МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Методические указания

по выполнению лабораторных работ по дисциплине
«БЕЗОПАСНОСТЬ СИСТЕМ БАЗ ДАННЫХ»
для студентов специальности 10.05.03 Информационная безопасность автоматизированных систем,
специализация Анализ безопасности информационных систем

Ставрополь
2026

ВВЕДЕНИЕ

Цель и задачи освоения дисциплины

Целью освоения дисциплины «Безопасность систем баз данных» является формирование у будущего специалиста по специальности 10.05.03 Информационная безопасность автоматизированных систем специализация «Анализ безопасности информационных систем» понимания основных методов, реализации успешной командной работы, применение ИТ-технологий для эффективного взаимодействия в команде и развития проектной деятельности.

Задачи дисциплины:

- изучение теоретических основ формирования и развития навыков командной работы и проектной деятельности;
- формирование умений удаленного управления групповыми проектами;
- овладение навыками эффективного социального взаимодействия, создания благоприятной и конструктивной атмосферы в команде средствами доступных онлайн-инструментов.

СТРУКТУРА И СОДЕРЖАНИЕ ЛАБОРАТОНЫХ ЗАНЯТИЙ

Лабораторная работа №1 Методы разработки безопасных информационных моделей.

Цель работы – исследовать методы разработки безопасных информационных моделей.

В **PostgreSQL** реализован (Multiversion Concurrency Control, MVCC) - многоверсионный контроль конкурентных транзакций, который управляет конкурентным доступом к данным на многоверсионной основе. На практике это означает, что при запросе к БД каждая транзакция видит как бы снимок данных (версию) на момент этого снимка, а не текущее состояние данных. Таким образом транзакции защищаются от просмотра нецелостных данных, которые могут ещѐ только формироваться другими конкурентными транзакциями в тех же самых строках таблицы. Этим же достигается изоляция транзакций для каждой сессии к БД. MMVC позволяет избегать методов явной блокировки, которые применяются в традиционных СУБД и таким образом, минимизирует блокировки данных и позволяет увеличить производительность в многопользовательской работе. Основное преимущество MMVC состоит в том, что чтение данных никогда не блокирует запись, а запись никогда не блокирует чтение.

Также в **PostgreSQL** реализованы традиционные схемы явных блокировок данных, применяющихся для изоляции транзаций, такие как:

блокировка на уровне таблицы

блокировка на уровне записи в таблице (строки)

advisory блокировки (для собственных блокировок на уровне приложе-ний)

Также реализовано отслеживание deadlocks (взаимных блокировок)

Журналы (логи) опережающей записи (WAL)

PostgreSQL реализует механизм WAL (журналов опережающей записи), что даѐт такие преимущества как:

Повышение производительности работы СУБД за счѐт того, что записываются только сделанные изменения без переписывания всех данных в таблицах.

Повышение надѐжности хранения данных за счѐт предварительного со- хранения буферизируемых данных в WAL

Возможность отката состояния БД на любой момент времени, путѐм применения WAL к существующей резервной копии

Репликация и технология Hot Standby

Начиная с 9.0, на основе развития WAL заработала репликация по технологии Hot Standby. Технология позволяет получить на сервере вторую базу данных, которая является актуальной копией оригинальной базы данных, доступной только на чтение. Технология может быть использована также и на удалѐнном сервере, который подключается к primary или master серверу и загружает с него WAL логи, предоставляя онлайновую репликацию базы данных и поддерживая копию базы данных на удалѐнном сервере в актуальном состоянии, а также делая эту копию доступной для запросов на чтение. Это некий аналог технологии Active DataGuard в СУБД Oracle.

Табличные пространства (tablespaces)

Табличные пространства в **PostgreSQL** позволяют задать место хранения объектов БД в файловой системе. Сперва создаѐтся табличное пространство с определѐнным именем. Далее, это имя может быть использовано при создании

таблиц, чтобы разместить эти таблицы именно в данном табличном пространстве

Гибкая настройка сервера

Основной конфигурационный файл postgresql.conf включает более настраиваемых 150 параметров по разделам:

Файлы и пути к ним Сетевые соединения Авторизация и безопасность Выделение ресурсов

WAL - логи обратной записи Планирование запросов Ошибки и протоколирование Статистика запросов

Оптимизация данных через VACUUM

Управление блокировками Совместимость версий и платформ Настройки клиента по умолчанию

Дополнительный конфигурационный файл pg_hba.conf включает в себя

настройки доступа к отдельным БД, такие как указание конкретных IP адресов и(или) сетей, с которых разрешѐн доступ, а также метод(ы) авторизации для до-

ступа к БД и возможность включения безопасных (зашифрованных соединений) через SSL.

Ограничения целостности

Поддерживаются следующие ограничения целостности: NOT NULL - не NULL

UNIQUE - уникальность (начиная с 9.0 введѐно понятие DEFFERABLE UNIQUE)

PRIMARY KEY - первичный ключ

FOREIGN KEY/REFERENCES - внешний ключ, ссылки CHECK - проверка

EXCLUDE - проверка уникальности по сложному условию (начиная с 9.0)

Хранимые процедуры

Хранимые процедуры в **PostgreSQL** могут быть написаны на любом из поддерживаемых встроенных языков. Хранимые процедуры могут быть ис-

пользованы в триггерах и могут возвращать любой из поддерживаемых типов данных, а также массивы и списки.

Начиная с 9.0, вызывать хранимые процедуры можно с указанием именованных параметров, что позволяет создавать хранимые процедуры с перемен- ным числом параметров и перегружаемые функции.

Начиная с 9.0, можно создавать функции без объявления имени (Аноним-

ные блоки) для выполнения блока операторов на любом встроенном языке, ко- торый поддерживает PostgreSQL прямо в командной строке.

Триггеры

Триггеры предназначены для автоматического выполнения отдельных процедур в зависимости от операции, для которой они были назначены. Триг-

геры могут быть назначены до или после операций INSERT, UPDATE или DELETE как для случаев изменения записи в таблице так и для случая выпол- нения оператора SQL. Если произошло событие, на которое был назначен триг- гер, то вызывается закреплѐнная за этим триггером процедура.

Начиная с 9.0.x есть триггеры на колонки (столбцы) и кроме того, при объ-

явлении триггера можно использовать ключевое слово WHEN, добавляющее дополнительное условие для срабатывания триггера.

Система правил

Система правил (более правильно говорить: система правил изменения запросов) позволяет изменять запросы согласно заданным правилам и затем пе-

редаѐт изменѐнный запрос планировщику запросов для планирования и выпол-

нения. Система правил является очень мощным инструментом и может быть использована во многих случаях таких как хранимые процедуры, представле-

ния (views) и версии.

Схемы

PostgreSQL поддерживает схемы. Схемы являются как бы дополнитель-

ными областями видимости внутри базы данных. Также схему можно сравнить и с дополнительным путѐм (название схемы должно указываться перед назва-

нием таблицы) и с каталогом, внутри которого можно разместить таблицы. В

любой базе данных по умолчанию существует схема *public*, в которой по умол-

чанию создаются все таблицы и которую не нужно указывать специально, но администратор БД может создавать другие схемы (и разграничивать доступ к ним), что обеспечивает ещѐ один уровень распределения прав доступа для пользователей, позволяет выделить каждому пользователю как бы персональ-

ный раздел внутри БД с теми же самыми названиями таблиц, что и у других пользователей.

Роли и привелегии

PostgreSQL управляет привелегиями в БД, используя концепцию *ролей*. Ролью может являться как отдельный пользователь БД, так и группа пользователей. Роли могут являться владельцами объектов в БД (например таблиц), а также могут назначать привелегии доступа к этим объектам для других ролей.

Возможно предоставить одной роли членство в другой роли и соответственно передать этой роли права той роли, членом которой она будет являться. Кон-

цепция ролей заменила старую концепцию пользователей и групп, предоставив эту же функциональность.

Начиная с 9.0 поддерживаются права на схемы, а также права по умолча- нию.

Сбор статистики

Чтобы построить производительный план запроса, планировщик запросов в **PostgreSQL** использует так называемую *статистику* или статистическую информацию, собранную на основе анализа данных в таблицах, которая соби-

рается с помощью команды ANALYZE, в свою очередь являющейся частью процесса обслуживания БД VACUUM. Начиная с версии 8.1, в **PostgreSQL** по-

явилась возможность вместо ручного вызова команд сбора статистики, работать

с новым инструментом, который назвали *autovacuum*. С помощью autovacuum весь необходимый сбор статистики и процесс обслуживания БД происходит в фоновом режиме автоматически. Исходя из настроек, **PostgreSQL** сам опреде-

ляет таблицы, для которых необходимо провести сбор статистики и выполнить обслуживание VACUUM.

Резервное копирование и восстановление

PostgreSQL предлагает несколько режимов резервного копирования и вос-

становления БД. Поскольку БД располагаются в файловой системе, вполне нормальным методом является резервное копирование на уровне файлов, т.е.

самого каталога где размещаются файлы БД. Единственное условие такого ре- жима - полный останов сервера **PostgreSQL**. Однако, для систем высокой готовности такой режим резервного копирования недопустим, поэтому **PostgreSQL** позволяет выполнять резервное копирование при запущенном сервере, не прерывая его обычной работы. Наиболее простой режим - это получе- ние дампа БД в текстовом виде (в форме операторов SQL) на стандартный вы- вод. Для экономии дискового пространства можно сразу же перенаправлять та- кой дамп на стандартный ввод утилите сжатия (например gzip). Также суще- ствует возможность создания дампа БД в двоичной форме, а также возмож- ность задавать специальные параметры для большего удобства в получении ре- зервной копии и еѐ последующего восстановления.

PostgreSQL также предоставляет возможность резервного копирования WAL и за счѐт этого, восстановление БД на конкретный момент времени, а также инкрементальное резервное копирование.

Контрольные вопросы:

1. Журналы (логи) опережающей записи.
2. Репликация и технология Hot Standby.
3. Табличные пространства.

Лабораторная работа №2 Методы технологии защиты инфраструктуры хранения информации систем баз данных.

Цель работы – исследовать методы технологии защиты инфраструктуры хранения информации систем баз данных.

Oracle Information Rights Management (IRM) - новая технология информационной безопасности, которая контролирует использование информации и защищает еѐ везде, где она хранится и используется. Обычные продукты по управлению информацией только управляют документами, электронными сообщениями и web-страницами, когда они хранятся в серверных репозиториях (хранилищах данных). Решение по управлению правами доступа к данным от компании Oracle использует кодирование и расширяет управление информацией за границы хранилищ для любых копий критической для организации информации, везде, где она хранится и используется - на рабочих станциях пользователей, лаптопах и мобильных переносных устройствах, в других репозиториях, внутри организации и снаружи, за пределами межсетевых экранов (вне периметра безопасности).

Oracle IRM обеспечивает безопасность самой информации, то есть напрямую защищает саму информацию, а не только хранилища, где она находится.

Решение Oracle IRM, являясь одним из сервисов линейки Oracle Fusion Middleware, глубоко интегрировано в спектр решений Oracle, в частности, в Content Management, Records Management, Identity and Access Management.

Oracle Information Rights Management вводит новые элементы в жизненный цикл документооборота, такие как «запечатывание» и классификацию документов, электронной почты и web-страниц и требует установки агента на рабочие станции пользователей и на каждое устройство, на котором эта закодированная информация хранится или используется. Выгоды подхода ориентированного на защиту информации так очевидны, а изменения в привычные механизмы жизненного цикла так минимальны, что Oracle IRM повсеместно используется в организациях и государственных структурах для обеспечения безопасности наиболее конфиденциальной информации.

В данном техническом обзоре Oracle IRM детально рассматриваются вопросы о том, какие проблемы решает данное решение, как оно работает, какие ключевые моменты необходимы для успешного внедрения в крупных организациях, обсуждаются архитектуры внедрения, различные компоненты, включая средства для разработчиков.

Проблема

Основной проблемой, которую позволяет решить Oracle Information Rights Management, является возможность полного управления информацией для ор-

ганизаций, которые до этого могли управлять только небольшим количеством данных, хранящихся в защищенных хранилищах. Основной же объѐм инфор-

мации оставался либо плохо управляемым, либо совсем не под контролем, так как эта информация циркулировала между серверами, рабочими станциями,

лаптопами, мобильными и беспроводными устройствами, снаружи и вне меж- сетевых экранов.

Ограничения существующих систем безопасности Общим методом построения систем защиты информации является опора

на понятие периметра безопасности, а не на саму информацию как таковую. Документы и сообщения электронной почты до некоторой степени защищены,

пока они находятся внутри контролируемых периметров, таких как файловые папки, контейнеры электронной почты, управляемые репозитории, системы до-

кументооборота и т.д. Но когда эти документы используются на тысячах рабо-

чих станций, лаптопах, мобильных и беспроводных устройствах внутри или вне зоны действия корпоративных межсетевых экранов, их можно легко и незамет-

но для администраторов открыть, скопировать, отправить... куда угодно. Системы защиты на основе периметра безопасности занимаются информа- цией пока она хранится внутри периметра и передаѐтся. В результате такие си- стемы имеют серьѐзные ограничения:

- Традиционная безопасность прекращает своѐ действие на межсетевых экранах, в то время как большинству современных компаний и государствен-

ных организаций приходится отдавать свою конфиденциальную информацию наружу, делясь ею с партнѐрами, клиентами, поставщиками и жителями.

- Копии одной и той же информации могут находиться внутри многих пе-

риметров с различными контролями доступа (например, прайс-лист может быть скопирован из отдела продаж в отдел маркетинга) или туда, где контроля нет совсем.

- На практике существует много различно управляемых периметров даже тогда, когда информация находится в одном месте.

• Возникают большие проблемы с попытками сделать вложенные периметры для того, чтобы делиться информацией или еѐ разделять.

Несмотря на все инвестиции в информационную безопасность, заметным фактом является то, что, как только требуется делиться информацией, никто уже не знает, кто завтра сможет ей воспользоваться.

Ограничения на управление информацией в хранилищах Компании инвестируют миллионы долларов в решения по управлению

информацией, которые оперируют терминами безопасности, аудита, хранения документов, управления версиями и др. Эти системы управления являются пре-

имущественно освнованными на хранилищах данных. Для того чтобы работать с этой информацией, организации хранят эти данные в управляемых репозито-

риях, которые могут быть базами данных (для структурированной информа-

ции), корпоративными системами управления содержимым (content management systems) или системами управления общими ресурсами (для неструктурирован-

ной информации).

Основной проблемой является то, что, когда эта контролируемая инфор-

мация из хранилища используется в повседневной работе, много копий неми-

нуемо передаются и используются вне репозитория, где уже нет возможности ими управлять. Но, кроме отсутствия безопасности и аудита вне репозитория,

которое обсуждалось ранее, хорошими иллюстрациями ограниченности подхо- да ориентации на хранилища являются управление версиями и жизненным цик- лом данных-.

• Управление жизненным циклом основано на политиках хранения и пере-

мещения, например, необходимо быть уверенным, что критические для бизнеса документы сохраняются (скажем) семь лет, в течение которых их нельзя изме-

нять, а затем они должны быть уничтожены, чтобы избежать ненужных рисков при возможных судебных расследованиях. Но, снова, при уничтожении данных в управляемых хранилищах сотни рассеянных в других местах (на других сер-

верах и рабочих станциях) копий продолжают существовать, и их можно будет найти современны ми поисковыми системами.

• Системы управления содержимым предоставляют мощную поддержку управления версиями, так что пользователи легко могут получить последние версии документов. Но много пользователей всѐ ещѐ будут использовать старые версии, хранящиеся локально, вне репозитория,

например, на их рабочих станциях или в почтовых ящиках. Это может привести к серьѐзным ошибкам, бесполезной работе, нарушению инструкций и сбоям в текущей работе.

Фактически, решения, основанные на репозиториях, управляют только некоторой частью документов компании, тем самым, подрывая выгоды от их возможного использования. Например, приложения, в идеале приложения, работающие с информацией должны управлять всеми копиями информации, несмотря на то, где она хранится и используется.

Решение

Oracle Information Rights Management использует «запечатывание» для того, чтобы обеспечить реальный периметр управления документами электрон- ными сообщениями и web-страницами в любом месте, где бы они не были.

Oracle называет процесс кодирования «запечатыванием», потому что он реально выполняет три функции:

• Происходит упаковывание информации слоем кодирования, так что, несмотря на то, как много копий сделано и где они хранятся, они бесполезны без соответствующих данных для раскодирования

• В кодируемый документ встраивается набор ссылок (URL links), так что каждая копия ссылается на Oracle IRM сервер, на котором информация была «запечатана»

• Используется цифровая подпись, так что любая попытка подделки доку- ментов будет определена и предотвращена

Подход, ориентированный на защиту информации

Сами права доступа на запечатанные документы хранятся отдельно от самих данных, на расположенном в сети сервере Oracle IRM, который обслуживает организация-собственник документов. Это привносит несколько революционных преимуществ, так что, где бы информация не хранилась передавалась или использовалась:

Рисунок 1 – «Запечатанная» информация остаѐтся управляемой в любом месте

• Неавторизованные пользователи не могут получить к ней доступ (наибо- лее значимая выгода)

• Только авторизованные пользователи могут открывать или модифициро- вать документы в соответствии с их правами (например, право на распечатыва- ние особо секретной информации)

• Вся работа с запечатанной информацией (и попытки доступа к ней) централизованно протоколируются

• Доступ к удалѐнно хранимой информации может быть централизованно отменѐн, например, если отношения с пользователем, работником по контрак-

ту, партнѐром завершились (если он сделал эти копии, используя DVD, USB и др. средства)

Вероятно, наиболее уникальным свойством Oracle Information Rights Management по сравнению с другими решениями безопасности является его способность управлять информацией снаружи межсетевых экранов, даже когда информация хранится в сети других организаций или дома. Это является прин-

ципиально важным, ввиду того, что современному бизнесу необходимо вовле- кать других участников, партнѐров, подрядчиков, поддерживающие подразде- ления (например, при аутсорсинге), внешних консультантов, домашних работ- ников и т.д.

Управление информацией вне репозитория

Решение Oracle IRM позволяет не только расширить безопасность и аудит за пределы контролируемых хранилищ данных. Оно также расширяет другие аспекты управления информацией за пределы репозитория, такие как управле-

ние жизненным циклом и версиями документов, расширяет выгоды решений по управлению документами на каждую копию корпоративной информации, где бы она ни находилась и использовалась - на рабочих станциях пользователей,

лаптопах и мобильных беспроводных устройствах, в других репозиториях, внутри и снаружи периметра сети.

Например:

• Если документы или почтовые сообщения, являющиеся собственностью компании, запечатаны, то они не только защищены от подделки (никто не име-

ет права их редактировать), но и, когда приходит время их удалить, каждая их копия может быть эффективно изъята с помощью удаления ключа раскодиро-

вания с сервера Oracle IRM. Это является естественным расширением решения Oracle Universal Record Management (универсальное управление записями), ко- торое управляет документами, находящимися в мультивендорных репозитори-

ях, распространяя политику даже дальше, до применения к каждой отдельной копии.

• Если запечатывание применятся к документам в репозитории (таком, как

Oracle Universal Content Management), имеющим версии, то Oracle IRM может быть сконфигурирован так, чтобы автоматически изымать доступ к старым вер-

сиям, если в репозитории находится более новая версия. Если пользователи ло-

кально, вне репозитория сохранили старые версии документов, они не только не получат доступа к старым версиям, но и находящиеся внутри документов ссылки (URL) их приведут к новым версиям, хранящимся в контролируемом репозитории. Своевременное обеспечение пользователей самой новой инфор-

мацией способно заметно сократить расходы компаниям и государственным агентствам и быть уверенным в более полном соблюдении инструкций и пра-

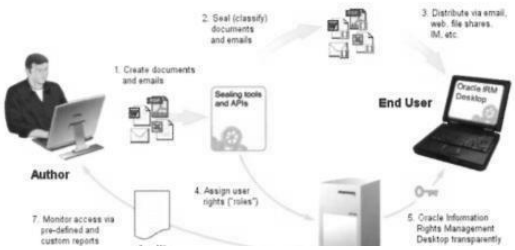
вил.

Как работает **управление** правами

Oracle Information Rights Management имеет запатентованную архитектуру распределения прав между центральным сервером Oracle IRM и агентами

(Oracle IRM Desktop agents), которые должны быть инсталлированы на каждом устройстве, которое создаѐт или использует запечатанную информацию.

Рисунок 2 – Архитектура Oracle Information Rights Management



На рисунке 2 показано, как работает архитектура Oracle IRM (для большей ясности некоторые элементы опущены, например, интеграция между Oracle IRM сервером и корпоративной инфраструктурой по управлению идентификацией и аутентификацией).

1. Авторы продолжают создавать документы и электронные сообщения с помощью существующих приложений, таких как Microsoft Office, Microsoft Outlook, Adobe Reader, Lotus Notes и т.д.
2. Oracle IRM позволяет автоматически или в ручном варианте запечатывать документы на различных стадиях их жизненного цикла с помощью средств, интегрированных в Windows desktop, приложения по созданию документов, клиентские программы электронной почты и общих репозиториях. Запечатывание защищает документы и электронные сообщения с помощью механизмов кодирования и цифровых электронных подписей, вшивая неудаляемые привязки (links) к находящимся в сети серверам Oracle IRM (управляемым организацией, которой принадлежат документы), которые хранят информацию для раскодирования и права доступа к документам.
3. Запечатанные документы и электронные сообщения могут распространятся любым доступным способом (email, web, через файловые сервера и т.д.)
4. Права на запечатанные документы могут быть жѐстко привязаны ко времени запечатывания, хотя в сложных корпоративных системах эта функция редко используется ввиду того, что пользователи не хотят использовать слишком сложные механизмы распределения прав. Права сохраняются отдельно от запечатанных документов и почтовых сообщений на сервере Oracle IRM, поз-

воляя менять права в любое удобное время.

5. Для создания и использования документов и электронных сообщений привычными средствами конечные пользователи должны загрузить и инстал-

лировать универсальный агент, называемый Oracle IRM Desktop. Агент Oracle IRM Desktop имеет небольшой размер, прост в инсталляции и отвечает за аутентификацию пользователей, прозрачным образом запрашивая права с сер-

вера Oracle IRM, защищая и протоколируя работу с запечатанными документа- ми и почтовыми сообщениями, когда они используются встроенными сред- ствами рабочих станций.

Заметим: Запатентованная компанией Oracle распределѐнная архитектура синхронизации прав доступа и аудита событий между сервером Oracle IRM и

клиентом Oracle IRM Desktop обеспечивает возможность пользователям рабо- тать off-line, не заботясь о синхронизации.

6. Сервер и агенты Oracle IRM совместно обеспечивают аудит всех попыток об-

ращения к запечатанным документам и электронным письмам (on-line и off- line), всех административных операций, таких как назначение или изъятие прав.

Можно управлять степенью детализации аудита, а записи аудита могут хра-

ниться в базе данных на сервере Oracle IRM, посланы через очередь сообщений внешним приложениям для обработки или могут экпортироваться в журналь-

ные файлы для дальнейшего импорта в стандартные средства генерации отчѐ-тов.

7. Консоль управления Oracle IRM Management Console и Oracle IRM Web Service SDK обеспечивают отчѐтность на основе запросов с преднастроенными полезными отчѐтами, такими как «Активность пользователей» или «События по определѐнному документу», а также и отчѐты определяемые пользователем.

Средства аудита Oracle IRM открывают беспрецедентные возможности аудита использования (или злоупотребления) корпоративных документов на рабочих станциях пользователей, и только одно это свойство оправдывает инвестиции в

Oracle IRM (не говоря о прочих выгодах для обеспечения безопасности). Ключевые архитектурные отличия от других решений

Два аспекта в архитектуре Oracle Information Rights Management, имеющие чрезвычайно важное значение для корпоративного использования, отличают решение Oracle от продуктов других вендоров:

1. Oracle использует модель прав, основанную на классификации документов, что позволяет присваивать права не отдельным файлам, а наборам. Как результат, необходимо оперировать с меньшим количеством прав. А это, в свою очередь, делает возможным периодическую или автоматическую синхронизацию прав и событий аудита между сервером и агентами Oracle IRM. 2.Автоматическая синхронизация делает возможным полностью прозрачную работу off-line с запечатанной информацией (мобильные пользователи), оставляя в силе централизованное аннулирование или изменение прав.

Конкурирующие решения управляют правами на основе индивидуальных файлов. Для корпоративных объѐмов информации это означает слишком большой объѐм прав для автоматической синхронизации с рабочими станциями. Администраторы таких решений вынуждены выбирать между механизмом кэширования прав (чтобы позволить off-line работу), постоянно находящихся на рабочих станциях, тем самым, жертвуя возможностью изъятия прав, или, всѐ же, оставляя отмену прав и жертвуя работой off-line. Конкурирующие решения не могут обеспечить одновременно работу off-line и возможность изъятия прав.

Успешное применение системы управления правами Для того, чтобы система по управлению правами доступа к данным могла быть успешно внедрена в сети гетерогенных серверов и рабочих станций в современных крупных организациях, имеющих партнѐров, клиентов, подрядчиков, такое решение должно быть **безопасным, удобным** рядовым пользователям и корпоративно **управляемым** (конечными пользователями, владельцами бизнес-процессов и администраторами).

Заметим: Даже в случае первоначального использования такой системы управления правами доступа для определѐнного набора приложений и ограниченного количества пользователей, покупателю лучше всего рассматривать применение его для организации в целом (чтобы получить все выгоды от использования решения).

Безопасность

Конечно, не бывает стопроцентно неуязвимой системы защиты против грамотных профессиональных хакеров (особенно действующих изнутри компании). Но Oracle IRM предоставляет механизмы эффективной многоуровневой защиты на основе индустриальных стандартов и лидирующих технологий безопасности. Как результат, это решение очень легко в использовании авторизо- ванными пользователями и очень сложно для компрометации. Элементы мно- гоуровневой модели безопасности включают в себя: постоянный контроль, аутентификацию, кодирование, электронные подписи и механизмы контроля уязвимостей.

Постоянный контроль

Oracle Information Rights Management может контролировать каждый аспект использования запечатанных документов на рабочих станциях пользовате- лей:

- *Кто:* контроль, кто смог и кто не смог открыть документы

- *Что:* контроль доступа к набору (согласно классификации) или к любому конкретному документу

- *Когда:* контроль того, когда доступ начался и закончился с возможностью отмены права доступа в любой момент

- *Где:* предотвращение возможности доступа к критическим документам снаружи сети

- *Как.* контроль того, как именно пользователи работают с документами на своих рабочих станциях (с глубоким контролем открытия, аннотирования, внесения изменений, трассировкой изменений, контролем копирования, отправки на печать, работы с полями и ячейками форм, просмотром табличных формул и т.д.)

Во всех случаях этот постоянный контроль осуществляется на протяжении всего жизненного цикла документов и электронных сообщений вне зависимости от того, где они находятся и используются.

Аутентификация

Oracle Information Rights Management поддерживает следующие три механизма аутентификации:

- Windows аутентификация (для Single SignOn)

- Имя пользователя и пароль (для внешних пользователей)

- Web-аутентификация

Windows аутентификация используется в качестве прозрачной аутентификации при нормальном заходе пользователей в сессию Windows. Аутентифика- ция по имени и паролю встроена в Oracle IRM для поддержки внешних пользо- вателей, когда не нужно требовать захода в Windows-домен. Web-аутентификация означает заход на сервер Oracle IRM основываясь на аутентификации сессии браузера (схема работы web-сервера и web-браузера). При некоторой доработке Web-аутентификации еѐ можно использовать для интеграции Oracle IRM с любыми системами аутентификациями, такими как RSA SecurID, PK1 сертификаты и др.

Контрольные вопросы:

1. Перечислить ограничение системы безопасности.

2. Ограничения на управление информацией в хранилищах

3. Какие три функции выполняет процесс кодирование по мнению Oracle?

Лабораторная работа №3 Технологии защиты инфраструктуры хранения информации систем баз данных.

Цель работы – исследовать технологии защиты инфраструктуры хранения информации систем баз данных.

Oracle Information Rights Management использует алгоритмы кодирования для выполнения функций защиты сообщений от несанкционированного доступа, а именно:

- Запечатывания документов и электронных сообщений. Обычно это повышает размер файла менее чем на 1%.
- Защиты сетевых телекоммуникаций между сервером и агентами Oracle IRM.
- Для защиты прав доступа на агенте Oracle IRM desktop.
- Для работы с контрольными суммами программных компонент Oracle IRM.

В дополнение, Oracle IRM использует другие дополнительные алгоритмы, например запутывание программного кода (software obfuscation).

Tamper-proofing

Кодирование не предотвращает от копирования с экрана, от попыток за-

лезть внутрь программного обеспечения. Поэтому Oracle IRM имеет дополни- тельные средства для защиты от попыток взлома программного обеспечения:

- Низкоуровневый контроль «лазеек» в приложениях и вызовах функций операционной системы, таких как доступ к виртуальной памяти, видеопамяти или копирования экрана

- Традиционные техники электронных подписей кода, такие как Microsoft Authenticode

- Layered code and interface obfuscation

- Поддержка доверенных часов, а не часов на локальных рабочих станциях

- Незащищѐнная информация никогда не пишется на диск

Обновления системы безопасности

В случае появления нарушений безопасности (бреши) надѐжная система должна иметь возможности их исправления. Фундаментальным свойством

Oracle IRM является то, что агент Oracle IRM Desktop постоянно связывается с серверами Oracle IRM. В функционал этого решения включена дополнительная возможность (в случае необходимости) автоматически загружать и инсталлиро-

вать на компьютерах пользователей необходимые обновления безопасности. Возможности системы

Поддержка гетерогенной инфраструктуры организации

Широкая и глубокая поддержка современных и унаследованных операци-

онных систем и приложений является серьѐзным аргументом при внедрении в реальных распределѐнных гетерогенных системах корпоративного уровня.

Важно также, чтобы внедряемые решения работали на самых последних верси- ях, чтобы при возможных обновлениях функционирование всех систем не пре-

рывалось. Поэтому Oracle поддерживает широкий диапазон последних и уста- ревших версий приложений и операционных систем Microsoft и других компа- ний:

- Microsoft Office 2000-2003 (Word, Excel, PowerPoint)

- Adobe Acrobat или Reader 6.0+
- Email: Microsoft Outlook 2000-2003, Lotus Notes 6.5+ и Novell GroupWise 6.5-7.0

- Email: BlackBerry for Exchange and Domino, BES 4.1+

- HTML и XML (Internet Explorer 6.0+)

- .TXT и .RTF документы •GIF,JPEG и PNG

- TIFF и 2D CAD (требует соответствующих программ-просмотра)

Лѐгкая интеграция в существующие бизнес-процессы

Возможность управления корпоративными документами на рабочих станциях пользователей является важным для администраторов бизнес-процессов, но желательно, чтобы влияние на существующие процессы было минималь- ным. Oracle Information Rights Management имеет несколько ключевых свойств,

которые позволяют с лѐгкостью интегрировать запечатывание документов в существующие бизнес-процессы:

- Очень небольшой единый пакет инсталлятора агента Oracle IRM требует минимальных административных привилегий
- Конечные пользователи могут создавать, открывать и использовать запе- чатанные документы с помощью своих привычных приложений
- Создание стандартных электронных сообщений происходит в обычном email- клиенте, а отправка - по кнопке «запечатать и отправить» (вместо «отправить»)
- Правая кнопка мышки позволяет запечатывать, менять печать и создавать запечатанные документы в интерфейсе Windows Explorer

- Осуществляет единый вход (Single SignOn) в NT-домены и автоматический заход для не NT-аутентификации.
- Осуществляет обработку ошибок и исключений (таких как «нет прав») с помощью интегрированной программы самообслуживания.

• Встроенная интеграция полнотекстового индексирования и поиска в запечатанных файлах.

Поддержка off-line работы

Довольно значительная часть работы является мобильной, и есть необходимость использовать запечатанные документы и электронные сообщения в режиме off-line. Oracle Information Rights Management является единственным решением, предлагающим возможность off-line работы объединѐнной со способностью изъятия доступа к запечатанным документам.

Oracle IRM агент автоматически синхронизирует права конечных пользователей на рабочих станциях без вмешательства самих пользователей (вместо других непрактичных схем, где, например, требуется процесс идентификации требуемых документов до того, как переходить к off-line работе). Для каждой роли Oracle IRM можно сконфигурировать off-line периоды, устанавливая, тем самым, баланс между удобством off-line работы и безопасностью (возможностями быстрого отзыва прав наиболее конфиденциальных документов). Запечатанные документы и электронные письма могут быть созданы и использованы в режиме off-line и операции, такие как открытие и распечатка протоколируются в защищенный буфер для дальнейшей передачи на сервер Oracle IRM. В результате создаѐтся полный хронологический протокол off-line доступа пользо- вателей к запечатанным документам на удалѐнных рабочих станциях.

Контрольные вопросы:

1. Tamper-proofing
2. Поддержка off-line работы
3. Поддержка гетерогенной инфраструктуры организации

Лабораторная работа №4 Методы защиты инфраструктуры обработки информации систем баз данных

Цель работы – исследовать методы защиты инфраструктуры обработки информации систем баз данных.

Управление на основе классификаций прав

Уникальный подход компании Oracle по управлению доступом к информации на основе классификации прав позволяет организации легко контроли-

ровать доступ к большому количеству документов в терминах существующих бизнес-процессов или уже имеющейся классификации информации (например,

по степени секретности - «секретно», «совершенно секретно»), по существую- щим бизнес-ролям (таким как «рецензент»), по существующим группам поль- зователей, определѐнным в каталоге пользователей (например, «менеджеры от- дела продаж»).

Диаграмма на рисунке 3 иллюстрирует простоту и удобство управления на основе классификации прав. Восемь файлов были запечатаны с использованием двух предопределѐнных классов («рабочие документы» и «объявления компа-

нии»). Пользователи финансовый директор (CFO), начальник отдела кадров

(HR Director) и группа «Все пользователи» (all employees) получили необходи- мые права на эти классы, что в результате породило четыре разных права до-

ступа на эти восемь документов. Масштабируемость и управляемость системы Oracle IRM проявляются в том, что количество запечатанных документов легко

со временем может вырасти от восьми до восьми тысяч, при этом, не изменяя количества прав доступа (всего четыре), потому что привязка идѐт на классы, а

не на индивидуальные файлы. А так как нет необходимости управления огром-

ным числом прав (как у некоторых конкурентов), данное решение Oracle имеет не имеющие себе равных надѐжность и масштабируемость, работая при этом на относительно скромном аппаратном обеспечении.

Oracle 1RM поддерживает возможность иметь исключения из общих пра-

вил (неизбежные в реальной жизни) для классификации прав доступа, что поз-

воляет администраторам конфигурировать систему на основе индивидуальных пользователей и отдельных файлов. А эта политика использования намного бо-

лее эффективна, нежели работа с миллионами индивидуальных прав каждого пользователя к каждому файлу. Как только классификация и роли определены в системе, автору документа остаѐтся только принять решение о том, какой печа-

тью он хочет запечатать свой документ или сообщение электронной почты. Большинству пользователей нет необходимость принимать такие решения, по-

тому что они только читают, рецензируют или меняют уже запечатанные документы и сообщения электронной почты.

Стандартная модель прав

Для IT-продуктов очень важным является правильное конфигурирование. Особенно важно это для решений типа управления правами. Никто не хочет потерять контроль над закодированной информацией или сделать ненужные барь-

еры дополнительных аутентификаций и авторизаций между пользователями и той информацией, которая им нужна для выполнения их работы.

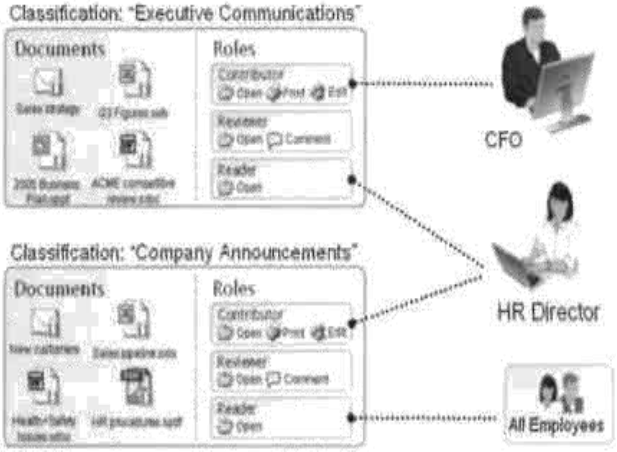


Рисунок 3 – Управление на основе классификации ролей

Oracle Information Rights Management является единственным решением,

построенным в результате пятилетних консультаций, и которое прямо включает в себя наилучшие практики - это называется «стандартной моделью прав Oracle IRM» - лѐгкое в использовании web-приложение, включающее в себя пред-

определѐнные пользовательские и административные роли, шаблоны классификаций, автоматическое управление пользователями и управлением электронными сообщениями, on-line помощь и учебники администраторов.

Рисунок 4 – Стандартная модель прав

Интуитивно понятная стандартная модель прав (Oracle IRM Standard Rights Model) является ключевой составной частью, позволяющей быстро и эффек-

тивно адаптировать Oracle IRM прямо на этапе внедрения с доказанной опытом моделью управления правами.

Административная модель на основе ролей

Oracle Information Rights Management отличается от других подобных ре-

шений тем, что имеет и ролевую и основанную на классификации администра-

тивные модели, которые также глубоки и удобны, как и модели управления правами конечных пользователей. Административные операции, такие как со-

здание классификаций безопасности, определение ролей и присваивание прав на основании типов или на основании отдельного документа могут быть выде-

лены и присвоены отдельным пользователям или группам пользователей. Вла- дельцы бизнес-процессов и их помощники легко могут управлять безопасно-стью их наиболее секретной информации без привлечения IT-администраторов (то есть можно обойтись без дополнительных суперпользователей).

Аудит

Oracle Information Rights Management осуществляет аудит всего on-line и off-line доступа конечных пользователей к запечатанным документам и элек-

тронным письмам, а также и все административные операции, такие как присваивание или изъятие прав. Можно настраивать уровень детализации аудита, а

записи аудита могут сохраняться в базе данных на сервере Oracle IRM, посланы в качестве сообщений (message queues)для использования внешними

системами мониторинга или экспортированы в журнальные файлы для дальнейшего им-

порта стандартными генераторами отчѐтов.

Консоль управления Oracle IRM и Oracle IRM Web Service SDK дают возможность производить отчѐты по аудиту на основе запросов. Они имеют пред-

определѐнные отчѐты типа «Протокол работы пользователя» и «Все операции по данному документу», а также могут использовать отчѐты, определяемые пользователем. Аудит Oracle IRM открывает беспрецедентные возможности по контролю использования или злоупотребления корпоративной информацией на рабочих станциях пользователей, и одно это свойство часто заставляет инве-

стировать средства в это решение (не говоря уже о том, что оно даѐт с точки зрения безопасности).

Интеграция с информационной инфраструктурой

Шлюз Oracle IRM Directory Gateway интегрируется с корпоративными

LDAP-каталогами, такими как Microsoft Active Directory и Sun ONE Directory Server для синхронизации сервера Oracle IRM с централизованными определе-

ниями пользователей и групп. Oracle IRM Directory Gateway также поддержива- ет расширения и плагины для синхронизации пользователей и групп пользова- телей с корпоративными базами данных, доменами Windows и другими источ-

никами. Oracle IRM включает в себя также функционально полный и лѐгкий в испоьзовании Oracle IRM Web Service SDK для программирования интеграции с дополнительными инфраструктурами, такими как web-приложения. Системы управления содержимым, сканерами фильтрации содержимого и т.д.

Производительность и масштабируемость

Экстенсивное кэширование, присущее патентованной распределѐнной ар-

хитектуре Oracle Information Rights Management, комбинированное с моделью прав на основе классификации (а не на основе отдельных файлов) позволяют избежать сильного сетевого трафика и загрузки сервера Oracle IRM (в отличие от других аналогичных продуктов). Это позволяет достичь дополнительного масштабирования и весьма скромных требований к аппаратной части. Типовая конфигурация Oracle IRM на простом сервере показывает способность обслу-

живать свыше 50,000 пользователей. Технологические характеристики и спецификации

Oracle Information Rights Management имеет четыре ключевых компонента:

- **Сервер Oracle IRM Server** - хранит ключи раскодирования и управляет правами пользователей к запечатанным документам и электронным письмам

- **Агент Oracle IRM Desktop** - позволяет авторизованным пользователям создавать и использовать запечатанную информацию в зависимости от прав, полученных с сервера Oracle IRM
- **Консоль управления Oracle IRM Management console** - позволяет ад- министратору управлять всеми аспектами решения Oracle IRM.
- **Oracle IRM Standard Rights Model** - web-приложение, позволяющее бизнес или IT-администраторам создавать новых пользователей, присваивать роли и т.д.

Топология примера внедрения Oracle Information Rights Management Рисунок 5 иллюстрирует типичную конфигурацию Oracle IRM для корпоративного использования. На одном сервере, обычно расположенном в демили- таризованной зоне, работает серверная часть Oracle IRM и web-приложение Oracle IRM Standard Model. Сервер Oracle IRM использует высоконадѐжную ба- зу данных, расположенную в кластере внутри корпоративной сети. Все пользо- ватели используют агент Oracle IRM Desktop, а пользователи с административ- ными правами - ещѐ и консоль управления Oracle IRM Management Console.

Обмен информацией происходит по защищенному каналу, используя 80 порт (рекомендация).

Эту простую, но очень наглядную топологию легко можно масштабировать для работы с очень большим числом пользователей и при высокой нагрузке. В более усложнѐнных схемах возможно применение основного и запасного сервера Oracle IRM в локальной (приватной) сети, что даѐт дополнительную гарантию того, что внешние пользователи не могут получить доступ к информа- ции внутреннего пользования. Сервер Oracle IRM хранит всѐ своѐ текущее со- стояние во внутренней базе данных, механизмы кэширования могут быть от- ключены, а запасной сервер сконфигурирован для использования в случае сбоя (failover).

Интеграция Oracle Information Rights Management

Хотя в решении Oracle Information Rights Management встроены большинство требуемых функций, но в целях более расширенных возможностей в продукт встроены возможности интеграции с корпоративной инфраструктурой и решениями других компаний.

Oracle IRM Web Services SDK имеет документированные примеры по использованию SOAP/WSDL Web-сервисов (применѐнных в сервере Oracle IRM), дающие разработчикам доступ к сервисам работы с документами и административным возможностям. Типичными применениями библиотеки Oracle IRM Web Services SDK являются:

- Динамическое «запетатывание» файлов, когда они покидают репозиторий, например, файловые сервера, системы управления содержимым, репозито- рии общей работы и т.д.
- Временное снятие защиты с файлов для полнотекстового индексирования и поиска, преобразование к другим форматам и т.д.
- Запечатывание и снятие защиты с файлов как часть управления жизненным циклом организации
- Интегрирование Oracle IRM с системами управления идентификацией, напри- мер, добавление/удаление пользователей, присваивание ролей и т.д.

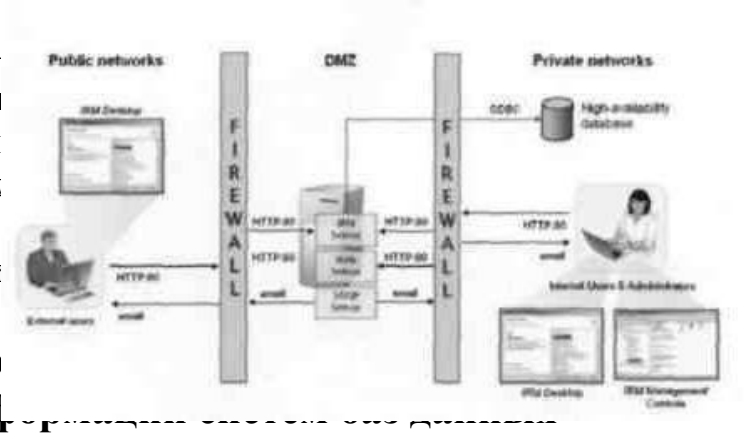


Рисунок 5 - I

Контрольн

1. Упра
2. Стан

3. Адми ролей.

Лаборато нфраструктуры обработки инф

Цель работы – исследовать технологии защиты инфраструктуры обработ-ки информации систем баз данных.

В настоящее время задачи ускорения бизнеса, возврата на инвестиции в ИТ, вопросы обеспечения безопасности информации становятся приоритетными направлениям работы ИТ- подразделений. Основными препятствиями на пути реализации этих задач являются:

• Разобщенность средств управления доступом к информационным систе-

мам, что приводит к существенному замедлению на предоставление доступа сотрудников к информационным ресурсам. Следствием является невозмож-

ность выполнения сотрудником своих бизнес-функций в полном объеме и рост затрат на сопровождение ИТ- инфраструктуры.

• Трудности при интеграции унаследованных и новых приложений в си-

стему управления ИТ. Следствием является замедление в выполнении требова-

ний бизнеса по реализации новых функций и повышение затрат на поддержку ИТ-инфраструктуры.

• Отсутствие интеграции систем управления ИТ с HR-системами. След-

ствием является неадекватные привилегии сотрудников в ИТ-системе, не соот-

ветствующие их должностным обязанностям, что может привести к утечкам конфиденциальной информации и нарушениям в работе ИТ-систем.

Решения Oracle

Для решения важнейших задач по построению системы управления ин-

формационной безопасностью предлагаем использовать систему управления учетными записями на базе решений Oracle Identity Management.

Рисунок 6 – Продукты Oracle Identity Management

Продукты Oracle Identity Management обладают всеми основными функци-ями способными решить поставленные перед ИТ задачи по управлению досту- пом к информационным ресурсам, а именно:

• Согласованное управление доступом на основе должностных обязанно- стей с интеграцией с HR-системой, что сокращает предоставление необходи- мых полномочий до нескольких минут.

• Встроенные средства документооборота и возможность интеграции в имеющиеся системы документооборота, что позволяет интегрировать процессы управления безопасностью в имеющиеся бизнес-процессы управления ИТ.

• Поддержка всех основных платформ и бизнес-приложений (Microsoft, SAP, Sun Microsytems, HP, IBM, Novell и др.) и простота интеграции с имею-

щимися приложениями на основе специального инструментария Oracle Adapter Factory, что сокращает затраты на интеграцию приложений в единую систему управления безопасностью и сохраняет инвестиции в ИТ-инфраструктуру.

• Контроль соблюдения политики безопасности с использованием гибких средств аудита и отчетности, что снижает риски и позволяет удовлетворить требованиям руководящих документов в области информационной безопасно-

сти.

Признание на рынке

По информации аналитического агентства Gartner (2007 Gartner UP MQ)

решения Oracle Identity Management являются лидерами среди решений подобного класса. На основании исследований независимых аналитиков The Radicati Group, опыта реализации подобных проектов и учитывая особенности ИТ в России и СНГ, срок окупаемости системы Oracle Identity Management мо-

жет не превышать 3-х **лет** при достижении ROI не менее **60-70%.**

Oracle уделяет большое внимание стратегическому развитию линейки Identity Management и ее интеграции с бизнес-приложениями. Основными стратегическими направлениями развития технологий Oracle являются:

• **Полнота.** Спектр решений Oracle по управлению безопасностью уровня предприятия является самым полным на рынке (по информации Gartner, "Oracle

предлагает полный портфель решений");

• **Интеграция с бизнес-приложениями.** Решения Oracle Identity Management уже интегрированы с большинством бизнес-приложений и инфра-

структурных решений (от Oracle, SAP, Siebel, PeopleSoft, Microsoft, IBM, HP, Sun и др.). Использование технологии SOA, на базе которых развиваются ре-

шения Oracle Identity Manager, позволит интегрировать технологии безопасно- сти в бизнес приложения на уровне Web-сервисов, что резко сократит

стои- мость и сроки создания интегрированной системы управления безопасностью.

• **Ориентация на открытые стандарты.** Oracle поддерживает и активно участвует в разработке практически всех стандартов управления безопасностью (OASIS, Liberty Alliance и др.).

Таблица 1. Решения Oracle по обеспечению информационной безопасности

Компо-	**Описание**
Identity & Access manageme	Комплекс решений по управлению безопасностью ИТ
Oracle Identity Manager	Средство согласованного управления ролями и учетными за- писями в гетерогенной среде с поддержкой управляющего документооборота (workflow) на основании должностных обязанностей сотрудника. Решаемые задачи: • Автоматиче- ское создание учетных записей пользователей в соответствии с должностными обязанностями; •Назначение / отзыв / изменение привилегий; •Контроль действий адми- нистраторов целевых систем; •Отчетность (оперативная / ис- торическая); •Проверка неизбыточности полномочий пользо- вателей; •Выявление «сиротских» учетных записей; •Разделение / делегирование полномочий;

	•Самообслуживание пользователей.
Oracle Access Manager	Средство согласованного ролевого управления доступом к гетерогенным Web-ресурсам с поддержкой управляющего документообо- рота (workflow). Решаемые задачи: •Единая точка доступа к Web-ресурсам; •Однократная аутентификация (SSO) для Web-ресурсов;
Oracle En-terprise Single Sign-on (ESSO)	Решение, обеспечивающее однократную аутентификацию в распределенных гетерогенных системах. Решаемые задачи: •Пользователю необходимо знать ОДИН пароль; •Пользователь вводит пароль ОДИН раз и получает доступ к необходимым

Web Services Manager	Решение по настройке параметров безопасности (аутентифи- кация, правила доступа) Web-сервисов в рамках архитектуры SOA Решаемые задачи: •Единая политика безопасности для всех SOA-приложений; •Функции безопасности универсальны и могут быть исполь- зованы всеми приложениями; •Централизованные контроли и мониторинг безопасности Web-сервисов.
Oracle Identity Federation	Средство установления доверительных отношений между автономными системами для совместного использования учетной информации. Решаемые задачи: •Доверенные отношения между автономными системами разных организаций; •Самостоятельность управления поли- тикой безопасности для каждой организации; •Однократная аутентификация пользователя во всех дове- ренных системах;
Oracle Virtual Directory	Средство организации единого представления данных из различных хранилищ информации. Решаемые задачи: •Универсальное представление данных из различных источ- ников в виде LDAP-каталога; •Нет необходимости синхронизации данных между источни- ками;

Management Pack for Identity Management	Средство управления и мониторинга SLA для компонент Identity management. Решаемые задачи: • Отслеживание уровня сервиса (SLA) компонент Identity Managementна функциональном уровне; • Мониторинг ключевых показателей (KPI) сервисов Identity Management; • Интеграция в единую консоль управления Oracle Enterprise Manager
Identity and Access management Suite	Комплексное решение, включающее в себя следующие ком- поненты: • Oracle Identity Manager • Oracle Internet Directory • Oracle Virtual Directory • Oracle Access Manager • Oracle Identity Federation
Database Security	Комплекс решений обеспечения безопасности баз данных
Database Vault	Решение, обеспечивающее дополнительное разграничение полномочий внутри базы данных с реализацией ограничения доступа к данным защищаемых приложений со стороны администратора, а также ограничения доступа и контроль вы- полнения команд в зависимости от времени, IP-адреса, опе- рации и т.д.
Label	Решение по реализации мандатного доступа (с использова- нием меток) к данным в базе данных.

Security	
Advanced Security Options	Решение, реализующее усиленную аутентификацию при до- ступе к базе данных (X.509, Kerberos, Radius), защиту клиентского трафика (SSL), прозрачное шифрование критиче- ской информации в базе данных и их защиту на физических носителях информации.
Secure Backup	Решение, обеспечивающее дополнительную защиту резерв- ных копий данных, баз данных и файлов операционной системы на магнитных лентах.
Audit Vault	Решение консолидации, хранения и анализа данных аудита из различных

Контрольные вопросы:

1. Основные препятствия реализации задач безопасности в БД.

2. Решения Oracle.

Лабораторная работа №6 Методы защиты инфраструктуры передачи информации систем баз данных.

Цель работы – исследовать методы защиты инфраструктуры передачи информации систем баз данных.

Выполнение требований законодательства РФ

Решения Oracle по безопасности позволяет существенным образом облегчить выполнение технических требований Законодательства РФ по защите конфиденциальной информации, в частности **Закона о персональных данных. С**одержится анализ выполнения требований Закона о персональных данных на базе решений Oracle.

Применение решений Oracle по информационной безопасности для выполнения требований Федерального закона Российской Федерации от 27 июля 2006 г. N 152-ФЗ "О персональных данных"

Данный документ содержит анализ соответствия решений Oracle по обеспечению информационной безопасности (ИБ) требованиям Федерального зако- на Российской Федерации от 27 июля 2006 г. N 152-ФЗ "О персональных данных".

Таблица 2. Применение решений Oracle

	Требование закона	**Реализация с помощью**
	Статья 5 п.5 (Недопустимость объединения созданных для несовместимых между собой целей баз данных информационных систем персональных данных)	Использование **Database Vault** позволяет разделить массивы различных данных на основе создания защищенных областей (realms).
	Статья 6 п.3 (Обезличивание персональных данных)	Использование **Ad- vanced Security Options (ASO)** позволяет обезличить персональные данные путем зашифрования колонок дан- ных, идентифицирующих субъекта персональных дан- ных.

	Статья 7 (Конфиденциальность персональных данных)	Использование решений **Database Vault, Audit Vault Label Security, Identity and Access Management Suite (IAMS), ASO, SecureBackup** обеспечивает конфиденциальность персональных данных за счет использования следующих технологий Oracle. • **IAMS** реализует механизм разделения полномочий, ролевой механизм управления в масштабах всей информационной системы на основе принципа need-to-know; • **Database Vault** обеспечивает разграничение доступа к полям данных с помощью

	Статья 10 (Специальные категории персональных данных)	**IAMS** позволяет разграничить права доступа к персональным данным из различных приложений, **Database Vault и Label Security** обеспечивают гарантию этого доступа на уровне базы данных.
	Статья 11 (Биометрические персональные данные)	**IAMS, Database Vault и Label Security** обеспечивают сквозной ролевой контроль доступа к биометрическим данным на уровне приложений и уровне базы данных.
	Статья 14 п.4 (получение информации, касающейся обработки персональных данных)	**IAMS** позволяет ответить на вопрос о том, кто именно, когда, и на основании каких документов имел право доступа к персональным данным. **Audit Vault** позволяет определить кто и когда обращался к персональным данным.

	Статья 19 (Меры по обеспече- нию безопасности персональных данных при их обработке)	Advanced Security Options обеспечивает усилен- ную аутентификацию и шифрование трафика и самих данных, Secure Backup - усилен- ную защиту резервных копий, Database Vault - защиту дан- ных от администраторов и разработчиков, IAMS - разграничение прав и привилегий со стороны приложений, сервисов и пользователей.
	Статья 20 п.3 (Возможность ознакомления субъекта с персо- нальными данными)	IAMS позволяет обеспечить доступ только к персо- нальным данным, относящимся к субъекту. Database Vault и Label Security обеспечивают гарантию защиты этого досту- па на уровне базы данных.

	Статья 21 пп. 1,2 (блокирова- ние и разблокирование персональ- ных данных)	Блокирование на уровне приложений может быть га- рантировано на уровне базы данных с помощью Label Security и Database Vault.
	Статья 21 п.3 (Выявление не- правомерных действий с персо- нальными данными)	Audit Vault позволяет оперативно обнаружить не- правомерные действия с перс- ональными данными и опо- вестить о необходимых дей- ствиях персонал оператора.

Контрольные вопросы:

1. Выполнение требований законодательства РФ

Лабораторная работа №7 Технологии защиты инфраструктуры передачи информации систем баз данных.

Цель работы – исследовать технологии защиты инфраструктуры передачи информации систем баз данных.

По заявлению аналитиков Gartner "Стратегия Oracle выглядит лучшей среди всех вендоров" (2006 Gartner UP MQ). По словам аналитиков Burton Group "Oracle сейчас может рассматриваться как основной поставщик решений Identity and Access Management".

По отношению к финансовым организациям решения Oracle способствуют выполнению требований Стандарта Банка России СТО БР ИББС-1.0-2006. содержится анализ выполнения требований данного Стандарта на базе решений Oracle.

Применение решений Oracle по информационной безопасности для выполнения требования Стандарта Банка России СТО БР ИББС-1.0-2006 Данный документ содержит анализ соответствия решений Oracle по обес-

печению информационной безопасности (ИБ) требованиям СТО БР ИББС-1.0- 2006.

Таблица 3. Применение решений Oracle

	Требование закона	**Реализация с помощью решений Oracle**
	п. 5.4 (угрозы со стороны персонала)	**Database Vault** обеспечивает защиту информации в базах данных, в том числе и от администратора баз данных. Средства защиты **Database Vault** позволяют разграничить доступ к защищенным объектам базы данных с использованием дополнительных условий, например, времени доступа, адреса рабочей станции и т. д. **Label Security** реализует мандатный механизм кон- троля доступа с использованием меток конфиденци- альности информации.
	п.5.6 (злоумышленник изуча- ет объект нападения)	Средствами контроля выполнения правил доступа (аудит в **Database Vault)** предотвращается возмож- ность исследования объекта нападения путем экспе- римента.
	п.5.9 (повышение сложности управления ИБ порождает новые уязвимости)	Внедрение прозрачного шифрования полей таблиц баз данных и использование **Database Vault** не тре- буют каких-либо изменений в ранее разработанных средствах защиты информации и прикладных систе- мах.

	пп. 5.12,5.13 (требование наличия процессного подхо- да)	**Identity and Access Management Suite (1AMS)** обеспечивает бизнес-процессы управления учетны- ми записями и доступом к информационным ресур- сам. Бизнес-процессы проектируются с помощью специального инструментария и автоматически ис- полняются системой **IAMS.**
	п.7.6 и п.8.2.3.10 (доступ к оборудованию)	Применение опции прозрачного шифрования **(TDE)** обеспечивает защиту информации на физических носителях.

6	пп.7.13,7.14,8.2.3.8,8.2.3.9 (управление операционными рисками и защита от угроз)	Использование решений **Database Vault, Audit Vault, Label Security, IAMS, Advanced Security Options (ASO), SecureBackup** снижает операцион- ные риски (ошибочных действий пользователя, дей- ствий внутренних злоумышленников, отказ в обслу- живании) за счет реализации: •оптимизации прав до- ступа и привилегий (need to know) **(IAMS, Database Vault, Label Security);** • разграничения полномочий (Segregation of duties) **(IAMS, Database Vault);** •мандатного доступа **(Label Security);** •дополнительных механизмов защиты СУБД **(Database** данных аудита и своевременного **Vault, Label Security);** •консолидации обнаружения злоумышленных действий **(Audit Vault);** •защиты резервных копий **(Secure Backup);**	

•защиты клиентского трафика **(ASO);** •прозрачного шифрования данных в СУБД (ASO); •усиленной аутентификации, в т.ч. с использованием сертификатов X.509 **(ASO, IAMS);** •горячего резер-

вирования компонент системы

защиты **(Real Application Cluster);** •оперативного мониторинга уровня обслуживания

(SLA) для сервисов безопасности **(IAMS).**

	пп. 8.2.2.1-8.2.2.27 (ролевой доступ)	**IAMS** реализует ролевой механизм управления в масштабах АБС. Даже если какое-то приложение не реализует ролевой доступ, IAMS интегрирует его в глобальную систему ролевого управления. **IAMS** обеспечивает создание иерархии ролей, синхронизированной с наборами бизнес-функцией, необходи- мых сотруднику, в соответствии с организационной структурой. 1AMS обеспечивает импорт организа- ционной структуры из HR-систем. **IAMS** обеспечивает наличие ролей по обеспечению ИБ с правами доступа, необходимыми для управления параметра- ми ИБ. **IAMS** реализует механизм разделения полномочий (Segregation of Duties) для избежания ис- полнения критичных операций одним сотрудником. **Database Vault** интегрирует механизмы защиты информации СУБД в систему ролевого доступа.

	п. 8.2.3.1 (обеспечение ИБ на стадиях жизненного цикла)	**IAMS** реализует разграничение ролей разработчи- ков, заказчиков и других участников жизненного цикла АБС по доступу к ресурсам АБС. **Database Vault** обеспечивает разграничение доступа к полям данных, используемых при разработке, тестировании и эксплуатации с помощью создания защищенных областей (realms) в СУБД.
	п. 8.2.3.4 (ввод и снятие АБС с эксплуатации)	**IAMS** обеспечивает автоматическое создания или удаление учетных записей пользователей и разра- ботчиков при вводе и снятии с эксплуатации. При этом данный процесс может производиться с участи- ем подразделения ИБ на уровне согласования дан- ных операций. Процедура согласования реализуется в составе единой системы документооборота по управлению безопасностью из состава **IAMS.**
0	8.2.4.1 (принципы безопасно- сти)	**IAMS** обеспечивает выполнение принципов "need to know", "know your Customer", "know your Employee" за счет: •четкого определения ролей пользователя в АБС; •соответствия ролей **IAMS** бизнес- ролям пользова- теля;

1	п. 8.2.4.3, раздел 1 (требования к парольной по- литике)	**IAMS** реализует синхронизацию паролей в компо- нентах АБС и единую парольную политику в АБС. **Enterprise Single Sign-On (ESSO)** обеспечивает од- нократную аутентификацию (Single-Sign-On) в ком- понентах АБС с реализацией проверки качества па- ролей и возможностью самостоятельной смены и восстановления пароля пользователем. При этом **ESSO** полностью интегрирована с **IAMS,** что позво- ляет обеспечить согласованное упраление учетными записями и параметрами аутентификации.
2	п. 8.2.4.3, раздел 2 (требова- ния к порядку назначения доступа)	**IAMS** обеспечивает документооборот по согласова- нию назначения прав и привилегий в АБС. При этом правила согласования доступа, в том числе участие функционального руководителя, задаются централи- зованно, с помощью инструментария дизайна биз- нес-процессов на стадии разработки политики без- опасности.
3	п. 8.2.4.3, раздел 5 (требова- ния к регистрации)	**IAMS** обеспечивает регистрацию истории назначе- ния прав и привилегий пользователя как в разрезе компоненты АБС, так и в разрезе времени соответ- ствующего назначения. **Audit Vault** реализует консолидацию журналов ре- гистрации различных компонент АБС, их хранение и оперативную отчетность об активности пользовате- лей с выдачей предупреждений об инцидентах и трендах активности пользователей АБС.

4	п. 8.2.8.5 (требование о недопустимости полноты полномочий у одного сотрудника) п. 8.2.9.6 (разграничение полномочий администратора АБС и администратора информаци- онной безопасности)	**IAMS** реализует разделение полномочий (Segregation of Duties) на уровне ролей АБС. **Label Security** реализует мандатный способ разграничения полномочий с использованием меток конфиденциальности информации. **Database Vault** обеспечивает разделение полномочий по доступу к критичной ин- формации в СУБД с использованием меток безопас- ности, в том числе, на уровне администратора баз данных.
5	п. 9.7.1 (функции службы информационной безопасности)	Решения Oracle **(IAMS, Database Vault, Audit Vault)** предоставляет службе ИБ инструмент управления и контроля политики безопасности.
6	п. 10 (порядок аудита АБС)	Решения Oracle **(IAMS, Database Vault, Audit Vault)** обеспечивают генерацию, консолидацию и хранение данных аудита действий пользователя в АБС, а так- же инструмент построения отчетов для аудиторов АБС.
7	п. 11 (модель зрелости менедж- мента ИБ)	**IAMS** способствует достижению 4-го уровня зрело- сти за счет встроенных возможностей автоматиче- ской оптимизации (Attestation) имеющейся ролевой модели доступа в АБС.

Контрольные вопросы:

1. Применение решений Oracle по информационной безопасности для выполнения требования Стандарта Банка России СТО БР ИББС-1.0-2006.

Лабораторная работа №8 Методы защиты инфраструктуры управления систем баз данных.

Цель работы – исследовать методы защиты инфраструктуры управления систем баз данных.

Мандатная защита

Средства мандатной защиты предоставляются специальными (trusted) версиями СУБД.

Мандатное управление доступом (mandatory access control) — 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.

Средства произвольного управления доступом характерны для уровня безопасности С. Как правило, их, в принципе, вполне достаточно 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 чего данные оказываются «обезличенными» и ничто не мешает передать их кому угодно даже средствами самой СУБД; для этого нужно лишь получить доступ к таблице или представлению.

Физическая защита СУБД главным образом характеризует данные (их принадлежность, важность, представительность и пр.). Это в основном метки безопасности, описывающие группу принадлежности и уровни конфиденциальности и ценности данных объекта (таблицы, столбца, строки или поля). Метки безопасности (физическая защита) неизменны на всем протяжении существования объекта защиты (они уничтожаются только вместе с ним) и территориально (на диске) располагаются вместе с защищаемыми данными, а не в системном каталоге, как это происходит при логической защите.

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

фиденциальности, защищенные СУБД предоставляют средства слежения за до- ступом субъектов к объектам защиты (аудит).

Использование СУБД с возможностями мандатной защиты позволяет разграничить доступ собственно к данным, хранящимся в информационной систе- ме, от доступа к именованным объектам данных. Единицей защиты в этом слу- чае будет являться, в частности, запись о договоре N, а не таблица или пред-

ставление, содержащее информацию об этом договоре. Пользователь, который будет пытаться получить доступ к договору, уже никак не сможет обойти метку конфиденциальности. Существуют реализации, позволяющие разграничивать

доступ вплоть до конкретного значения конкретного атрибута в конкретной строке конкретной таблицы. Дело не ограничивается одним значением метки конфиденциальности — обычно сама метка представляет собой набор значе-

ний, отражающих, например, уровень защищенности устройства, на котором хранится таблица, уровень защищенности самой таблицы, уровень защищенно-

сти атрибута и уровень защищенности конкретного кортежа.

За исключением атрибута собственности (логическая защита), разбивающего данные (таблицы) на собственные (принадлежащие данному субъекту) и чужие, физическая защита разбивает данные более тонко. Но можно ли обой- тись без физической защиты или, по крайней мере, попытаться, реализовав,

например, сложный набор хранимых процедур. В общем-то некоторое подобие такой защиты реализуемо в случае, когда метки добавляются в таблицу в каче-

стве дополнительного атрибута, доступ к таблицам запрещается вообще и ни одно приложение не может выполнить интерактивный SQL-запрос, а только хранимую процедуру и т.п. Ряд реализаций подобного уровня защиты исполь-

зует вызов набора хранимых процедур с весьма абстрактными (что очень жела-

тельно) именами. Система реализации защиты информации в данном случае достаточно сложна и предполагает определенный уровень доверия к админи-

стратору 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 секретны- ми данными.

Кроме того, защищенные СУБД позволяют разграничить доступ к информационной системе с тех или иных рабочих станций для тех или иных зарегистрированных пользователей, определить режимы работы, наложить ограниче-

ния по времени работы тех или иных пользователей с тех или иных рабочих станций. В случае реализации данных опций на прикладном уровне задача, как правило, сводится к созданию сервера приложений, который занимается отсле-

живанием, «кто и откуда пришел». Отдельный комплекс серверных приложе-

ний (обычно — хранимых процедур, если в СУБД отсутствует мандатная защи- та) обеспечивает аудит.

Рассмотрим мандатную защиту подробнее. В качестве

примера возьмем мандатную защиту СУБД «Линтер», которая получила признание в весьма спе-

цифическом секторе — силовых структурах, как единственная СУБД, имеющая сертификат по второму классу защиты от несанкционированного доступа, что соответствует классу В3 по американскому национальному стандарту.

Во-первых, все перечисленные объекты (независимо от их иерархии в базе данных) разбиваются здесь на группы принадлежности. Объект может принад-

лежать только одной из групп (это может быть, например, разбиение по отде-

лам организации). Группы принадлежности напрямую связаны с группами субъектов (см. ниже). Субъект вправе видеть только данные своей группы, если между группами субъектов не установлены отношения доверия.

Во-вторых, все объекты выстроены в иерархию по уровням конфиденци-

альности и по уровням ценности или важности. Уровень конфиденциальности разбивает объекты по доступности на чтение (и даже на просмотр). Пользова-

тель с более низким уровнем доступа не будет знать даже о существовании объектов с более высоким уровнем конфиденциальности. Уровень ценности,

напротив, разбивает данные (объекты) по важности, ограничивая возможность их удаления и модификации.

В уже упоминавшихся «Критериях оценки надежных компьютерных систем» применительно к системам уровня безопасности В описан механизм ме- ток безопасности, реализованный в рассматриваемых СУБД.

Метка объекта включает следующее:

1. Группа субъекта, который внес данный объект.

2. Уровень доступа на чтение — RAL (Read Access Level).

3. Уровень доступа на запись — WAL (Write Access Level). Метка субъекта выглядит аналогично:

1. Группа, к которой принадлежит субъект.

2. RAL-уровень субъекта, который представляет собой максимальный RAL- уровень доступной субъекту информации.

3. WAL-уровень субъекта, то есть минимальный RAL-уровень объекта, ко- торый может быть создан этим субъектом.

Все пользователи базы данных считаются разбитыми на непересекающие-

ся группы. Группа описывает область доступных пользователю данных. Для каждой группы существует администратор группы (уровень DBA для группы),

созданный администратором системы. При этом пользователи одной группы не видят данных, принадлежащих пользователям другой группы. В этом плане у СУБД «Линтер» имеется особенность: в системе реализовано такое понятие,

как «уровень доверия между группами». При этом уровни доверия не могут быть вложенными. Группа представляет собой числовое значение в диапазоне

[1-250]. Группа 0 — группа администратора системы. Только администратор системы может создать пользователя в группе, отличной от своей. Все данные,

созданные от имени пользователя, помечаются его группой.

Уровни доступа вводятся для проверки прав на осуществление чтения-записи информации. Вводятся следующие уровни доступа:

1. Для пользователя (субъекта):

RAL — уровень доступа; пользователь может получать (читать) инфор-мацию, RAL-уровень которой не выше его собственного уровня доступа; WAL — уровень доверия на понижение уровня конфиденциальности;

пользователь не может вносить информацию с уровнем доступа (RAL-уровнем) более низким, чем данный WAL-уровень пользователя. Иными словами, поль- зователь не может сделать доступную ему информацию менее конфиденциаль- ной, чем указано в данном параметре.

2. Для информации:

RAL — уровень чтения; пользователь может получать (читать) информацию, RAL-уровень которой не выше его собственного RAL-уровня (может читать менее конфиденциальные данные);

WAL — уровень ценности или уровень доступа на запись (модификацию, удаление); пользователь может модифицировать (удалять) информацию, WAL-уровень которой не выше его RAL-уровня.

Создать пользователя с произвольными уровнями может только администратор системы. Остальные администраторы (DBA) могут создавать пользователей (или изменять уровень пользователям) лишь в пределах отведенных им уровней. Пользователь может принудительно пометить вводимые данные, указав в списке атрибутов уровни доступа для соответствующих записей и полей (при выполнении операторов INSERT или UPDATE). По умолчанию вносимые данные наследуют уровни пользователя, вносящего/изменяющего данные. Защищаемые объекты: пользователи, таблицы, столбцы, записи (вносится при выполнении INSERT), поля записей (изменяются при выполнении UPDATE).

Уровни, как и группы, нельзя использовать в случае, если они не созданы специальными запросами.

Конфигурация, к которой имеет доступ хотя бы один программист, не может считаться безопасной. Поэтому обеспечение информационной безопасности баз данных — дело весьма сложное, и во многом вследствие самой природы реляционных СУБД.

Помимо систематического применения арсенала средств, описанных выше, необходимо использовать административные и процедурные меры, в частности регулярное изменение паролей пользователей, предотвращение доступа к физическим носителям информации и т.п.

Контрольные вопросы:

1. Мандатная защита

2. Что характеризует физическая защита СУБД?

Уровни доступа.

Лабораторная работа №9 Технологии защиты инфраструктуры управления систем баз данных.

Цель работы – исследовать технологии защиты инфраструктуры управления информации систем баз данных.

Пользователей СУБД можно разделить на три группы:

1. Прикладные программисты - отвечают за создание программ, использу- ющих 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 опреде-

ляет администратор безопасности или администратор базы данных (если это одно и то же должностное лицо).

3. Администраторы баз данных - образуют особую категорию пользовате- лей СУБД. Они создают сами базы данных, осуществляют технический кон-

троль функционирования СУБД, обеспечивают необходимое быстродействие системы. В обязанности администратора, кроме того, входит обеспечение поль-

зователям доступа к необходимым им данным, а также написание (или оказание помощи в определении) необходимых пользователю внешних представле-

ний данных. Администратор определяет правила безопасности и целостности данных.

Дискреционная защита

В современных СУБД достаточно развиты средства дискреционной защи- ты.

Дискреционное управление доступам (discretionary access control) — раз-

граничение доступа между поименованными субъектами и поименованными объектами. Субъект с определенным правом доступа может передать это право любому другому субъекту.

Дискреционная защита является многоуровневой логической защитой. Логическая защита в СУБД представляет собой набор привилегий или ро-

лей по отношению к защищаемому объекту. К логической защите можно отне- сти и владение таблицей (представлением). Владелец таблицы может изменять

(расширять, отнимать, ограничивать доступ) набор привилегий (логическую защиту) . Данные о логической защите находятся в системных таблицах базы данных и отделены от защищаемых объектов (от таблиц или представлений).

Информация о зарегистрированных пользователях базы данных хранится в ее системном каталоге. Современные СУБД не имеют общего синтаксиса SQL-

предложения соединения с базой данных, так как их собственный синтаксис сложился раньше, чем стандарт ISO. Тем не менее часто таким ключевым пред-

ложением является CONNECT. Ниже приведен синтаксис данного предложе-

ния для Oracle и IBM DB2 соответственно: CONNECT [[logon] [AS {SYSOPER|SYSDBA}]]

пользователь/пароль[@база_данных]

CONNECT TO база_данных USER пользователь USING пароль

В данных предложениях отражен необходимый набор атрибутов, а также показано различие синтаксиса. Формат атрибута база_данных, как правило,

определяется производителем СУБД, так же как и имя пользователя, имеющего по умолчанию системные привилегии (SYSDBA/SYSOPER в случае Oracle).

Соединение с системой не идентифицированных пользователей и пользо-

вателей, подлинность идентификации которых при аутентификации не под-

твердилась, исключается. В процессе сеанса работы пользователя (от удачного прохождения идентификации и аутентификации до отсоединения от системы)

все его действия непосредственно связываются с результатом идентификации. Отсоединение пользователя может быть как

нормальным (операция DISCONNECT), так и

насильственным (исходящим от пользователя-

администратора, например в случае удаления пользователя или при аварийном обрыве канала связи клиента и сервера). Во втором случае пользователь будет проинформирован об этом, и все его действия аннулируются до последней фик-

сации изменений, произведенных им в таблицах базы данных. В любом случае на время сеанса работы идентифицированный пользователь будет

субъектом доступа для средств защиты информации от несанкционированного доступа

(далее - СЗИ НСД) СУБД.

Следуя технологии открытых систем, субъект доступа может обращаться посредством СУБД к базе данных только из программ, поставляемых в дистри-

бутиве или подготовленных им самим, и только с помощью штатных средств системы.

Все субъекты контроля системы хранятся в таблице полномочий системы и разделены для системы на ряд категорий, например CONNECT, RESOURCE и DBA. Набор таких категорий определяется производителем СУБД. Мы не слу-

чайно предлагаем указанный порядок рассмотрения — именно так происходит нарастание возможностей (полномочий) для каждого отдельного вида подклю-

чения:

CONNECT — конечные пользователи. По умолчанию им разрешено только соединение с базой данных и выполнение запросов к данным, все их действия регламентированы выданными им привилегиями;

RESOURCE — привилегированные пользователи, обладающие правом создания собственных объектов в базе данных (таблиц, представлений, сино-

нимов, хранимых процедур).

Пользователь — владелец объекта обладает полным набором привилегий для управления данным объектом;

DBA — категория администраторов базы данных. Включает возможно-

сти обеих предыдущих категорий, а также возможность вводить (удалять) в си- стему (из системы) субъекты защиты или изменять их категорию.

Следует особо отметить, что в некоторых реализациях административные действия также разделены, что обусловливает наличие дополнительных катего-

рий. Так, в Oracle пользователь с именем DBA является администратором сер-

вера баз данных, а не одной-единственной базы данных. В СУБД «Линтер» компании РЕЛЭКС понятие администратора сервера баз данных отсутствует, а

наличествует только понятие администратора конкретной базы данных. В IBM DB2 существует ряд категорий администраторов: SYSADM (наивысший уро-

вень; системный администратор, обладающий всеми привилегиями); DBADM (администратор базы данных, обладающий всем набором привилегий в

рамках конкретной базы данных). Привилегии управления сервером баз данных име-

ются у пользователей с именами SYSCTRL (наивысший уровень полномочий управления системой, который применяется только к операциям, влияющим на системные ресурсы; непосредственный доступ к данным запрещен, разрешены операции создания, модификации, удаления базы данных, перевод базы данных или экземпляра (instance) в пассивное состояние (quiesce), создание и удаление

табличных пространств), SYSMAINT (второй уровень полномочий управления системой, включающий все операции поддержки работоспособности экземпля-

ра (instance); непосредственный доступ к данным этому пользователю запрещен, разрешены операции изменения конфигурационных файлов базы данных,

резервное копирование базы данных и табличных пространств, зеркалирование базы данных). Для каждой административной операции в IBM DB2 определен необходимый набор административных категорий, к которым должен принад-

лежать пользователь, выполняющий тот или иной запрос администрирования. Так, выполнять операции назначения привилегий пользователям может

SYSADM или DBADM, а для того чтобы создать объект данных, пользователь должен обладать привилегией CREATETAB.

Администратор каждой базы занимается созданием круга возможных пользователей создаваемой им БД и разграничением полномочий этих пользо-

вателей. Данные о разграничениях располагаются в системном каталоге БД. Очевидно, что данная информация может быть использована для несанкционированного доступа и поэтому подлежит защите. Защита этих данных осуществляется средствами самой СУБД.

СУБД позволяет зарегистрировать пользователя и хранить информацию о его уникальном идентификаторе. Например, подсистема безопасности Oracle

позволяет создавать пользователей базы данных посредством предложения: CREATE USER IDENTIFIED пользователь BY пароль

Подсистема безопасности IBM DB2 может использовать идентификаторы пользователей операционной системы; ее синтаксис SQL не содержит предло-

жения, аналогичного предложению CREATE USER. Microsoft SQL Server мо-

жет использовать аутентификацию как базы данных, так и операционной си-

стемы. Но мы не станем здесь обсуждать достоинства и недостатки выбранных производителями способов аутентификации — все они позволяют строить кор-

ректные схемы определения подлинности пользователей. Использование до-

полнительных средств аутентификации в рамках информационной системы не запрещается.

Набор привилегий можно определить для конкретного зарегистрированно-

го пользователя или для группы пользователей (это могут быть собственно группы пользователей, роли и т.п.). Объектом защиты может являться таблица,

представление, хранимая процедура и т.д. (подробный список объектов защиты

имеется в документации к используемой СУБД). Субъектом защиты может быть пользователь, группа пользователей или роль, а также хранимая процеду-

ра, если такое предусматривается используемой реализацией. Если из исполь- зуемой реализации следует, что хранимая процедура имеет «двойной статус» (она и объект защиты, и субъект защиты), то нужно очень внимательно рас-

смотреть возможные модели нарушителей разграничения прав доступа и предотвратить эти нарушения, построив, по возможности, соответствующую систему защиты.

При использовании хранимых процедур следует обращать особое внима-

ние на то, от имени какого пользователя выполняется данная хранимая проце- дура в каждом конкретном случае. Так, в Oracle до недавнего времени храни-

мые процедуры выполнялись от имени владельца хранимой процедуры, а не от имени пользователя, выполнившего ее вызов. Текущая версия Oracle предо-

ставляет возможность указать, под чьим именем будет выполняться вызванная хранимая процедура, пользователь же должен иметь только привилегию

EXECUTE для данной процедуры. В «Линтер», например, выполнение храни- мых процедур всегда происходит от имени пользователя, вызвавшего процеду- ру.

Привилегии конкретному пользователю могут быть назначены администратором явно и неявно, например через роль. Роль — это еще один возмож- ный именованный носитель привилегий. С ролью не ассоциируют перечень допустимых пользователей — вместо этого роли защищают паролями, если, ко-

нечно, такая возможность поддерживается производителем СУБД. Роли удобно использовать, когда тот или иной набор привилегий необходимо выдать (или отобрать) группе пользователей. С одной стороны, это облегчает администратору управление привилегиями, с другой — вносит определенный порядок в случае необходимости изменить набор привилегий для группы пользователей сразу. Нужно особо отметить, что при выполнении хранимых процедур и интерактивных запросов может существовать зависимость набора привилегий пользователя от того, как они были получены: явно или через роль. Имеют место и реализации, например в Oracle, где в хранимых процедурах используются привилегии, полученные пользователем явно. Если используемая вами реализация обладает подобным свойством, то изменение привилегий у группы пользователей следует реализовать как набор команд или как административную процедуру (в зависимости от предпочтений администратора).

Предложения управления привилегиями: назначение привилегии:

GRANT привилегия [ON объект] TO субъект [WITH GRANT OPTION] отмена привилегии:

REVOKE привилегия [ON объект] FROM субъект

Если субъект=пользователь, то привилегия назначается ему явно. Если субъект=роль, то для управления привилегиями используются соответственно: GRANT ROLE имя_роли [ON объект] TO субъект [WITH GRANT OPTION]

REVOKE ROLE имя_роли [ON объект] FROM субъект

Назначение привилегии всем пользователям системы осуществляется следующим образом:

GRANT привилегия [ON объект] TO PUBLIC

В этом случае каждый новый созданный пользователь автоматически получит такую привилегию. Отмена привилегии осуществляется так: REVOKE привилегия [ON объект] FROM PUBLIC

Необходимо иметь в виду, что некоторые реализации, например IBM DB2, используют группы пользователей, определенные в операционной системе. Поэтому следует обращать внимание на особенности реализации аналогов ролей в СУБД. Нужно выяснить, содержит ли реализация SQL-предложения вида:

CREATE ROLE имя_роли

DROP ROLE имя_роли

При управлении доступом к таблицам и представлениям набор привилегий в реализации СУБД определяется производителем.

Привилегии выборки и модификации данных:

SEL — привилегия на выборку данных;

INSERT — привилегия на добавление данных;

DELETE — привилегия на удаление данных;

UPDATE — привилегия на обновление данных (можно указать определенные столбцы, разрешенные для обновления).

Привилегии изменения структуры таблиц:

ALTER — изменение физической/логической структуры базовой таблицы (изменение размеров и числа файлов таблицы, введение дополнительного столбца и т.п.);

INDEX — создание/удаление индексов на столбцы базовой таблицы; ALL — все возможные действия над таблицей.

В реализациях могут присутствовать другие разновидности привилегий, например:

CONTROL (IBM DB2) — комплексная привилегия управления структурой таблицы,

REFERENCES — привилегия создания внешних ключей, RUNSTAT — выполнение сбора статистической информации по таблице и дру- гие.

Однако дискреционная защита является довольно слабой, так как доступ ограничивается только к именованным объектам, а не собственно к хранящимся данным. В случае реализации информационной системы с использованием реляционной СУБД объектом будет, например, именованное отношение (то есть таблица), а субъектом — зарегистрированный пользователь. В этом случае нельзя в полном объеме ограничить доступ только к части информации, хранящейся в таблице. Частично проблему ограничения доступа к информации решают представления и использование хранимых процедур, которые реализуют тот или иной набор бизнес-действий.

Представление (view) — это сформированная выборка кортежей, хранящихся в таблице (таблицах). К представлению можно обращаться точно так же, как и к таблицам, за исключением операций модификации данных, поскольку

некоторые типы представлений являются немодифицируемыми. Часто в реали-

зациях view хранится как текст, описывающий запрос выборки, а не собственно выборка данных; выборка же создается динамически на момент выполнения предложения SQL, использующего view. Но разграничить доступ, например, к

двум документам, которые удовлетворяют одному и тому же условию выборки, уже нельзя. Это связано с тем, что даже если ввести отдельный атрибут, кото-

рый будет хранить информацию о метке конфиденциальности документа, то средствами SQL можно будет получить выборку данных без учета атрибута данной метки. Фактически это означает, что либо сам сервер баз данных дол-

жен предоставить более высокий уровень защиты информации, либо придется реализовать данный уровень защиты информации с помощью жесткого ограни-

чения операций, которые пользователь может выполнить посредством SQL. На некотором уровне такое разграничение можно реализовать с помощью храни-

мых процедур, но не полностью — в том смысле, что само ядро СУБД позволя- ет разорвать связь «защищаемый объект Ц метка конфиденциальности».

Контрольный вопрос:

1. Как можно разделить пользователей СУБД на группы?
2. Дискреционная защита.

Лабораторная работа №10 Настройка механизмов аутентификации на основе ролей

Цель работы – исследовать настройки механизмов аутентификации на основе ролей.

В современных условиях любая деятельность сопряжена с оперированием большими объемами информации, которое производится широким кругом лиц.

Защита данных от несанкционированного доступа является одной из приоритетных задач при проектировании любой информационной системы. Следстви- ем возросшего в последнее время значения информации стали высокие требо- вания к конфиденциальности данных. Системы управления базами данных, в

особенности реляционные СУБД, стали доминирующим инструментом в этой области. Обеспечение информационной безопасности СУБД приобретает ре-

шающее значение при выборе конкретного средства обеспечения необходимого уровня безопасности организации в целом.

Для СУБД важны три основных аспекта информационной безопасности - конфиденциальность, целостность и доступность. Общая идея защиты базы данных состоит в следовании рекомендациям, сформулированным для класса безопасности С2 в «Критериях оценки надежных компьютерных систем»1.

Политика безопасности определяется администратором данных. Однако решения защиты данных не должны быть ограничены только рамками СУБД.

Абсолютная защита данных практически не реализуема, поэтому обычно довольствуются относительной защитой информации - гарантированно защищают ее на тот период времени, пока несанкционированный доступ к ней влечет какие-либо последствия. Разграничение доступа к данным также описывается в базе данных посредством ограничений, и информация об этом хранится в ее системном каталоге. Иногда дополнительная информация может быть запрошена из операционных систем, в окружении которых работают сервер баз данных и клиент, обращающийся к серверу баз данных.

Некоторые термины

Конфиденциальная информация (sensitive information) - информация, которая требует защиты.

Доступ к информации (access to information) - ознакомление с информаци- ей, ее обработка (в частности, копирование), модификация, уничтожение. Субъект доступа (access subject) - лицо или процесс, действия которого ре- гламентируются правилами разграничения доступа.

Объект доступа (access object) - единица информации автоматизированной системы, доступ к которой регламентируется правилами разграничения доступа. Объектами доступа (контроля) в СУБД является практически все, что содержит конечную информацию: таблицы (базовые или виртуальные), представления, а также более мелкие элементы данных: столбцы и строки таблиц и даже поля строк (значения). Таблицы базы данных и представления имеют владельца или создателя. Их объединяет еще и то, что все они для конечного пользователя представляются как таблицы, то есть как нечто именованное, содержащее информацию в виде множества строк (записей) одинаковой структуры. Строки таблиц разбиты на поля именованными столбцами.

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 использо-

ванием штатных средств, предоставляемых средствами вычислительной техни- ки или автоматизированными системами.

Идентификатор доступа (access identifier) - уникальный признак объекта или субъекта доступа.

Идентификация (identification) - присвоение объектам и субъектам доступа идентификатора и (или) сравнение предъявляемого идентификатора с перечнем присвоенных идентификаторов.

Пароль (password) - идентификатор субъекта, который является его секретом.

Аутентификация (authentification) - проверка принадлежности субъекту доступа предъявленного им идентификатора, подтверждение подлинности.

В СУБД на этапе подключения к БД производится идентификация и про-

верка подлинности пользователей. В дальнейшем пользователь или процесс по- лучает доступ к данным согласно его набору полномочий. В случае разрыва со-

единения пользователя с базой данных текущая транзакция откатывается, и при восстановлении соединения требуется повторная идентификация пользователя и проверка его полномочий.

Уровень полномочий субъекта доступа (subject privilege) - совокупность прав доступа субъекта доступа (для краткости в дальнейшем мы будем исполь-

зовать термин «привилегия»).

Нарушитель правил разграничения доступа (security policy violator) - субъект доступа, который осуществляет несанкционированный доступ к информа-

ции.

Модель нарушителя правил разграничения доступа (security policy violator model) - абстрактное (формализованное или неформализованное) описание нарушителя правил разграничения доступа.

Целостность информации (information integrity) - способность средства вы-

числительной 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.

Контрольные вопросы:

1. Определение доступа к информации.
2. Определение субъект доступа.
3. Определение объект доступа.

Лабораторная работа №11 Настройка механизмов авторизации на основе ролей

Цель работы – исследовать настройки механизмов авторизации на основе ролей.

В современных СУБД достаточно развиты средства дискреционной защи- ты.

Дискреционное управление доступам (discretionary access control) — разграничение доступа между поименованными субъектами и поименованными объектами. Субъект с определенным правом доступа может передать это право любому другому субъекту.

Дискреционная защита является многоуровневой логической защитой. Логическая защита в СУБД представляет собой набор привилегий или ролей по отношению к защищаемому объекту. К логической защите можно отне- сти и владение таблицей (представлением). Владелец таблицы может изменять

(расширять, отнимать, ограничивать доступ) набор привилегий (логическую защиту) . Данные о логической защите находятся в системных таблицах базы данных и отделены от защищаемых объектов (от таблиц или представлений).

Информация о зарегистрированных пользователях базы данных хранится в ее системном каталоге. Современные СУБД не имеют общего синтаксиса SQL-предложения соединения с базой данных, так как их собственный синтаксис сложился раньше, чем стандарт ISO. Тем не менее часто таким ключевым предложением является CONNECT. Ниже приведен синтаксис данного предложения для Oracle и IBM DB2 соответственно: CONNECT

[[logon] [AS {SYSOPER|SYSDBA}]]

пользователь/пароль[@база_данных]

CONNECT TO база_данных USER пользователь USING пароль

В данных предложениях отражен необходимый набор атрибутов, а также показано различие синтаксиса. Формат атрибута база_данных, как правило,

определяется производителем СУБД, так же как и имя пользователя, имеющего по умолчанию системные привилегии (SYSDBA/SYSOPER в случае Oracle).

Соединение с системой не идентифицированных пользователей и пользователей, подлинность идентификации которых при аутентификации не под-

твердилась, исключается. В процессе сеанса работы пользователя (от удачного прохождения идентификации и аутентификации до отсоединения от системы)

все его действия непосредственно связываются с результатом идентификации. Отсоединение пользователя может быть как

нормальным (операция DISCONNECT), так и

насильственным (исходящим от пользователя-

администратора, например в случае удаления пользователя или при аварийном обрыве канала связи клиента и сервера). Во втором случае пользователь будет проинформирован об этом, и все его действия аннулируются до последней фик-

сации изменений, произведенных им в таблицах базы данных. В любом случае на время сеанса работы идентифицированный пользователь будет субъектом доступа для средств защиты информации от несанкционированного доступа

(далее - СЗИ НСД) СУБД.

Следуя технологии открытых систем, субъект доступа может обращаться посредством СУБД к базе данных только из программ, поставляемых в дистри-

бутиве или подготовленных им самим, и только с помощью штатных средств системы.

Все субъекты контроля системы хранятся в таблице полномочий системы и разделены для системы на ряд категорий, например CONNECT, RESOURCE и

DBA. Набор таких категорий определяется производителем СУБД. Мы не слу-

чайно предлагаем указанный порядок рассмотрения — именно так происходит нарастание возможностей (полномочий) для каждого отдельного вида подклю-

чения:

CONNECT — конечные пользователи. По умолчанию им разрешено только соединение с базой данных и выполнение запросов к данным, все их действия регламентированы выданными им привилегиями;

RESOURCE — привилегированные пользователи, обладающие правом создания собственных объектов в базе данных (таблиц, представлений, синонимов, хранимых процедур).

Пользователь — владелец объекта обладает полным набором привилегий для управления данным объектом;

DBA — категория администраторов базы данных. Включает возможности обеих предыдущих категорий, а также возможность вводить (удалять) в си- стему (из системы) субъекты защиты или изменять их категорию.

Следует особо отметить, что в некоторых реализациях административные действия также разделены, что обусловливает наличие дополнительных категорий. Так, в Oracle пользователь с именем DBA является администратором сервера баз данных, а не одной-единственной базы данных. В СУБД «Линтер» компании РЕЛЭКС понятие администратора сервера баз данных отсутствует, а наличествует только понятие администратора конкретной базы данных. В IBM DB2 существует ряд категорий администраторов: SYSADM (наивысший уровень; системный администратор, обладающий всеми привилегиями); DBADM (администратор базы данных, обладающий всем набором привилегий в рамках конкретной базы данных). Привилегии управления сервером баз данных имеются у пользователей с именами SYSCTRL (наивысший уровень полномочий управления системой, который применяется только к операциям, влияющим на системные ресурсы; непосредственный доступ к данным запрещен, разрешены операции создания, модификации, удаления базы данных, перевод базы данных или экземпляра (instance) в пассивное состояние (quiesce), создание и удаление табличных пространств), SYSMAINT (второй уровень полномочий управления системой, включающий все операции поддержки работоспособности экземпляра (instance); непосредственный доступ к данным этому пользователю запрещен, разрешены операции изменения конфигурационных файлов базы данных,

резервное копирование базы данных и табличных пространств, зеркалирование базы данных). Для каждой административной операции в IBM DB2 определен необходимый набор административных категорий, к которым должен принад-

лежать пользователь, выполняющий тот или иной запрос администрирования.

Так, выполнять операции назначения привилегий пользователям может

SYSADM или DBADM, а для того чтобы создать объект данных, пользователь должен обладать привилегией CREATETAB.

Администратор каждой базы занимается созданием круга возможных пользователей создаваемой им БД и разграничением полномочий этих пользо-

вателей. Данные о разграничениях располагаются в системном каталоге БД. Очевидно, что данная информация может быть использована для несанкционированного доступа и поэтому подлежит защите. Защита этих данных осуществляется средствами самой СУБД.

СУБД позволяет зарегистрировать пользователя и хранить информацию о его уникальном идентификаторе. Например, подсистема безопасности Oracle

позволяет создавать пользователей базы данных посредством предложения: CREATE USER IDENTIFIED пользователь BY пароль

Подсистема безопасности IBM DB2 может использовать идентификаторы пользователей операционной системы; ее синтаксис SQL не содержит предло-

жения, аналогичного предложению CREATE USER. Microsoft SQL Server может использовать аутентификацию как базы данных, так и операционной си-

стемы. Но мы не станем здесь обсуждать достоинства и недостатки выбранных производителями способов аутентификации — все они позволяют строить кор-

ректные схемы определения подлинности пользователей. Использование до-

полнительных средств аутентификации в рамках информационной системы не запрещается.

Набор привилегий можно определить для конкретного зарегистрированно-

го пользователя или для группы пользователей (это могут быть собственно группы пользователей, роли и т.п.). Объектом защиты может являться таблица,

представление, хранимая процедура и т.д. (подробный список объектов защиты имеется в документации к используемой СУБД). Субъектом защиты может быть пользователь, группа пользователей или роль, а также хранимая процеду-

ра, если такое предусматривается используемой реализацией. Если из исполь- зуемой реализации следует, что хранимая процедура имеет «двойной статус» (она и объект защиты, и субъект защиты), то нужно очень внимательно рассмотреть возможные модели нарушителей разграничения прав доступа и предотвратить эти нарушения, построив, по возможности, соответствующую систему защиты.

При использовании хранимых процедур следует обращать особое внимание на то, от имени какого пользователя выполняется данная хранимая проце- дура в каждом конкретном случае. Так, в Oracle до недавнего времени храни-

мые процедуры выполнялись от имени владельца хранимой процедуры, а не от имени пользователя, выполнившего ее вызов. Текущая версия Oracle предо-

ставляет возможность указать, под чьим именем будет выполняться вызванная хранимая процедура, пользователь же должен иметь только привилегию

EXECUTE для данной процедуры. В «Линтер», например, выполнение храни- мых процедур всегда происходит от имени пользователя, вызвавшего процеду- ру.

Привилегии конкретному пользователю могут быть назначены администратором явно и неявно, например через роль. Роль — это еще один возмож- ный именованный носитель привилегий. С ролью не ассоциируют перечень допустимых пользователей — вместо этого роли защищают паролями, если, ко-

нечно, такая возможность поддерживается производителем СУБД. Роли удобно использовать, когда тот или иной набор привилегий необходимо выдать (или отобрать) группе пользователей. С одной стороны, это облегчает администра-

тору управление привилегиями, с другой — вносит определенный порядок в

случае необходимости изменить набор привилегий для группы пользователей сразу. Нужно особо отметить, что при выполнении хранимых процедур и ин-

терактивных запросов может существовать зависимость набора привилегий пользователя от того, как они были получены: явно или через роль. Имеют ме-

сто и реализации, например в Oracle, где в хранимых процедурах используются привилегии, полученные пользователем явно. Если используемая вами реали-

зация обладает подобным свойством, то изменение привилегий у группы поль-

зователей следует реализовать как набор команд или как административную процедуру (в зависимости от предпочтений администратора).

Контрольные вопросы:

1. Дискреционное управление доступом.

Лабораторная работа №12 Настройка механизмов управление правами и привилегиями пользователей на основе ролей

Цель работы – исследовать настройки механизмов управление правами и привилегиями пользователей на основе ролей.

В СУБД Oracle9*i* введено понятие *идентификатора клиента* (client identifier), который является значением, устанавливаемым в качестве атрибута сеанса. Все, что находится в этом атрибуте, в нашей ситуации мы можем использовать для представления имени *фактического* пользователя, такого, как JANE. И даже если имя пользователя базы данных зарегистрировано как APPUSER или что-то подобное, идентификатор клиента может иметь значение JANE.

Вы можете установить значение идентификатора клиента для сеанса с помощью поставляемой корпорацией Oracle пакетной процедуры dbms_session.set_identifier. Следующий оператор устанавливает в сеансе идентификатор SCOTT:

```
EXECUTE DBMS_SESSION.SET_IDENTIFIER ('SCOTT')
```

После установки этот идентификатор виден для данного сеанса в представлении V$SESSION:

```
SELECT CLIENT_IDENTIFIER FROM V$SESSION
WHERE AUDSID = USERENV('SESSIONID');
```

Здесь будет показано значение SCOTT, которое и было установлено ранее. Итак, какова важность этого значения? В дополнение к тому, что эта информа- ция находится в представлении V$SESSION, она также обнаруживается в среде FGA в таблице FGA_LOG$и, впоследствии, в представлении DBA_FGA_AUDIT_TRAIL в столбце CLIENT_ID, как показано ниже:

```
SELECT DB_USER, CLIENT_ID FROM DBA_FGA_AUDIT_TRAIL;
```

И даже если столбец DB_USER показывает *пользователя базы данных*, в журнале аудита FGA может быть найден *реальный пользователь приложения*, если его имя установлено в идентификаторе клиента. Это очень важно для по- нимания и применения во время аудита пользователей приложения.

Идентификатор клиента также находится в журнале обычного аудита, а не только в журнале детального аудита. Итак, во всех типах аудита, от обычного до расширенного детального, это значение обеспечивает недостающую связь для представления фактического пользователя. В случае обычного аудита иден-

тификатор клиента, если он установлен в сеансе, записывается в таблицу AUD$ в столбец CLIENTID. Это значение находится в зависимых представлениях, та- ких, как DBA_AUDIT_TRAIL, в столбце CLIENT_ID.

Проблема: как установить это значение должным образом. Помните, в веб-

приложениях пул соединений сеансы базы данных обслуживают много пользо-

вательских сеансов, а значит, установка идентификатора только один раз в начале сеанса не поможет. Вместо этого приложение должно устанавливать идентификатор клиента перед каждым обращением к базе данных, чтобы сред-

ства FGA могли видеть фактический идентификатор пользователя приложения.

Когда сеанс базы данных повторно используется другим потоком приложения,

в идентификатор клиента вводится идентификатор данного пользователя, что позволяет "опознать" его в журнале аудита FGA.

Другой подход состоит в том, чтобы создать "куку" (cookie) и использо-

вать это значение для установки идентификатора клиента. Этот способ более безопасен, поскольку "кука" может содержать другую аутентификационную информацию, которая не доступна в обычном сеансе. Значение в идентифика-

торе в этом случае может быть должным образом декодировано для получения реального имени пользователя, но аутентификационная информация добавляет к значению идентификатора клиента компонент целостности данных.

Контексты приложений

Использование идентификатора клиента имеет свои преимущества, но также и представляет серьезную угрозу безопасности: эта установка предпола-

гает, что пользователь установит в идентификаторе значение реального иден-тификатора пользователя, но этого нельзя гарантировать. Злонамеренный поль-

зователь может соединиться с базой данных и установить значение другого идентификатора пользователя, что серьезно подрывает достоверность журнала аудита. В веб-приложениях использование "куков" для хранения идентифика-

торов клиентов делает эту задачу трудной, если не невозможной; но в обычных приложениях использование только клиентских идентификаторов не обеспечи-

вает удовлетворительной защиты. Нам требуется более безопасное средство фиксации пользователей приложения в журнале аудита.

Предложим решение: *контексты приложений* (application contexts). Кон-

тексты приложений похожи на атрибуты сеанса: после установки к ним можно обращаться в сеансе в любое время. В другом сеансе может быть установлено другое значение, но эти значения не будут видны в других сеансах. Контекст имеет атрибуты, подобные столбцам таблицы; но, в отличие от таблиц, контек-

сты – не объекты сегментов, и атрибуты могут быть определены во время выполнения, а не во время проектирования.

Контекст приложения может быть создан следующим SQL- оператором: create context my_app_ctx using set_my_app_ctx;

Обратите внимание на предложение using set_my_app_ctx, указывающее,

что атрибутами внутри контекста можно манипулировать только с помощью процедуры с именем set_my_app_ctx, определяемой следующим образом:

```
create or replace procedure set_my_app_ctx (
p_app_user in varchar2 := USER
)
is begin
dbms_session.set_context('MY_APP_CTX','APP_USERID', p_app_user); end;
```

Эта процедура просто устанавливает в атрибуте APP_USERID значение,

передаваемое в ее входном параметре, вызывая для этой установки процедуру dbms_session.set_context. Что случится, если пользователь непосредственно вы-

зовет процедуру dbms_session.set_context?

```
SQL> exec dbms_session.set_context('MY_APP_CTX','APP_USERID', 'JUNE')
BEGIN dbms_session.set_context('MY_APP_CTX','APP_USERID', 'JUNE'); END;
*
ERROR at line 1:
ORA-01031: insufficient privileges
ORA-06512: at "SYS.DBMS_SESSION", line 78
ORA-06512: at line 1
```

Обратите внимание на ошибку ORA-01031:insufficient privileges (недостаточные привилегии), которая в данное время немного вводит в заблуждение.

Пользователь не имеетпривилегиивыполнения процедуры SYS.DBMS_SESSION, установка атрибута контекста вызовом этой процедуры запрещена, поэтому возникает ошибка. Однако когда пользователь для установки атрибута вызовет доверенную процедуру:

SQL> execute set_my_app_ctx ('AAAA') PL/SQL procedure successfully completed.

…она выполнится успешно. Атрибуты контекста могут устанавливаться только с помощью этой процедуры, поэтому она и называется соответствующим образом – *доверенная* процедура (trusted procedure). Это важное свойство контекста приложения, и оно может использоваться в FGA.

Атрибут контекста, если установлен (в приведенном выше коде), может быть извлечен вызовом функции SYS_CONTEXT:

select sys_context('MY_APP_CTX','APP_USERID') from dual;

Этот оператор возвращает значение атрибута. Если контекст может быть установлен безопасным способом, его можно использовать для установки идентификатора клиента.

Исходя из наших знаний, возможное решение может выглядеть следую- щим образом:

приложение выполняет процедуру, которая автоматически устанавливает правильное значение в контексте приложения. В вышеупомянутом примере атрибут контекста использовался для установки идентификатора пользователя,

но можно также использовать и другой атрибут, такой, как роль пользователя. В контексте вы можете определять более одного атрибута. Атрибут может использоваться как включаемая роль. В доверенную процедуру можно включить все типы проверок безопасности, обеспечивая защиту и аутентичность. Если эти проверки безопасности будут неудачными, требуемая роль не будет включена. Так, даже если пользователь может успешно соединиться с базой данных,

используя учетную запись APPUSER, он будет не в состоянии манипулировать данными, так как соответствующая роль не будет включена. Обратите внима-

ние: роли должны быть ролями, аутентифицируемыми процедурой, а не обыч-

ными. Такие роли создаются оператором CREATE ROLE <имя_роли> USING <имя_процедуры>; и пользователь включает роль, вызывая <имя_процедуры>,

а *не* оператором SET ROLE;

эта процедура также устанавливает идентификатор клиента, так что нет никакой потребности предоставлять привилегии выполнения пакета dbms_session всем пользователям (public), они не нужны даже данному пользо-

вателю. Поскольку пользователи не имеют привилегий вызова процедур пакета dbms_session, они не могут непосредственно устанавливать идентификаторы клиентов – те будут устанавливаться и переноситься в журнал детального ауди-

та автоматически.

Этот метод может быть в дальнейшем усовершенствован за счет использования определяемых пользователем записей аудита, в которых в идентификато- рах клиентов могут устанавливаться значения, извлекаемые из контекста при-

ложения. Мы обсудим данный метод в третьей (и последней) части этой серии статей.

Контрольные вопросы:

1. Идентификатор клиента в Oracel.

Лабораторная работа №13 Управление журналами событий и анализ результатов работы функций безопасности.

Цель работы – исследовать управление журналами событий и анализ результатов работы функций безопасности.

Локальная база

Самая простая база данных – локальная. В этом случае база и программа расположены на одном компьютере. Соединение с файлом базы данных проис-

ходит через спец драйвер или напрямую. Даже если подключение происходит через драйвер, то он умеет только обрабатывать простые запросы SQL стандар-

та 92-го года и предоставлять данные программе или сохранять изменения в таблице. Все остальные манипуляции могут выполняться только программой.

Таким образом, логика, данные и приложение работают как единое целое, и не могут быть разделены.

Рисунок 7 – Локальная база данных

Работа локальной базы данных

Яркими и наиболее распространенными представителями такого рода баз являются Dbase (файлы с расширением dbf), Paradox (расширение db) и Access (расширение mdb). Форматы Dbase и Paradox это даже не базы данных, а таб-

лицы, потому что в одном файле может храниться только одна таблица данных. Индексы, ускоряющие поиск и осуществляющие сортировку находятся в от- дельных файлах. Таким образом, одна база данных может состоять из множе-

ства файлов и это иногда приводит к определенным проблемам при поставке приложения конечному юзеру.

Файлы Access являются гибридом таблиц и баз данных. Здесь уже все таб-

лицы и индексы хранятся в одном файле, что намного удобнее в управлении. К

тому же среда управления базами Access наиболее удобна и доступна в любом офисном пакете от MS. В остальном, MS Access обладает теми же недостатка-

ми, что и остальные представители этого сословия.

Самый главный недостаток локальных баз данных, как говорит юморист М. Задорнов – «они тупые». Да, да. Качество и скорость доступа напрямую за-

висит от драйвера. Большинство из них не имели оптимизаторов SQL запросов,

и отсутствовало какое-либо кэшировании. Возможности железа использовались минимально, поэтому на больших базах запросы выполняются сверх медленно.

Таблицы Dbase и Paradox были разработаны слишком давно и самое слабое звено это индексы. В этих таблицах нет транзакций и соответствующего жур-

нала. После добавления новой записи, если драйвер не успел обработать изме-

нения в индексах, и произошла ошибка, пропал свет или зависон, то индекс рушится и для восстановления приходится использовать спец утилиты или пе-

реформировывать индексы. В базах Access у меня таких проблем не было, по- тому что тут индексы более защищены.

Что такое разрушенный индекс? Индекс – это колонка, в которой все зна-

чения строк обязательно уникальны. Чаще всего для этих целей используется простой счетчик. Допустим, что пользователь добавил запись и счетчик при-

своил ей значение 195, но само значение счетчика не изменилось. При добавле-

нии следующей записи счетчик снова пытается втулить нам число 195, но так как такая запись уже есть, происходит ошибка. Это и есть нарушение индекса,

и лечить его достаточно просто – переформировать индекс, но нудно.

Сетевая база данных

Почему локальные базы называют локальными? Да потому что с данными работает только один пользователь, а также база данных и программа находятся на одном компьютере. В случае с небольшими проектами, это нормальная си-

туация, но для больших объемов данных один оператор не справится с задачей и нужно, чтобы несколько человек могли работать с общими данными.

Решением этой проблемы стали сетевые базы данных. В принципе, это те же локальные базы, только выложены они на сетевой диск сервера (это может быть простой файловый сервер или компьютер с шарами), и несколько клиен-

тов обращаются к одной базе по сети.

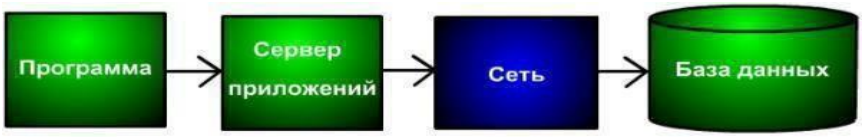


Рисунок 8 – Сетевая база данных

Сетевая модель доступа к данным

Давайте посмотрим, как происходит обращение к базе данных. Программа и драйвер находятся на клиенте, а данные находятся на сервере или просто на удаленном компьютере. Теперь подумаем, как программа получает данные.

Клиент передает драйверу SQL запрос, который должен быть выполнен, но данные ведь находятся удаленно. Чтобы отработать запрос, вся нужная таблица

(в случае с Access вся база данных, потому что все в одном файле) выкачивает- ся на компьютер клиента, где драйвер обрабатывает данные.

Я бы убил того, кто придумал такую технологию, потому что это самое настоящее издевательство над системой. Представляешь, что будет, если надо выполнить запрос на базе данных в 1 гига с телефонным соединением в 34кб/с?

Это то же самое, что заставить ЮКОС добывать нефть через трубочку для мо- лочных коктейлей.

А ведь некоторые российские компании (не будет тыкать пальцем) тулили нам сетевые решения на основе dbf файлов в области бухгалтерии, делопроизводства и экономики. Это же издевательство. Меня несколько раз просили восстановить умершие базы складской программы, после того, как встроенные в программу средства не справлялись с задачей.

Но страшнее всего начали вести себя индексы. У таблиц Paradox, если они находились на расшаренном диске Win95, мне приходилось ремонтировать индексы как минимум раз в неделю. Но когда я убрал файлы базы данных на сетевой диск сервера NetWare 3.11 (это был где-то 1998-й год), как сразу проблемы с нарушением индексации исчезли. Просто это действительно сервер, а не корявый Windows 9x.

При сетевом соединении многопользование получилось не полное. Изменения одного пользователя небыли видны другим, пока они не перезапустят программу или не переконнектятся. Почему? Да потому что именно в момент коннекта прога сосет все данные с сетевого диска. Потом мы работаем как бы с локальной версией и чужие изменения не видны.

Клиент сервер

Лохонувшись с сетевыми базами, монотонную модель наконец-то решили разделить на два уровня – приложение и база данных. Теперь база данных – это не просто таблица с данными, а целый движок, в задачи которого входит не только хранение данных, но и обработка запросов. Это уже не просто таблицы с информацией, это программа-сервер, которая берет на себя множество функций по обеспечению хранения, целостности данных и предоставлению удобных интерфейсов доступа к информации.

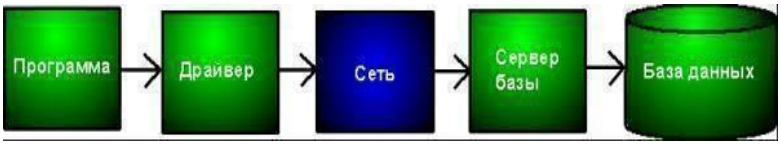


Рисунок 9 – Доступ к данным по технологии Клиент-Сервер

Доступ к данным по технологии Клиент-Сервер

В технологии клиент сервер, драйвер уже изменил свое назначение, и те- перь он уже должен только знать, как подключится к серверу и передать ему

запрос. Остальное уже ложится на плечи сервера. Такая технология намного сокращает трафик, особенно, при хорошем программировании. Допустим, что юзеру нужно увидеть, все данные, в которых имя определенной колонки со-

держит слова на букву «А». Клиенту достаточно направить серверу всего лишь такой текст:

SELECT *

FROM Имя таблицы WHERE Колонка LIKE _A%‘

Я думаю, что не надо даже считать, сколько кило занимает этот текст, и

как долго он будет отправляться по сети. Даже через медный провод с железом на 2400бод все произойдет практически мгновенно.

Сервер базы данных, получив запрос, разбирает его и придумывает для се- бя оптимальный план выполнения, в данном случае поиска нужных строк. Получив нужные данные, сервер возвращает только их и ничего больше.

Таким образом, клиент в любой момент может запросить у сервера нужные данные и нет необходимости гонять по сети всю базу данных. При хорошо по-

строенном приложении и оптимальных запросах, клиент сможет работать с ба-

зой данных любого размера даже через модем в 56 кбит/с. Неплохо? Я тоже сказал: «неужели мозги у особо умных заработали в нужном направлении».

Главное, запрашивать только то, что нужно и маленькими кусками и все будет в ажуре.

Особенности клиент-сервера

Возможности клиент серверных баз данных зависят от производителя. Са- мые простые возможности предоставляют такие базы как My SQL. В них сер-

вер имеет встроенный движок обработки запросов и основные возможности по обеспечению безопасности, распределении прав.

В более солидных клиент-серверных базах (MS SQL Server, Oracle и т.д.) есть следующие дополнительные возможности:

1. Вьюшки – более подробное о них мы поговорим в статье по безопасности;
2. Триггеры – это как бы функции, которые могут вызываться на определенные события (вставка, изменение и удаление данных). В этих функциях мо- жет производиться какая-то логика по обеспечению целостности данных
3. Репликация – объединение баз данных. Допустим, что у фирмы есть два офиса и в каждом из них своя база. Настроив репликацию, обе базы могут автоматически сливаться в одну в главном офисе или обмениваться изменениями по расписанию.
4. Хранимые процедуры и функции, которые выполняются на сервере по мизерному запросу клиента. Они могут содержать целые подпрограммы с логикой, которые будут выполнять какие-либо действия. Для написания таких программ используется уже не просто язык SQL, а его расширение – Transact-SQL (для MS баз) и PL/SQL (для Oracle и др.).

Список возможностей зависит от конкретной базы данных, ее навороченности и может быть больше или меньше.

Индексирование данных на сервере

За счет наличия в серверных базах данных управления транзакциями, про проблемы с индексами можно забыть. Допустим, что пользователь добавил запись. В этот момент начинается транзакция (не явная), в течение которой происходят все необходимые действия по сохранению данных. Если что-то пошло неправильно и сохранение не прошло до конца, то все изменения откатываются, и ничего в работе сервера не нарушается.

Транзакции могут быть и явными, когда программист сам указывает, где начало и конец и в них может выполняться несколько операций изменения или добавления данных. В этом случае сервер в случае ошибки в указанном блоке откатит любые изменения всех операций, сделанные во время выполнения явной транзакции.

В локальных базах данных индексы хранятся линейно. Это как колонка из упорядоченных данных, и для строк, это то же самое, что выстроить все слова по алфавиту. Конечно же, такой индекс упрощает поиск. Когда происходит сканирование по индексу и программа видит, что уже пошло слово больше, чем

задано в условии поиска, сканирование может прекращаться и не придется про-

сматривать всю базу данных. Например, вам надо найти слово «Абажур». Оно будет где-то в начале и чтобы его найти, нужно просканировать всего лишь начало таблицы, не дальше, чем все слова на букву А. За счет того, что данные упорядочены, мы можем быть уверенными, что все остальные слова будут на буквы Б, В и т.д.

В случае с серверной базой, индексы чаще всего (зависит от базы и типа индекса) хранятся немного по-другому – в виде дерева. Сколько слов надо про-

верить для поиска слова Якорь в базе данных при линейном индексе? Да прак- тически все :(. При древовидном хранении индекса не более чем для слова

«Абажур». Для пояснения древообразного индекса рассмотрим классическую задачу (в реальности все немного сложнее, но идея такая же). В самом верху дерева хранятся алфавит. Программа находит букву А, и спускается на уровень ниже. Здесь она находит все слова на буквы АБ и двигается еще ниже. И так пока не найдется нужное слово.

Таким образом, даже если нужное слово находится в самом конце, его по- иск будет не намного дольше, чем поиск слова из начала таблицы.

Трехуровневые базы данных

Многие программисты, которых я знаю, способны работать только с двух уровневой моделью, т.е. клиент-серверными приложениями. Нет, это не пото-

му, что они больше ничего не знают, просто не видят преимуществ 3-х уровне- вой модели. К тому же не хотят мучиться с лишними проблемами, а ведь буду- щем, во время сопровождения программ, три уровня спасают от лишних болез- ней анального отверстия.

Я работал на одной фирме (не будем в нее тыкать вилами), у которой было несколько офисов по России и в каждом из них парк компьютеров из 20-30

штук. В московском офисе эта цифра превышала сотню. Корпоративные про-

граммы обновлялись каждые две недели (вносились изменения, добавления и т.д.). Бедные админы в момент обновлений работали по субботам, чтобы про-

патчить софт на каждой машине и убедится в функциональности. Как решить эту проблему?

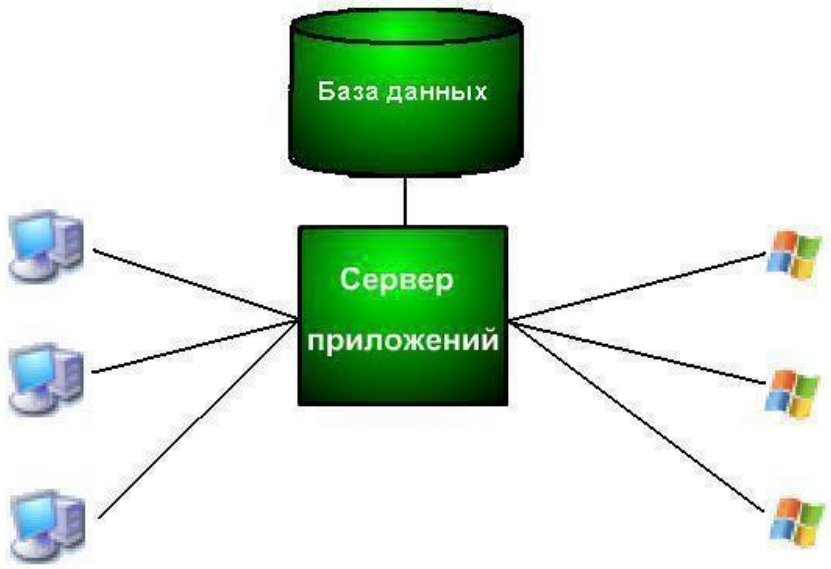


Рисунок 10 – Трехуровневая система

Трехуровневая система

Самое простое – использовать трех уровневую систему: клиент, сервер логики (умники любят выражаться «бизнес логика») и сервер приложения. В такой системе вся логика собрана в сервере приложений. Если что-то изменилось в базе данных или в логике обработки данных, достаточно обновить его и все клиенты будут работать по-новому без каких-либо патчей.

Преимущество такой системы состоит еще и в том, что на клиентских машинах не нужно держать драйвера доступа к каким либо базам. Клиенты должны только знать, где находится сервер приложений, уметь к нему подключится и правильно отобразить данные.

Представим себе классическую задачу – появление новой версии базы данных или переход на базу качественно более нового уровня. Ну не хватает уже возможностей MySQL и захотелось получить всю мощь от Oracle!!! Для этого переустанавливается сервер баз данных, изменяется сервер приложений на подключение к новой базе и клиенты готовы к работе. Их обновлять не надо!!!

Но самое интересное, клиентская программа может быть какой угодно. Можно написать сценарии, которые позволят работать с сервером приложении

прямо из браузера. В этом случае с базой смогут работать пользователи на лю- бой платформе (Windows, Linux и т.д.).

Логика 3-х уровневых приложений

Не смотря на наличие сервера приложений, нет смысла засовывать в него всю логику обработки данных. Если вы используете мощную базу данных, ко-

торая поддерживает хранимые процедуры и функции, то лучше переложить часть логики на сервер базы. В этом случае, внесенные изменения в хранимый код вступают в силу моментально и не надо даже обновлять сервер приложе-

ний.

Если в сети не так уж и много компьютеров (не более 20), и достаточно мощный сервер, то можно сервер приложений и базу данных расположить на одном физическом сервере. В этом случае обмен данными между сервером приложений и базой будет происходит внутри одного компьютера, а не по сети,

что может существенно снизить нагрузку на сетевое оборудование. Допустим, что сервер приложений и база данных находятся на разных сер- верах. Результаты запросов будут сначала идти через коммутатор от базы дан-

ных к серверу приложений, а затем через тот же коммутатор к компьютерам клиентов. Таким образом, по сети дважды пролетают одни и те же данные.

Чтобы от этого избавится, я чаще всего объединяю в одном физическом сервере логику и данные.

Контрольные вопросы:

1. Локальная база

2. Сетевая база данных

Лабораторная работа №14 Методы обеспечения целостности систем баз данных

Цель работы – исследовать методы обеспечения целостности систем баз данных.

Исследования проблем информационной безопасности показывают, что среди максимальных по объему потерь различных организаций от атак разных видов есть такие, которые напрямую являются следствием недостаточной за-

щиты информации в базах данных (БД), а именно: кража конфиденциальных данных; неавторизованный доступ к данным.

Проблема защиты данных в СУБД, как и в любой другой информационной системе, может быть разложена на три задачи по обеспечению:

конфиденциальности данных; целостности данных; доступности данных.

Рассмотрим задачи обеспечения конфиденциальности и целостности данных. В данной статье под целостностью будем подразумевать неизменность данных в процессе их хранения и обработки, а не логическую целостность (непротиворечивость) данных, которой будет посвящена отдельная статья.

Объекты защиты

Для начала рассмотрим, на каких уровнях можно защитить информацию в БД, т. е. определим объекты защиты. Во-первых, мы можем защищать данные, хранящиеся в различных структурных элементах БД, т. е. защищать (см. рис. 1): целиком базы данных;

отдельные таблицы БД; записи конкретных таблиц БД;

значения конкретных полей таблиц или записей.

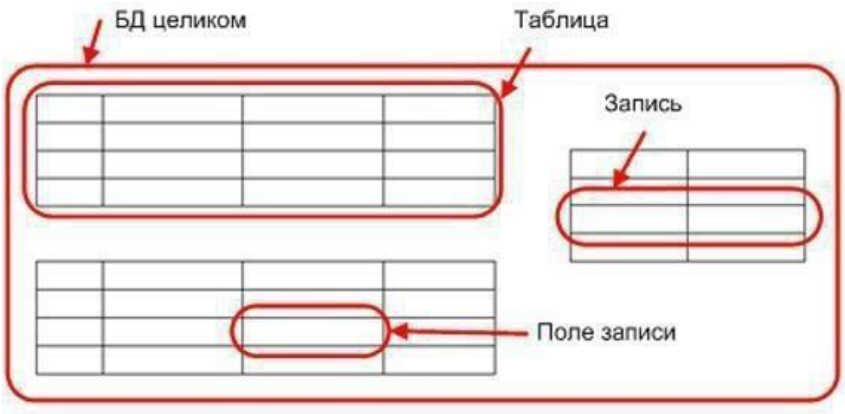


Рисунок 11 – Защита элементов БД

Во-вторых, логические структурные элементы БД физически хранятся на жестких дисках и передаются по компьютерным сетям, следовательно, мы можем защищать данные по месту их физического воплощения, например: файлы данных;

программные модули СУБД (от модификации и внедрения программных закладок, изменяющих логику работы СУБД в каких-либо целях атакующего);

каналы взаимодействия между компонентами СУБД (особенно актуально в случае распределенных СУБД) и каналы взаимодействия между СУБД и пользователями.

Защищать программные модули, а также информацию, хранящуюся на жестком диске и передаваемую по сети, можно точно так же, как и любую аналогичную информацию, не относящуюся к СУБД. Например, с помощью моде- лей доступа, а также с помощью криптографических методов:

шифрование файлов и сетевого трафика поможет обеспечить конфиденциальность;

методы контроля целостности (например, хэширование) позволят сохранять целостность данных и программных модулей.

Более интересна задача выборочной защиты данных, хранящихся в конкретных полях или записях БД. В данном случае какие-либо методы защиты на файловом уровне не помогают и кажется необходимым использовать специфичные для БД методы защиты, в частности: представления (views);

триггеры;

встроенные функции шифрования данных.

Рассмотрим поочередно данные методы. Представления

«Представление – это поименованная динамически поддерживаемая сервером выборка из одной или нескольких таблиц» [1]. Иными словами, представ- ление является виртуальной таблицей, все записи которой формируются в про- цессе обращения пользователя к представлению согласно тому запросу, кото- рый был сопоставлен данному представлению. Таким образом, пользователь получает доступ не к целой таблице (или к не- скольким таблицам), а, в простейшем случае, к совокупности столбцов или за- писей, которые определены в представлении.

Например, существует таблица, которая содержит основную информацию о преподавателях института:

Таблица 4. Основная информация о преподавателях института

Prep

NP	***Name***	***Position***	***ChartNum***	***Salary***
1	Иванов	Ст. преп.	403	20 000
2	Петров	Доцент	556	10 000
3	Сидоров	Профессор	212	30 000

И есть три сотрудника института, которым необходимы данные из этой таблицы:

работник кадровой службы, который не должен иметь доступ к столбцу зарплаты (salary);

работник бухгалтерии, который должен иметь доступ ко всем полям, но только в том случае, если преподаватель не относится к высокооплачиваемым ;

главный бухгалтер, который должен иметь доступ ко всем полям всех записей.

Такое разграничение доступа решается с помощью представлений очень просто:

главный бухгалтер работает непосредственно с таблицей Prep; работник кадровой службы работает с представлением PrepView1, в ко- тором отсутствует столбец Salary;

работник бухгалтерии работает с представлением PrepView2, в котором отсутствуют записи о преподавателях с высокой зарплатой.

Упомянутые представления PrepView1 и PrepView2 могут быть созданы следующими простыми запросами (использована нотация СУБД MySQL 5.0 [2]):

CREATE VIEW PrepView1 AS SELECT NP, Name, Position, ChartNum FROM Prep;

CREATE VIEW PrepView2 AS SELECT NP, Name, Position, ChartNum, Salary FROM Prep WHERE Salary <= 20000;

Исходя из приведенного выше примера, можно смело утверждать, что представление является простейшим в использовании методом защиты как конфиденциальности, так и целостности данных, который позволяет:

четко ограничить данные, доступные пользователю;

контролировать тот набор данных, который пользователь имеет право модифицировать.

Контрольные вопросы:

1. Объекты защиты.

Лабораторная работа №15 Технологии обеспечения целостности систем баз данных

Цель работы – исследовать технологии обеспечения целостности систем баз данных.

Триггеры

«Триггер (trigger) – это хранимая процедура особого типа, которую пользователь не вызывает непосредственно, а исполнение которой обусловлено наступлением определенного события (действием) – по сути добавлением INSERT или удалением DELETE строки в заданной таблице, или модификации UPDATE данных в определенном столбце заданной таблицы реляционной базы данных» [3].

Таким образом, триггер – это некая подпрограмма, срабатывающая автоматически в случае модификации данных в таблице. Причем, триггер может срабатывать как до, так и после наступления конкретного события, что определяется ключевыми словами AFTER и BEFORE соответственно.

С точки зрения защиты данных в БД триггеры позволяют:

проверить полномочия пользователя – имеет ли пользователь право на модификацию конкретных данных; важно, что такая проверка может быть выполнена до модификации данных, которую триггер может отменить при недостаточности прав пользователя на данную операцию; протоколировать (например, в отдельной таблице аудита) все события, связанные с модификацией данных в защищаемой таблице – это может быть крайне полезным, например, при поиске пользователя, выполнившего несанкционированную модификацию данных.

В [3] приведен пример простейшего триггера, который записывает в таблицу аудита (info) информацию о произошедшем изменении строки таблицы district, для которой определяется триггер (нотация СУБД Oracle):

```
CREATE OR REPLACE TRIGGER DistrictUpdatedTrigger AFTER UPDATE ON district FOR EACH ROW
BEGIN
INSERT INTO info VALUES ('one string in table "district" has changed'); END;
```

И снова отметим, что триггеры крайне просты в использовании, но при этом позволяют эффективно контролировать целостность данных, а также протоколировать события их модификации.

Функции шифрования

Стоит отметить, что встроенные функции шифрования присутствуют далеко не во всех СУБД. Следовательно, универсальным данный метод назвать нельзя.

Рассмотрим функции шифрования на примере СУБД MySQL 5.0 [2]. Данная СУБД предлагает два однотипных набора функций шифрования, в одном из которых реализован алгоритм DES, а в другом – AES. Кроме того, в MySQL реализовано несколько алгоритмов хэширования.

Таблица 5. Набор криптографических функций данной СУБД

	Функция	*Назначение*
	AES_ENCRYPT()	Зашифрование данных алгоритмом AES.
	AES_DECRYPT()	Расшифрование данных алгоритмом AES.
	DES_ENCRYPT()	Зашифрование данных алгоритмом DES.
	DES_DECRYPT()	Расшифрование данных алгоритмом DES.
	ENCRYPT()	Зашифрование данных функцией crypt().
	MD5()	Хэширование данных алгоритмом MD5.
	SHA1() или SHA()	Хэширование данных алгоритмом SHA-1.

Функции шифрования данных алгоритмом AES используют 128-битный ключ шифрования, т. е. шифрование ключами размером 192 и 256 бит, предусмотренными стандартом AES [4], в MySQL не реализовано. Ключ шифрования задается явным образом как один из параметров функции.

В отличие от них, функции DES_ENCRYPT() и DES_DECRYPT(), которые шифруют алгоритмом TripleDES, помимо явного задания ключа шифрования,

допускают простейший вариант управления ключами в виде ключевого файла,

содержащего пронумерованные значения ключей. Однако, данные функции по умолчанию выключены, для их использования необходимо включить поддержку протокола SSL в конфигурации СУБД.

Функция ENCRYPT() может быть использована только в операционных системах семейства Unix, поскольку она шифрует данные с помощью системного вызова crypt().

Что касается используемых функций хэширования, то в документации на MySQL [2] содержится предупреждение о том, что лежащие в их основе алгоритмы взломаны (подробно об этом написано, в частности, в [5]), поэтому использовать их следует с осторожностью. Однако, MySQL пока не предлагает более стойких функций хэширования взамен существующих.

Перечисленные выше криптографические функции также весьма просты в ис- пользовании. Например, следующий запрос помещает в таблицу table значение ―text‖, зашифрованное на ключе ―password‖ [2]:

INSERT INTO table VALUES (1, AES_ENCRYPT('text', 'password')). Отметим, что формат поля, в которое записывается зашифрованное значение, должен соответствовать ограничениям, накладываемым используемым криптоалгоритмом – в данном случае, оно должно быть двоичным (например, типа VARBINARY) и предполагать выравнивание в соответствии со 128- битным размером блока алгоритма AES.

Контрольные вопросы:

1. Триггеры.

Список рекомендуемой литературы:

1. А. Лихоносов. Безопасность баз данных. Учебное пособие. М., 2010.

2. MySQL 5.0 Reference Manual..

3. Триггер (базы данных). Материалы из Википедии – свободной энцикло- педии.

4. FIPS Publication 197. Specification for the Advanced Encryption Standard.– November 26, 2001.

S. Panasenko. SHA Hash Functions: History & Current State.– 2009