по формату журнала аудита больших двоичных объектов.

Просмотреть журналы аудита больших двоичных объектов можно несколькими способами:

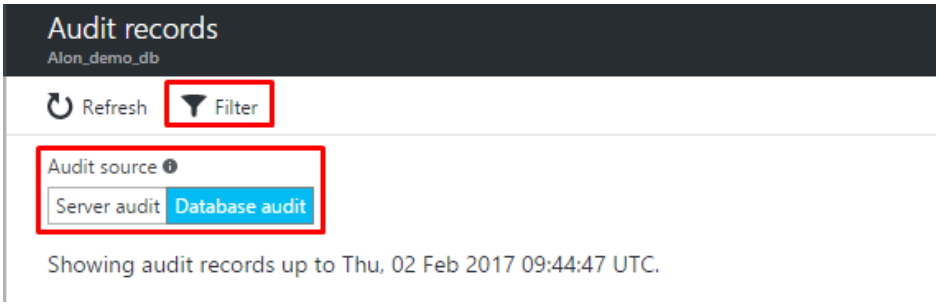
– Используйте портал Azure. Откройте соответствующую базу данных. В верхней области колонки **Auditing & Threat detection** (Аудит и обнаружение угроз) щелкните **Ознакомиться с журналами аудита**.



Откроется колонка **Записи аудита**, в которой вы сможете просматривать журналы.

– Для просмотра записей за определенную дату щелкните **Фильтр** в верхней области колонки **записей аудита**.

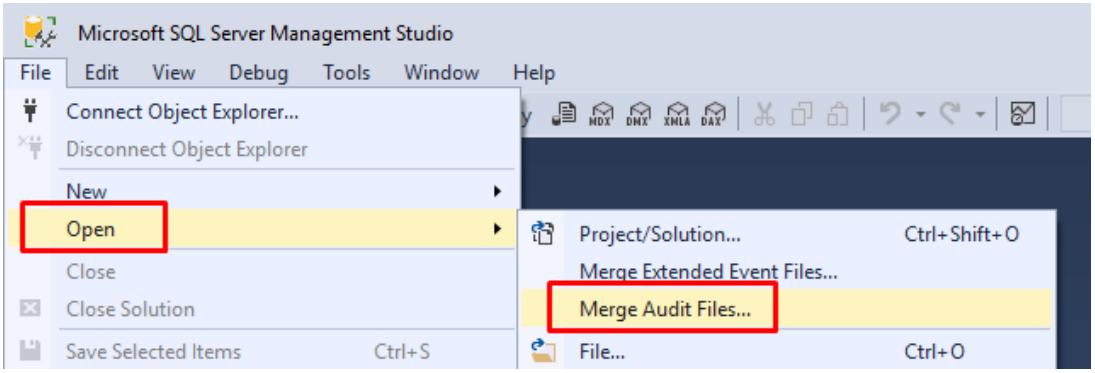
– Вы можете переключаться между записями аудита, созданными политиками аудита для сервера и для базы данных.



∗ Используйте системную функцию **sys.fn_get_audit_file** (T-SQL), чтобы вернуть данные журнала аудита в табличном формате. Дополнительные сведения об использовании этой функции см. в статье sys.fn_get_audit_file (Transact-SQL).

∗ Используйте **объединение файлов аудита** в SQL Server Management Studio (начиная с SSMS 17):

1. В меню SSMS выберите **Файл** > **Открыть** > **Объединение файлов аудита**.



2. Откроется диалоговое окно **Добавление файлов аудита.** Выберите один из способов **добавления**. Вы можете использовать объединение файлов аудита из локального диска или импортировать их из службы хранилища Azure. Необходимо предоставить сведения о службе хранилища Azure и ключ учетной записи.

3. После добавления всех файлов для слияния нажмите кнопку **OK** для завершения операции слияния.

4. Объединенный файл откроется в SSMS, где вы сможете его просмотреть и проанализировать, а также экспортировать в XEL- или CSV-файл или в таблицу.

– Используйте созданное приложение синхронизации. Оно работает в Azure и использует общие API-интерфейсы службы Log Analytics для Operations Management Suite (OMS) для передачи журналов аудита SQL в OMS. Приложение синхронизации отправляет журналы аудита SQL в OMS Log Analytics для использования через панель мониторинга OMS Log Analytics.

– Используйте Power BI. Вы можете просматривать и анализировать данные журнала аудита в Power BI. Вы можете узнать больше о Power BI, а также скачать шаблон.

– Скачайте файлы журналов из контейнера больших двоичных объектов хранилища Azure из портала или с помощью обозревателя хранилища Azure.

* После загрузки файла дважды щелкните по нему, чтобы открыть, просмотреть и проанализировать файл в среде SSMS.

– Вы также можете загрузить одновременно несколько файлов через обозреватель хранилища Azure. Щелкните правой кнопкой мыши конкретную вложенную папку и выберите **Сохранить как**, чтобы сохранить ее в локальной папке.

– Дополнительные методы

* После загрузки нескольких файлов или вложенной папки, содержащей файлы журнала, можно объединить их локально, как описано выше в инструкциях по объединению файлов аудита в SSMS.

* Чтобы просмотреть журналы аудита больших двоичных объектов программным способом:

* Используйте библиотеку C# для считывания расширенных событий.

* Используйте запросы к файлам расширенных событий с помощью PowerShell.

Рекомендации для рабочей среды

Аудит георепликированных баз данных

* При использовании георепликированных баз данных, если включить аудит для базы данных-источника, к базе данных-получателю будут применяться идентичные политики

аудита. Кроме того, можно настроить аудит базы данных-получателя, включив аудит для **сервера-получателя**, независимо от базы данных-источника.

* Аудит на уровне сервера (**рекомендуется**). Включите аудит на **сервере-источнике**и **сервере-получателе** — все базы данных-источники и базы данных-получатели пройдут независимый аудит на основе своей соответствующей политики на уровне сервера.

* Аудит на уровне базы данных. Аудит на уровне базы данных для баз данных-получателей можно настроить только с помощью параметров аудита в базе данных-источнике.

* Его необходимо включить для самой базы данных-источника, а не для сервера.

* После включения аудита больших двоичных объектов в базе данных-источнике он станет доступным в базе данных-получателе.

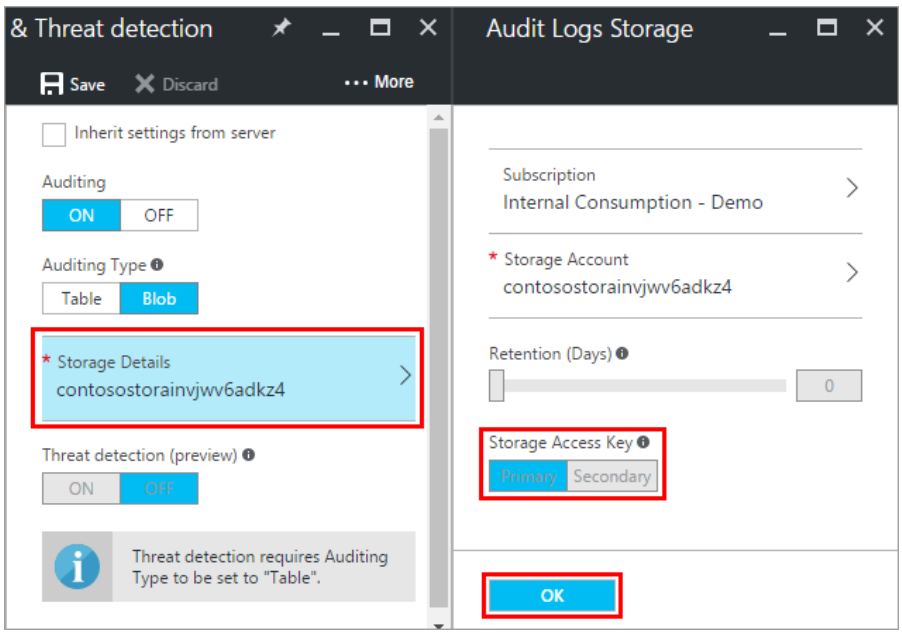
Важно!

При использовании аудита на уровне базы данных настройки хранилища для базы данных-получателя будут совпадать с настройками для базы данных-источника, что приведет к появлению трафика между регионами. Рекомендуем включать аудит только на уровне сервера, а на уровне баз данных отключить все параметры аудита для всех баз данных.

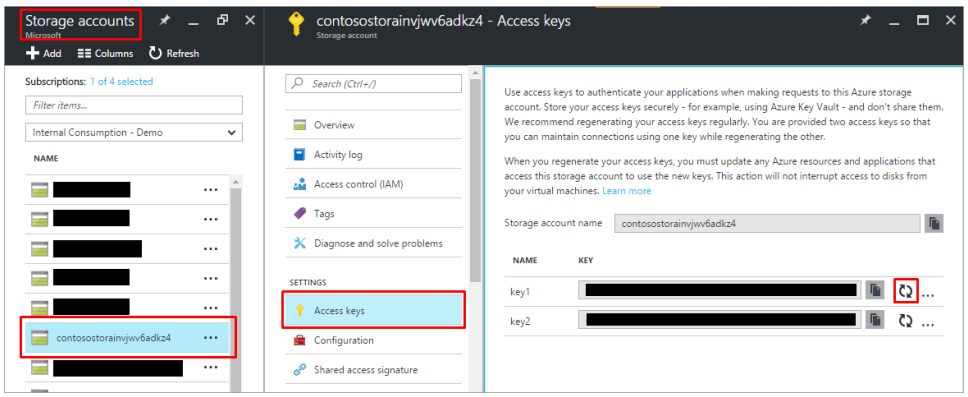
Повторное создание ключа к хранилищу данных

Обычно в рабочей среде приходится периодически обновлять ключи хранилища. При обновлении ключей необходимо повторно сохранить политику аудита. Процесс таков:

1. Откройте колонку **Сведения о хранилище**. В поле **Ключ доступа к хранилищу**выберите **Вторичный**, а затем нажмите кнопку **OK**. Затем щелкните **Сохранить** в верхней области колонки настройки аудита.



2. Перейдите в колонку конфигурации хранилища и повторно создайте первичный ключ доступа.



3. Вернитесь в колонку настройки аудита, измените значение ключа доступа к хранилищу с вторичного на первичный, затем нажмите кнопку **OK**. Затем щелкните **Сохранить** в верхней области колонки настройки аудита.

4. Вернитесь в колонку настройки хранилища и повторно создайте ключ доступа получателя (для подготовки к следующему циклу обновления ключей).

Контрольные вопросы:

1. Что понимается под аудитом информационной системы?
2. Какие активы организации рассматривается при проведении аудита информационной безопасности?
3. Перечислите виды аудита информационной безопасности.
4. На какие части разделен аудит в SQL?

Лекция 12: Особенности проведения аудита безопасности в системах управления базами данных

План:

Под аудитом информационной системы (ИС) или информаци-' онной технологии (ИТ) понимается системный процесс получения и оценки объективных данных о текущем состоянии системы, технологии, действиях и событиях, происходящих в ней, устанавливающий уровень их соответствия определенному критерию и предоставляющий результаты заказчику.

Проведение аудита позволяет оценить текущую безопасность функционирования ИТ, оценить риски и управлять ими, прогнозировать их влияние на бизнес-процессы организации, корректно и обоснованно подходить к вопросу обеспечения безопасности информационных активов организации, основными из которых являются:

- идеи;
- знания;
- проекты;
- результаты внутренних обследований.

Общее понятие аудита. Датой рождения аудита принято считать 1844г., когда в Англии приняли закон об акционерных компаниях, согласно которому их правления должны были ежегодно отчитываться перед аукционерами, причем отчет должен был быть проверен и подтвержден специальным человеком - независимым аудитором.

В настоящее время аудит, пройдя несколько этапов своего развития стал частью хозяйственной жизни страны. От проверки бухгалтерских счетов акционерных компаний отдельными профессиональными аудиторами аудит развивался до комплексного понятия, включающего в себя ряд услуг (проверку бухгалтерской отчетности, финансовый анализ, консультирование), оказываемых профессиональными аудиторами и аудиторскими фирмами. Среди таких фирм есть и небольшие, включающие в себя десяток сотрудников, и гиганты с численностью до нескольких тысяч человек.

Виды аудита. Аудит безопасности информационных систем обычно подразделяют на внешний и внутренний.

Внешний аудит проводится в основном вне организации и, как правило, специализированными организациями, занимающимися аудитом информационной безопасности. В этом случае, анализируются меры риска от внешних атак и атак со стороны (даже если организация защищена межсетевыми экранами). При проведении внешнего аудита осуществляется сканирование портов, поиск уязвимостей в сетевом и прикладном программном обеспечении,

Осуществляются попытки взаимодействия с Web-серверами, почтовым и файловыми серверами, попытки вхождения в локальные сети организации. По желанию руководства организации, может проводиться специальный вид внешнего аудита - Ethical Hacking. В этом случае, специальная организация (в мире это является широко распространенной практикой, такие подразделении имеют специальное название Tiger Team) осуществляет избранные атаки на серверы, сайты и хосты организации. Такие атаки могут продемонстрировать уязвимости ИС организации.

Внутренний аудит, как правило, проводился специальной командой из числа персонала организации. Его задачей является оценка риска существующей технологии применения ИС. Этот вид аудита выполняется с привлечением средств автоматизации аудита, реализующих какой-либо стандарт. Внутренний аудит проводится внутри сетевого пространства, ограниченного межсетевым экраном организации. Он также включает в себя сканирование портов и уязвимостей внутренних хостов организации. Кроме того, анализируются организация и выполнение установленной политики безопасности, контроль и управление доступом к ресурсам, парольная политика персонала организации и ее выполнение, Данный вид аудита дополняет стандартные методики проведения аудита более исчерпывающим рассмотрением сетевых уязвимостей.

Проведения аудита безопасности баз данных на примере СУБД Oracle. СУБД Oracle - функционально развитый продукт, и в нем существует несколько возможностей проведения аудита.

Аудит Oracle может помочь в определении неавторизованного доступа или внутреннего злоупотребления по отношению к информации содержащейся в базе данных.

Аудит в Oracle разделен на три части:

- аудит таких выражений как CREATE TABLE или CREATE SESSION,
- аудит привилегий ALTER USER
- аудит на объект на объектном уровне SELECT TABLE.

Основная конфигурация. Записи аудита могут помещаться ли- оо в аудиторскую таблицу базы данных, либо в аудиторский журнал операционной системы. Запись аудита в журнал операционной системы в некоторых случаях более защищена, но эта возможность

доступна не для всех платформ. Аудит включается для записи в базу данных добавлением следующей строки в файле init.ora.

audittrail = db

Примеры. Пример включения аудита для попыток доступа к базе данных:

```
SQL> audit create session;
Audit succeeded.
SQL>
```

Приведенная команда будет отслеживать доступ всех пользователей, независимо от того успешен он или нет.

Формат всех команд аудита по документации Oracle выглядит следующим образом:

```
audit {statement_option/privilege_option}
[by user] [by
{session/access}] [ whenever {successful/unsuccessful}]
```

Обязательными являются только лишь statement option и privilege_option части выражения. Другие части являются опционными и их использование позволяет сделать аудит более специфичным.

Чтобы пользователь мог задать команду аудита, необходимым условием для него является наличие привилегии «AUDIT SYSTEM». Найти пользователей, которые имеют эту привилегию, можно выполнив следующее:

```
SQL> select *
–from dba_sys_privs
–where privilege like '%AUDIT%';
  GRANTEE                        PRIVILEGE                      ADM
  CTXSYS                         AUDIT ANY                      NO
  CTXSYS                         AUDIT SYSTEM                   NO
  DBA                            AUDIT ANY                      YES
  DB                             AUDIT SYSTEM                   YES
  IMPFULLDATABASE                NO
AUDIT ANY
  MDSYS                          AUDIT ANY                      YES
  MDSYS                          AUDIT SYSTEM                   YES
  WKSYS                          AUDIT ANY                      NO
  WKSYS                          AUDIT SYSTEM                   NO
9 rows selected.
SQL>
```

Выше приведенные результаты принадлежат базе данных Oracle 9i. Пользователи по умолчанию MDSYS, CTXSYS и WKSYS были бы неплохой мишенью для атакующего, так как любые действия аудита могут быть выключены любым из этих пользователей, что бы скрыть любые предпринятые действия.

Теперь аудит будет отслеживать все попытки доступа, необходимо подождать, когда какие-нибудь пользователи войдут в систему что бы выполнить свою работу.

Пример включения аудита для контроля изменений в базе данных.

Для краткости, не все изменения объектов схемы будем отслеживать, в этом примере. Хотя в принципе, можно отслеживать изменения любых объектов БД: таблиц, индексов, кластеров, представлений, последовательностей, процедур, триггеров, библиотек и т.д. В этом примере аудит будет включен на выборочной группе объектов. Настройка аудита может быть выполнена за два этапа, создание команд аудита и запуск на исполнение, как показано ниже:

```
set head off
set feed off
set pages 0
spool aud.lis
select 'audit '//name//';'
from system_privilege_map
where (name like 'CREATE%TABLE%'
or name like 'CREATE%INDEX%'
/or name like 'CREATE%CLUSTER%'
or name like 'CREATE%SEQUENCE%'
or name like 'CREATE%PROCEDURE%' or name like ’CREATE%TRIGGER%’
or name like 'CREATE%LIBRARY%')
union
select 'audit '//name//';'
from system_privilege_map
where (name like 'ALTER%TABLE%'
or name like 'ALTER%INDEX%'
or name like 'ALTER%CLUSTER%'
or name like 'ALTER%SEQUENCE%'
or name like 'ALTER%PROCEDURE%'
or name like 'ALTER%TRIGGER%' or name like 'ALTER%LIBRARY%')
union
select 'audit '//пате//[1];' from system_privilege_map
where (name like 'DROP%TABLE%'
or name like 'DROP%INDEX%'
or name like 'DROP%CLUSTER%'
or name like T)ROP%SEQUENCE%'
or name like 'DROP%PROCEDURE%'
or name like 'DROP%TRIGGER%'
or name like 'DROP%LIBRARY%') union
select 'audit '//name//';'
from system_privilege_map
where (name like 'EXECUTE%INDEX%'
or name like 'EXECUTE%PROCEDURE%'
or name like 'EXECUTE%LIBRARY%')
/spool off @@aud.lis
```

Данный скрипт выведет набор команд аудита в спул файл, который затем запустится для выполнения команд аудита.

После этого базу данных необходимо перезапустить. Простая проверка покажет, что аудит действительно включен.

```
SQL> select name,value from vSparameter
2 where name like 'audit%';
        NAME                VALUE
        audit_trail         DB
        audit_file_dest     ?/rdbms/audit

        SQL>
```

Но контролируемые действия не отслеживаются до тех пор, пока эти действия не заданы явно; это верно, кроме случаев привилегированного доступа к базе данных, запуска и останова базы данных, структурных изменений, таких как добавление файла данных. Эти действия отслеживаются в файле операционной системы в $ORACLE_HOME/rdbms/audit до тех пор пока audit_file_dest не переопределено в файле init.ora . В Windows эти события появляются в Event Viewer.

Для того, что бы проверить наличие того, что какие-нибудь привилегии или выражения уже используются для аудита следующее:

```
SQL> select * from dba_stmt_audit_opts
union
select * from dba_priv_audit_opts;
no rows selected SQL>
```

Что бы найти какие объекты уже контролируются аудитом, необходимо запросить представление dba_obj_audit_opts.

Контрольные вопросы:

1. Что понимается под аудитом информационной системы?
2. Какие активы организации рассматривается при проведении аудита информационной безопасности?
3. Перечислите виды аудита информационной безопасности.
4. На какие части разделен аудит в SQL?

Лекция 13: Восстановление базы данных, процесс синхронизаций репликации в современных системах управления базами данных

План:

1. Резервное копирование базы данных
2. Основные вида резервного копирования

Для того чтобы избежать потери важной информации, необходимо проводить регулярное резервное копирование или бэкап (от англ. backup - защита, поддержка) данных. *Резервное копирование* - это периодическое дублирование или создание запасных копий критически важных данных с целью их восстановления в случае потери оригинала. Можно сказать, что резервное копирование - это страхование от потери информации в случае поломки оборудования или случайного удаления файлов пользователем. Существует два основных метода резервного копирования - это копирование файловой системы компьютера и копирование образа жесткого диска.

Копирование образа жесткого диска - это создание точной копии всего жесткого диска, что позволяет восстанавливать не только данные пользователя, но Windows и всю информацию о состоянии операционной системы, такую как данные системного реестра, драйверы, профили пользователей, системные настройки, программы и приложения.

Файловое копирование — это копирование файловой системы компьютера, то есть папок и файлов, хранящихся на компьютере. Такое копирование поможет восстановить папки и файлы пользователя, но не сможет вернуть систему в рабочее состояние. Что касается конкретных способов реализации этих двух типов копирования, то тут также можно выделить несколько основных видов: полное копирование, дифференциальное копирование и инкрементальное копирование. Полное копирование - это копирование всех указанных данных целиком и полностью, будь то образ диска или файловая система, без учета изменений, произошедших в промежутках между копированиями.

Под дифференциальным резервным копированием понимается копирование изменившейся информации со времени последнего полного бэкапа. То есть, каждое последующее копирование включает в себя все файлы, которые изменились со времени первого бэкапа. Таким образом, чтобы сделать восстановление резервной копии нужно взять первый полный и последний бэкап. Инкрементальный бэкап копирует только новые и изменившиеся файлы со времени последнего копирования, а не первого. Поэтому он занимает меньше места на носителе, чем дифференциальный. Но инкрементальный бэкап сложнее восстанавливать, так как приходится учитывать не только первый и последний бэкап-файлы, но и все промежуточные. Существует еще один способ копирования: «зеркальное копирование». Этот способ предполагает, что как только на диске появляется новый файл, он тут же появляется и в копии (в режиме реального времени). Некоторые специалисты называют это способ копирования двухсторонней синхронизацией.

Требования к системе резервного копирования:

– надёжность хранения информации. Обеспечивается применением отказоустойчивого оборудования систем хранения, дублированием информации и заменой утерянной копии другой в случае уничтожения одной из копий;

– простота в эксплуатации - автоматизация (по возможности минимизировать участие человека: как пользователя, так и администратора);

– быстрое внедрение (простая установка и настройка программ, быстрое обучение пользователей).

По способу передачи резервной копии на устройство хранения можно выделить следующие методы резервного копирования данных: через локальную вычислительную сеть, через сеть хранения данных без участия сервера резервного копирования, через сеть хранения данных с использованием механизма «мгновенных копий». Преимущества «классического» способа резервного копирования через КС на базе TCP/IP - простота реализации и внесения изменений в инфраструктуру резервного копирования: достаточно установить программное обеспечение сетевого агента резервного копирования на сервере-клиенте и настроить взаимодействие по сети с сервером резервного копирования. Резервное копирование происходит по следующей схеме: сервер резервного копирования отправляет по КС команду агенту резервного копирования на сервере-клиенте, который выполняет операции по подготовке и отправке данных на сервер резервного копирования. Сервер резервного копирования принимает и записывает данные на устройство хранения. При каких-либо изменениях в сетевой инфраструктуре сетевой агент может быть достаточно

быстро перенастроен, и система резервного копирования продолжит работу. Кроме того, технология дешева в реализации и не требует прямого подключения серверов- клиентов к сети хранения данных, т. е. может быть использована для защиты данных серверов, не подключенных к SAN (Storage Area Network - сеть хранения данных), или серверов с виртуализи- рованным подключением к сети хранения данных. К недостаткам метода относятся загрузка КС трафиком резервного копирования, нагрузка на сервер-клиент, зависимость от пропускной способности КС и интерфейсов сервера резервного копирования, необходимость выполнять резервное копирование в течение заданного «временного окна», ограниченные возможности по масштабированию. Конечно, можно организовать выделенную КС для передачи трафика резервного копирования, установить дополнительные управляемые серверы резервного копирования, агрегировать сетевые интерфейсы сервера резервного копирования в единый «толстый» интерфейс, однако эти меры ведут к усложнению инфраструктуры защиты данных и ухудшению ее масштабируемости. Тем не менее, указанная технология вполне подходит для резервного копирования редко изменяемых данных, таких как системные данные ОС и приложений.

В базе данных можно создавать файлы данных двух типов.

Первичный файл данных (primary data file) обязательным. В нём хранится загрузочная информация каталога базы данных и указатели на другие файлы базы данных. Первичный файл данных может также содержать объекты и пользовательские данные. Для имени первичного файла рекомендуется расширение mdf.

Вторичные файлы данных (Secondary data file) не являются обязательными и определяются пользователем. В них содержатся объекты и пользовательские данные. Для повышения производительности вторичные файлы рекомендуется размещать на разных дисках. В базе данных может быть не более 32 766 вторичных файлов данных. Для имени вторичного файла данных рекомендуется расширение ndf.

Модель восстановления (recovery model) - это параметр конфигурации базы данных, который управляет регистрацией транзакций, созданием резервных копий журнала транзакций и параметрами восстановления базы данных. Выбор модели восстановления оказывает существенное влияние как на восстановление данных, так и на производительность в зависимости от того, выполняет модель восстановление регистрации транзакций или нет.

Модель полного восстановления означает, что ядро базы данных регистрирует в журнале транзакций все операции и никогда не выполняет усечение журнала. Это модель позволяет восстановить базу данных до ее состояния на момент сбоя.

Простая модель восстановления регистрирует минимум данных о большинстве транзакций и выполняет усечение журнала транзакций после каждой контрольной точки. Это модель восстановления не поддерживает резервное копирование и восстановление журнала транзакций. Более того, не позволяет восстанавливать отдельные страницы данных.

Модель с неполным протоколированием означает, что ядро базы данных ведет минимальную регистрацию массовых операций, таких как SELECT INTO и BULKINSERT. Если в резервной копии журнала содержатся какие-либо массовые операции, базу данных можно восстановить до состояния, соответствующего концу резервной копии журнала транзакций, а не до определенного момента времени. Это модель восстановления используются только для больших массовых операций.

Основные виды резервного копирования. Выделяют два основных вида резервного копирования:

– Непротиворечивое (холодное) резервное копирование, когда копии создаются, в случае закрытой для пользователей БД (close). Копия базы данных, созданной в автономном режиме, содержит все файлы данных, журналы повторов и управляющие файлы. После остановки БД, все файлы базы копируются на один из backup дисков. По окончании копирования осуществляется перезагрузка базы данных.

– Резервное (горячее) копирование в оперативном режиме, к примеру, когда БД всё время находится в оперативном режиме и доступна пользователям.

Резервные копии журнала транзакций можно создавать только для баз данных, в которых установлена модель полного восстановления или модель восстановления с неполным протоколированием. Также резервное копирование журнала транзакций возможно только после выполнения резервного копирования. Резервная копия журнала транзакций содержит только часть данных, поэтому для восстановления базы данных требуется также ее полная копия.

Из двух видов резервирования наибольший выигрыш надежности достигается при резервировании замещением. Однако для реализации этого вида резервирования требуется автомат контроля состояния системы и коммутации при отказе работающей системы. База данных может быть «разнесена» на множество файловых групп, каждая из которых может включать множество дополнительных файлов данных и журналов транзакций (рис. 4.1). Наибольший эффект от разнесения базы данных на файловые группы достигается применением RAID массивов. Применение КАШ массивов повышает производительность файловой подсистемы, уменьшается время отклика системы, ее доступность и надёжность.

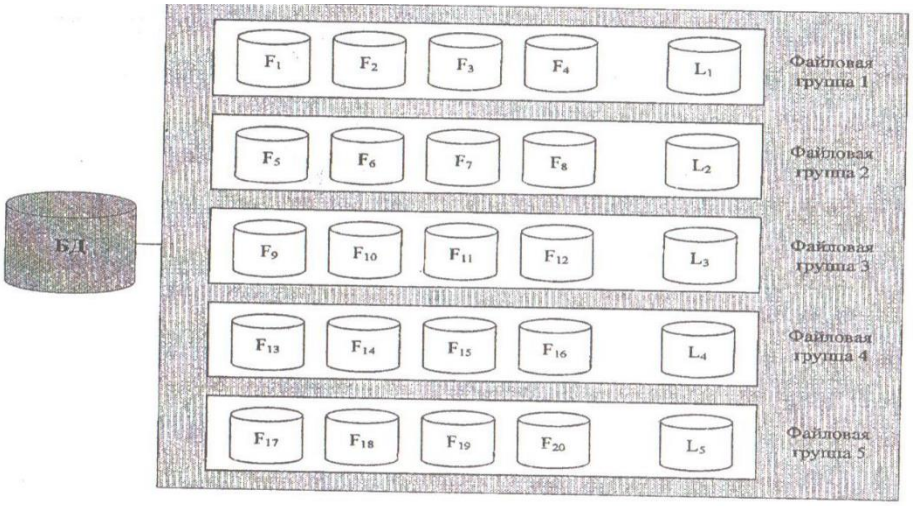


Рис. 4.1. Разбиения файла базы данных на файловые группы.

При наличии автомата контроля и коммутации структурная схема резервированной системы с кратностью m=1 будет выглядеть так, как показано на рис.4.2:

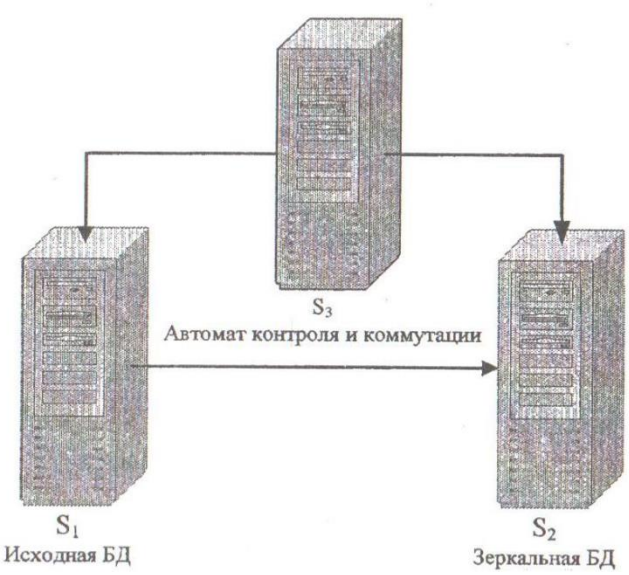


Рис.4.2. Структурная схема модуля резервного копирования с автоматом контроля и коммутации.

На рис.4.2 приняты следующие обозначения:

SI, S2 - основная и резервная системы защиты информации;

S3 - автомат контроля правильности функционирования системы защиты и коммутации при обнаружении отказа основной системы.

Процесс синхронизаций репликации в современных системах управления базами данных

Во многих случаях узким местом распределенных систем, построенных на основе технологий «Клиент-сервер» или объектного связывания данных, является недостаточно высокая производительность из-за необходимости передачи по сети большого количества данных. Определенную альтернативу построения быстродействующих распределенных систем предоставляют технологии реплицирования данных.

Репликой называют *особую копию базы данных для размещения на другом компьютере сети с целью автономной работы пользователей с одинаковыми (согласованными) данными общего пользования.*

Основная идея реплицирования заключается в том, что пользователи работают автономно с одинаковыми (общими) данными, растиражированными по локальным базам данных, обеспечивая с учетом отсутствия необходимости передачи и обмена данными по сети максимальную для своих вычислительных установок производительность. Программное обеспечение СУБД для реализации такого подхода соответственно дополняется функциями тиражирования (реплицирования) баз данных, включая тиражирование как самих данных и их структуры, так и системного каталога с информацией о размещении реплик.

При этом, однако, возникают две проблемы обеспечения одного из основополагающих принципов построения и функционирования распределенных систем, а именно — *непрерывности согласованного состояния данных*:

– обеспечение согласованного состояния во всех репликах количества и значений общих данных;

– обеспечение согласованного состояния во всех репликах структуры данных.

– Обеспечение согласованного состояния общих данных, в свою очередь, основывается на реализации одного из *двух принципов:*

– принципа непрерывного размножения обновлений (любое обновление данных в любой реплике должно быть немедленно размножено);

– принципа отложенных обновлений (обновления реплик могут быть отложены до специальной команды или ситуации).

Принцип непрерывного размножения обновлений является основополагающим при построении так называемых «*систем реального времени»,* таких, например, как системы управления воздушным движением, системы бронирования билетов пассажирского транспорта и т. п., где требуется непрерывное и точное соответс-, твие реплик или других растиражированных данных во всех узлах и компонентах подобных распределенных систем.

Реализация принципа *непрерывного размножения обновлений* заключается в том, что *любая транзакция считается успешно завершенной, если она успешно завершена на всех репликах системы.* На практике реализация этого принципа встречает существенные Затруднения, связанные с *тупиками.* Предположим, что на одной вычислительной установке пользователь обновляет данные в своей реплике. На время осуществления транзакции (транзакций) соответствующие записи в базе данных этой реплики ядром локальной СУБД заблокированы от изменения другими пользователями. Вместе с тем транзакция может быть зафиксирована и, следовательно, разблокированы соответствующие данные только тогда, когда данная транзакция послана и также завершена на других репликах системы. Предположим также, что в другой реплике системы, находящейся на другом компьютере сети, в это же время другой пользователь проводит свои обновления (транзакции) с теми же записями, которые, естественно, в этот момент также заблокированы от изменений для других пользователей. Так образуется тупик. Одна транзакция не может быть зафиксирована в своей реплике, потому что заблокированы соответствующие записи в другой реплике. А разблокировка этих записей в другой реплике также невозможна до тех пор, пока не разблокируются соответствующие записи в первой реплике, т. е. когда завершится транзакция в первой реплике. Создается тупиковая ситуация.

Для обнаружения тупиков в реплицированных системах применяются такие же алгоритмы, которые были разработаны в мониторах транзакций централизованных систем «Клиент-сервер».

В целом ряде предметных областей распределенных информационных систем режим реального времени с точки зрения непрерывности согласования данных не требуется. Такие системы автоматизируют те организационно-технологические структуры, в которых информационные процессы не столь динамичны. Если взять, к примеру, автоматизированную информационную систему документооборота, то традиционная «скорость» перемещения и движения служебных документов соответствует рабочему дню или в лучшем случае рабочим часам. В этом случае обновление реплик распределенной информационной системы, если она будет построена на технологии реплицирования,

требуется, скажем, только лишь один раз за каждый рабочий час, или за каждый рабочий день.

Такого рода информационные системы можно строить на основе *принципа отложенных обновлений.* Накопленные в какой-либо реплике изменения данных *специальной командой* пользователя направляются для обновления всех остальных реплик систем. Такая операция называется синхронизацией реплик. Возможность конфликтов и тупиков в этом случае при синхронизации реплик существенно снижается, а немногочисленные подобные конфликтные ситуации легко разрешить организационными мерами.

Решение второй проблемы согласованности данных, а именно — согласованности структуры данных, осуществляется через частичное отступление, как и в системах «Клиент-сервер», от принципа отсутствия центральной установки и основывается на технике «главной» реплики.

Суть этой техники заключается в том, что одна из реплик базы данных системы объявляется главной. При этом изменять структуру базы данных можно только в главной реплике. Эти изменения структуры данных тиражируются на основе принципа отложенных обновлений, т. е. через специальную синхронизацию реплик. Частичность отступления от принципа отсутствия центральной установки заключается в том, что в отличие от чисто централизованных систем, выход из строя главной реплики не влечет сразу гибель всей распределенной системы, так как остальные реплики продолжают функционировать автономно. Более того, на практике СУБД, поддерживающие технологию реплицирования, позволяют пользователю с определенными полномочиями (администратору системы) преобразовать любую реплику в главную и тем самым полностью восстановить работоспособность всей системы.

Процесс синхронизации реплик в современных СУБД. Этот процесс включает обмен только теми данными, которые были изменены или добавлены в разных репликах. С этой целью в системном каталоге базы данных создаются специальные таблицы текущих изменений и организуется система глобальной идентификации (именования) всех объектов распределенной системы, включая раздельное по именование одинаковых объектов в разных репликах. Такой подход несколько увеличивает объем базы данных, но позволяет существенно ограничить транспортные расходы на синхронизацию реплик.

Важным, с точки зрения гибкости и эффективности функционирования распределенных информационных систем, построенных на технологиях реплицирования, является возможность создания так называемых частичных реплик и включения в реплики как *реплицируемых*, так и *нереплицируемых объектов.* Частичной репликой называется *база данных, содержащая ограниченное подмножество записей полной реплики*. Распространенным способом создания частичных реплик является использование фильтров, устанавливаемых для конкретных таблиц полной (главной) реплики. Частичные реплики позволяют решить некоторые проблемы, связанные с разграничением доступа к данным и повышают производительность обработки данных. Так, к примеру, в реплику базы данных для определенного подразделения целесообразно реплицировать только те записи таблицы, которые относятся к данному подразделению, исключив тем самым доступ к другим записям. Техника частичных реплик также снижает затраты на синхронизацию реплик, так как ограничивает количество передаваемых по сети изменений данных.

Возможность включения в реплики объектов базы данных, которые не подлежат репликации, позволяет более гибко и адекватно настроить схему и прочие объекты БД

(запросы, формы и отчеты) на специфику предметной области, особенности ввода данных и решаемые информационные задачи по конкретному элементу распределенной системы.

На рис. 4.3 иллюстрируется подход к организации общей схемы распределенной информационной системы по делопроизводству некоторой организационной структуры на основе технологий репликации данных.

Технологии репликации данных в тех случаях, когда не требуется обеспечивать большие потоки и интенсивность обновляемых в информационной сети данных, являются экономичным решением проблемы создания распределенных информационных систем с элементами централизации по сравнению с использованием дорогостоящих «тяжелых» клиент- серверных систем.

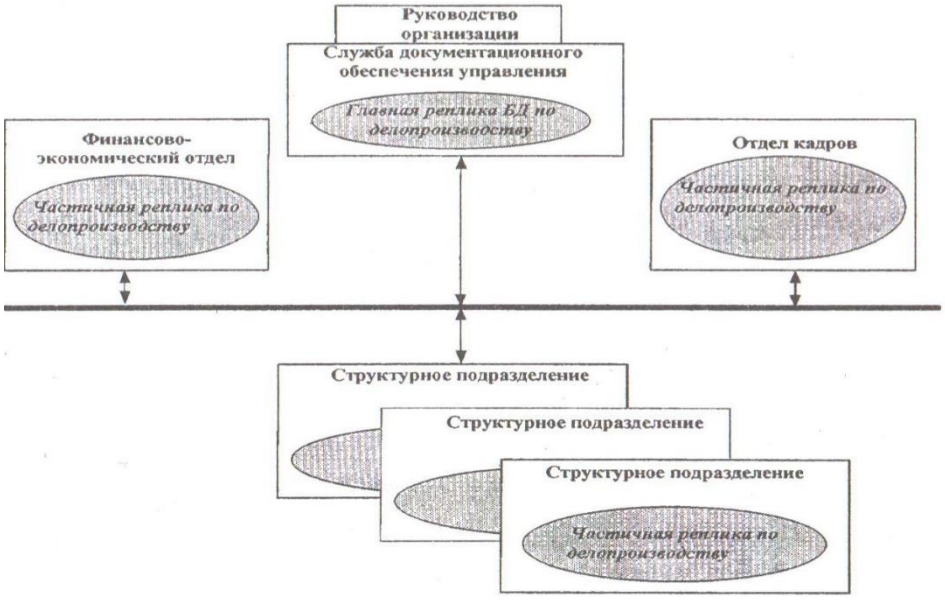


Рис. 4.3. Пример подхода к организации схемы распределенной информационной системы по делопроизводству на основе технологии реплицирования.

На практике для совместной коллективной обработки данных применяются смешанные технологии, включающие элементы объектного связывания данных, репликации и клиент-серверных решений. При этом дополнительно к проблеме логического проектирования, т. е. проектирования логической схемы организации данных (таблицы, поля, ключи, связи, ограничения целостности), добавляется не менее сложная проблема транспортно-технологического проектирования информационных потоков, разграничения доступа и т.д. К сожалению, пока не проработаны теоретико-методологические и инструментальные подходы для автоматизации проектирования распределенных информационных систем с учетом факторов как логики, так и информационно-технологической инфраструктуры предметной ооласти. Тем не менее развитие и все более широкое распространение распределенных информационных систем, определяемое самой распределенной природой информационных потоков и технологий, является основной перспективой развития автоматизированных информационных систем.

Контрольные вопросы:

1. Объясните процесс репликации данных.
2. Объясните процесс создания частичных реплик.
3. Объясните случаи возникновения «тупиковых» ситуаций.
4. Объясните процесс синхронизации реплик

Лекция 14: Стандарты и спецификации по обеспечению безопасности базы данных

План:

1. Стандартизация управления и обмена данными
2. Роль стандартов и спецификаций.

Стандартизация управления и обмена данными

Международная организация стандартизации ISO в рамках SC32 подкомитета JTC1 ("*Data Management* and *Interchange*") разрабатывает стандарты в области управления и обмена данными для локальных и распределенных информационных систем.

В круг вопросов, обсуждаемых SC32, входит рассмотрение моделей взаимодействия для существующих и появляющихся стандартов; *определение* структур и типов данных, семантики применения этих структур и типов; описание стандартов для языков, сервисов и протоколов, используемых для параллельного доступа и изменения данных, для обмена данными, а также для реализации хранения данных; стандартов на методы, языки, сервисы и протоколы, употребляемых для структурирования, организации и регистрации метаданных, а также других информационных ресурсов.

В рамках SC32 функционирует ряд рабочих групп:

WG01 - рабочая группа, специализирующаяся на выработке стандартов для идентификации и спецификации технологии формального описания разрабатываемых бизнес-сценариев и их компонентов, а также других стандартов, используемых в области *электронной коммерции*.

WG02 - рабочая группа, разрабатывающая и развивающая стандарты по спецификации и управлению метаданными, обмену метаданными в различных средах (в Internet, *Intranet* и в других средах). В число наиболее интересных проектов данной группы входят следующие:

* 1.32.16.01.02.00 ISO/*IEC* AWI 20943-2 "Информационные технологии - Применение XML структурированных данных для процедуры регистрации данных" (Information technology - Procedure for Achieving Data *Registry* Content *Consistency* - XML *Structured Data*). Де-факто язык XML уже используется web-серверами как язык описания дескриптора доставки модулей, располагаемых и регистрируемых на сервере;
* 1.32.17.01.00.00 ISO/*IEC* AWI 20944 "Информационные технологии - Сервисы доступа к метаданным" (Information technology - *Metadata*Access Service).

WG03 - рабочая группа, разрабатывающая стандарт языка взаимодействия с базами данных. Круг вопросов, рассматриваемых WG03, включает развитие языка для описания структуры и содержания базы данных в многопользовательских и многосерверных средах. Рассматриваемые спецификации определяют стандартные типы данных, механизмы для создания новых типов данных, включая определения их поведения. Кроме того, рабочая группа занимается вопросами стандартизации интерфейса разрабатываемого языка с

другими языками программирования, а также вопросами стандартизации типов данных и их поведения в рассматриваемом языке с другими языками представления и обработки данных. В число наиболее интересных проектов данной группы входят следующие:

* 1.32.03.05.09.00 ISO/*IEC* CD 9075-9 "ИТ- Язык SQL: Управление внешними данными" (Information technology - Database Languages - SQL - Part 9: Management of External Data (SQL/MED));
* 1.32.03.05.14.00 ISO/*IEC* WD 9075-14 "ИТ- Язык SQL: Взаимодействие SQL и XML" (Information technology - Database Language SQL - Part 14: SQL/XML (for SQL:200n)).

WG04 - рабочая группа, стандартизирующая пакеты абстрактных типов данных для использования в различных прикладных областях.

WG05 - рабочая группа, разрабатывающая стандарты в области взаимодействия приложений и баз данных, в которые включены вопросы удаленного доступа к данным и протоколы передачи данных. Среди наиболее интересных проектов группы отметим следующий:

* 1.32.05.04.00.00 ISO/*IEC* CD 9579 ed 4 "ИТ - Удаленный доступ к данным в SQL" (Information technology - Remote Database Access for SQL: (RDA/SQL). Edition 4).

В последнее время для обмена данными и представления информации все чаще используется язык *XML* (*eXtensible Markup Language*). Этот язык не привязан к какой-либо конкретной платформе или к конкретному производителю. Первая спецификация языка *XML* 1.0 получила статус рекомендации консорциума *W3C* в 1998 году. Далее консорциум *W3C* разработал и опубликовал ряд стандартов, связанных с *XML* (*ExtensibleMarkup Language* (*XML*) Version 1.0 (*Edition* 2): http://www.w3.org/TR/REC-xml), включая стандарт на механизм связывания XLink и XPointer, стандарт синтаксиса схемы, описывающей набор данных (Recommendation) *XML Schema Part* 1: Structures, 2 May, 2001, (Recommendation) *XMLSchema Part* 2: *Datatypes*, 2 May, 2001: http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/,http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/), спецификации *по* определению и использованию пространства имен (Namespaces in *XML*, 14 January, 1999:http://www.w3.org/TR/REC-xml-names).

Консорциум *W3C*, продолжая работу над стандартизацией *XML*, опубликовал рекомендации *по DOM XML* - объектной модели документа, представляющей *XML*-документ в виде объекта.

Вопросами стандартизации *XML* также частично занимается *OASIS* (Организация *по* продвижению стандартов структурирования информации - Organization for the Advancement of Structured *Information* Standards: http://www.oasis-open.org/).

Инженерной группой *IETF* был разработан стандарт *SOAP* (Simple *Object* Access *Protocol*), использующий язык *XML*, как язык для обмена данными. Фактически *SOAP* позволяет посредством применения *XML* реализовывать межплатформенный *доступ* к данным, связывая воедино применение таких технологий, как *CORBA, EJB* и *COM*.

Разрабатываемый в настоящее время консорциумом *W3C* стандарт XQL (*XML* Query *Language*: http://www.w3.org/TR/2001/WD-xquery-20011220/) включает вопросы, связанные со спецификацией методов выполнения запросов к набору *XML*-документов.

В рамках WG3 32 подкомитета JTC1 также разрабатывается стандарт, связанный с использованием языка *XML*: "ИТ - Язык *SQL* - часть 14: Спецификация *SQL*/*XML*" (*Information technology - Database* languages - *SQL - Part* 14: *XML*-Related Specifications (*SQL*/*XML*)). Разрабатываемый стандарт рассматривает *механизмы* преобразования данных, описываемых средствами языка *SQL*, в данные, представляемые языком *XML*, и обратно, включая алгоритмы соответствия *SQL*-таблиц типам данных *XML*-схемы, соответствия *SQL*-значений значениям *XML*, а также приводит описание *XML*-схемы для *SQL*/*XML*. *По* этому стандарту опубликован Final Committee *Draft ISO*/*IEC* FCD 9075-14.

Одним из наиболее значительных стандартов, разрабатываемых в настоящее время и предназначаемых для обмена данными, является стандарт *ISO*/*IEC* WD 9579, Fourth *Edition* "ИТ - удаленный *доступ* к базам данных для *SQL*" (*Information Technology - Remote Database* Access for *SQL with* Extended *Security*).

Рассматриваемый стандарт *RDA*/*SQL* базируется на уже существующих следующих стандартах *IETF* (http://www.internic.net):

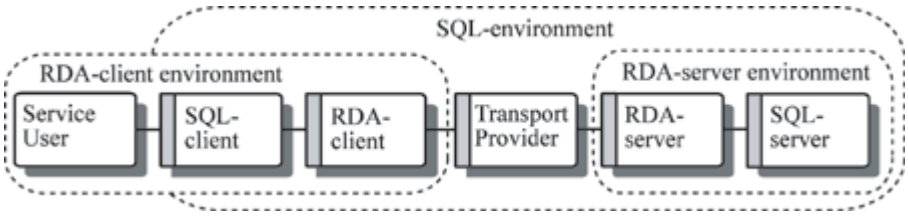
- RFC 791 Internet Protocol.
- RFC 793 Transmission Control Protocol.
- RFC 819 The Domain Naming Convention for Internet User Applications.
- RFC 1122 Requirements for Internet Hosts - Communication Layers.
- RFC 1123 Requirements for Internet Hosts - Application and Support.
- RFC 2246 The TLS Protocol.

RDA/*SQL* может быть использован для реализации удаленного доступа к *СУБД*, соответствующей стандарту *ISO*/*IEC* 9075 (*Database LanguageSQL*).

Стандарт *RDA*/*SQL* описывает модель для удаленного взаимодействия *SQL*-клиента с одним или несколькими *SQL*-серверами посредством коммуникационных протоколов.

RDA/*SQL* устанавливает соответствие *RDA*-протокола стандартным протоколам *TCP/IP* и *TLS* (*Transport Layer Security*), вводит понятия *RDA*-сообщения, *RDA*-оператора, *RDA*-протокола и *RDA*-передачи.

В стандарте определяется *RDA*-модель среды *SQL* (рис 1.1) и функциональные компоненты, составляющие среду *RDA*-клиента и среду *RDA*-сервера.



RDA-модель среды SQL

RDA-модель определяет провайдера транспортного уровня, реализующего взаимодействие между *RDA*-клиентом и *RDA*-сервером.

Стандарт *ISO*/*IEC* 9075-3 (*SQL/CLI*) описывает результирующий набор, определяемый на стороне сервера, а стандарт *RDA*/*SQL* описывает *RDA-операторы*, предназначаемые для взаимодействия с результирующим набором и соответствующие вызовам *SQL/CLI*. Наряду с *RDA*-операторами, данный стандарт вводит коды атрибутов, используемые *RDA*. К настоящему времени рабочей группой WR5 опубликована 4-я редакция разрабатываемого стандарта *RDA*/*SQL*.

Роль стандартов и спецификаций. Наиболее важные стандарты и спецификации в области информационной безопасности

Специалистам в области *информационной безопасности* (ИБ) сегодня почти невозможно обойтись без знаний соответствующих *стандартов* и *спецификаций*. На то имеется несколько причин.

Формальная состоит в том, что необходимость следования некоторым *стандартам* (например, криптографическим и/или Руководящим документам) закреплена законодательно. Однако наиболее убедительны содержательные причины. Во-первых, *стандарты* и *спецификации* - одна из форм накопления знаний, прежде всего о процедурном и программно-техническом уровнях ИБ. В них зафиксированы апробированные, высококачественные решения и методологии, разработанные наиболее квалифицированными специалистами. Во-вторых, и те, и другие являются основным средством обеспечения взаимной совместимости аппаратно-программных систем и их компонентов, причем в *Internet*-сообществе это средство действительно работает, и весьма эффективно.

- ***стандарт*** - документ, в котором в целях добровольного многократного использования устанавливаются характеристики продукции, правила осуществления и характеристики процессов производства, эксплуатации, хранения, перевозки, реализации и утилизации, выполнения работ или оказания услуг. *Стандарт* также может содержать требования к терминологии, символике, упаковке, маркировке или этикеткам и правилам их нанесения;
- ***стандартизация*** - деятельность по установлению правил и характеристик в целях их добровольного многократного использования, направленная на достижение упорядоченности в сферах производства и обращения продукции и повышение конкурентоспособности продукции, работ или услуг.

Примечательно также, что в число принципов *стандартизации*, провозглашенных в упомянутого закона, входит принцип применения международного *стандарта* как основы разработки национального, за исключением случаев, если "такое применение признано невозможным вследствие несоответствия требований международных *стандартов* климатическим и географическим особенностям, техническим и (или) технологическим особенностям или по иным основаниям, либо, в соответствии с установленными процедурами, выступала против принятия международного *стандарта* или отдельного его положения". С практической точки зрения, количество *стандартов* и *спецификаций* (международных, национальных, отраслевых и т.п.) в области *информационной безопасности* бесконечно. В курсе рассматриваются наиболее важные из них, *знание* которых необходимо всем или почти всем разработчикам и оценщикам защитных средств, многим сетевым и системным администраторам, руководителям соответствующих подразделений, пользователям.

Отбор проводился таким образом, чтобы охватить различные аспекты *информационной безопасности*, разные виды и конфигурации *информационных систем (ИС)*, предоставить полезные сведения для самых разнообразных групп целевой аудитории.

На верхнем уровне можно выделить две существенно отличающиеся друг от друга группы *стандартов* и спецификаций:

- оценочные *стандарты*, предназначенные для оценки и классификации *информационных систем* и средств защиты по требованиям безопасности;
- *спецификации*, регламентирующие различные аспекты реализации и использования средств и методов защиты.

Эти группы, разумеется, не конфликтуют, а дополняют друг друга. Оценочные *стандарты* описывают важнейшие, с точки зрения *информационной безопасности*, понятия и аспекты ИС, играя роль организационных и

архитектурных *спецификаций*. Другие *спецификации* определяют, как именно строить ИС предписанной архитектуры и выполнять организационные требования.

Технические *спецификации*, применимые к современным распределенным ИС, создаются, главным образом, "Тематической группой по технологии *Internet*" (*Internet Engineering Task Force, IETF*) и ее подразделением - рабочей группой по безопасности. Ядром рассматриваемых технических спецификаций служат документы по безопасности на *IP*-уровне (IPsec). Кроме этого, анализируется защита на *транспортном уровне* (*Transport Layer Security, TLS*), а также на уровне приложений (спецификации *GSS-API,* Kerberos). Необходимо отметить, что *Internet-сообщество* уделяет должное внимание административному и процедурному уровням безопасности ("Руководство по информационной безопасности предприятия", "Как выбирать поставщика Интернет-услуг", "Как реагировать на нарушения информационной безопасности").

В вопросах сетевой безопасности невозможно разобраться без освоения *спецификаций* X.800 "*Архитектура безопасности* для взаимодействия открытых систем", X.500 "*Служба директорий*: обзор концепций, моделей и сервисов" и X.509 "*Служба директорий*: каркасы сертификатов открытых ключей и атрибутов".

Британский стандарт BS 7799 "*Управление информационной безопасностью*. Практические правила", полезный для руководителей организаций и лиц, отвечающих за *информационную безопасность*, без сколько-нибудь существенных изменений воспроизведен в международном стандарте ISO/IEC 17799.

Таков, на наш взгляд, "стандартный *минимум*", которым должны активно владеть все действующие специалисты в области *информационной безопасности*.

Контрольные вопросы:

1. Что такое профили защиты систем управления базами данных?
2. Что такое аутентификационные пакеты?
3. Перечислите способы конфигурирования системы управления базами данных.
4. Перечислите угрозы на безопасность объекта оценки и меры, противостоящие этим угрозам.
5. Перечислите нарушителей профиля защиты в системе управления базами данных.
6. Какие задачи приведены в документе «Профиль защиты системы управления базами данных»?

Лекция 15: Архитектура и принцип функционирования подсистемы безопасности базы данных

План:

1. Идентификация и проверка подлинности пользователей
2. Привилегии безопасности позволяют выполнять административные действия

Системы управления базами данных, в особенности реляционные СУБД, стали доминирующим инструментом хранения больших массивов информации. Сколько-нибудь развитые информационные приложения полагаются не на файловые структуры операционных систем, а на многопользовательские СУБД, выполненные в технологии клиент/сервер. В связи с этим, обеспечение информационной безопасности СУБД, и в первую очередь, их серверных компонентов, приобретает решающее значение для безопасности организации в целом. Как мы уже отметили, для СУБД важны все три основных аспекта информационной безопасности - конфиденциальность, целостность и доступность. Общая идея защиты баз данных состоит в следовании рекомендациям,

сформулированным для класса безопасности С2 в «Критериях оценки надежных компьютерных систем». В принципе некоторые СУБД предлагают дополнения, характерные для класса В1, однако практическое применение подобных дополнений имеет смысл, только если все компоненты информационной структуры организации соответствуют категории безопасности В. Достичь этого непросто и с технической, и с финансовой точек зрения. Следует, кроме того, учитывать два обстоятельства. Во-первых, для подавляющего большинства коммерческих организаций класс безопасности С2 достаточен. Во-вторых, более защищенные версии отстают по содержательным возможностям от обычных «собратьев», так что поборники секретности по сути обречены на использование морально устаревших (хотя и тщательно проверенных) продуктов со всеми вытекающими последствиями в плане сопровождения.

Идентификация и проверка подлинности пользователей

Обычно в СУБД для идентификации и проверки подлинности пользователей применяются либо соответствующие механизмы операционной системы, либо SQL-оператор CONNECT. Например, в случае СУБД Oracle оператор CONNECT имеет следующий вид:

– CONNECT пользователь [/пароль] [@база данныхя].

Так или иначе, в момент начала сеанса работы с сервером баз данных, пользователь идентифицируется своим именем, а средством аутентификации служит пароль. Детали этого процесса определяются реализацией клиентской части приложения.

Некоторые операционные системы, такие как UNIX, позволяют во время запуска программы менять действующий идентификатор пользователя. Приложение, работающее с базой данных, как правило, имеет привилегии, значительно превосходящие привилегии обычных пользователей. Естественно, что при этом приложение предоставляет тщательно продуманный, строго фиксированный набор возможностей. Если пользователь сумеет тем или иным способом завершить приложение, но сохранить подключение к серверу баз данных, ему станут доступны по существу любые действия с данными.

Управление доступом. Для иллюстрации вопросов, связанных с управлением доступом, будет использоваться СУБД INGRES.

Обычно в СУБД применяется произвольное управление доступом, когда владелец объекта передает права доступа к нему (чаще говорят - привилегии) по своему усмотрению. Привилегии могут передаваться субъектам (отдельным пользователям), группам, ролям или всем пользователям.

Привилегии роли имеют приоритет над привилегиями пользователей и групп. Иными словами, пользователю как субъекту не обязательно иметь права доступа к объектам, обрабатываемым приложениям с определенной ролью. Отметим, что в СУБД Oracle под ролью понимается набор привилегий. Такие роли служат средством структуризации привилегий и облегчают их модификацию.

Совокупность всех пользователей именуется как PUBLIC. Придание привилегий PUBLIC - удобный способ задать подразумеваемые права доступа. Поручать администрирование различных баз данных разным людям имеет смысл только тогда, когда эти базы независимы и по отношению к ним не придется проводить согласованную политику выделения привилегий или резервного копирования. В таком случае каждый из администраторов будет знать ровно столько, сколько необходимо.

Можно провести аналогию между пользователем INGRES и администраторами баз данных с одной стороны, и супер пользователем операционной системы (root в случае ОС UNIX) и служебными пользователями (в ОС UNIX это могут быть bin, lp, uucp и т.д.) с другой стороны. Введение служебных пользователей позволяет администрировать функциональные подсистемы, не получая привилегий супер пользователя. Точно так же информацию, хранящуюся на сервере баз данных, можно разделить на отсеки, так что

компрометация администратора одного отсека не означает обязательной компрометации другого.

Виды привилегий. Привилегии в СУБД можно подразделить на две категории: привилегии безопасности и привилегии доступа.

Привилегии безопасности всегда выделяются конкретному пользователю во время его создания (оператором CREATE USER) или изменения характеристик (оператором ALTER USER). Таких привилегий пять:

– security - право управлять безопасностью СУБД и отслеживать действия пользователей. Пользователь с этой привилегией может подключаться к любой базе данных, создавать, удалять и изменять характеристики пользователей, групп и ролей, передавать права на доступ к базам данным другим пользователям, управлять записью регистрационной информации, отслеживать запросы других пользователей и, наконец, запускать INGRES-команды от имени других пользователей. Привилегия security необходима администратору сервера баз данных, а также лицу, персонально отвечающему за информационную безопасность. Передача этой привилегии другим пользователям (например, администраторам баз данных) увеличивает число потенциально слабых мест в защите сервера баз данных;

– createdb - право на создание и удаление баз данных. Этой привилегией, помимо администратора сервера, должны обладать пользователи, которым отводится роль администраторов отдельных баз данных;

– operator - право на выполнение действий, которые традиционно относят к компетенции оператора. Имеются в виду запуск и остановка сервера, сохранение и восстановление информации. Помимо администраторов сервера и баз данных этой привилегией целесообразно наделить также администратора операционной системы;

– maintainjocations - право на управление расположением баз администратора сервера баз данных и операционной системы;

– trace - право на изменение состояния флагов отладочной трассировки. Данная привилегия полезна администратору сервера баз данных и другим знающим пользователям при анализе сложных, непонятных ситуаций.

Привилегии безопасности позволяют выполнять административные действия.

Привилегии доступа, в соответствии с названием, определяют права доступа субъектов к определенным объектам. Привилегии доступа выделяются пользователям, группам, ролям или всем посредством оператора GRANT и изымаются с помощью оператора REVOKE. Эти привилегии, как правило, присваивает владелец соответствующих объектов (он же - администратор базы данных) или обладатель привилегии security (обычно администратор сервера баз данных).

Прежде чем присваивать привилегии группам и ролям, их (группы и роли) необходимо создать с помощью операторов CREATE GROUP и CREATE ROLE.

Для изменения состава группы служит оператор ALTER GROUP.

Оператор DROP GROUP позволяет удалять группы, правда, только после того, как опустошен список членов группы.

Оператор ALTER ROLE служит для изменения паролей ролей, а DROP ROLE - для удаления ролей.

Напомним, что создавать и удалять именованные носители привилегий, а также изменять их характеристики может лишь пользователь с привилегией security. При совершении подобных действий необходимо иметь подключение к базе данных iidbdb, в которой хранятся сведения о субъектах и их привилегиях.

Привилегии доступа можно подразделить в соответствии с видами объектов, к которым они относятся. В СУБД INGRES таких видов пять:

– таблицы и представления,

– процедуры,

- базы данных,
- сервер баз данных,
- события.

Присваивание привилегий доступа производится с помощью оператора GRANT. В самом общем виде оператор GRANT имеет следующий формат:

- GRANT привилегии,
- ON объекты,
- TO кому.

Применительно к таблицам и представлениям можно управлять следующими правами доступа:

- SELECT - право на выборку данных;
- INSERT - право на добавление данных;
- DELETE - право на удаление данных;
- UPDATE - право на обновление данных (можно указать определенные столбцы, разрешенные для обновления);
- REFERENCES- право на использование внешних ключей, ссылающихся на данную таблицу (можно указать определенные столбцы).

По умолчанию пользователь не имеет никаких прав доступа к таблицам и представлениям - их необходимо передать с помощью операторов GRANT.

По отношению к процедуре можно предоставить право на выполнение. При этом не нужно заботиться о выделении прав доступа к объектам, обрабатываемым процедурой - их наличие не обязательно. Таким образом, процедуры баз данных являются удобным средством предоставления контролируемого доступа для выполнения строго определенных действий над данными.

Права доступа к базе данных как к единому целому, может предоставлять ее администратор или пользователь с привилегией security. Эти «права» на самом деле устанавливают ряд ограничений на использование базы данных, то есть по сути являются запретительными. Имеется в виду ограничение на число операций ввода/вывода или число строк, возвращаемых одним запросом, ограничение права создания таблиц и процедур и т.п. По умолчанию пользователь не стесняется количественными лимитами и получает право на создание объектов в базе.

Отметим, что при создании базы данных указывается ее статус - общая или личная. Это влияет на подразумеваемые права доступа к базе. По умолчанию право на подключение к общей базе предоставляется всем. Право на подключение к личной базе нужно передавать явным образом. Право на подключение необходимо для выполнения всех прочих операций с базой и содержащимися в ней объектами.

Привилегии (которые в данном случае точнее было бы назвать ограничениями) QUERY_IO_LIMIT и QUERY_ROW_LIMIT проверяются на основании оценок, выданных оптимизатором запросов. Если оптимизатор предсказывает, что запрос превысит отведенный лимит числа операций ввода вывода или возвращаемых строк, он (запрос) отвергается. Наложение подобных количественных ограничений препятствует монополизации сервера одним клиентом и может использоваться как один из инструментов поддержания высокой готовности.

Для отмены привилегий, выданных ранее (как разрешительных, так и запретительных), служит оператор REVOKE.

Использование представлений для управления доступом. СУБД предоставляют специфическое средство управления доступом - представления. Представления позволяют сделать видимыми для субъектов определенные столбцы базовых таблиц (реализовать проекцию) или отобрать определенные строки (реализовать селекцию). Не предоставляя субъектам прав доступа к базовым таблицам и сконструировав подходящие представления,

администратор базы данных защитит таблицы от несанкционированного доступа и снабдит каждого пользователя своим видением базы данных, когда недоступные объекты как бы не существуют.

Приведем пример создания представления, содержащего два столбца исходной таблицы и включающего в себя только строга с определенным значением одного из столбцов:

– CREATE VIEW empview AS SELECT name, dept FROM employee WHERE dept = ’shoe’;

– Предоставим всем право на выборку из этого представления: GRANT SELECT ON empview TO PUBLIC;

Субъекты, осуществляющие доступ к представлению empview, могут пытаться запросить сведения об отделах, отличных от shoe, например:

– SELECT *

– FROM empview WHERE dept = ’toy’; но в ответ просто получат результат из нуля строк, а не код ответа, свидетельствующий о нарушении прав доступа. Это принципиально важно, так как лишает злоумышленника возможности получить список отделов косвенным образом, анализируя коды ответов, возвращаемые после обработки SQL-запросов.

Иерархия прав доступа. Оператор GRANT и другие средства управления доступом СУБД позволяют реализовать следующие виды ограничения доступа:

– операционные ограничения (за счет прав доступа SELECT, INSERT, UPDATE, DELETE, применимых ко всем или только? некоторым столбцам таблицы);
– ограничения по значениям (за счет механизма представлений);
– ограничения на ресурсы (за счет привилегий доступа к базам данных).

При обработке запроса СУБД сначала проверяет права доступа к объектам. Если операционные ограничения оказываются нарушенными, запрос отвергается с выдачей соответствующей диагностики. Нарушение ограничений на значения влияет только на количество результирующих строк; никакой диагностики при этом не выдается. Наконец, после учета двух предыдущих ограничений, запрос поступает на обработку оптимизатору. Если тот обнаружит превышение ограничений на ресурсы, запрос будет отвергнут с выдачей соответствующей диагностики.

На иерархию привилегий можно посмотреть и с другой точки зрения. Каждый пользователь, помимо собственных, имеет привилегии PUBLIC. Кроме этого, он может входить в различные группы и запускать приложения с определенными ролями. Как соотносятся между собой права, предоставленные различным именованным носителям привилегий?

Иерархия авторизации выглядит для СУБД INGRES следующим образом:

– роль (высший приоритет),
– пользователь,
– группа,
– PUBLIC (низший приоритет).

Для каждого объекта, к которому осуществляется доступ, INGRES пытается отыскать в иерархии привилегию, относящуюся к запрашиваемому виду доступа (SELECT, EXECUTE и т.п.). Например, при попытке доступа к таблице с целью обновления, INGRES проверяет привилегии роли, пользователя, группы и всех пользователей. Если хотя бы на одном уровне иерархии привилегия UPDATE имеется, запрос передается для дальнейшей обработки. В противном случае используется подразумеваемое право доступа, которое предписывает отвергнуть запрос.

Рассмотрим подробнее трактовку ограничений на ресурсы. Пусть, например, на всех четырех уровнях иерархии специфицированы свои ограничения на число результирующих строк запроса (привилегия QUERYROWLIMIT):

– роль 1700

- пользователь 1500
- группа2000
- PUBLIC 1000

Если пользователь в момент начала сеанса работы с СУБД задал и роль, и группу, будет использовано ограничение, накладываемое ролью 1700. Если бы привилегия QUERY_ROW_LIMIT для роли отсутствовала, или пользователь не задал роль в начале сеанса работы, пользователь смог бы получать результаты не более чем из 1500 строк и т.п. Если бы привилегия QUERY ROW LIMIT вообще не была специфицирована ни на одном уровне иерархии, СУБД воспользовалась бы подразумеваемым значением, которое в данном случае означает отсутствие ограничений на число результирующих строк.

Обычно используемая роль и группа задаются, соответственно, как аргументы опций -R и -G в командной строке запуска приложения.

Пример:

QBF -Gaccounting company_db

Если опция -G отсутствует, применяется подразумеваемая группа пользователя, если таковая имеется.

Наконец, если в командной строке sql задана опция:

- и пользователь, то в число проверяемых входят также привилегии указанного пользователя.

Метки безопасности и принудительный контроль доступа. Выше были описаны средства произвольного управления доступом, характерные для уровня безопасности С. Как уже указывалось, они в принципе достаточны для подавляющего большинства коммерческих приложений. Тем не менее, они не решают одной весьма. важной задачи - задачи слежения за передачей информации. Средства произвольного управления доступом не могут помешать авторизованному пользователю законным образом получить секретную информацию и затем сделать ее доступной для других, неавторизованных пользователей. Нетрудно понять, почему это так. При произвольном управлении доступом привилегии существуют отдельно от данных (в случае реляционных СУБД - отдельно от строк реляционных таблиц). В результате данные оказываются «обезличенными», и ничто не мешает передать их кому угодно даже средствами самой СУБД.

В «Критериях оценки надежных компьютерных систем», применительно к системам уровня безопасности В, описан механизм меток безопасности, реализованный в версии INGRES/Enhanced Security (INGRES с повышенной безопасностью). Применять эту версию на практике имеет смысл только в сочетании с операционной системой и другими программными компонентами того же уровня безопасности. Тем не менее, рассмотрение реализации меточной безопасности в СУБД INGRES интересно с познавательной точки зрения, а сам подход, основанный на разделении данных по уровням секретности и категориям доступа, может оказаться полезным при проектировании системы привилегий многочисленных пользователей по отношению к большим массивам данных.

В СУБД INGRES/Enhanced Security к каждой реляционной таблице неявно добавляется столбец, содержащий метки безопасности строк таблицы. Метка безопасности состоит из трех компонентов:

* уровень секретности. Смысл этого компонента зависит от приложения. В частности, возможен традиционный спектр уровней от «совершенно секретно» до «несекретно»;
* категории. Понятие “категории” позволяет разделить данные на «отсеки» и тем самым повысить надежность системы безопасности. В коммерческих приложениях категориями могут служить «финансы», «кадры», «материальные ценности» и т.п.; область. Является дополнительным средством деления информации на отсеки. На практике компонент «область» может действительно иметь географический смысл, обозначая, например, страну, к которой относятся данные.

Каждый пользователь СУБД INGRES/Enhanced Security характеризуется степенью благонадежности, которая также определяется меткой безопасности, присвоенной данному пользователю. Пользователь может получить доступ к данным, если степень его благонадежности удовлетворяет требованиям соответствующей метки безопасности: Более точно:

- уровень секретности пользователя должен быть не ниже уровня секретности данных;
- набор категорий, заданных в метке безопасности данных, должен целиком содержаться в метке безопасности пользователя;
- набор областей, заданных в метке безопасности пользователя, должен целиком содержаться в метке безопасности данных.

Специальная привилегия DOWNGRADE, позволяет изменять метки безопасности, ассоциированные с данными. Подобная возможность необходима, например, для коррекции меток, по тем или иным причинам оказавшихся неправильными.

Представляется естественным, что СУБД INGRES/Enhanced Security допускает не только скрытое, но и явное включение меток безопасности в реляционные таблицы. Появился новый тип данных, security label, поддерживающий соответствующие операции сравнения.

INGRES/Enhanced Security - первая СУБД, получившая сертификат, эквивалентный аттестации на класс безопасности B1. Вероятно, метки безопасности постепенно войдут в стандартный репертуар систем управления базами данных.

Поддержание целостности данных в СУБД. Для коммерческих организаций обеспечение целостности данных по крайней мере не менее важно, чем обеспечение конфиденциальности. Конечно, неприятно, когда кто-то подглядывает за суммами на счетах клиентов, но гораздо хуже, когда в процессе перевода денег со счета на счет часть суммы исчезает в неизвестном направлении.

Известно, что главными врагами баз данных являются не внешние злоумышленники, а ошибки оборудования, администраторов, прикладных программ и пользователей.

С точки зрения пользователя СУБД, основными средствами поддержания целостности данных являются ограничения и правила.

Ограничения. Ограничения могут относиться к таблицам или отдельным столбцам. Ограничения на столбцы задаются при создании таблицы, в операторах CREATE TABLE

Табличные ограничения относятся к группе столбцов и могут задаваться как при создании таблицы, так и позже, посредством оператора ALTER TABLE.

Следующий пример содержит именованное ограничение, связывающее значения в двух столбцах:

- CREATE TABLE dept dname char (10), budget money, expenses money,
- CONSTRAINT check_amount CHECK (budget > 0 and expenses <= budget));
- {Бюджет должен быть положительным, а расходы не должны выходить за рамки бюджета}.

Ссылочные ограничения отвечают за целостность связей между таблицами. Подобное ограничение требует, чтобы каждому значению в столбце или группе столбцов одной таблицы соответствовало ровно одно значение в другой таблице. Название ограничения объясняется тем, что такие значения играют роль ссылок между таблицами в реляционной модели.

Приведем пример ссылочного ограничения:

CREATE TABLE emp ename char (10),

edept char(10) references dept(dname);

{

Ни один работник не должен числиться в неизвестном отделе}. Ограничения всех видов накладываются владельцем таблицы и влияют на исход последующих операций с

данными. Перед завершением выполнения SQL-оператора производится проверка имеющихся ограничений. При обнаружении нарушений СУБД сигнализирует о ненормальном завершении и аннулирует внесенные оператором изменения.

Отметим, что для наложения ссылочного ограничения необходимо обладать привилегией REFERENCES по отношению к таблице, на которую делается ссылка (dept в примере выше).

Ограничения можно не только накладывать, но и отменять. При этом между ограничениями могут существовать зависимости, и отмена одного из них может потребовать ликвидации других (ссылочных) ограничений, зависящих от первоначального. Рассмотрим следующий пример:

- CREATE TABLE dept name char(10) NOT NULL, location char(20),
- CONSTRAINT dept_unique UNIQUE(name));
- CREATE TABLE emp name char(10), salary decimal(10,2),
- edept char(10) CONSTRAINT empref REFERENCES dept(name));

Если требуется удалить ограничение dept_unique, можно воспользоваться следующим оператором:

- ALTER TABLE dept
- DROP CONSTRAINT deptunique cascade;

Слово cascade означает, что следует удалить также все ограничения, прямо или косвенно зависящие от dept unique. В данном случае будет изъято ограничение empref. Если вместо cascade указать restrict, то есть сделать попытку удалить только ограничение dept unique, СУБД зафиксирует ошибку. Тем самым обеспечивается целостность системы ограничений.

В СУБД INGRES делается попытка примирить контроль ограничений и эффективность функционирования. При массовом копировании данных контроль ограничений отключается. Это значит, что необходимо дополнять копирование запуском процедуры глобальной проверки целостности.

Правила. Правила позволяют вызывать выполнение заданных действий при определенных изменениях базы данных. Обычно действие - это вызов процедуры. Правила ассоциируются с таблицами и срабатывают при изменении этих таблиц.

В отличие от ограничений, которые являются лишь средством контроля относительно простых условий, правила позволяют проверять и поддерживать сколь угодно сложные соотношения между элементами данных в базе. Как и в случае ограничений, проверка правил отключается при массовых операциях копирования. Администратор базы данных может также явным образом отменить проверку правил, воспользовавшись оператором:

SET NORULES;

Оператор

SET RULES

позволит затем восстановить работу механизма правил. По умолчанию этот механизм включен.

Для удаления правил служит оператор:

DROP RULE - правило;

СУБД обеспечивает автоматическое удаление правил в тех случаях, когда удаляется соответствующая таблица. Тем самым поддерживается целостность системы таблиц и правил.

В контексте информационной безопасности важно отметить, что создать правило, ассоциируемое с таблицей, может владелец этой таблицы, имеющий право на выполнение соответствующей процедуры. Пользователь, действия которого вызывают срабатывание правила, должен обладать лишь необходимыми правами доступа к таблице. Тем самым правила неявно расширяют привилегии пользователей. Подобные расширения нуждаются в строгом административном контроле, поскольку даже незначительное изменение правила

или ассоциированной процедуры может кардинально повлиять на защищенность данных. Ошибка же в сложной системе правил вообще чревата непредсказуемыми последствиями.

Средства поддержания высокой готовности. В коммерческих приложениях высокая готовность аппаратно-программных комплексов является важнейшим фактором. Применительно к СУБД средства поддержания высокой готовности должны обеспечивать нейтрализацию аппаратных отказов, особенно касающихся дисков, а также восстановление после ошибок обслуживающего персонала или прикладных программ.

Подобные средства должны с самого начала закладываться в архитектуру комплекса. Например, необходимо использовать тот или иной вид избыточных дисковых массивов. Конечно, это сделает аппаратно-программное решение более дорогим, но зато убережет от возможных убытков во время эксплуатации.

Кластерная организация сервера баз данных. Обычно кластер содержит также несколько дисковых подсистем, совместно используемых узлами-компьютерами, и избыточные связи между компонентами. С внешней точки зрения кластер выглядит как единое целое, а наличие нескольких узлов способствует повышению производительности и устойчивости к отказам.

Тиражирование данных. В контексте информационной безопасности тиражирование можно рассматривать как средство повышения доступности данных. Стала легендой история про бакалейщика из Сан-Франциско, который после разрушительного землетрясения восстановил свою базу данных за 16 минут, перекачав из другого города предварительно протиражированную информацию.

Развитые возможности тиражирования предоставляет СУБД INGRES. В Informix OnLine-DS 7.1 поддерживается модель тиражирования, состоящая в полном отображении данных с основного сервера на вторичные.

В конфигурации серверов Informix OnLine-DS с тиражированием выделяется один основной и ряд вторичных серверов. На основном сервере выполняется и чтение, и обновление данных, а все изменения передаются на вторичные серверы, доступные только на чтение (рис. 5.1). В случае отказа основного сервера вторичный автоматически или вручную переводится в режим доступа на чтение и запись (рис. 5.2). Прозрачное перенаправление клиентов при отказе основного сервера не поддерживается, но оно может быть реализовано в рамках приложений.

После восстановления основного сервера возможен сценарий, при котором этот сервер становится вторичным, а бывшему вторичному, который уже функционирует в режиме чтения-записи, придается статус основного; клиенты, которые подключены к нему, продолжают работу. Таким образом, обеспечивается непрерывная доступность данных.

Тиражирование осуществляется путем передачи информации из журнала транзакций (логического журнала) в буфер тиражирования основного сервера, откуда она пересылается в буфер тиражирования вторичного сервера. Такая пересылка может происходить либо в синхронном, либо в асинхронном режиме. Синхронный режим гарантирует полную согласованность баз данных - ни одна транзакция, зафиксированная на основном сервере, не останется незафиксированной на вторичном, даже в случае сбоя основного сервера. Асинхронный режим не обеспечивает абсолютной согласованности, но улучшает рабочие характеристики системы.

Побочный положительный эффект тиражирования - возможность вынести преимущественно на вторичный сервер ресурсоемкие приложения поддержки принятия решений. В этом случае, они могут выполняться с максимальным использованием средств параллельной обработки, не подавляя приложений оперативной обработки транзакций, сосредоточенных на основном сервере. Это также можно рассматривать как фактор повышения доступности данных.

Защита коммуникаций между сервером и клиентами. Проблема защиты коммуникаций между сервером и клиентами не является специфичной для СУБД, она присуща всем распределенным системам. Вполне естественно, что и решения здесь ищутся

общие, такие, например, как в распределенной вычислительной среде (Distributed Computing Environment, DCE) концерна OSF. Разработчикам СУБД остается «погрузить» свои программные продукты в эту среду, что и сделала компания Informix, реализовав Informix- DCE/Net.

Рис. 5.1. Тиражирование. Основной сервер доступен на чтение и запись, вторичный - только на чтение.

Рис. 5.2. Когда основной сервер выходит из строя, вторичный переводится в режим доступа: и на чтение, и на запись.

Informix-DCE/Net открывает доступ к сервисам DCE для всех инструментальных средств Informix, а также любых приложений или инструментальных комплексов от независимых поставщиков, которые используют интерфейс ODBC (рис. 5.3).

Ключевым компонентом в реализации взаимодействий клиент-сервер в среде DCE является сервис безопасности. Основные функции, предоставляемые этим сервисом, - аутентификация,, реализуемая средствами Kerberos, авторизация (проверка полномочий) и шифрование.

Informix-DCE/Net использует все средства обеспечения безопасности, имеющиеся в DCE. Например, для каждого приложения клиент-сервер администратор может задать один из пяти уровней защиты:

– защита пересылаемых данных только при установлении соединения клиента с сервером;

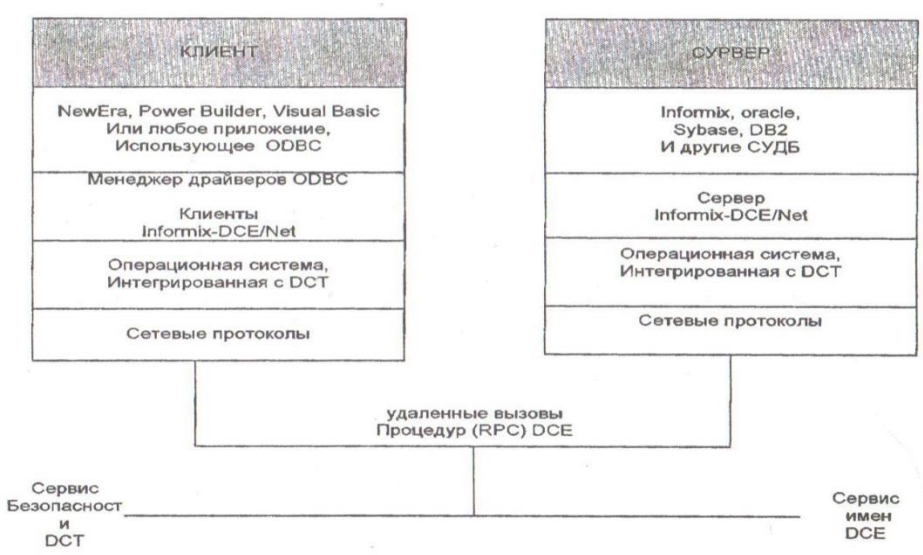


Рис. 5.3. Конфигурация прикладной или инструментальной среды клиент-сервер, использующей Informix-DCE/Net.

– защита данных только на начальном этапе выполнения удаленного вызова процедуры, когда сервер впервые получает запрос;

– подтверждение подлинности источника данных. Проверяется, что все поступающие на сервер данные получены от определенного клиента;

– подтверждение подлинности источника и целостности данных. Проверяется, что отправленные данные не были изменены; подтверждение подлинности источника, целостности и конфиденциальности данных. Выполняются проверки, предусмотренные на предыдущем уровне и осуществляется шифрование всех пересылаемых данных.

Сервис аутентификации DCE, поддерживаемый Informix- DCE/Net, существенно улучшает характеристики безопасности распределенной среды, упрощая, в то же время, деятельность как пользователей, так и администраторов. Достаточно иметь единое входное имя и пароль для DCE, чтобы обращаться к любой погруженной в эту среду базе данных. При запуске приложения Informix-DCE/Net запрашивает аутентификационную информацию пользователя у DCE, и подключает его к требуемой базе.

Наличие единой точки администрирования входных имен и прав доступа к базам данных и приложениям способствует упорядочению общей ситуации с безопасностью. Например, если уничтожается входное имя DCE, то администратор может быть уверен, что данный пользователь уже не сможет получить доступ ни к одному из системных ресурсов.

Контрольные вопросы:

1. Механизмы архитектуры подсистемы безопасности базы данных.
2. Виды привилегий в СУБД.
3. Объясните использование представлений при управлении доступом.
4. Иерархия прав доступа.

Лекция 16: Профили защиты систем управления базами данных, управление надежности проекта баз данных

СУБД, как и операционные системы, содержат комбинацию сервисов безопасности, однако, в отличие от ОС, не являются самодостаточными. СУБД используют механизмы и функции ОС. Такая двух уровневость ведет к появлению специфических уг роз и требует

привлечения соответствующих средств противодействия. Например, базы данных располагаются в файлах или на дисках, управляемых ОС, следовательно, к объектам БД можно обратиться как штатными средствами СУБД, так и с помощью механизмов ОС, получив доступ к файлу или устройству. Подобные возможности должны учитываться в профиле защиты (ПЗ) для СУБД (его прототип соответствует классу безопасности С2 «Оранжевой книги»). Здесь вводится понятие аутентификационного пакета, который предоставляет для СУБД механизм подтверждения подлинности заявляемого идентификатора пользователя. Еще одно проявление упомянутой выше двухуровневое™ - предположение безопасности базовой конфигурации, состоящее в том, что базовая система (операционная система, и/или сетевые сервисы безопасности, и/или специальное программное обеспечение) установлены, сконфигурированы и управляются безопасным образом. Аналогичную направленность имеют цели безопасности для среды, предусматривающие, что базовая система должна обеспечить механизмы управления доступом, которые позволят защитить от несанкционированного доступа все связанные с СУБД файлы; кроме того, ОС предоставит средства для изоляции функций безопасности и защиты процессов СУБД. Можно отметить, что в распределенной среде управление доступом и изоляция могут поддерживаться не только средствами базовой ОС, но и архитектурно, путем разнесения компонентов СУБД по узлам сети и использования межсетевых экранов.

Переходя к функциональным требованиям безопасности, можно указать на важность требований согласованности данных между функциями безопасности (FPT TDC), а также согласованности данных функций безопасности при дублировании в пределах распределенного объекта оценки (FPT_TRC). Согласованность достигается с помощью некоторой формы обработки распределенных транзакций или путем обновления дублируемых данных с применением какого- либо протокола синхронизации.

Для защиты от атак на доступность в профиле защиты предусмотрены реализация квот, выделяемых пользователям (FRU - RSA.1), а также базовые ограничения на параллельные сеансы (FTA_MCS.l).

Необходимо учесть специфику современных СУБД, в частности, требования обеспечения динамической целостности данных, реализуемые механизмом транзакций. Требования безопасного восстановления носят слишком общий характер. Защита от стандартных угроз, существующих в сетевой среде, целиком переложена на базовую систему.

Документ профиль защиты системы управления базами данных (Database Management System Protection Profile) определяет требования безопасности для систем управления базой данных в организациях, где имеются требования для защиты конфиденциальности, целостности и доступности информации, хранимой в базе данных. Несанкционированное раскрытие, модификация или отказ в обслуживании такой информации может оказать неблагоприятное воздействие на функционирование организации. Данный ПЗ определяет:

– совокупность *основных требований,* которым все совместимые с ПЗ базы данных должны удовлетворять;

– совокупность *аутентификационных пакетов* (один или более таких пакетов должны быть обеспечены в совместимой с ПЗ базой данных).

– Администраторы этих систем имеют возможность:

– управлять и постоянно контролировать действия конечных пользователей, чтобы способствовать предотвращению нарушения их прав в пределах системы;

– управлять потреблением ресурса индивидуальными пользователями, и учитывать действия пользователей.

– Аутентификационные пакеты предоставляют средства для подтверждения подлинности пользователя:

– аутентификация в ОС (пользователь аутентифицирован ОС хоста и идентифицирован в базе данных);

– аутентификация в базе данных (пользователь идентифицирован и аутентифицирован СУБД).

Подход разделения основных требований и аутентификационных пакетов был принят для того, чтобы упростить сопровождение этого профиля защиты. Это предполагает, что в будущих изданиях этого профиля защиты, список предлагаемых аутентификационных пакетов может расширяться, например, чтобы включить каталог, основанный на аутентификации.

Для того чтобы заявить соответствие этому профилю защиты в задании по безопасности, должно быть установлено аутентификационный пакет. Заявления соответствия ПЗ должны устанавливать либо «СУБД в режиме аутентификации средствами ОС», либо «СУБД в режиме аутентификации средствами базы данных», либо «СУБД в режиме аутентификации средствами ОС и средствами базы данных».

Профили защиты СУБД определены для некой обобщенной среды со средним уровнем риска для активов. Требования доверия и минимальная стойкость функций выбраны в соответствии с этим уровнем риска.

Обычно СУБД используется для того, чтобы предоставить многим пользователям одновременный доступ к базе данных.

СУБД может быть сконфигурирована многими способами:

- *автономная система* с одиночным пользователем базы данных (например, приложение, основанное на работе отдельного пользователя на персональном компьютере);
- много пользователей базы данных, работающих за *терминалами, которые связаны с центральной машиной* (например, традиционный вариант терминал - среда мэйнфрейма);
- *сеть интеллектуальных рабочих станций, поддерживающих связь с центральным сервером* (архитектура «клиент-сервер»);
- *сеть интеллектуальных клиентских рабочих станций, поддерживающих связь с приложением сервера,* которое, в свою очередь, связано с СУБД (например, Web-браузер, поддерживающий связь с Web-сервером, который формирует динамические страницы с помощью СУБД).

В любой из вышеупомянутых конфигураций сами данные могут постоянно находиться на одном сервере или могут быть распределены среди многих независимых серверов.

СУБД представляет собой приложение, использующее функции нижележащей базовой системы (операционной системы хоста и/или сетевых сервисов, и/или специально заказанного программного обеспечения), и является ИТ-компонентом конкретной системы.

Приложение СУБД может состоять из одного или нескольких выполняемых загрузочных модулей и одного или нескольких файлов данных. Они будут подчинены администрированию основных системных прав, как и любые другие основные системные процессы и файлы.

СУБД может расширять функциональные возможности средств обеспечения безопасности базовой системы. Например, база данных может реализовать гораздо лучший разветвленный механизм привилегий, чем операционная система хоста.

Аутентификационные пакеты. Аутентификационный пакет предоставляет для базы данных механизм подтверждения подлинности заявляемого идентификатора пользователя. В пределах данного профиля защиты это может быть обеспечено следующими двумя механизмами.

Внешним - с помощью операционной системы хоста (аутентификация средствами ОС). В этой схеме аутентификации при идентификации и аутентификации пользователя база данных полагается на операционную систему хоста, которая в этом случае обеспечивает подтверждение подлинности пользователя в базе данных. База данных использует

предоставленный операционной системой идентификатор для установления идентификатора базы данных.

Непосредственно в пределах базы данных (аутентификация средствами базы данных). В этой схеме аутентификации база данных верифицирует заявляемый идентификатор пользователя, используя свой собственный механизм аутентификации.

По крайней мере, один из вышеупомянутых сервисов аутентификации должен быть предоставлен соответствующей базой данных. Активы ИТ, требующие защиты, состоят из информации, хранимой в пределах СУБД: конфиденциальность, целостность или доступность которой может быть скомпрометирована. Активами ИТ являются объекты базы данных и данные, содержащиеся в пределах этих объектов базы данных.

Объектами БД могут быть объединения частей данных, содержащихся в других объектах базы данных. Данные управления базой данных используются СУБД для того, чтобы организовать и защитить объекты базы данных. Данные аудита базы данных генерируются СУБД в процессе ее функционирования.

Принятые угрозы безопасности СУБД наряду с нарушителями, которые могли бы провоцировать эти угрозы, определены ниже.

Этим угрозам будут противостоять:

- технические меры безопасности, предоставленные СУБД, вместе с? б) техническими мерами безопасности, предоставленными базовой системой;
- нетехнические операционные меры безопасности в среде (процедурные, физические меры и относящиеся к персоналу).

Нарушителями могут быть:

- лица, которые не являются уполномоченными пользователями базовой системы (операционной системы и/или сетевых сервисов, и/или специального программного обеспечения);
- лица, которые являются уполномоченными пользователями СУБД;
- лица, которые являются уполномоченными пользователями базовой системы.

Пользователями этой системы могут быть: а) лица, которые не являются пользователями базы данных; б) лица, которые являются пользователями базы данных. *Угрозы, предотвращаемые* СУБД Нарушители могут инициировать следующие типы угроз СУБД (или СУБД должна противостоять следующим угрозам):

T.ACCESS - *несанкционированный доступ к базе данных.* Посторонний или пользователь системы, который в настоящее время не является уполномоченным пользователем базы данных, обращается к СУБД.

T.DATA - *несанкционированный доступ к информации.* Уполномоченный пользователь базы данных обращается к информации, содержащейся в пределах СУБД, без разрешения пользователя базы данных, который является собственником данных или который отвечает за защиту данных. Эта угроза включает несанкционированный доступ к информации СУБД, остаточной информации, хранящейся в памяти или в ресурсах хранения, управляемых СУБД, или к данным управления БД.

T.RESOURCE - *чрезмерное использование ресурсов.* Аутентифицированный пользователь базы данных использует глобальные ресурсы базы данных путем, который ставит под угрозу возможность других пользователей базы данных получить доступ к СУБД. Эта угроза относится к доступности информации в пределах СУБД. Например, пользователь базы данных мог выполнять действия, связанные с использованием чрезмерных ресурсов, периодически препятствуя законному доступу других пользователей базы данных к данным, ресурсам и сервисам. Такие нападения могут быть злонамеренными, происходить в результате невнимательности или небрежности, или в случае, когда пользователь базы данных может просто не сознавать потенциальные

последствия своих действий. Воздействие таких нападений на готовность и надежность системы может быть усилена многими пользователями, действующими одновременно.

T. ATTACK - *необнаруженное нападение.* Необнаруженная компрометация СУБД происходит в результате действий нарушителя (уполномоченного или неуполномоченного пользователя базы данных), пытающегося выполнить действия, которые он не уполномочен выполнять. Эта угроза включена, потому что независимо от обеспечения контрмер, адресованным другим угрозам, все же имеется еще остаточная угроза нарушения политики безопасности нарушителями, пытающимися противостоять этим контрмерам.

T.ABUSE.USER - *неправильное использование привилегий.* Необнаруженная компрометация СУБД происходит в результате действий пользователя базы данных (преднамеренных или нет), Например, пользователь базы данных может предоставить доступ к объекту БД, ответственным за который он является, другому пользователю базы данных, способному использовать эту информацию для мошеннических целей. Отметим, что эта угроза не распространяется на пользователей базы данных с высоким уровнем доверия.

Угрозы, предотвращаемые средой:

T.OPERATE - *опасная операция.* Компрометация базы данных может произойти из-за неправильной конфигурации, администрирования и/или функционирования сложной системы.

T.CRASH - *внезапные прерывания.* Внезапные прерывания функционирования СУБД могут приводить к потере или разрушению данных, связанных с безопасностью, таких как данные управления БД и данные аудита. Такие прерывания могут являться результатом ошибки оператора или сбоев программного обеспечения, аппаратных средств, источников питания или носителей данных.

T.PHYSICAL - *физическое нападение.* Критичные к безопасности части СУБД или базовой операционной системы и/или сетевых сервисов могут быть подвергнуты физическому нападению, которое может нарушить безопасность.

P.ACCESS - Доступ к объектам БД определяется:

- владельцем объекта БД;
- идентификатором субъекта базы данных, пытающегося получить доступ;
- привилегиями доступа к объекту БД, которыми владеет субъект базы данных;
- административными привилегиями субъекта базы данных;
- ресурсами, выделенными субъекту.
  - Заметим, что эта политика включает следующее:
- ***владение*** - владельцы объектов БД ответственны за свои объекты;
- ***дискреционное управление доступом*** - владельцы объектов БД могут предоставлять другим пользователям базы данных доступ или управление своими объектами БД на основе дискреционного управления доступом;
- ***ресурсы*** - пользователи базы данных уполномочены использовать только те ресурсы, которые распределены им.
  - P.ACCOUNT - Пользователи базы данных ответственны за:
- операции на объектах, которые определены владельцем объекта;
- действия, определенные администраторами базы данных.

СУБД зависит как от технических аспектов ИТ, так и от функциональных аспектов ее среды.

Предположения по СУБД:

A.TOE.CONFIG - СУБД инсталлирован, сконфигурирован и управляется в соответствии со своей оцененной конфигурацией.

Основные системные предположения.

Физические предположения:

A.PHYSICAL - ресурсы функционирования СУБД и базовой системы расположены в пределах управления средствами доступа, которые предотвращают несанкционированный физический доступ посторонних, пользователей системы и пользователей базы данных.

Предположения конфигурации:

A.SYS.CONFIG - базовая система (операционная система и/или сервисы безопасности сети, и/или специальное программное обеспечение) инсталлированы, сконфигурированы и управляются в соответствии со своей безопасной конфигурацией.

A. ACCESS - базовая система конфигурирована так, что только санкционированная группа лиц может получить доступ к системе.

A.MANAGE - будут назначены одно или более компетентных доверенных лиц для того, чтобы управлять СУБД, базовой системой и безопасностью информации.

Предположения связности:

A.PEER - предполагается, что любые другие компоненты ИТ, с которыми взаимодействует СУБД, будут под тем же самым управлением и функционируют под той же самой политикой безопасности.

A.NETWORK - предполагается, что в распределенной среде базовые сервисы сети будут основаны на безопасных протоколах взаимодействия, которые обеспечат аутентичность пользователей.

В таблице 5.1 представлено отношение целей безопасности СУБД к каждой из угроз и политик безопасности и показано, что всякой угрозе соответствует, по крайней мере, одна цель безопасности ИТ, и что всякая политика безопасности удовлетворена, по крайней мере, одной целью безопасности ИТ. В таблице слово «ДА» указывает, что указанная цель безопасности ИТ уместна для определенной угрозы или политики безопасности.

O.ACCESS - СУБД должен обеспечить конечных пользователей и администраторов возможностью управления доступом к их собственным данным или ресурсам или к тем, за которые они отвечают в соответствии с политикой безопасности P.ACCESS. Для этого СУБД имеет следующие более конкретные цели:

Таблица 5.1

Взаимосвязь угроз и политик с целями безопасности СУБД

Политики / Угрозы	O.I&A.TOE	O.ACCESS	O.AUDIT	O.RESOURCE	O.ADMIN.TOE
T.ACCESS	ДА	ДА		ДА	ДА
T.DATA	ДА	ДА			ДА
T.RESOURCE	ДА	ДА		ДА	ДА
T.ATTACK	ДА	ДА	ДА		ДА
T.ABUSE.USER	ДА	ДА	ДА		ДА
P.ACCESS		ДА		ДА	
P.ACCOUNT		ДА	ДА		

O.ACCESS.OBJECTS - СУБД должен предотвратить несанкционированное или непредусмотренное раскрытие, ввод, модификацию или уничтожение данных и объектов базы данных, а также просмотр базы данных, управление данными и аудит данных базы данных.

O.ACCESS.CONTROL - СУБД должен предоставить возможность пользователям базы данных, которые являются собственниками или ответственными за данные, управлять доступом к этим данным других уполномоченных пользователей базы данных.

O.ACCESS.RESIDUAL - СУБД должен предотвратить несанкционированный доступ к остаточным данным, остающимся в объектах и ресурсах после использования этих объектов и ресурсов.

O.RESOURCE - СУБД должен предоставить средства управления использованием ресурсов базы данных уполномоченными пользователями СУБД.

O.I&A.TOE - СУБД с поддержкой или без поддержки базовой системы должен предоставить средства идентификации и аутентификации пользователей СУБД.

O.AUDIT - СУБД должен предоставить средства подробной регистрации значимых для безопасности событий для того, чтобы в достаточной мере помочь администратору СУБД:

- обнаруживать предпринятые нарушения безопасности или потенциальную ошибку в конфигурации средств безопасности СУБД, которые оставили бы базу данных незащищенной от компрометации;
- обязать индивидуальных пользователей базы данных быть ответственными за любые выполняемые ими действия, которые являются значимыми для безопасности базы данных в соответствии с политикой P.ACCOUNT.

O.ADMIN.TOE - там, где необходимо, СУБД вместе с базовой системой должен предоставить функции, позволяющие уполномоченному администратору эффективно управлять СУБД и его функциями безопасности, обеспечивая, чтобы только уполномоченные администраторы могли получать доступ к такой функциональности.

Цели безопасности для среды. Следующие цели безопасности ИТ должны быть удовлетворены средой, в которой СУБД используется:

O.ADMIN.ENV - там где необходимо, СУБД вместе с базовой системой должен предоставить функциональные возможности, позволяющие уполномоченному администратору эффективно управлять

СУБД и его функциями безопасности, обеспечивая, чтобы только уполномоченные администраторы могли получать доступ к такой функциональности.

O.FILES - базовая система должна обеспечить механизмы управления доступом, которые позволят защитить от несанкционированного доступа все связанные с СУБД файлы и каталоги (включая выполняемые программы, библиотеки рабочих программ, файлы базы данных, экспортируемые файлы, файлы повторной регистрации, управляемые файлы, файлы с трассировкой и файлы с дампом).

O.I&A.ENV - базовая операционная система должна предоставить средства идентификации и аутентификации пользователей, когда требуется с помощью СУБД надежно подтвердить подлинность пользователей.

O.SEP - базовая операционная система должна предоставить средства для изоляции функций безопасности СУБД и уверенность в том, что ее компоненты не будут искажаться. Составляющими, реализующие функции безопасности СУБД, являются: 1) файлы, используемые СУБД для того, чтобы хранить базу данных и 2) процессы СУБД, управляющие базой данных.

Следующие, не связанные с ИТ, цели безопасности должны быть удовлетворены процедурными и другими мерами, предпринятыми в пределах среды ОО.

O.INSTALL - Ответственные за СУБД должны обеспечить, чтобы:

- ОО был поставлен, инсталлирован, управлялся и использовался в соответствии с эксплуатационной документацией СУБД;

– базовая система была инсталлирована и использовалась в соответствии с ее эксплуатационной документацией. Если элементы системы сертифицированы, то они должны быть инсталлированы и использоваться в соответствии с необходимой документацией сертификации.

O.PHYSICAL - ответственный за СУБД должен обеспечить, чтобы те части СУБД, которые являются критичными к политике безопасности, были защищены от физического нападения.

O.AUDITLOG - администраторы базы данных должны обеспечить, чтобы средства аудита использовались и управлялись эффективно. Эти процедуры должны применяться в журнале аудита базы данных и/или журнале аудита для базовой операционной системы, и/или для сетевых сервисов безопасности. В особенности:

– должны быть предприняты необходимые действия для того, чтобы обеспечить продолжительное функционирование аудита, например, для того, чтобы обеспечить достаточную свободную память, необходимую для регулярной архивации журнала аудита;
– журналы регистрации событий аудита должны регулярно просматриваться и необходимо определить действия или события, которые могут привести к нарушению безопасности в будущем;
– системные часы должны быть защищены от несанкционированной модификации (так, чтобы целостность меток времени аудита не была скомпрометирована).

O.RECOVERY - ответственный за СУБД должен предоставить возможность для процедур и/или механизмов восстановления функционирования на месте после системного сбоя или другого прерывания.

O.QUOTA - администраторы базы данных должны обеспечивать, чтобы каждый пользователь СУБД имел необходимые квоты, которые:

– достаточны для выполнения операций, к которым пользователь имеет доступ;
– достаточно ограничены, чтобы пользователь не мог нарушить режим эксплуатации, доступ к ресурсам и монополизировать ресурсы.

O.TRUST - ответственный за СУБД должен обеспечить, чтобы только у пользователей с высоким уровнем доверия была привилегия, которая позволяет им:

– устанавливать или изменять конфигурацию журнала аудита для базы данных;
– изменять или удалять любую запись аудита в журнале аудита базы данных;
– создавать любые учетные данные пользователя или изменять любые атрибуты безопасности пользователя;
– предоставлять полномочия на использование административных привилегий.

O.AUTHDATA - ответственный за СУБД должен обеспечить, чтобы данные аутентификации для любых учетных данных пользователя СУБД, так же как и для базовой системы, надежно поддерживалась и не раскрывались лицам, которые не уполномочены использовать эту учетную запись. В особенности:

– носители, на которых хранятся данные аутентификации для базовой операционной системы и/или сетевых сервисов безопасности, не должны быть физически устранимы из базовой платформы несанкци- онированными пользователями;
– пользователи не должны раскрывать свои пароли другим лицам;
– пароли, сгенерированные администратором системы, должны быть распределены безопасным способом.

O.MEDIA - ответственный за СУБД должен обеспечить, чтобы конфиденциальность, целостность и доступность данных, хранимых на носителях данных, были адекватно защищены. В особенности:

– сетевые и автономные устройства хранения данных, на которых располагается база данных и данные, связанные с безопасностью (такие как: резервные копии операционной системы, резервные копии базы данных, журналы транзакций и журналы аудита),

физически не могли быть несанкционированно устранены из базовой платформы пользователями;

- сетевые и автономные устройства хранения данных должны подходящим образом сохраняться и поддерживаться, а также регулярно проверяться для того, чтобы обеспечить целостность и доступность связанных с безопасностью данных;
- носители, на которых хранятся связанные с базой данных файлы (включая файлы базы данных, экспортные файлы, файлы повторной регистрации, файлы управления, файлы трассировки и файлы дампов) будут очищены до того, как они будут повторно использоваться в целях не связанных с базой данных.

Таблица 5.2

Отображение целей безопасности среды на угрозы, цели безопасности СУБД, политику и предположения о безопасном использовании

				использовании
O.INSTALL	T.OPERATE			A.TOE.CONFIG, A.SYS.CONFIG, A.MANAGE
O.PHYSICAL	T.PHYSICAL			A.ACCESS, A.PEER, A.PHYSICAL
O.AUDITLOG		O.AUDIT	P.ACCOUNT	A.MANAGE
O.RECOVERY	T.CRASH			A.MANAGE
O.QUOTA		O.RESOURCE		A.MANAGE
O.TRUST			P.ACCESS	A.MANAGE
O.AUTHDATA		O.I&A.TOE	P.ACCESS	A.MANAGE, A.PEER, A.NETWORK
O.MEDIA	T.CRASH			A.MANAGE
O.ADMIN.ENV		O.ADMIN.TOE		A.MANAGE
O.FILES	T.ACCESS		P.ACCESS	A.MANAGE
O.I&A.ENV	T.ACCESS	O.I&A.TOE	P.ACCESS	A.MANAGE
O.SEP	T.ACCESS		P.ACCESS	A.MANAGE

Любой заявленный для соответствия этому ПЗ объект оценки должен, как минимум, обеспечивать выполнение всех функциональных требований безопасности, как определено в основных требованиях.

Дополнительно, любой соответствующий СУБД должен идентифицировать и обеспечить выполнение, по крайней мере, одного из указанных пакетов аутентификации. Для каждого заявленного пакета аутентификации СУБД должен обеспечивать выполнение всех соответствующих функциональных требований безопасности.

Контрольные вопросы:

1. Метка безопасности и принудительный контроль доступа.
2. Из каких компонентов состоит метка безопасности?
3. Объясните суть кластерной организации сервера баз данных.
4. Что понимается под тиражированием данных?
5. Каким образом происходит защита коммуникации между сервером и клиентами?