Основы построения защищенных баз данных

Ваша команда по спасению компьютерной безопасности

16 апреля 2024 г.

Самый надежный в мире алгоритм
Всегда делать исключительно то, что горит
В самый прекрасный последний момент
Ведь для самых важных дел в принципе лучше времени нет

Anacondaz - Факап

Содержание

1 Концепция безопасности БД

1.0.1 Понятие безопасности БД

Для того чтобы иметь общую точку старта нам придется дать пару определений, я постараюсь быстро разобраться с обязательной копипастой и перейти к делу. Итак:

База данных[1]– это организованная коллекция данных, обычно хранящихся и доступных в электронном виде из компьютерной системы. Там, где базы данных более сложны, они часто разрабатываются с использованием формальных методов проектирования и моделирования.

Система управления базами данных (СУБД)[1] – это программное обеспечение, которое взаимодействует с конечными пользователями, приложениями и самой базой данных для сбора и анализа данных. Программное обеспечение СУБД дополнительно включает в себя основные средства, предоставляемые для администрирования базы данных. Общая сумма базы данных, СУБД и связанных приложений может называться «системой базы данных». Часто термин «база данных» также используется для обозначения любой СУБД, системы баз данных или приложения, связанного с базой данных.

Но если вас вдруг спросят, то смело отвечайте:

Согласно гражданскому кодексу Российской Федерации (часть четвертая) от 18.12.2006 N 230-ФЗ (ред. от 18.07.2019), базой данных является представленная в объективной форме совокупность самостоятельных материалов (статей, расчетов, нормативных актов, судебных решений и иных подобных материалов), систематизированных таким образом, чтобы эти материалы могли быть найдены и обработаны с помощью электронной вычислительной машины (ЭВМ).

И если с просто базами данных все более-менее понятно, то в тот момент, когда нам приходиться говорить о вопросах безопасности, все превращается в классическую задачу о двух стульях: как надо и как в законе написано. (Ну или пафосно перефразируя) Вопросы информационной безопасности баз данных целесообразно рассматривать с двух взаимодополняющих позиций (Лихоносов А. Г. 2011)[2]:

- оценочные стандарты, направленные на классификацию информационных систем и средств их защиты по требованиям безопасности
- технические спецификации, регламентирующие различные аспекты реализации средств защиты

Попытка защищать все и сразу обречена на провал, так что довольно логичным кажется сосредоточится на обеспечении безопасности четырех уровней информационной системы (Лихоносов А. Г. 2011) :

1. уровня прикладного программного обеспечения, отвечающего за взаимодействие с пользователем
2. уровня системы управления базами данных, обеспечивающего хранение и обработку данных информационной системы
3. уровня операционной системы, отвечающего за функционирование СУБД и иного прикладного программного обеспечения
4. уровня среды доставки, отвечающего за взаимодействие информационных серверов и потребителей информации

Теперь уже можно ввести недостающие определения. Я вижу как вы соскучились по определениям[3].

Если БД рассматривать только как совокупность данных, то можно использовать следующее определения:

- **Безопасность информации [данных]**: Состояние защищенности информации [данных], при котором обеспечены ее [их] конфиденциальность, доступность и целостность.
- **Информационная система (ИС)** – система, предназначенная для хранения, поиска и обработки информации, и соответствующие организационные

[1] Нагло скопипизженно с вечно загнивающей
[2] Мне стыдно указывать ресурс, где я взял этот кусок, так что пусть будет pornhub
[3] А эти определения я взял с прошлого года

ресурсы (человеческие, технические, финансовые и т. д.), которые обеспечивают и распространяют информацию.

Если БД рассматривать как информационную систему, то можно использовать следующие определения:

- **Безопасность ИС (БД)** можно определить как состояние защищенности ИС от угроз ее нормальному функционированию. Под защищенностью понимается наличие средств ИС и методов их применения, обеспечивающих снижение или ликвидацию негативных последствий, связанных с реализацией угроз. Изложенный подход к определению понятия безопасности ИС предполагает, что перечень и содержание угроз достаточно хорошо определены и достаточно стабильны во времени.
- **Безопасность ИС (БД)** можно определить как свойство системы адаптироваться к агрессивным проявлениям среды, в которой функционирует система, обеспечивающее поддержку на экономически оправданном уровне характеристики качества системы. В сформулированном определении основной акцент делается не на перечне и содержании угроз, нейтрализация которых обеспечивается, а на особую характеристику качества системы. При этом основной критерий качества ИС является экономическим, т.е. оценка средств и методов обеспечения безопасности осуществляется на основе затрат на реализацию механизмов безопасности и потенциальных выгод от недопущения ущерба, связанного с целенаправленным или случайным агрессивным проявлением среды.

Ну и не одно введение не может обойтись без заклинания

Проблема обеспечения безопасности автоматизированных информационных систем может быть определена как решение трех взаимосвязанных задач по реализации требуемого уровня:

- **конфиденциальности** – обеспечения пользователям доступа только к данным, для которых пользователь имеет явное или неявное разрешение на доступ
- **целостности** – обеспечения защиты от преднамеренного или непреднамеренного изменения информации или процессов ее обработки
- **доступности** – обеспечения возможности авторизованным в системе пользователям доступа к информации в соответствии с принятой технологией

1.0.2 Угрозы безопасности БД: общие и специфичные

Интуитивное понимание угрозы безопасности можно сформулировать как нарушение великолепной тройки: ~~Трус, Балбес и Бывалый~~ конфиденциальность, целостность, доступность. Для фор-

мального диалога можно использовать что-то около[4]:

> **Угрозой информационной безопасности** автоматизированный информационной системе (АИС) назовем возможность воздействия на информацию, обрабатываемую в системе, приводящего к искажению, уничтожению, копированию, блокированию доступа к информации, а также возможность воздействия на компоненты информационной системы, приводящего к утрате, уничтожению или сбою функционирования носителя информации или средства управления программно-аппаратным комплексом системы.

Для того, чтобы разобраться во всем зоопарке перечисленных воздействий на систему нам придется все это дело классифицировать. Для удобства будем сразу смотреть на это со стороны обороняющего, то есть по источнику воздействия. В этом контексте они довольно естественно разбиваются на внутренние и внешние. Для описания внешних угроз необходимо учитывать объекты воздействия. (Под объектами воздействия понимаются объекты, которые могут подвергнуться атакам или могут стать причиной их возникновения.) (Утебов Данияр Рашидович 2008)

Внешними дестабилизирующими факторами, создающими угрозы безопасности функционированию систем баз данных и СУБД, являются:

- умышленные, деструктивные действия лиц с целью искажения, уничтожения или хищения программ, данных и документов системы, причиной которых являются нарушения информационной безопасности защищаемого объекта
- искажения в каналах передачи информации, поступающей от внешних источников, циркулирующих в системе и передаваемой потребителям, а также недопустимые значения и изменения характеристик потоков информации из внешней среды и внутри системы
- сбои и отказы в аппаратуре вычислительных средств
- вирусы и иные деструктивные программные элементы, распространяемые с использованием систем телекоммуникаций, обеспечивающих связь с внешней средой или внутренние коммуникации распределенной системы баз данных
- изменения состава и конфигурации комплекса взаимодействующей аппаратуры системы за пределы, проверенные при тестировании или сертификации системы

Внутренними источниками угроз безопасности баз данных и СУБД являются:

- системные ошибки при постановке целей и задач проектирования автоматизированных информационных систем и их компонент, допущенные при формулировке требований к функциям и характеристикам средств обеспечения безопасности системы
- ошибки при определении условий и параметров функционирования внешней среды, в которой предстоит использовать информационную систему и, в частности, программно-аппаратные средства защиты данных
- ошибки и несанкционированные действия пользователей, административного и обслуживающего персонала в процессе эксплуатации системы

[4]Нагло взято отсюда

- недостаточная эффективность используемых методов и средств обеспечения информационной безопасности в штатных или особых условиях эксплуатации системы

Если у вас разбегаются глаза, это нормально – здесь перечислены атаки на всех уровнях . Что за уровни? Напоминаю:

- На уровне сети
  - Activex-объект
  - Интерфейсы: OLE DB, ADO, ODBC, JDBC
  - Протоколы: TCP/IP, IPX/SPX, Named Pipes, Multiprotocol
  - Рабочие станции
  - Серверы
  - Маршрут
  - URL
- На уровне ОС
  - Аппаратное обеспечение
  - Программное обеспечение
  - Файлы базы данных
  - Файлы журнала транзакций
  - Файлы резервного копирования
  - Transact-SQL, PLSQL
  - Службы: MSSQLServer, SQLServerAgent, TNSListener и т. д.
- На уровне БД
  - Пользователи
  - Роли
  - Роли приложения
  - Диаграммы
  - Представления
  - Таблицы
  - Хранимые процедуры
  - Определения по умолчанию
  - Правила
  - Функции
  - Тип данных

Дальше, также легко и непринужденно можно разбить все атаки на СУБД на:

1. атаки на уровне ОС
2. атаки на уровне сети
3. атаки на уровне БД

Атаки на ОС, в которых функционирует СУБД, возникают гораздо чаще, так как защитить ОС гораздо сложнее, чем СУБД. Это обусловлено тем, что число различных типов защищаемых объектов в современных ОС может достигать нескольких десятков, а число различных типов защищаемых информационных потоков – нескольких сотен. Возможность практической реализации той или иной атаки на ОС в значительной мере определяется архитектурой и конфигурацией ОС. Тем не менее существуют атаки, которые могут быть направлены практически на любые ОС:

1. Кража ключевой информации (паролей)
2. Подбор пароля
3. Сканирование жестких дисков компьютера
4. Превышение полномочий
5. Атаки класса «Отказ в обслуживании»

Наиболее опасные атаки на СУБД исходят из сетей. На уровне сетевого программного обеспечения возможны следующие атаки на СУБД:

1. Прослушивание канала
2. Перехват пакетов на маршрутизаторе
3. Создание ложного маршрутизатора
4. Навязывание пакетов
5. Атаки класса «Отказ в обслуживании»

Теперь спускаемся на уровень самой БД. Для простоты восприятия разобьем все кучкам угроз конфиденциальности, целостности и доступности(Опять же, согласно (Утебов Данияр Рашидович 2008)):

- К угрозам конфиденциальности информации можно отнести следующие :
    1. Инъекция SQL. Во многих приложениях используется динамический SQL – формирование SQL-предложений кодом программы путем конкатенации строк и значений параметров. Зная структуру базы данных, злоумышленник может либо выполнить хранимую программу в запросе, либо закомментировать «легальные» фрагменты SQL-кода, внедрив, например, конструкцию UNION, запрос которой возвращает конфиденциальные данные. В последнее время злоумышленник может использовать специальные программы, автоматизирующие процесс реализации подобных угроз.

2. Логический вывод на основе функциональных зависимостей. Пусть дана схема отношения: $R(A_1,\ldots,A_n)$. Пусть $U = \{A_1,\ldots,A_n\}$, X,Y – подмножества из U. X функционально определяет Y, если в любом отношении r со схемой $R(A_1,\ldots,A_n)$ не могут содержаться два кортежа с одинаковыми значениями атрибутов из X и с различными из Y. В этом случае имеет место функциональная зависимость, обозначаемая $X \Rightarrow Y$. В реальных БД при наличии сведений о функциональных зависимостях злоумышленник может вывести конфиденциальную информацию при наличии доступа только к части отношений, составляющих декомпозированное отношение.

3. Логический вывод на основе ограничений целостности. Для кортежей отношений в реляционной модели данных можно задать ограничения целостности – логические условия, которым должны удовлетворять атрибуты кортежей. При этом ограничение целостности может быть задано в виде предиката на всем множестве атрибутов кортежа. В случае попытки изменить данные в таблице, СУБД автоматически вычисляет значение этого предиката, и в зависимости от его истинности операция разрешается или отвергается. Многократно изменяя данные и анализируя реакцию системы, злоумышленник может получить те сведения, к которым у него нет непосредственного доступа. К этому виду угроз можно отнести также анализ значений первичных/вторичных ключей.

4. Использование оператора UPDATE для получения конфиденциальной информации. В некоторых стандартах SQL пользователь, не обладая привилегией на выполнение оператора SELECT, мог выполнить оператор UPDATE со сложным логическим условием. Так как после выполнения оператора UPDATE сообщается, сколько строк он обработал, фактически пользователь мог узнать, существуют ли данные, удовлетворяющие этому условию.

- К угрозам целостности информации, специфические для СУБД можно отнести следующие :

  1. С помощью SQL-операторов UPDATE, INSERT и DELETE можно изменить данные в СУБД. Опасность заключается в том, что пользователь, обладающий соответствующими привилегиями, может модифицировать все записи в таблице.

- К угрозам доступности для СУБД можно отнести следующие:

  1. Использование свойств первичных и внешних ключей. В первую очередь отнесем сюда свойство уникальности первичных ключей и наличие ссылочной целостности. В том случае, если используются натуральные, а не генерируемые системой значения первичных ключей, может создаться такая ситуация, когда в таблицу невозможно будет вставить новые записи, так как там уже будут записи с такими же значениями первичных ключей. Если в БД поддерживается ссылочная целостность, можно организовать невозможность удаления родительских записей, умышленно создав подчиненные записи.

  2. Блокировка записей при изменении. Заблокировав записи или всю таблицу, злоумышленник может на значительное время сделать ее недоступной для обновления.

  3. Загрузка системы бессмысленной работой. Злоумышленник может выполнить запрос, содержащий декартовое произведение двух больших отношений. Мощность декартового произведения двух отношений мощности N_1 и N_2 равна $N_1 \cdot N_2$. Это означает, что при

выдаче злоумышленником запроса вида SELECT * FROM Tab1, Tab1 ORDER BY 1, где мощность отношения (количество строк в таблице Tab1) $N_1 = 10000$, мощность результирующего отношения будет $N = N_1^2 = 10000^2$. Вычисление соединения и сортировка результирующего отношения потребуют значительных ресурсов системы и отрицательно скажутся на производительности операций других пользователей.

4. Использование разрушающих программных средств. Например, атака типа «троянский конь» – запуск пользователями программ, содержащих код, выполняющий определенные действия, внедренный туда злоумышленником.

1.0.3 Требования безопасности БД

Если сильно постараться то, на основании угроз можно выделить следующий список требований к безопасности БД[5]:

- Функционирование в доверенной среде. Под доверенной средой следует понимать инфраструктуру предприятия и ее защитные механизмы, обусловленные политиками безопасности. Таким образом, речь идет о функционировании СУБД в соответствии с правилами безопасности, применяемыми и ко всем прочим системам предприятия

- Организация физической безопасности файлов данных. Требования к физической безопасности файлов данных СУБД в целом не отличаются от требований, применяемых к любым другим файлам пользователей и приложений

- Организация безопасной и актуальной настройки СУБД. Данное требование включает в себя общие задачи обеспечения безопасности, такие как своевременная установка обновлений, отключение неиспользуемых функций или применение эффективной политики паролей

- Безопасность пользовательского ПО. Сюда можно отнести задачи построения безопасных интерфейсов и механизмов доступа к данным

- Безопасная организация и работа с данными. Вопрос организации данных и управления ими является ключевым в системах хранения информации. В эту область входят задачи организации данных с контролем целостности и другие, специфичные для СУБД проблемы безопасности. Фактически эта задача включает в себя основной объем зависящих от данных уязвимостей и защиты от них

Но настало время отставить логику в сторону и поговорить ~~на эльфийском~~ о юридических требованиях. В целом, в России пытаются регламентировать эту отрасль, но в основном это басни про «кругом враги», «безопасная безопасность» и «импортозамещать до потери пульса». Даже «вертикаль власти» не забыли [(Мысев Алексей Эдуардович 2019)].[6]

Для беглого ознакомления достаточно заглянуть в ФЗ «Об информации, информационных технологиях и о защите информации» и в Указ президента «О мерах по обеспечению информационной безопасности Российской Федерации при использовании информационно-телекоммуникационных сетей международного информационного обмена». Хоть что-то более-менее содержательное начинается с требований ФСТЭК.

[5] За оригиналом сюда

[6] Если очень хочется, то можно преисполниться вот этой монографией

Итак, согласно ФСТЭКу автоматизированная система управления, имеет многоуровневую структуру :

1. уровень операторского (диспетчерского) управления (верхний уровень)
2. уровень автоматического управления (средний уровень)
3. уровень ввода (вывода) данных исполнительных устройств (нижний (полевой) уровень)

В автоматизированной системе управления объектами защиты являются:

1. информация (данные) о параметрах (состоянии) управляемого (контролируемого) объекта или процесса
2. программно-технический комплекс, включающий технические средства, программное обеспечение, а также средства защиты информации

Принимаемые организационные и технические меры защиты информации:

- должны обеспечивать доступность обрабатываемой в автоматизированной системе управления информации (исключение неправомерного блокирования информации), ее целостность (исключение неправомерного уничтожения, модифицирования информации), а также, при необходимости, конфиденциальность (исключение неправомерного доступа, копирования, предоставления или распространения информации)
- должны соотноситься с мерами по промышленной, физической, пожарной, экологической, радиационной безопасности, иными мерами по обеспечению безопасности автоматизированной системы управления и управляемого (контролируемого) объекта и (или) процесса
- не должны оказывать отрицательного влияния на штатный режим функционирования автоматизированной системы управления.

Организационные и технические меры защиты информации, реализуемые в автоматизированной системе управления в рамках ее системы защиты, в зависимости от класса защищенности, угроз безопасности информации, используемых технологий и структурно-функциональных характеристик автоматизированной системы управления и особенностей ее функционирования должны обеспечивать:

- идентификацию и аутентификацию (ИАФ)
- управление доступом (УПД)
- ограничение программной среды (ОПС)
- защиту машинных носителей информации (ЗНИ)
- аудит безопасности (АУД)
- антивирусную защиту (АВЗ)

- предотвращение вторжений (компьютерных атак) (СОВ)
- обеспечение целостности (ОЦЛ)
- обеспечение доступности (ОДТ)
- защиту технических средств и систем (ЗТС)
- защиту информационной (автоматизированной) системы и ее компонентов (ЗИС)
- реагирование на компьютерные инциденты (ИНЦ)
- управление конфигурацией (УКФ)
- управление обновлениями программного обеспечения (ОПО)
- планирование мероприятий по обеспечению безопасности (ПЛН)
- обеспечение действий в нештатных ситуациях (ДНС)
- информирование и обучение персонала (ИПО)

Для разнообразия можно ознакомиться с положением дел в Беларуси. Там есть вот такие прикольные законы[7]:

- Об информатизации
- О научно-технической информации;
- О национальном архивном фонде и архивах в Республике Беларусь
- О печати и других средствах массовой информации
- О правовой охране программ для ЭВМ и баз данных
- О введении в действие Единой системы классификации и кодирования технико- экономической и социальной информации Республики Беларусь
- и др.

1.0.4 Защита от несанкционированного доступа

Когда мы начинаем говорить о безопасности бд, в первую очередь в голову приходит мысль, что не плохо было бы подумать о конфиденциальности (а уже потом про доступность и тд.). С нее и начнем.

К основным средствам защиты информации относят следующие[8]:

- идентификация и аутентификации
- установление прав доступа к объектам БД

[7] А это взято с лекций по администрированию баз данных
[8] За оригиналом сюда

- защита полей и записей таблиц БД
- шифрование данных и программ

Простейший пример аутентификации это парольная защита. Парольная защита представляет простой и эффективный способ защиты БД от несанкционированного доступа. Пароли устанавлюваются конечным пользователями или администраторами БД и хранятся в определенных системных файлах СУБД в зашифрованном виде. Улучшенным вариантом является использование механизма SSL-аутентификации с использованием сертификатов.

В целях контроля использования основных ресурсов СУБД во многих системах имеются средства установления прав доступа к объектам БД. Права доступа определяют возможные действия над объектами. Владелец объекта, а также администратор БД имеют все права. Остальные пользователи к разным объектам могут иметь различные уровни доступа.

К данным, имеющимся в таблице, могут применяться меры защиты по отношению к отдельным полям и отдельным записям. В реляционных СУБД отдельные записи специально не защищаются. Применительно к защите данных в полях таблицы можно выделить такие уровни прав доступа, как полный запрет доступа, только чтение, разрешение всех операций (просмотр, ввод новых значений, удаление, изменение). Более мощным средством защиты данных является их шифрование. Для расшифрования информации пользователи, имеющие санкционированный доступ к зашифрованным данным, имеют ключ и алгоритм расшифрования.

Итак, для минимизации риска несанкционированного доступа необходима реализация комплекса нормативных, организационных и технических защитных мер, в первую очередь: введение ролевого управления доступа, организация доступа пользователей по предъявлению сертификата, выборочное шифрование для сегментов базы данных.

1.0.5 Защита от вывода

Мы уже говорили об этом в угрозах безопасности 1.0.2 на 9 странице. У (Утебов Данияр Рашидович 2008) есть только объяснение логического вывода на основе функциональных зависимостей или ограничений целостности, а за решением нас посылают в (Смирнов 2007). О! Я тут неожиданно понял что все, что я находил в интернете, в статьях и других книгах это копипаста различной степени наглости из (Смирнов 2007). Так что не буду отличаться оригинальностью и я. Помимо очевидного мониторинга с целью свести количество таких связей к минимуму, надо очень внимательно следить за привилегиями (принцип минимальных привилегий).

> Привилегия – это разрешение на выполнение в системе определенного действия. Не имея соответствующей привилегии, пользователь не может получить доступ к данным или выполнить какое-либо действие.

Все привилегии могут быть разделены на два класса: системные привилегии и привилегии доступа к объектам.

> Системная привилегия – это привилегия, которая дает пользователю право на выполнение какой-либо операции в масштабе базы данных. Например, пользователь с системной привилегией ALTER TABLESPACE может изменять любую табличную область (за исключением некоторых ограничений на табличную область

> SYSTEM).
> Привилегия доступа к объекту – это разрешение пользователю на выполнение определенной операции над определенным объектом, например выполнение выборки из некоторой таблицы. При этом пользователь может формировать любые запросы к данной таблице, но не имеет права модифицировать данные этой таблицы или формировать какой-либо запрос к другой таблице.

1.0.6 Целостность БД

Когда мы говорили о защите от НСД, речь шла о конфиденциальности. Теперь настало время целостности.

По (Смирнов 2007):

> Задача обеспечения целостности предусматривает комплекс мер по предотвращению непреднамеренного изменения или уничтожения информации, используемой информационной системой управления или системой поддержки принятия решений. Изменение или уничтожение данных может быть следствием неблагоприятного стечения обстоятельств и состояния внешней среды (стихийные бедствия, пожары и т. п.), неадекватных действий пользователей (ошибки при вводе данных, ошибки операторов и т. п.) и проблем, возникающих при многопользовательской обработке данных.
> Угроза нарушения целостности включает в себя любое умышленное или случайное изменение информации, обрабатываемой в информационной системе или вводимой из первичного источника данных. К нарушению целостности данных может привести как преднамеренное деструктивное действие некоторого лица, изменяющего данные для достижения собственных целей, так и случайная ошибка программного или аппаратного обеспечения, приведшая к безвозвратному разрушению данных.

Мы уже обсуждали основные угрозы целостности в 1.0.2 на 10.

Средства обеспечения целостности баз данных включают автоматическую поддержку некоторой системы правил, описывающих допустимость и достоверность хранимых и вводимых значений. Реляционная модель включает некоторые характерные правила, вытекающие из существа модели: ограничения домена и ограничения таблицы.

- Целостность домена предполагает, что допустимое множество значений каждого атрибута является формально определенным. То есть существуют формальные способы проверки того, что конкретное значение атрибута в базе данных является допустимым. Строка не будет вставлена в таблицу, пока каждое из значений ее столбцов не будет находиться в соответствующем домене (множестве допустимых значений).

- Целостность таблицы означает, что каждая строка в таблице должна быть уникальной. Хотя не все СУБД промышленного уровня требуют выполнения такого ограничения, возможность уникальной идентификации каждой строки представляется необходимой для большинства реальных приложений.

Ограничения целостности позволяют гарантировать, что требования к данным будут соблюдаться независимо от способа их загрузки или изменения. В большинстве СУБД предусмотрена поддержка следующих типов статических ограничений целостности[9]:

1. ограничение на определенность значения атрибута (NOT NULL)
2. ограничение на уникальность значения атрибутов (UNIQUE)
3. ограничение – первичный ключ
4. ограничение – внешний ключ
5. ограничение целостности, задаваемое предикатом

1.0.7 Аудит

Мало того, что все это великолепие надо долго и скрупулезно настраивать, так еще и за всем этим надо следить. О том, как, мы, опять же, находим у (Смирнов 2007).

> Важнейшей составляющей процесса обеспечения безопасности ИС является проведение квалифицированного аудита. Проведение профессионального независимого аудита позволяет своевременно выявить существующие недостатки в системе обеспечения безопасности ИС и объективно оценить соответствие параметров, характеризующих режим обеспечения информационной безопасности, требуемому решаемыми задачами организации уровню.

Полноценная система обеспечения безопаности ИС должна обладать развитыми средствами аудита, то есть, как минимум, СУБД должна обладать средствами автоматического ведения протоколов действий пользователей системы.

В СУБД средство ведения аудита может быть реализовано в виде независимой утилиты (IBM DB2) или возможностей, управляемых языковыми средствами системы (Oracle). Средства аудита обычно ассоциированы с экземпляром, т. е. для каждого экземпляра сервера баз данных может быть запущен собственный файл аудита.

Средства аудита выполняют фиксацию информации об активности пользователей системе в словаре данных или специальном файле – журнале аудита. Информация о настройках системы аудита хранятся в специальном конфигурационном файле. Файл настройки или параметры команды активизации аудита определяют перечень событий, которые фиксируются системой аудита.

Пользователь, обладающий необходимыми полномочиями, может выполнять следующие действия со средствами аудита:

- запускать и останавливать средства аудита;
- просматривать состояние конфигурации средства аудита и настраивать средства аудита на фиксацию определенных событий;
- переписывать данные аудита во внешние файлы операционной системы для проведения независимого анализа.

[9]Немного другая точка зрения: https://studfiles.net/preview/6354061/page:55/ (Роскомнадзор будет ругаться, но вас это пугать не должно)

1.0.8 Многоуровневая защита

Мы уже успели поговорить про уровни в связке интерфейс-СУБД-ОП-среда [1.0.1], сеть-ОС-БД [1.0.2] и даже ФСТЭКовские уровни [1.0.3].

Самое полное описание, пожалуй, сеть-ОС-БД, так что дальше будем отталкиваться от него. Вполне очевидно что защищать все и сразу сложно, каждый уровень имеет свою специфику, но какие то общие принципы выделить можно. Так как все труды на эту тему каскадно ссылаются на (Смирнов 2007), то просто процитирую:

> Анализ наиболее успешных решений в области обеспечения информационной безопасности баз данных позволил сформулировать несколько полезных принципов, которыми можно руководствоваться при проектировании систем защиты:
>
> - экономическая оправданность механизмов защиты
> - открытое проектирование
> - распределение полномочий между различными субъектами в соответствии с правилами организации
> - минимально возможные привилегии для пользователей и администраторов
> - управляемость системы при возникновении отказов и сбоев
> - психологическая приемлемость работы средств защиты данных

1.0.9 Типы контроля безопасности: потоковый, контроль вывода, контроль доступа

Про все три типа мы уже успели поговорить. В частности 1.0.2 и 1.0.3. Единственное, что осталось уточнить – это контроль доступа. Идейно он этот вопрос распадается на два: *кому* и **что** мы будем разрешать?

Кому? Получение доступа к ресурсам информационной системы предусматривает выполнение трех процедур: идентификации, аутентификации и авторизации.

Общепринято, что технологии идентификации и аутентификации являются обязательным элементом защищенных систем, так как обеспечивают аксиоматический принцип персонализации субъектов и, тем самым, реализуют первый (исходный) программно-технический рубеж защиты информации в компьютерных системах.

Под идентификацией понимается различение субъектов, объектов, процессов по их моделям, существующим в форме имен. Под аутентификацией понимается проверка и подтверждение подлинности образа идентифицированного субъекта, объекта, процесса. Как вытекает из самой сути данных понятий, в основе технологий идентификации и аутентификации лежит идеология вероятностного распознавания образов, обуславливая, соответственно, принципиальное наличие ошибок первого и второго рода.

1. Сущность процедуры идентификации состоит в назначении пользователю, т. е. объекту – потребителю ресурсов сервера баз данных – имени. Имя пользователя – это некоторая уникальная метка, соответствующая принятым соглашениям именования и обеспечивающая од-

нозначную идентификацию объекта реального мира в пространстве отображаемых объектов. С позиций ИС источники, предъявившие идентификатор, неразличимы.

2. Сущность процедуры аутентификации состоит в подтверждении подлинности пользователя, представившего идентификатор.

3. Сущность процедуры авторизации состоит в определении перечня конкретных информационных ресурсов, с которыми аутентифицированному пользователю разрешена работа.

Процедуры идентификации, аутентификации и авторизации являются обязательными для любой защищенной ИС.

Что? В целом используют одну из трех моделей безопасности: дискреционная, мандатная и ролевая.

1. Простейшая (одноуровневая) модель безопасности данных строится на основе дискреционного (избирательного) принципа разграничения доступа, при котором доступ к объектам осуществляется на основе множества разрешенных отношений доступа в виде троек – «субъект доступа – тип доступа – объект доступа».

   Наглядным и распространенным способом формализованного представления дискреционного доступа является матрица доступа, устанавливающая перечень пользователей (субъектов) и перечень разрешенных операций (процессов) по отношению к каждому объекту базы данных (таблицы, запросы, формы, отчеты).

2. Мандатная модель доступа характерна для случая, когда возможность конкретных действий с данными или документами определяется внешним, обычно глобальным собственником информации. Исторически в роли такого глобального собственника чаще всего выступало государство. Основная идея мандатной модели доступа состоит в приписывании объектам и субъектам доступа меток. Если метки объекта и субъекта соответствуют (в некотором смысле), то субъект получает право на выполнение определенных действий с объектом. В роли метки, приписываемой субъектам в государственных структурах, выступает, например, форма допуска. В реляционной модели в качестве структуры, обладающей меткой, естественно выбрать кортеж.

3. В основу ролевой модели положена идея принадлежности всех данных системы некоторой организации, а не пользователю, как в случае моделей дискреционного и мандатного доступа. В целом модель ориентирована на упрощение и обеспечение формальной ясности в технологии обеспечения политики безопасности системы. Управление доступом в ролевой модели осуществляется как на основе матрицы прав доступа для ролей, так и с помощью правил, регламентирующих назначение ролей пользователям и их активацию во время сеансов.

   В ролевой модели классическое понятие субъект разделяется на две части: пользователь и роль. Пользователь — это человек, работающий с системой и выполняющий определенные служебные обязанности. Роль – это активно действующая в системе абстрактная сущность, с которой связан ограниченный, логически связанный набор привилегий, необходимых для осуществления определенной деятельности. (Самым распространенным примером роли является присутствующая почти в каждой системе учетная запись администратора (например, root

для UNIX и Administrator для Windows), который обладает специальными полномочиями и может использоваться несколькими пользователями.

Ролевая модель включает три компонента: модель отображения пользователь – роль, модель отображения привилегия – роль и модель отображения роль – роль. Для упрощения логической структуры объектов управления вводится понятие иерархии ролей. Роль, входящая в иерархию, может включать другие роли, наследуя все привилегии включаемых ролей.

2 Теоретические основы безопасности в СУБД

2.1 Критерии защищенности БД

2.1.1 Критерии оценки надежных компьютерных систем (TCSEC)

«Критерии безопасности компьютерных систем» (Trusted Computer System Evaluation Criteria), получившие неформальное название «Оранжевая книга», были разработаны Министерством обороны США в 1983 году с целью определения требований безопасности, предъявляемых к аппаратному, программному и специальному обеспечению компьютерных систем и выработки соответствующей методологии и технологии анализа степени поддержки политики безопасности в компьютерных системах военного назначения. В данном документе были впервые нормативно определены такие понятия, как «политика безопасности», «ядро безопасности» (TCB) и т.д.

Предложенные в этом документе концепции защиты и набор функциональных требований послужили основой для формирования всех появившихся впоследствии стандартов безопасности.

В «Оранжевой книге» предложены три категории требований безопасности – политика безопасности, аудит и корректность, в рамках которых сформулированы шесть базовых требований безопасности. Первые четыре требования направлены непосредственно на обеспечение безопасности информации, а два последних – на качество самих средств защиты.

Рассмотрим эти требования подробнее:

1. Политика безопасности

   - **Политика безопасности** Система должна поддерживать точно определённую политику безопасности. Возможность осуществления субъектами доступа к объектам должна определяться на основе их идентификации и набора правил управления доступом. Там, где необходимо, должна использоваться политика нормативного управления доступом, позволяющая эффективно реализовать разграничение доступа к категорированной информации (информации, отмеченной грифом секретности — типа «секретно», «сов. секретно» и т.д.).
   - **Метки** С объектами должны быть ассоциированы метки безопасности, используемые в качестве атрибутов контроля доступа. Для реализации нормативного управления доступом система должна обеспечивать возможность присваивать каждому объекту метку или набор атрибутов, определяющих степень конфиденциальности (гриф секретности) объекта и/или режимы доступа к этому объекту.

2. Аудит

   - **Идентификация и аутентификация** Все субъекты должен иметь уникальные идентификаторы. Контроль доступа должен осуществляться на основании результатов идентификации субъекта и объекта доступа, подтверждения подлинности их идентификаторов (аутентификации) и правил разграничения доступа. Данные, используемые для идентификации и аутентификации, должны быть защищены от несанкционированного доступа, модификации и уничтожения и должны быть ассоциированы со всеми активными компонентами компьютерной системы, функционирование которых критично с точки зрения безопасности.

- **Регистрация и учет** Для определения степени ответственности пользователей за действия в системе, все происходящие в ней события, имеющие значение с точки зрения безопасности, должны отслеживаться и регистрироваться в защищенном протоколе. Система регистрации должна осуществлять анализ общего потока событий и выделять из него только те события, которые оказывают влияние на безопасность для сокращения объема протокола и повышения эффективность его анализа. Протокол событий должен быть надежно защищен от несанкционированного доступа, модификации и уничтожения.

3. Корректность

- **Контроль корректности** Средства защиты должны содержать независимые аппаратные и / или программные компоненты, обеспечивающие работоспособность функций защиты. Это означает, что все средства защиты, обеспечивающие политику безопасности, управление атрибутами и метками безопасности, идентификацию и аутентификацию, регистрацию и учёт, должны находиться под контролем средств, проверяющих корректность их функционирования. Основной принцип контроля корректности состоит в том, что средства контроля должны быть полностью независимы от средств защиты.
- **Непрерывность защиты** Все средства защиты (в т.ч. и реализующие данное требование) должны быть защищены от несанкционированного вмешательства и/или отключения, причем эта защита должна быть постоянной и непрерывной в любом режиме функционирования системы защиты и компьютерной системы в целом. Данное требование распространяется на весь жизненный цикл компьютерной системы. Кроме того, его выполнение является одним из ключевых аспектов формального доказательства безопасности системы.

2.1.2 Понятие политики безопасности

Согласно МЕТРОЛОГИИ 2008: политика безопасности – совокупность документированных правил, процедур, практических приемов или руководящих принципов в области безопасности информации, которыми руководствуется организация в своей деятельности (то есть как организация обрабатывает, защищает и распространяет информацию). Причём, политика безопасности относится к активным методам защиты, поскольку учитывает анализ возможных угроз и выбор адекватных мер противодействия.

2.1.3 Совместное применение различных политик безопасности в рамках единой модели

Проблема совмещения различных политик безопасности возникает достаточно часто при администрировании компьютерных систем. Стандарты защиты информации в автоматизированных системах подразумевают наличие более одной политики разграничения доступа.

Так, в «Оранжевой книге» использование только дискреционного разделения доступа относит компьютерную систему к одному из классов безопасности группы «С», тогда как добавление мандатного контроля доступа позволяет претендовать на более высокий класс защищенности группы «В». Причем «Оранжевой книгой» подразумевается именно добавление мандатной политики безопасности (МПБ) с сохранением возможностей дискреционной политики безопасности (ДПБ).

В качестве еще одного примера совмещения политик безопасности можно привести системы управления базами данных, функционирующие на базе операционных систем семейства Windows. В системах управления базами данных наиболее распространенной является ролевая политика безопасности, но при этом данные хранятся в файлах, доступ к которым разграничивается операционной системой. В операционных системах базовой является ДПБ, но при этом реализуется на определенном уровне МПБ. Таким образом, требуется сопряжение трех различных политик безопасности.

2.1.4 Интерпретация TCSEC для надежных СУБД (TDI)

В дополнение к «Оранжевой книге» TCSEC, регламентирующей вопросы обеспечения безопасности в ОС, существуют аналогичный документ Национального центра компьютерной безопасности США для СУБД – TDI, («Пурпурная книга»).

2.1.5 Оценка надежности СУБД как компоненты вычислительной системы

2.1.6 Монитор ссылок

Контроль за выполнением субъектами (пользователями) определённых операций над объектами, путем проверки допустимости обращения (данного пользователя) к программам и данным разрешенному набору действий.

Обязательные качества для монитора обращений:

- Изолированность (неотслеживаемость работы)
- Полнота (невозможность обойти)
- Верифицируемость (возможность анализа и тестирования)

2.1.7 Применение TCSEC к СУБД непосредственно

2.1.8 Элементы СУБД, к которым применяются TDI: метки, аудит, архитектура системы, спецификация, верификация, проектная документация

Пурпурная книга

2.1.9 Критерии безопасности ГТК/ФСТЭК

Некоторое время ФСТЭК (в прошлом ГТК, до 2004 г.) использовал критерии безопасности баз данных, заданные документом «Безопасность информационных технологий. Критерии оценки безопасности информационных технологий» от 19.06.02 г. № 187. Этот документ был разработан в ГТК и сейчас выведен из употребления.

Однако, в настоящее время (на 2023 год), ФСТЭК работает по еще более старому документу, которым заменили более новый. Не ищите тут логику. Называется он так: «Средства вычислительной техники. Защита от несанкционированного доступа к информации. Показатели защищенности от несанкционированного доступа к информации», утвержденного решением Государственной технической комиссии при Президенте Российской Федерации от 30 марта 1992 г.

ФСТЭК 187 (устаревший)

ФСТЭК Средства вычислительной техники. Защита от несанкционированного доступа к информации. Показатели защищенности от несанкционированного доступа к информации (актуальный)

Всего существует семь классов СВТ (средств вычислительной техники), в них входит что угодно, СВТ - это совокупность ПО и аппаратной части в любых сочетаниях. Шесть классов «рабочие» и седьмой «мусорный», для тех, которые не дотягивают даже до шестого.

Классы дополнительно делятся на четыре группы, отличающиеся качественным уровнем защиты:

- Первая группа содержит только один седьмой класс;
- Вторая группа характеризуется дискреционной защитой и содержит шестой и пятый классы;
- Третья группа характеризуется мандатной защитой и содержит четвертый, третий и второй классы;
- Четвертая группа характеризуется верифицированной защитой и содержит только первый класс.

Сами требования весьма базовые и к функционированию непосредственно баз данных относятся опосредованно. Но, если в вашей БД будет лежать что-то ценное, вы автоматом попадаете под раздачу, так что эти требования надо выполнять.

Оценка класса защищенности СВТ проводится в соответствии с «Положением о сертификации средств и систем вычислительной техники и связи по требованиям защиты информации» и «Временным положением по организации разработки, изготовления и эксплуатации программных и технических средств защиты информации от несанкционированного доступа в автоматизированных системах и средствах вычислительной техники» и другими документами.

Le meme в том, что первый документ на сайте ФСТЭК отсутствует и видимо был заменен на «Положение о системе сертификации средств защиты информации». Куда делись требования ко всему остальному не ясно, это еще предстоит выяснить. Второй документ (тоже образца 92 года) на сайте есть в чистом виде и все еще актуален.

Положение о системе сертификации средств защиты информации

Временное положение по организации разработки, изготовления и эксплуатации программных и технических средств защиты информации от несанкционированного доступа в автоматизированных системах и средствах вычислительной техники

Не уверены в том, что юзать? Спросите старших товарищей на работе или напишите на горячую линию ФСТЭК-а, они там для этого и сидят.

Контакты ФСТЭК

В соответствии с все тем же документом образца 92 года, выбор класса защищенности СВТ для автоматизированных систем, создаваемых на базе защищенных СВТ, зависит от грифа секретности обрабатываемой в АС информации, условий эксплуатации и расположения объектов системы.

Эти документы регулируют в первую очередь разработку продукта. Защита систем в целом подчиняется приказу 239 (меры ИБ, как защищать) и Методике оценки угроз безопасности информации от 2021 года, еще актуальной на 2023 год (как правильно бояться).

Итак, в документе «Средства вычислительной техники. Защита от несанкционированного доступа к информации. Показатели защищенности от несанкционированного доступа к информации»

вы найдете список вещей которые надо держать в уме и требования по их реализации. Классификации нет, механизмы подлежащие реализации смешаны с требованиями к комплекту документации. Список следующий:

- Дискреционный принцип контроля доступа
- Мандатный принцип контроля доступа
- Очистка памяти
- Изоляция модулей
- Маркировка документов
- Защита ввода и вывода на отчуждаемый физический носитель информации
- Сопоставление пользователя с устройством
- Идентификация и аутентификация
- Гарантии проектирования
- Регистрация
- Взаимодействие пользователя с комплексом средств защиты
- Надежное восстановление
- Целостность комплекса средств защиты
- Контроль модификации
- Контроль дистрибуции
- Гарантии архитектуры
- Тестирование
- Руководство для пользователя
- Руководство по комплексу средств защиты
- Тестовая документация
- Конструкторская (проектная) документация

Сертификация ФСТЭК – процедура получения документа, подтверждающего, что средство защиты информации соответствует требованиям нормативных и методических документов ФСТЭК России.

Сертификация ФСТЭК для средств защиты информации создана для того, чтобы обеспечить:

- Защиту конфиденциальной информации строго определенного уровня.
- Возможность для потребителей выбирать качественные и эффективные средства защиты информации.

- Содействие формированию рынка защищенных информационных технологий и средств их обеспечения.

Сертификация ФСТЭК необходима для строго определенных сфер деятельности (примерно Всех важных и делающих деньги, кроме закладок, вебкама и геймдизайна, и то не факт) и ее необходимость зависит от того, какая именно информация будет обрабатываться в той или иной информационной системе.

Сферы деятельности с обязательной сертификацией средств защиты информации:

- Государственные информационные системы (ГИС).
- Автоматизированные системы управления технологическим процессом (АСУ ТП).
- Персональные данные (ЗПДн).
- Критическая информационная инфраструктура (КИИ). Их тоже целый перечень, см. ниже.
- Государственная тайна (ЗГТ).
- Конфиденциальная информация (служебная информация).

Список сфер КИИ:

- Здравоохранение;
- Наука;
- Транспорт;
- Связь;
- Энергетика;
- Банковская и иные сферы финансового рынка;
- Топливно-энергетический комплекс;
- Атомная энергия;
- Оборонная и ракетно-космическая промышленность;
- Горнодобывающая, металлургическая и химическая промышленность.

Несертифицированный продукт невозможно использовать в тех сферах деятельности, где обязательно использование сертифицированных продуктов.

Использование компанией несертифицированной продукции в сфере деятельности, требующей обязательной сертификации средств защиты информации, может повлечь за собой серьезные последствия: от больших штрафов до уголовной ответственности для руководителей.

В пределах одной компании может быть несколько информационных систем. Некоторые из них нуждаются в сертифицированных средствах защиты, а некоторые – нет. Все зависит от характера информации, которая обрабатывается в этих системах. В случае, если данные касаются вышеперечисленных сфер деятельности, сертификация обязательна.

ФСТЭК контролирует практически все области защитф информации, исключениями являются линии связи (их контролирует Минскомсвязи), и средства криптографической защиты (их контролирует ФСБ).

Ситуация с шифрованием очень зыбкая, какое-то время ФСБ наступала даже на антивирусы, поскольку обновления в них защищаются в том числе криптографическими методами. Но, процесс этот к счастью заглох. Требования ФСБ - тайна покрытая мраком и процедура сертификации у них крайне непрозрачная, осуществляют ее всего несколько особо доверенных фирм. Если криптографические функции вашего продукта не являются его основным назначением, то сертификация от ФСБ необязательна, для выхода на рынок вам хватит сертификата ФСТЭК.

Ваша дорога из желтого кирпича изложена в двух вышеупомянутых документах.

«Полжение о системе сертификации средств защиты информации» - это просто инструкция, описывающая процесс сертификации и ответсвенность сторон.

Положение определяет состав участников системы сертификации средств защиты информации, создаваемой ФСТЭК России, а также организацию и порядок сертификации продукции, используемой в целях защиты сведений, составляющих государственную тайну или относимых к охраняемой в соответствии с законодательством Российской Федерации иной информации ограниченного доступа, являющейся государственным информационным ресурсом и (или) персональными данными, продукции, сведения о которой составляют государственную тайну, подлежащей сертификации в рамках указанной системы.

Список участников такой:

- федеральный орган по сертификации;
- организации, аккредитованные ФСТЭК России в качестве органа по сертификации (органы по сертификации);
- организации, аккредитованные ФСТЭК России в качестве испытательной лаборатории (далее - испытательные лаборатории);
- изготовители средств защиты информации (вы находитесь вот тут)

Окей, что такое «временное положение?». Ну, одно точно, нет ничего постояннее временного.

Положение устанавливает единый на территории Российской Федерации порядок исследований и разработок в области:

- Защиты информации, обрабатываемой автоматизированными системами различного уровня и назначения, от несанкционированного доступа;
- Создания средств вычислительной техники общего и специального назначения, защищенных от утечки, искажения или уничтожения информации за счет НСД, в том числе программных и технических средств защиты информации от НСД;
- Создания программных и технических средств защиты информации от НСД в составе систем защиты секретной информации в создаваемых АС.

Положение определяет следующие основные вопросы:

- Организационную структуру и порядок проведения работ по защите информации от НСД и взаимодействия при этом на государственном уровне;

- Систему государственных нормативных актов, стандартов, руководящих документов и требований по этой проблеме;
- Порядок разработки и приемки защищенных СВТ, в том числе программных и технических (в частности, криптографических) средств и систем защиты информации от НСД;
- Порядок приемки указанных средств и систем перед сдачей в эксплуатацию в составе АС, порядок их эксплуатации и контроля за работоспособностью этих средств и систем в процессе эксплуатации.

Вот так, как-то. Добро пожаловать в увлекательный мир нормативно-правовой документации в области защиты информации.

2.2 Модели безопасности в СУБД

Сегодня существует большое число различных теоретических моделей, которые позволяют описать практически все аспекты безопасности и обеспечивают средства защиты информации формально подтверждённой алгоритмической базой. Однако на практике воспользоваться результатами данных исследований не всегда удается, потому что слишком часто теория защиты информации не согласуется с реальной жизнью. Дело в том, что теоретические исследования в области защиты информационных систем носит разрозненный характер и не составляет комплексной теории. Существующие технические разработки основаны на различных подходах проблеме, поэтому методы её решения существенно различаются. Наибольшее развитие получили два подхода – это формальное моделирования политики безопасности и криптография. Эти различные по происхождению и решаемым задачам подходы взаимно дополняют друг друга. В отличии от криптографии, формальные модели безопасности предоставляют разработчикам защищенных систем принципы, которые лежат в основе архитектуры защищенной системы и определяют концепцию её построения.

Под *политикой безопасности* понимается совокупность норм и правил, определяющих меры по обеспечению безопасности даанных, обрабатываемых в информационной системе. Для реализации политики безопасности, сформулированной в терминах естественного языка, необходимо представить ее формальное выражение. Его называют **модель политики безопасности** (или просто **модель безопасности**).

Далее мы разберем основные модели безопасности. Наиболее важные модели мы разберём подробно, а оставшиеся затронем кратко, отослав заинтересованного читателя к первоисточникам.

Определения Перед дальнейшим рассмотрением моделей безопасности сформулируем основные определения:

- Политика безопасности – совокупность норм и правил, определяющих меры по обеспечению безопасности даанных, обрабатываемых в информационной системе.
- Модель безопасности – формальное выражение политики безопасности.
- Монитор безопасности – механизм реализации политики безопасности в автоматизированной системе, совокупность аппаратных, программных и специальных компонент системы, реализующих функции защиты и обеспечения безопасности (общепринятое сокращение – Trusted Computing Base, TCB).

- Правила разграничения доступа – совокупность правил, регламентирующих права доступа субъектов доступа к объектам доступа.
- Объект доступа – единица информационного ресурса, доступ к которой регламентируется правилами разграничения доступа.
- Субъект доступа – лицо или процесс, действия которого регламентируются правилами разграничения доступа.
- Состояние системы – совокупность множеств субъектов, объектов и отношений между ними. Каждое состояние системы является либо безопасным, либо небезопасным в соответствии с предложенными в модели критериями безопасности.

Основная теорема безопасности (Basic Security Theorem) Во время работы над моделью безопасности Белла-ЛаПадулы авторы пошли несколько дальше, и основываясь на формальном математическом аппарате, попытались строго доказать свойство защищенности своей модели. В результате они сформулировали и доказали "Основную теорему безопасности"(Basic Security Theorem):

> Система безопасна тогда, и только тогда, когда ее начальное состояние безопасно, и любые переходы между ее состояниями также безопасны. Тогда и любое последующее состояние этой системы также будет безопасным, вне зависимости от обрабатываемой информации и конкретных процессов ее обработки.

Теорема предполагает, что имеется возможность проверить безопасность начального состояния и всех возможных переходов состояний, чтобы доказать безопасность всей системы.

За моделью Белла-ЛаПадулы последовали другие модели управления доступом, создатели которых доказывали основную теорему безопасности и для них. Считалось, что доказательство этой теоремы доказывает безопасность и надежность самой модели.

Так продолжалось до середины 1980-х годов, когда Дж. Маклеан (J. McLean) подверг основную теорему безопасности критике. Маклеан предположил и обосновал, что она абсолютно бесполезна в силу своей тривиальности. Теорема никак не адресует собственно понятие безопасности, но демонстрирует общее свойство всех систем, допускающих формальное представление в виде конечного автомата с индуктивными переходами между последовательными состояниями. Безопасность же, скорее неформальное понятие.

После этого, попытки формального математического обоснования надежности и безопасности различных моделей управления доступом прекратились. Вопрос о возможности формализации и строгом математическом доказательстве свойства защищенности при обработке информации в вычислительной системе остается открытым до настоящего времени.

2.2.1 Аспекты исследования моделей безопасности

Основная цель создания и формального описания политики безопасности – это определение условий, которым должно подчиняться поведение системы, выработка критерия безопасности и проведение формального доказательства соответствия системы этому критерию при соблюдении установленных правил и ограничений.

Исследование моделей безопасности выполняется с разных сторон. Можно выделить следующие аспекты:

- Сущности системы – субъекты и объекты.
- Возможные связи между сущностями системы.
- Разрешенные функции перехода между состояниями системы.
- Выполнение основной теоремы безопасности.
- Реализация монитора безопасности.

2.2.2 Классификация моделей безопасности

Обычно говоря о моделях безопасности имеют ввиду модели разграничения доступа. Различают три основные модели контроля доступа:

1. **Дискреционная** (англ. discretionary access control, DAC);
2. **Мандатная** (англ. mandatory access control, MAC);
3. **Ролевая** (англ. role-based access control, RBAC).

2.2.3 Дискреционная модель

Политика дискреционного управления достуом реализована в большинстве защищенных информационных систем и исторически является первой проработанной в теоретическом и практическом плане. Первые описания моделей дискреционного доступа появились еще в 60-х годах и подробно представлены в литературе.

На практике дискреционная модель доступа предполагает, что для каждого объекта в системе определен субъект-владелец. Этот субъект может самостоятельно устанавливать необходимые, по его мнению, права доступа к любому из своих объектов для остальных субъектов, в том числе и для себя самого. Логически владелец объекта является владельцем информации, находящейся в этом объекте. При доступе некоторого субъекта к какому-либо объекту система контроля доступа лишь считывает установленные для объекта права доступа и сравнивает их с правами доступа субъекта. Кроме того, предполагается наличие в ОС некоторого выделенного субъекта – администратора дискреционного управления доступом, который имеет привилегию устанавливать дискреционные права доступа для любых, даже ему не принадлежащих объектов в системе.

Кратко рассмотрим основные модели безопасности.

HRU Модель безопасности Харрисона-Руззо-Ульмана (HRU-модель) (середина 1970-х гг.) реализует произвольное управление доступом субъектов к объектам и контроль за распространением прав доступа (Зегжда Д.П. 2000). Названа в честь трёх его авторов: Майкла Харрисона, Уолтера Руззо и Джеффри Ульмана.

Главной особенностью является матрица доступа с полным описанием пользовательских прав к файлам. Изменения в эту матрицу вводятся с помощью специальных команд.

Представления модели. Введем некоторые обозначения:

- S —- множество субъектов;
- O —- множество объектов;
- R = (r_1, r_2, ... , r_n) -— множество прав доступа.

Для реализации этих прав в данной модели используется матрица доступов M, строки которой соответствуют субъектам, а столбцы — объектам. На пересечении строчек и столбцов указаны права доступа R, которыми обладает данный субъект по отношению к данному объекту. Тогда текущее состояние системы Q можно однозначно записать в таком виде: Q = (S, O, M). Также вводят множество возможных операций A = (a_1, a_2, ... , a_k).

Критерий безопасности системы. Для заданной системы исходное состояние Q_0 = (S_0, O_0, M_0) называется безопасным относительно права r, если не существует такой последовательности команд, которая изменила бы заданное начальное состояние системы так, что право r записалось бы в ячейку M[s;o], в которой оно отсутствовало в начальном состоянии Q_0 (Зегжда Д.П. 2000). Если это условие не выполнено, то произошла утечка информации.

Определение. Монооперационная команда – команда, состоящая из не более чем одной элементарной операции (Зегжда Д.П. 2000).

Определение. Монооперационная система – система, все команды которой являются монооперационными.

Теорема. Существует алгоритм, проверяющий на безопасность исходное состояние Q_0 монооперационной системы на безопасность относительно права r (*Модель Харрисона-Руззо-Ульмана — Википедия* 2021).

Теорема. Задача определения безопасности исходного состояния Q_0 системы общего вида для данного права r является неразрешимой (*Модель Харрисона-Руззо-Ульмана — Википедия* 2021).

Основным преимуществом модели над другими дискреционными является строгость критерия безопасности (*Модель Харрисона-Руззо-Ульмана — Википедия* 2021).

Take-Grant Несмотря на недостатки дискреционных моделей безопасности, относительно простая реализация их на практике побудила исследователей к усовершенствованию моделей типа HRU, в которых проблема контроля распространения прав доступа алгоритмически не разрешима (Уральский федеральный университет 2021).

Например, *система передачи прав доступа встраивается в систему разграничения в виде дополнительных прав*, образуя модель передачи прав доступа Take-Grant (1976). В этой модели доказывается безопасное состояние системы после изменения прав доступа к объекту или субъекту путем преобразования графов доступа (Уральский федеральный университет 2021).

Модель представляет всю систему как направленный граф, где узлы — либо объекты, либо субъекты. Дуги между ними маркированы, и их значения указывают права, которые имеет объект или субъект (узел). В модели доминируют два правила: «брать» и «давать». Они играют в ней особую роль, переписывая правила, описывающие допустимые пути изменения графа (*Модель Take-Grant — Википедия* 2021). В общей сложности существует 4 правила преобразования:

- правило «брать»;
- правило «давать»;

- правило «создать»;
- правило «удалить».

Используя эти правила, можно воспроизвести состояния, в которых будет находиться система в зависимости от распределения и изменения прав доступа. Следовательно, можно проанализировать возможные угрозы для данной системы.

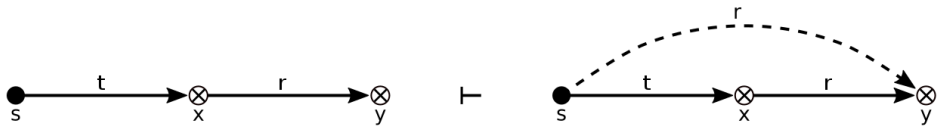
Представления модели. Введем некоторые обозначения:

- S —- множество субъектов;
- O —- множество объектов;
- R = (r_1, r_2, ... , r_n) -— множество прав доступа.

Правило «брать».

Брать = take(r, x, y, s), r ∈ R. Пусть s ∈ S, x,y ∈ O — вершины графа G.

Тогда граф G:

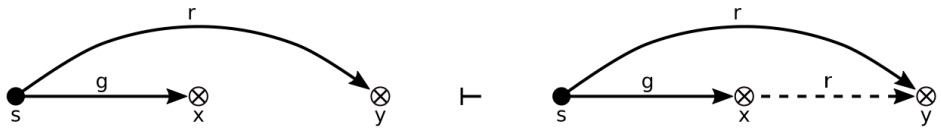


То есть субъект *S* берет у объекта *X* права r на объект Y.

Правило «давать».

Давать = grant(r, x, y, s), r ∈ R. Пусть s ∈ S, x,y ∈ O — вершины графа G.

Тогда граф G:



То есть субъект *S* дает объекту *X* права r на объект Y.

Правило «создать».

Создать = create(p, x, s), r ∈ R. Пусть s ∈ S, x,y ∈ O — вершины графа G.

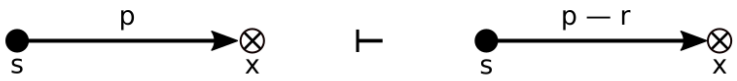
Тогда граф G:

То есть субъект *S* берет r-доступный объект X.

Правило «удалить».

Удалить = remove(r, x, s), r ∈ R. Пусть s ∈ S, x,y ∈ O — вершины графа G.

Тогда граф G:



По-сути критерием безопасности системы при использовании модели Take-Grant является невозможность нарушения прав доступа при выполнении установленных правил.

Результатом модели становится представление системы как направленного графа с дугами - правами между субъектами и объектами.

Следующие две модели намного менее популярны, поэтому при их рассмотрении ссылаемся только на англоязычные источники.

Action-Entity Модель Acten (Action Entity) была предложена Буссолати и др. (Jalili 2021) в 1983 г. Является расширением модели Take-Grant (*Security models* 2021):

- Дополнительные административные привилегии (использование, создание, удаление, ...)
- Предикаты авторизации – условия, которые должны быть выполнены для предоставления доступа
- Представление с использованием двух графиков (один для авторизации для статических действий и один для авторизации для динамических действий)

Цели расширения (Jalili 2021):

- Исправление неизбирательности админских привилегий
- Увеличение контроля над транспортным потоком

Каждый элемент системы, имеющий отношение к безопасности, рассматривается как ***сущность (Entity)*** *(Jalili 2021)*:

- Сущность касается как субъектов, так и объектов
- Сущности – это типы ресурсов

Выделяют два режима доступа – ***действия (Action)*** *(Jalili 2021)*:

- статический;
- динамический.

Статические действия (режим доступа). Их выполнение не изменяет аутентификацию и состояние системы. Состоят из (*Security models* 2021):

- «Использование»: между пользователями, операторами и ресурсами ввода-вывода, приложениями, ...
- «Чтение»: для доступа к содержимому объекта

- «Написание» («Обновление»)
- «Создание»: сущностей
- «Удаление»: объекта

Динамические действия (режим доступа). Их выполнение изменяет аутентификацию и/или состояние системы. Состоят из (*Security models* 2021):

- «Грант» (предоставить, Grant): удержание субъектом Ei статического режима m по отношению к Ek позволяет Ei предоставить субъекту Ej режим m на Ek
- «Отозвать»
- «Делегировать»: позволяет Ei предоставлять Ej динамическое разрешение на «грант», относящееся к режиму m на Ek. Авторизация для делегирования/отмены ограничена несколькими объектами
- «Отменить»

Вводится иерархия отношений между статическими режимами (Jalili 2021):

- «Создание»/«Удаление» – 4
- «Обновление» – 3
- «Чтение» – 2
- «Использование» – 1

И между динамическими режимами (административными правами) (Jalili 2021):

- «Делегировать»/«Отменить» – 2
- «Грант»/«Отозвать» – 1

В соответствии с этими иерархиями в модели Action Entity строят два графа:

- статических режимов;
- динамических режимов.

Вводят определённые правила для двух графов, которые являются критерием безопасности системы (*Security models* 2021). Подробное изложение можно найти в (*Security models* 2021) и в (Jalili 2021).

Wood Единственным источником, где удалось найти хоть какую-то информацию о модели Woods является (Jalili 2021). Согласно нему модель была разработана в 1979 г. Вудсом. Ориентация модели на управлении аутентификацией и контролем доступа в БД многоуровневой схемы. Модель рассматривает трехуровневую модель ANSI / SPARC, включая:

- Внешний уровень: ближайший к пользователю

- Концептуальный уровень: представление данных
- Внутренний уровень: ближайший к физическому хранению, не привязанный к аппаратным элементам.

Субъекты – пользователи системы:

- Авторизатор, который управляет авторизацией
- Пользователи, имеющие доступ к данным

Объекты – объекты концептуального уровня: каждый O принадлежит к категории типа, заданной функцией n(O), указывает свою категорию типа (n (O)):

- Категории типов на концептуальном уровне: набор сущностей, тип отношения и атрибут
- Атрибут может быть связан с одним набором объектов и сопоставляет набор объектов с набором значений
- Тип отношения – это двунаправленная ассоциация между двумя наборами сущностей и может иметь два имени

Состояние авторизации указывается в правилах доступа. Правила доступа имеют вид <s, o, t, p>, где субъект s реализует режим доступа t к объекту o при условии p. Правила доступа определяются авторизатором. Основные правила указаны в концептуальном уровне, который представляет собой глобальное представление данных организации. Определение таблицы (на внешнем уровне) включает ограничения доступа, определяющие, какие типы доступа допустимы для каждый внешний объект e (каждая таблица и ее поля). Эти ограничения хранятся в ограничении доступа (Access Constraint, AC).

2.2.4 Мандатная модель

Приведём классическое определение мандатной модели из *Оранжевой книги*.

> Мандатная модель контроля доступа – это модель, в которой используются средства ограничения доступа к объектам, основанные на секретности информации, содержащейся в объектах, и обязательная авторизация действий субъектов для доступа к информации с присвоенным уровнем важности.

Важность информации определяется уровнем доступа, который приписываетя всем объектам и субъектам. В сравнении с дискреционным мандатное управление доступом устанавливает гораздо более строгие правила определения прав доступа субъектов к объектам. Предполагается, что некоторый выделенный привилегированный субъект – администратор мандатного управления доступом – определяет необходимые, и согласованные с установленной в системе политикой безопасности, права доступа каждого из субъектов к каждому из объектов. Никакой другой субъект системы не может устанавливать или изменять свои, либо чьи-то еще права доступа к объектам. В такой системе права доступа пользователей определяются, прежде всего, интересами безопасности владельца всей системы.

Далее рассмотрим наиболее популярные мандатные модели безопасности.

Модель Белла-ЛаПадулы В середине 1970-х годов, два математика из MITRE Corporation – Д. Белл (D. Bell) и Л. ЛаПадула (L. LaPadula) – работая по проекту Министерства Обороны США, разработали мандатную модель, названную затем по их именам.

Основной задачей проекта была разработка и формальное обоснование модели управления доступом, функционально согласованную с методиками работы, принятым при обработке бумажной документации конфиденциального характера, т.е. имеющей грифы "секретно "совершенно секретно"и т.п. Сложившаяся на тот момент практика обработки бумажной документации позволяла избежать несанкционированного доступа к информации, а также обладала приемлемой степенью удобства, поэтому было решено применить подобную схему и в вычислительной системе. Суть проекта состояла именно в этом. Интересно, что данный факт порой ускользает от внимания специалистов в области защиты информации.

Ниже дается упрощенное математическое описание модели Белла-ЛаПадулы.

Вычислительная система рассматривается в виде набора субъектов S_i, набора объектов O_j и матрицы управления доступом $M[i * j]$.

Имеется набор упорядоченных уровней секретности – иерархия уровней. Каждый субъект и каждый объект имеет назначенный ему *индивидуальный уровень секретности*. Кроме того, каждый субъект имеет еще *текущий уровень секретности*, который не может превосходить его индивидуальный уровень секретности. Таким образом, субъект может произвольно изменять свой текущий уровень секретности в диапазоне, не выше своего индивидуального уровня. Обычно используют четыре уровня секретности, каждый из которых доминирует над уровнем, находящиимся ниже в ииерархии: совершенно секретно ($top - secret$), секретно ($secret$), конфиденциально ($confidential$) и несекретно ($unclassified$). При этом верно следующее: $top - secret \supset secret \supset confidential \supset unclassified$.

Предполагается, что уровень секретности S_1 доминирует над уровнем S_2, если он больше или равен S_2, т.е. $S_1 \geq S_2$.

Функция всех возможных в системе взаимных уровней секретностей субъектов и объектов F задается полным набором троек ($f_S(S_i)$, S_i, $f_O(O_j)$). Здесь $f_S(S_i)$ – индивидуальный уровень секретности субъекта S_i, $f_O(O_j)$ – индивидуальный уровень секретности объекта O_j, а $f_C(S_i)$ – текущий уровень секретности субъекта Si. Очевидно, что $f_S(S_i)$ должен доминировать над $f_C(S_i)$ для всех S_i. Рассматриваются четыре типа прав доступа субъекта к объекту:

1. **только чтение** (read-only, ro) произвольных участков данных;
2. **дополнение** (append, ad) - субъект может только записывать данные, размещая их в конце объекта;
3. **исполнение** (execute, ex) - субъект не может ни читать, ни записывать данные объекта, но может исполнять этот объект, как прикладную программу;
4. **чтение и запись** (read-write, rw) - субъект может и читать, и записывать данные объекта, по принципу произвольного доступа.

Набор всех типов прав доступа обозначают A, а любой его элемент – x.

Матрица управления доступом $M[i; j]$ содержит в каждой из своих ячеек M_{ij} текущие дискреционные права доступа x (поднабор из A) субъекта S_i к объекту O_j. Каждый столбец матрицы

составляет атрибут управления объекта O_j, и является совокупным набором текущих прав доступа к нему всех субъектов. Атрибут управления для объекта O_j выдается субъекту S_k во время создания им этого объекта. Таким образом, субъект-создатель может устанавливать и отменять текущие дискреционные права доступа для любых субъектов, в том числе и для себя самого. Для этого он модифицирует соответствующие ячейки столбца M_j матрицы доступа $M[i; j]$, изменяя этим атрибут управления объекта O_j. Однако он не может передать право изменения атрибута управления какому-либо другому субъекту.

Всякое обращение субъекта S_i к объекту O_j с типом доступа x описывается тройкой (S_i , O_j, x), а совокупность всех текущих обращений в системе – набором доступов b.

Теперь всю систему защиты можно описать в виде автомата конечных состояний с набором функций перехода между состояниями. Состояние модели z задается набором из четырех элементов (b, $M[i; j]$, F, O_j). С течением времени система принимает некоторую последовательность состояний z, с начальным состоянием z_0.

Разрешены следующие функции перехода между состояниями:

1. изменение набора текущих доступов b;
2. изменение матрицы управления доступом $M[i; j]$;
3. изменение набора объектов set O_j;
4. субъект S_i может изменять свой текущий уровень секретности, но не выше его индивидуального уровня секретности.

Чтобы модель обеспечивала защиту, нужно чтобы всякий раз для функции перехода между состояниями удовлетворялся ряд дополнительных условий. Эти условия называется свойствами защиты.В модели Белла – ЛаПадулы устанавливаются и поддерживаются два мандатных и одно дискреционное свойства защиты[10]:

1. Субъект с определённым уровнем секретности не может иметь доступ на чтение объектов с более высоким уровнем секретности (англ. no read-up).
2. Субъект с определённым уровнем секретности не может иметь доступ на запись объектов с более низким уровнем секретности (англ. no write-down).
3. Использование матрицы доступа субъектов к объектам для описания дискреционного доступа.

Описанные три свойства защиты в обязательном порядке реализуются в системе защиты. Также требуется, чтобы каждое из свойств обязательно удовлетворялось для любого состояния системы защиты. Такие состояния называются безопасными состояниями.

Свойства защиты служат лишь общей директивой для действующих правил управления доступом. Текущий доступ (S_i, O_j, x) может быть предоставлен только в том случае, если выполнены следующие условия:

1. $ss - property$: $fC(Si)$ доминирует над $f_O(O_j)$, если $x = ro$ или $x = rw$.
2. $* - property$:

[10]В англоязычной литературе первое свойство называют simple security property, а второе – *-property.

2.1 $f_O(O_j)$ доминирует над $f_C(S_i)$, если $x = ad$.

2.2 $f_O(O_j)$ равен $f_C(S_i)$, если $x = rw$.

2.3 $f_C(S_i)$ доминирует над $f_O(O_j)$, если $x = ro$.

3. $ds - property$: x находится в ячейке M_{ij} текущих прав доступа матрицы $M[i * j]$.

Недостатки модели Белла-ЛаПадулы Легко увидеть, что из описания модели Белла-ЛаПадулы нельзя определить формальных принципов назначения атрибутов безопасности субъектам и объектам. Действительно, в ней определены и обоснованы все правила управления доступом на основе атрибутов безопасности, но не способы их назначения при создании субъектов и объектов, импорте объектов в систему и т.д. Данный факт вызывает некоторые проблемы при практическом использовании модели.

Другим недостатком модели Белла-ЛаПадулы принято считать ее излишнюю строгость – возможность передачи информации только в сторону увеличения ее уровня секретности: "write up - read down". Это не всегда совместимо с нормальным функционированием реальной вычислительной системы. В силу тех же причин, несколько субъектов с разным уровнем секретности не могут равноправно получать доступ к одному объекту, например, совместно использовать один принтер для вывода информации на печать. Для устранения описанных ограничений в реальных системных обычно реализуют доверенных субъектов или доверенные программы, которые являются посредниками, гарантирующими безопасность несогласованных с моделью видов доступа.

Еще одним основанием для критики модели является проблема скрытых каналов (covert channels). Суть ее в том, что большинство ресурсов вычислительных систем, в том числе операционных систем UNIX - машинное время, устройства хранения информации - совместно и равноправно используются субъектами. Ресурсы выделяются субъектам по мере необходимости. Это рационально, однако также позволяет двум различным субъектам по предварительной договоренности произвольно обмениваться информацией без учета правил управления доступом, заложенных в модели защиты. Например, один из субъектов модулирует ресурсы системы, усиливая и уменьшая вычислительную нагрузку на компьютер, или записывая и удаляя данные во внешней памяти. Это можно рассматривать как закодированную информацию, которую другой субъект считывает, измеряя периодически скорость выполнения своей задачи, или пытаясь также записать данные во внешнюю память.

Борьба со скрытыми каналами чрезвычайно трудна, так сами каналы являются следствием стремления к оптимальному использованию ресурсов системы. Обычно считается, что пропускная способность скрытых каналов слишком мала, и они не могут нанести серьезный ущерб безопасности. К тому же, они предполагают предварительный сговор двух субъектов, что маловероятно. Наличие скрытых каналов просто игнорируется, что можно в некоторой степени считать решением проблемы. В то же время, системы обработки особо конфиденциальной информации должны проектироваться с учетом проблемы скрытых каналов, возможно, в ущерб их производительности.

К числу недостатков модели Белла-ЛаПадулы также относят проблему целостности обрабатываемой информации, определяемую как невозможность искажения и/или уничтожения информации несанкционированными субъектами. Действительно, в рамках правил модели происходит передача информации в сторону повышения ее секретности. Однако разумно предположить, что менее секретная информация заслуживает меньшего доверия, равно как и менее секретный субъект, который может передать ее выше, исказив или уничтожив при этом конфиденциальные данные. Одновре-

менно, с точки зрения системы, повышается (необоснованно) уровень доверия к передаваемой информации. То есть задача защиты информации в модели Белла-ЛаПадулы решается не полностью - конфиденциальная информация может быть искажена и даже уничтожена субъектом с низким уровнем секретности; хотя модель одновременно эффективно препятствует несанкционированному доступу к информации, ее утечке.

Целостность информации принято гарантировать реализацией в вычислительных системах специальных дополнительных моделей защиты. Дискреционное управление доступом также может обеспечить целостность информации в объекте, если его субъект-владелец позаботится об этом, установив соответствующие права доступа к нему.

Важно понимать, что сама по себе модель Белла-ЛаПадулы обладает рядом существенных недостатков, которые проявляются на практике. Тем не менее, она стала отправной точкой в области разработки мандатных моделей безопасности.

Модель Биба Модель Биба (англ., Biba model) оригинальным образом поддерживает целостность информации в системе, представимой в виде набора взаимодействующих субъектов и объектов. Она строится на основе иерархии уровней целостности (доверенности, достоверности) информации и двух основных правил ее обработки – аксиом целостности:

- аксиома простой целостности запрещает чтение информации более доверенным субъектом из менее достоверного объекта;
- аксиома целостности запрещает запись информации менее доверенным субъектом в более достоверный объект.

Таким образом, общие правила данной модели обозначить "no write up - no read down что показывает ее логическую инверсию к модели Белла-ЛаПадулы.

Легко убедиться, что модель Биба позволяет решить проблемы поддержания целостности обрабатываемой информации и сохранения доверия к ней. Действительно, система запрещает произвольное повышение достоверности информации, более доверенные субъекты не могут пользоваться менее достоверной информацией, а общий уровень достоверности комплексной информации, находящейся в объекте, равен минимальному уровню достоверности ее составляющих.

Как модель Белла-ЛаПадулы, так и модель Биба, часто называют моделями информационных потоков (information-flow model), так как информация в вычислительной системе может перемещаться при обработке только в виде потоков, разрешенных в рамках модели: "write up - read down"; "no write up - no read down".

Таким образом, единственной проблемой модели Белла-ЛаПадулы, к решению которой не существует общего подхода, является определение принципов назначения атрибутов безопасности субъектам и объектам.

Модель Дион Предлагается в качестве обязательной политики, которая одновременно обеспечивает секретность и целостность. Сочетает в себе принципы модели Биба (политика строгой последовательности). В модели нет дискреционной политики. Не позволяет передавать информацию от объектов к субъектам (через операции записи). Поток данных разрешен только между объектами, но никогда между объектами и субъектами. Содержит введение концепции связи между объектами для обмена информацией между ними.

Субъекты – это сущности, выполняемые в системе от имени пользователей. Каждому системному объекту назначается 3 уровня безопасности и 3 уровня целостности:

- ASL (Absolute Security Level) указывается при создании и фиксируется на всю жизнь предмета. Это второй уровень пользователя, соответствующий теме
- RSL (Read Security Level) – это самый высокий уровень безопасности, с которого субъекту разрешено читать
- WSL – это самый низкий уровень безопасности, на который субъекту разрешено писать
- AIL (Absolute Integrity Level) присваивается при создании и фиксируется на всю жизнь субъекта. Это уровень целостности пользователя, соответствующий предмету
- RIL: самый низкий уровень целостности, с которого субъекту разрешено читать
- WIL: наивысший уровень целостности, на который субъекту разрешено писать.

Устанавливаются ограничения – для каждого субъекта s:

$$WSL(s) \leqslant ASL(s) \leq RSL(s)$$
$$RIL(s) \leqslant AIL(s) \leq WIL(s)$$

Субъект для которого строго соблюдается хотя бы одно неравенство удовлетворенный считается доверенным. Субъект может быть доверенным с точки зрения секретности, целостности или того и другого. Субъекты, для которых 4 отношения удовлетворяются знаком равенства, считаются недоверенными.

Объекты – любая сущность хранения данных в системе. Каждому объекту назначается 3 уровня безопасности и 3 уровня целостности:

- ASL (Absolute sec level) – уровень безопасности данных, содержащихся в объекте; фиксируется на весь срок службы объекта
- MSL (Migration sec level) – это самый высокий уровень безопасности, на который могут передаваться данные в объекте
- CSL (Corruption sec level) - это самый низкий уровень безопасности, с которого данные могут поступать в объект
- AIL (Absolute int level) - это уровень целостности данных в объекте
- MIL (Migration int level) - это самый низкий уровень целостности, на который могут передаваться данные в объекте
- CIL (Corruption int level) - это самый высокий уровень целостности, с которого данные могут поступать в объект

Аналогично:

$$CSL(o) \leqslant ASL(o) \leq MSL(o)$$
$$MIL(o) \leqslant AIL(o) \leq CIL(o)$$

В модели вводится ряд дополнительных аксиом и правил на отношения между субъектами и объектами.

Многозначность в БД с многоуровневой секретностью Как уже было показано, принципы модели безопасности Белла-ЛаПадулы сами по себе вполне здравые, но на практике поддержание нескольких уровней защиты в пределах одной таблицы (или в одной базе данных) сопряжено с рядом серьезных проблем. Рассмотрим, например, следующую ситуацию.

Сотрудник военного ведомства, чьи персональные данные хранятся в базе данных с многоуровневой защитой, может иметь "настоящую"миссию, например быть сотрудником отдела по борьбе с терроризмом, и в то же время некоторое "прикрытие согласно которому он является, допустим, членом отряда десантников. При этом в базе данных должна быть представлена и реальная информация, и прикрытие.

Мы могли бы расширить модель безопасности, добавив свойство секретности не только для каждой строки, но и на уровне элементов. Но это расширение приведет к новым серьезным проблемам, поскольку каждый элемент в строке может принимать одно из нескольких возможных значений, в зависимости от уровня доступа субъекта. Такое положение дел на фундаментальном уровне нарушает один из важнейших принципов реляционных баз данных – требование первой нормальной формы (1NF)[11].

Решение проблемы поэлементной классификации основано на идее многозначности – когда в рамках одной таблицы может существовать множество записей с одним и тем же значением первичного ключа, но разными полями. Многозначность впервые была применена в 1988 г. в модели безопасной базы данных SeaView (Secure Data Views).

SeaView Модель SeaView (Secure Data View) (*SeaView* б.г.) была предложена Дороти Деннинг в 1987 г. Она состоит из мандатной, дискреционной моделей, а также вспомогательной для добавления и классификации новой информации поступающей в базу данных. SeaView состоит из двух слоев:

- **MAC** обязательный контроль доступа. Этот уровень соответствует обязательной политике безопасности моделей Bell-LaPadula и Biba.
- **Trusted Computing Base (TCB)** определяет концепцию многоуровневых отношений, поддерживает дискреционные средства управления для многоуровневых отношений и представлений, а также формализует поддерживающие политики.

Класс доступа в модели SeaView состоит из:

- **Уровень секретности** – иерархический уровень секретности.
- **Классы целостности** – набор неирархических классов,между которыми определены отношения вложенности.

В рамках модели MAC каждому субъекту и объекту назначаются классы:

- Класс для записи

[11] Таблица считается приведенной к первой нормальной форме, если выполняются следующие условия:
- В таблице не должно быть дублирующихся строк;
- В каждой ячейке таблицы должно храниться атомарное значение (одно не составное значение);
- В столбце должны храниться данные одного типа;
- Не должно быть массивов и списков.

- Класс для чтения

Причем для каждого объекта класс чтения должен быть выше класса записи. Субъект считается доверенным, если его класс чтения строго больше класса записи. Для недоверенных субъектов классы чтения и записи равны.

В рамках модели TCB реализуется многоуровневость. Модель TCB определяет политику дискреционного управления доступом и поддерживающие политики. Она определяет компоненты многоуровневой защищенной системы реляционной базы данных, включая многоуровневые отношения, представления, ограничения целостности и дискреционные полномочия. Информация в модели TCB секретна и должна храниться в объектах, доступ к которым ограничен.

SeaView представляет собой набор одноуровневых БД. Одноуровневые БД связанны с конкретным классом доступа. Таким образом, для реализации модели SeaView могут использоваться обыкновенные реляционные базы данных. А доступ к ним осуществляет отдельное ПО — монитор безопасности. Он при попытке доступа субъекта к объекту сранивает классы доступа и принимает решение — разрешить действие или нет. Оригинальная статья формулирует следующее требование к монитору безопасности — он должен быть легковесным и легкопроверяемым на уязвимости. Требование подкрепляется ссылкой на Оранжевую книгу и объясняется тем, что монитор безопасности ключевой элемент, отвечающий за безопасность данных в модели SeaView.

Jajodia&Sandhu Модель Джаджодиа–Сандху является производной от модели SeaView. Она модифицирует алгоритм, который разлагает многоуровневое отношение на одноуровневые фрагменты, а также модифицирует алгоритм восстановления, который восстанавливает исходное многоуровневое отношение (Faragallah 2015).

В модели Джаджодиа–Сандху алгоритм декомпозиции использует только горизонтальную фрагментацию, поскольку вертикальной фрагментации не требуется. Это приводит к улучшению алгоритма восстановления в модели Джаджодиа–Сандху по сравнению с алгоритмом восстановления в модели SeaView, поскольку можно восстановить многоуровневое отношение без необходимости выполнять операции соединения; для восстановления многоуровневого отношения требуются только операции объединения (Faragallah 2015).

В модели Джаджодиа–Сандху есть две основные проблемы (Faragallah 2015):

- Семантическая неоднозначность: предположим, что есть два кортежа в отношениях с уровнями безопасности U и S, и нет кортежа с уровнем безопасности TS. Если пользователю с уровнем безопасности TS необходимо получить информацию из отношения, он не может решить, какая информация является правильной, потому что значения из кортежа U и кортежа S в отношении будут извлечены в результате запроса

- Операционная неполнота: предположим, что существует два несравнимых уровня безопасности, M1 и M2, наименьшая верхняя граница которых соответствует уровню безопасности S, а наибольшая нижняя граница – уровню безопасности U. Пользователь с уровнем безопасности S не может вставлять кортежи, которые содержат атрибуты с уровнями безопасности U, M1 и M2

Smith&Winslett В модели Смита–Уинслетта многоуровневая реляционная база данных рассматривается как набор обычных реляционных баз данных, в которых все базы данных используют одну

и ту же схему. Эта модель не поддерживает безопасность на уровне каждого отдельного атрибута. Уровень безопасности может быть назначен только атрибутам первичного ключа и кортежам в целом (Faragallah 2015).

Многоуровневая реляционная схема задается как R(A_{pk}, C_{pk}, A_1, ..., A_n, T_C), где A_{pk} обозначается как атрибут данных первичного ключа, C_{pk} - атрибут классификации первичного ключа, который содержит уровень безопасности первичного ключа. Атрибут данных $A_1...A_n$ обозначается как атрибуты данных, а T_C обозначается как атрибут классификации кортежа, который содержит уровень безопасности кортежа (Faragallah 2015).

Согласно этим правилам определяется доступ для обновления и чтения. Модификация базы данных (вставка, удаление и обновление) пользователем может изменить данные только на уровне безопасности пользователя. Запрос от пользователя с уровнем безопасности L может получить доступ к данным именно из тех баз данных, уровень которых не выше уровня L (Faragallah 2015).

В этой модели была добавлена семантика, основанная на концепции веры. Модель Смита–Уинслетта также известна как модель семантики, основанная на убеждениях, а также представила концепцию базового кортежа. Базовый кортеж – это самый низкий уровень безопасности кортежа базы данных, в котором утверждается существование объекта. Таким образом, процедура обновления устраняет проблемы, присутствующие в модели Джаджодиа–Сандху, но ограничивает объем обновления одним объектом (Faragallah 2015).

Решеточная Управление доступом на основе решеток (Lattice-Based Access Control, LBAC) - это сложная модель управления доступом, основанная на взаимодействии между любой комбинацией объектов (например, ресурсов, компьютеров и приложений) и субъектов (таких как отдельные лица, группы или организации) (*Lattice-based access control - Wikipedia* 2021).

В этом типе модели управления принудительным доступом на основе меток решетка используется для определения уровней безопасности, которые может иметь объект и к которым субъект может иметь доступ. Субъекту разрешен доступ к объекту только в том случае, если уровень безопасности субъекта больше или равен уровню безопасности объекта (*Lattice-based access control - Wikipedia* 2021).

Математически уровень безопасности доступа также может быть выражен в терминах решетки (набор частичного порядка), где каждый объект и субъект имеют наибольшую нижнюю границу (совпадение) и наименьшую верхнюю границу (соединение) прав доступа. Например, если двум субъектам A и B требуется доступ к объекту, уровень безопасности определяется как соответствие уровней A и B. В другом примере, если два объекта X и Y объединены, они образуют другой объект Z, которому присваивается уровень безопасности, образованный объединением уровней X и Y (*Lattice-based access control - Wikipedia* 2021).

LBAC также известен как ограничение управления доступом на основе меток (или управление доступом на основе правил) в отличие от управления доступом на основе ролей (RBAC) (*Lattice-based access control - Wikipedia* 2021).

Модели управления доступом на основе решетки были впервые формально определены Деннингом (1976).

2.2.5 Ролевая модель

Ролевые модели безопасности нельзя отнести ни к дискреционным, ни к мандатным моделям, потому что управление доступом в них осуществляется как на основе матрицы прав доступа для ролей, так и с помощью правил, регламентирующих назначение ролей пользователям и их активации во время сеансов. Поэтому ролевая модель представляет собой совершенно особый тип политики, основанный на компромиссе между гибкостью и простотой управления доступом (характерным для дискреционных моделей), так и жёсткостью правил контроля доступа (соответствующие мандатным моделям).

В ролевой модели классическое понятие *субъект* заменяется понятиями *пользователь* и *роль*. Пользователь – это человек, работающий с системой и выполняющий определённые обязанности. Роль – это активно действующая в системе абстрактная сущность, с которой связан ограниченный, логический связный набор полномочий, необходимый для осуществления определённой деятельности.

Такой подход в политике безопасности позволяет учесть разделение обязанностей и полномочий между участниками прикладного информационного процесса, так как с точки зрения ролевой политики имеет значение не личность пользователя, осуществляющего доступ к информации, а то какие полномочия ему необходимы для выполнения его служебных обязанностей (Зегжда Д.П. 2000).

В качестве критерия безопасности ролевой модели используется следующее правило (Зегжда Д.П. 2000): *система считается безопасной, если любой пользователь системы, работающий в определённом сеансе S, может осуществлять действия, требующие полномочия P только в том случае, если P принадлежит разрешениям S.*

2.2.6 Особенности реализации моделей безопасности в СУБД

Почти все крупные производители СУБД ограничиваются развитием концепции конфиденциальности, целостности и доступности данных, а их действия направлены, в основном, на преодоление существующих и уже известных уязвимостей, реализацию основных моделей доступа и рассмотрение вопросов, специфичных для конкретной СУБД. Такой подход обеспечивает решение конкретных задач, но не способствует появлению общей концепции безопасности для такого класса ПО, как СУБД. Это значительно усложняет задачу по обеспечению безопасности хранилищ данных на предприятии.

Обычно пользователей СУБД (субъектов) можно разделить на три группы, что влияет на применение определённых моделей безопасности (разграничения доступа) (Лилия Козленко 2021):

- Прикладные программисты – отвечают за создание программ, использующих базу данных. Программист может быть как пользователем, имеющим права создания и управления объектами (данными), так и пользователем, имеющим права только управления данными
- Конечные пользователи базы данных – работают с БД непосредственно через терминал или рабочую станцию. Как правило, конечные пользователи имеют строго ограниченный набор прав управления данными. Этот набор может определяться при конфигурировании интерфейса конечного пользователя и не изменяться. Политику безопасности в данном случае определяет администратор безопасности или администратор БД (если это одно и то же должностное лицо)

- Администраторы баз данных – образуют особую категорию пользователей СУБД. Они создают сами БД, осуществляют технический контроль функционирования СУБД, обеспечивают необходимое быстродействие системы. В обязанности администратора, также входит обеспечение пользователям доступа к необходимым им данным, написание (или оказание помощи в определении) необходимых пользователю внешних представлений данных. Администратор определяет политику безопасности

Отметим, что в настоящее время для контроля доступа в СУБД как правило используются дискреционные и ролевые модели. Мандатные модели значительно менее популярны в силу ряда недостатков в их использовании в СУБД.

Дискреционные и ролевые модели доступа При использовании дискреционных моделей в СУБД:

- Субъект – системный идентификатор, от имени которого СУБД выполняет определенные действия над определенными объектами. Понятие субъекта отличается от понятия пользователь компьютерной системы, поскольку инициировать изменение информации могут также и системные процессы

- Объект защиты – часть БД, на которую распространяется действие конкретного правила безопасности; это может быть группа отношений, отдельное отношение, подмножества атрибутов и т.д.

- Привилегия – действие над объектом защиты, которое может быть совершено от имени конкретного идентификатора (по сути элемент из множества отношений между субъектами и объектами – право на доступ)

Чаще всего владельцем базы данных является Администратор БД. Ему предоставлены все системные и объектные привилегии. В частности, он имеет право регистрации новых пользователей и предоставления им привилегий, как системных, так и объектных. Пользователь, имеющий системные привилегии, является владельцем всех созданных им объектов, имеет по отношению к ним все привилегии и может предоставлять их полностью или частично другим пользователям. Минимальной системной привилегией является право подключения к СУБД - привилегия входа (В.В. 2017).

Пользователи могут быть объединены в специальные группы пользователей. Один пользователь может входить в несколько групп. Для пользователей с минимальным стандартным набором прав вводится понятие группы PUBLIC. По умолчанию предполагается, что каждый вновь создаваемый пользователь, если специально не указано иное, относится к группе PUBLIC (как минимум в PostgreSQL).

Привилегии конкретному пользователю могут быть назначены администратором явно и неявно, например, через роль. Роль - это еще один возможный именованный носитель привилегий. Существует ряд стандартных ролей, которые проектируются при разработке СУБД. Также имеется возможность создавать новые роли, группируя в них произвольные полномочия. Введение ролей позволяет упростить управление привилегиями пользователей, структурировать этот процесс. Кроме того, введение ролей не связано с конкретными пользователями, поэтому роли могут быть определены и сконфигурированы до того, как определены пользователи системы.

Операторы SQL предоставления и отмены привилегий. В стандарте SQL определены два оператора GRANT и REVOKE для предоставления и отмены привилегий соответственно. ***Оператор предоставления привилегий*** имеет следующий формат (В.В. 2017):

```
GRANT {<action list>|ALL PRIVILEGES} ON <object name>
    TO {<user list>|PUBLIC} [WITH GRANT OPTION];
```

, где:

- <action list>(<список действий>) определяет набор действий из доступного списка действий над объектом данного типа (параметр ALL PRIVILEGES указывает, что разрешены все действия, допустимые для объектов данного типа),
- <object name>(<имя объекта>) определяет имя объекта защиты: таблицы, представления, хранимой процедуры или триггера,
- <user list>(<список пользователей>) определяет список идентификаторов пользователей, кому предоставляются данные привилегии. Вместо списка идентификаторов можно воспользоваться параметром PUBLIC.

Параметр WITH GRANT OPTION является необязательным и определяет режим, при котором передаются не только права на указанные действия, но и право передавать эти права другим пользователям. Передавать права в этом случае пользователь может только в рамках разрешенных ему действий. В общем случае набор привилегий зависит от реализации СУБД (определяется производителем). Выделяются привилегии манипулирования данными:

- SELECT — просматривать данные;
- INSERT [(<список полей>)] -– добавлять данные;
- UPDATE [(<список полей >)] -– обновлять данные;
- DELETE — удалять данные;
- REFERENCES [(<список полей >)] – ссылаться на указанные поля при определении ссылочной целостности;
- USAGE — использовать домены и ограничители целостности;
- EXECUTE — выполнять сохраненные процедуры и функции.

Среди привилегии создания/изменения объектов БД выделим наиболее часто используемые:

- CREATE <тип объекта> – создание объекта некоторого типа;
- ALTER <тип объекта> – изменение структуры объекта;
- DROP <тип объекта> – удаление объекта;
- ALL – все возможные действия над объектом.

Для отмены ранее назначенных привилегий в стандарте SQL определен оператор REVOKE. ***Оператор отмены привилегий*** имеет следующий синтаксис (В.В. 2017):

```
REVOKE {<action list>|ALL PRIVILEGES} ON <object name>
    FROM {<user list>|PUBLIC} {CASCADE|RESTRICT};
```

Параметры CASCADE или RESTRICT определяют, каким образом должна производиться отмена привилегий. Параметр CASCADE отменяет привилегии не только пользователя, который непосредственно упоминался в операторе GRANT при предоставлении ему привилегий, но и всем пользователям, которым этот пользователь присвоил привилегии, воспользовавшись параметром WITH GRANT OPTION.

Дискреционная модель является очень популярной у разработчиков СУБД. Она реализована в практически всех SQL-совместимых СУБД. Операторы SQL GRANT, REVOKE, DENY, реализующие дискреционную модель разграничения доступа, определены в стандарте языка SQL.

Ролевая модель в СУБД. Рассмотренная дискреционная модель в СУБД очень похожа на ролевую модель. Основное отличие ролевой модели в СУБД – это обязательное использование принципа наименьших привилегий и то, что после авторизации пользователя в системе для него создается сессия. При реализации ролевой модели может быть введен ряд ограничений (В.В. 2017): например, назначение роли главного администратора (суперпользователя) может быть предоставлено только одному пользователю, вводится запрет совмещения одним пользователем определенных ролей, ограничивается количество пользователей, одновременно выполняющих определенную роль и т.п.

К недостаткам ролевого разграничения доступа в СУБД относится отсутствие формальных доказательств безопасности компьютерной системы, возможность внесения дублирования и избыточности при предоставлении пользователям прав доступа и сложность конструирования ролей (В.В. 2017).

Наряду с дискреционной, ролевая модель разграничения доступа, реализована во множестве развитых коммерческих СУБД, например: Microsoft SQL Server, Oracle, IBM DB2 (В.В. 2017).

Мандатные модели доступа в СУБД Обычно мандатную модель доступа в СУБД реализуют с помощью модели Белл-ЛаПадула, о которой мы поговорим позже при рассмотрении конкретных примеров мандатных моделей.

Перечислим основные недостатки применения мандатных моделей доступа в СУБД (В.В. 2017):

- Невозможность автоматизации назначения уровней секретности и определения границ защищаемых данных, что в больших системах может приводить к практически бесконечному ручному процессу конфигурации системы
- Снижение эффективности работы компьютерной системы, так как проверка прав доступа субъекта к объекту выполняется не только при открытии объекта в процессе субъекта, но и перед выполнением любой операции чтения из объекта или записи в объект
- Создание дополнительных неудобств в работе пользователей компьютерной системы, связанных с невозможностью изменения информации в неконфиденциальном объекте, если тот же самый процесс использует информацию из конфиденциального объекта (его уровень конфиденциальности больше нуля). Это зачастую решается путем разрешения пользователю выступать от имени субъекта с меньшим уровнем доступа, что в свою очередь приводит к деградации системы защиты.

Из-за отмеченных недостатков мандатного разграничения доступа в реальных СУБД множество объектов, к которым применяется мандатное разграничение, является подмножеством множества объектов, доступ к которым осуществляется на основе дискреционного разграничения.

Примером реализации мандатного подхода разграничения доступа можно считать компонент Oracle Label Security (OLS), реализованный в СУБД Oracle, начиная с 9 версии. Примером Российской СУБД, реализующей стандарт SQL-92 является СУБД ЛИНТЕР (В.В. 2017).

Проблемы реализации многозначности Даже если отвлечься от проблемы хранения в базе данных разного рода легенд и прикрытий, многозначность все равно должна стать неотъемлемым свойством баз данных с многоуровневой защитой. Порассуждаем, каким образом следовало бы обрабатывать отношение с многоуровневой защитой в отсутствие многозначности. База данных может скрывать значения, недоступные пользователю или процессу по соображениям безопасности, и такое маскирование обычно осуществляется при помощи пустых значений (null values).

Напомним, что каждый кортеж существует в единственном экземпляре, поэтому непривилегированный процесс может попытаться записать неклассифицированные данные в столбцы кортежа, содержащие как бы пустые значения. В связи с этим в СУБД с многоуровневой защитой возникает проблема обработки таких операций, поскольку на самом деле значение столбца непустое, и замещать его нельзя. Естественно, что подобные попытки должны отвергаться средствами системы безопасности.

Однако неудача при выполнении операции модификации открывает определенные возможности, получившие название косвенных каналов (для получения закрытой информации), т. е. пути для снижения классификации данных (преднамеренного или случайного). Отказ в выполнении вроде бы вполне законной транзакции на модификацию пустых значений в базе данных с многоуровневой защитой выдает информацию о том, что эти элементы на самом деле не пусты, а содержат классифицированные данные. Уведомляя пользователя или процесс, не наделенных соответствующими полномочиями, о наличии скрытых данных в некотором кортеже, мы тем самым открываем косвенный канал.

Допустим, что СУБД с многоуровневой защитой симулирует выполнение подобных модификаций, чтобы не раскрывать скрытый канал. Если пользователь, только что изменивший некоторые данные, попытается обратиться к ним, ему ответят, что они отсутствуют. Тогда мы опять создаем косвенный канал.

А что если попытаться поддерживать в СУБД с многоуровневой защитой нечто вроде журнала таких модификаций и использовать его для последующих выборок? Знакомая идея, и неудивительно – ведь мы только что говорили о многозначности. Даже если мы предусмотрим какое-нибудь специфическое решение для реализации многозначности, например хранить два экземпляра кортежей, – это всего лишь одна из реализаций. Важно, что многозначность – это существенное свойство данных с многоуровневой защитой, а как она будет реализована – это вопрос второстепенный, не имеющий отношения к самой идее по-разному представлять информацию для пользователей разных уровней благонадежности; хотя способы реализации многозначности остаются важным направлением для исследовательской работы.

Рассматривая проблему реализации многозначности, полезно обратить внимание на концептуальное сходство между многозначными кортежами в базе данных с многоуровневой защитой и кор-

тежами с множественными значениями в хронологических базах данных. Напомним, что в большинстве реляционных баз данных с расширениями для поддержки хронологизма должны существовать несколько экземпляров одного кортежа с одним и тем же значением первичного ключа, чтобы обеспечить отображение и текущего состояния объекта, и исторической информации о нем. В реляционной СУБД с расширениями для поддержки многоуровневой защиты по существу присутствует такой же тип представления с той разницей, что определяющим признаком экземпляров некоторого кортежа в хронологической базе данных является время, а в базе данных с многоуровневой защитой – класс доступа. Хотя с других точек зрения, помимо реализации многозначности, хронологические СУБД и СУБД с многоуровневой защитой имеют мало общего, в связи с коммерциализацией СУБД обоих типов может оказаться полезным применять в них общие алгоритмы представления и доступа к многозначным кортежам.

Итак, многозначность, примененная впервые в проекте SeaView, стала общепринятой моделью реализации баз данных с многоуровневой защитой, в особенности реляционных. Многозначность активно исследуется и в качестве механизма обеспечения безопасности в объектно-ориентированных базах данных (развитие средств защиты в СУБД этого типа пока еще значительно отстает по сравнению с реляционными СУБД, но их значение будет неуклонно расти вместе со стремлением позиционировать ООСУБД на рынке продуктов для коммерческих и правительственных организаций).

3 Механизмы обеспечения целостности СУБД

3.1 Угрозы целостности СУБД

Задача обеспечения целостности предусматривает комплекс мер по предотвращению непреднамеренного изменения или уничтожения информации, используемой информационной системой управления или системой поддержки принятия решений. Изменение или уничтожение данных может быть следствием неблагоприятного стечения обстоятельств и состояния внешней среды (стихийные бедствия, пожары и т. п.), неадекватных действий пользователей (ошибки при вводе данных, ошибки операторов и т. п.) и проблем, возникающих при многопользовательской обработке данных (Лихоносов А. Г. 2011).

Например, с помощью SQL-операторов UPDATE, INSERT и DELETE можно изменить данные в СУБД. Опасность заключается в том, что пользователь, обладающий соответствующими привилегиями, может модифицировать все записи в таблице (Утебов Д. Р. 2008).

Основные виды и причины возникновения угроз целостности

Перечислим основные угрозы целостности информации (Пирогов 2009):

1. **Отказ пользователей**. Под пользователями в данном случае мы понимаем широкий круг персонала, работающего с системой: операторы, програм- мисты, администраторы и т. д.

    - Непреднамеренные ошибки. Непреднамеренная ошибка может вызвать непосредственно порчу данных или средств доступа, либо создать условия для реализации другой угрозы, например вторжения злоумышленника.

- Нежелание работать с информационной системой. Причиной нежелания может, например, быть необходимость освоения новых возможностей системы или несоответствие системы запросам пользователей.
- Невозможность работать с системой. Причиной невозможности работать с системой может быть как отсутствие соответствующей подготовки персонала, так и отсутствие необходимой документации по системе.

2. **Внутренние отказы информационной системы**. Основными источниками внутренних отказов могут быть:

   - случайное или умышленное отступление от правил эксплуатации. Например, правила могут предусматривать определенный набор параметров сервера (объем памяти, производительность процессора, объем дискового пространства, версия операционной системы и т. п.), на котором предполагается использовать ИС;
   - выход системы из штатного режима эксплуатации в силу случайных или преднамеренных действий пользователей;
   - ошибки конфигурирования системы. В сложных системах конфигурирование выполняется при установке и настройке системы. При неправильной настройке могут возникнуть проблемы в эксплуатации;
   - отказ программного обеспечения. Программное обеспечение может содержать ошибки, в том числе и такие, которые могут привести к серьезным повреждениям данных. Кроме этого преднамеренно может быть изменен алгоритм программы. Таким образом, следует защищать программное обеспечение информационной системы и от случайного повреждения, и от исправления непосредственно исполняемых модулей;
   - разрушение данных (возможно, преднамеренное). На заре компьютерной революции, когда не слишком заботились о безопасности данных, часто сталкивались с ситуацией, когда не совсем компетентный пользователь просто стирал важные (но плохо защищенные) данные с диска.

3. **Внешние источники возможного нарушения доступа к данным**. Дан- ные источники могут быть вызваны и злонамеренными действиями людей, и стихийными бедствиями или авариями.

   - Отказ, повреждение или разрушение аппаратных средств (носителей информации, компьютеров, каналов связи). При интенсивном использовании отказ аппаратных средств случается совсем не редко, и меры безопасности должны учитывать такую возможность.
   - Нарушение условий работы (системы связи, электропитание, отопление и т. п.). Отключение электричества совсем недавно было серьезной проблемой функционирования компьютерных систем. Источники бесперебойного питания должны защищать не только сами компьютеры, но все устройства в сети.
   - Разрушение или повреждение помещений. Конечно, такая ситуация на первый взгляд кажется мало возможной, но это вполне вероятно в регионах, например, с сейсмической неустойчивостью.

- Невозможность или отказ обслуживающего персонала выполнять свои обязанности (стихийные бедствия, волнения, забастовка и т. п.). Отказ персонала выполнять свои обязанности для некоторых систем может привести к катастрофическим последствиям.
- Сетевые атаки, вирусные программы и другое вредоносное программное обеспечение. Сетевые атаки последнее время стали сильнейшим фактором риска информационных систем, работающих в Интернете.
- Разрушение информации намеренными действиями человека (диверсия). В данном случае речь идет о действиях людей, не являющихся обслуживающим персоналом данной системы.

Способы противодействия

Основными средствами защиты целостности информации в ИС являются (Пирогов 2009):

- транзакционные механизмы, позволяющие восстановить целостность дан- ных в случае незначительных сбоев;
- контроль ввода данных. Много ошибок можно было бы избежать, если программы не пропускали бы заведомо противоречивые данные;
- использование средств защиты целостности СУБД;
- резервное копирование данных;
- периодическое тестирование системы на предмет нарушения целостности.

3.2 Метаданные и словарь данных

Метаданные – Это данные, описывающие другие данные. Это важный элемент хранилища данных, который предоставляет возможность показывать пользователю предметно-ориентированную структуру, а не набор абстрактно-связанных таблиц. Метаданные предназначены для хранения информации о происхождении данных, о любых изменениях данных, о расположении данных, об ограничениях на данные, о соответствии данных тем или иным объектам предметной области и т. д. (Пирогов 2009)

Назначение словаря данных

Согласно канонам реляционной модели данных описание структуры базы данных (описание объектов, входящих в базу данных) также должно храниться в таблицах. Такая структура называется словарем данных или системным каталогом. По сути, такие данные следует назвать метаданными, т. е. данными, описывающими структуру других данных. В 1992 году в стандарте языка SQL было дано описание такого системного каталога. Но к этому времени в различных СУБД были приняты свои принципы хранения метаданных, отказаться от которых не так-то просто. В результате в одних СУБД вообще отсутствуют стандартные подходы к хранению метаданных, в других эти подходы в значительной степени дополнены своими особенностями. (Пирогов 2009)

Типы словарей данных

Существует два типа словарей данных. Активные и пассивные, они отличаются уровнем автоматической синхронизации.

Активные словари данных. Это словари данных, созданные в описываемых ими базах данных, которые автоматически отражают любые изменения внутри этих баз, что позволяет избежать любых несоответствий между словарями данных и описываемыми данными. **Словари пассивных данных.** Это словари данных, созданные как отдельные от описываемых ими баз данных сущности. Пассивные словари данных требуют дополнительной логики для синхронизации с базами данных, которые они описывают, и с ними нужно обращаться осторожно. Компоненты словаря данных Конкретное содержимое словаря данных может варьироваться. Как правило, эти компоненты представляют собой различные типы метаданных, предоставляющие информацию о данных. (Chai 2021)

Доступ к словарю данных

По 4 правилу Кодда (Dynamic On-Line Catalog Based on the Relational Model) словарь данных должен сохраняться в форме реляционных таблиц, и СУБД должна поддерживать доступ к нему при помощи стандартных языковых средств.

Для доступа к словарю данных используются инструкции SQL. Так как словарь данных доступен только для чтения, допускается выполнять только запросы таблиц и представлений. Можно запрашивать представления словаря, которые основаны на таблицах словаря, чтобы найти сведения, такие как (Sql-oracle б.г.):

- определения всех объектов схемы в словаре (таблицы, представления, индексы, синонимы, последовательности, процедуры, функции, пакеты, триггеры и так далее);
- значения по умолчанию для столбцов;
- сведения об ограничениях целостности;
- имена пользователей Oracle;
- привилегии и роли, предоставленные каждому пользователю;
- другие общие сведения о базе данных.

Состав словаря

В состав словаря данных базы данных могут входить: (Chai 2021)

- списки объектов данных (имена и определения);
- свойства элемента данных (такие как тип данных, уникальные идентификаторы, размер, допустимость значений NULL, индексы и параметр required);
- ER-диаграммы[12];

[12]Схема «сущность-связь» (также ERD или ER-диаграмма) — это разновидность блок-схемы, где показано, как разные «сущности» (люди, объекты, концепции и так далее) связаны между собой внутри системы.

- диаграммы системного уровня;
- справочные данные — доступные пользователю представления, которые суммируют и отображают в удобном формате информацию из базовых таблиц словаря. Эти представления декодируют информацию базовых таблиц, представляя ее в полезном виде, таком как имена пользователей или таблиц, чтобы добиться человекочитаемости данных. Большинство пользователей имеют доступ к этим представлениям вместо диаграмм системного уровня;
- бизнес-логика (например, для проверки качества данных и объектов схемы);

Словарь данных базы данных Oracle имеет два основных применения (Кирилов 2009):

- Oracle обращается к словарю данных каждый раз, когда выдается предложение DDL;
- каждый пользователь Oracle может использовать словарь данных как только читаемый справочник по базе данных.

При этом словарь данных всегда доступен при открытой базе данных. Он размещается в табличном пространстве SYSTEM, которое всегда находится в состоянии online, когда база данных открыта.

В MongoDB подобными методами получения информации о БД или таблицах являются dbStats, collStatsm rolesInfo и тому подобные.(MongoDB 2021)

Представления словаря

Рассматривается СУБД от вышеупомянутого Oracle. Словарь данных состоит из нескольких наборов представлений. Во многих случаях такой набор состоит из трех представлений, содержащих аналогичную информацию и отличающихся друг от друга своими префиксами (Кирилов 2009):

- USER — представление для пользователя;
- ALL — расширенное представление для пользователя;
- DBA — представление администратора.

Столбцы в каждом представлении набора идентичны, но имеются исключения. В представлениях с префиксом USER обычно нет столбца с именем OWNER (владелец); в представлениях USER под владельцем подразумевается пользователь, выдавший запрос. Некоторые представления DBA имеют дополнительные столбцы, которые содержат информацию, полезную для АБД.

Представления с префиксом USER:

- отражают окружение пользователя в базе данных, включая информацию о созданных им объектах, предоставленных им грантах и т. д.;
- выдают только строки, имеющие отношение к пользователю;
- имеют столбцы, идентичные с другими представлениями, с тем исключением, что столбец OWNER подразумевается (текущий пользователь);

- возвращают подмножество информации, предоставляемой представлениями ALL;
- могут иметь сокращенные общие синонимы для удобства.

Представления с префиксом ALL отражают общее представление о базе данных со стороны пользователя. Эти представления возвращают информацию об объектах, к которым пользователь имеет доступ через общие или явные гранты, помимо тех объектов, которыми владеет этот пользователь.

Представления с префиксом DBA показывают общее представление о базе данных, и предназначены для администраторов базы данных. Во время своей работы Oracle поддерживает набор "виртуальных"таблиц, в которых регистрируется текущая информация о базе данных. Эти таблицы называются динамическими таблицами производительности. Так как динамические таблицы производительности не являются истинными таблицами, большинство пользователей не должно обращаться к ним. Динамические таблицы производительности принадлежат схеме SYS, а их имена начинаются с V_$. По этим таблицам создаются представления, а для представлений создаются синонимы, имена которых начинаются с V$.

3.3 Понятие транзакции

> Последовательность действий над данными, обрабатываемая СУБД как единая операция, будем называть **транзакцией**.

Из определения ясно, что транзакция может состоять из одной или нескольких команд языка SQL. При этом команды языка SQL могут перемежаться командами, не выполняющими непосредственно операций над данными (не читающими данные, не изменяющими данные, не изменяющими структуру данных). Таким образом, в качестве транзакции может быть выбрана любая часть программы, содержащая команды чтения или записи данных. В частности, транзакция может состоять всего из одной команды, например INSERT, или из сотен команд, изменяющих или не изменяющих данные. Понятие транзакции чрезвычайно емкое. Оно в действительности тесно связано с концептуальной моделью данных и всей информационной системы. Дело в том, что на уровне пользователя операции над данными носят предметный характер: начислить зарплату, уволить работника, перевести деньги на другой счет и т. п. Для выполнения таких операций часто необходимо выполнить десятки, и даже сотни команд языка SQL. Однако вся эта последовательность команд должна быть подчинена одной цели: операция уровня пользователя должна быть обязательно выполнена. Что будет, если при выполнении такой цепочки команд произойдет сбой? Как рассматривать тогда выполняемую операцию — как частично выполненную? Но как в дальнейшем система узнает, что операция была только частично выполнена, и как ее закончить? Все эти вопросы, в конце концов, и приводят к понятию **"транзакция"** (Пирогов 2009).

Фиксация транзакции

Согласно требованиям ACID (atomic, consistent, isolated, durable) транзакция должна быть устойчивой. После своего завершения она сохраняется в системе, которую ничто не может вернуть в исходное (до начала транзакции) состояние, т. е. происходит фиксация транзакции, означающая,

что ее действие постоянно даже при сбое системы. При выполнении отдельных операций транзакции могут быть нарушены какие-либо требования целостности данных (в первую очередь имеются в виду корпоративные правила целостности, см. главу 2). Однако по окончании выполнения транзакции (фиксация транзакции) все правила целостности базы данных будут соблюдены. Согласно стандарту транзакция начинается с первой команды, которая обращается к данным. После этого транзакция продолжается до тех пор, пока не будет выполнена команда COMMIT WORK (или просто COMMIT) либо не будет закрыто соединение к базе данных. При выполнении команды COMMIT WORK происходит фиксация транзакции, другими словами, после этой команды откат транзакции уже будет невозможен (Пирогов 2009).

Прокрутки вперед и назад

В результате сбоя СУБД могут возникнуть две потенциальные ситуации (Карпова 2009):

- Блоки, содержащие подтверждённые модификации, не были записаны в файлы данных, так что эти изменения отражены лишь в журнале транзакций. Следовательно, журнал транзакций содержит подтверждённые данные, которые должны быть переписаны в файлы данных.
- Журнал транзакций и блоки данных содержат изменения, которые не были подтверждены. Изменения, внесенные неподтверждёнными транзакциями, во время восстановления БД должны быть удалены из файлов данных.

Для того чтобы разрешить эти ситуации, СУБД автоматически выполняет два этапа при восстановлении после сбоев: прокрутку вперед и прокрутку назад.

1. Прокрутка вперед заключается в применении к файлам данных всех изме-нений, зарегистрированных в журнале транзакций. После прокрутки вперед файлы данных содержат все как подтверждённые, так и неподтверждённые изменения, которые были зарегистрированы в журнале транзакций.
2. Прокрутка назад заключается в отмене всех изменений, которые не были подтверждены. Для этого используются журнал транзакций и сегменты отката, информация из которых позволяет определить и отменить те транзакции, которые не были подтверждены, хотя и попали на диск в файлы БД.

После выполнения этих этапов восстановления БД находится в согласованном состоянии и с ней можно работать.

Контрольная точка

Критическим моментом в отказе системы является потеря содержимого основной (оперативной) памяти (в частности, буферов базы данных). Поскольку точное состояние любой выполнявшейся в момент отказа системы транзакции остается неизвестным, такая транзакция не может быть успешно завершена. Поэтому при перезапуске системы любая такая транзакция будет отменена (т.е. будет выполнен ее откат). Более того, при перезапуске системы, возможно, потребуется повторно выполнить некоторые транзакции, которые были успешно завершены до аварийного останова, но выполненные ими обновления еще не были перенесены из буферов оперативной памяти в физическую базу данных во вторичной памяти. Возникает очевидный вопрос: как система определяет в

процессе перезапуска, какую транзакцию следует отменить, а какую выполнить повторно? Ответ заключается в том, что система автоматически создает контрольные точки с некоторым наперед заданным интервалом (обычно, когда в журнале накапливается определенное число записей). Для создания контрольной точки требуется, во-первых, выполнить принудительное сохранение содержимого буферов оперативной памяти в физической базе данных, и, во-вторых, осуществить принудительное сохранение специальной записи контрольной точки в журнале на физическом носителе. Запись контрольной точки содержит список всех транзакций, выполняемых в тот момент, когда создавалась контрольная точка (Дейт 2005).

Откат

Для управления транзакциями в системах, поддерживающих механизм транзакций и язык SQL, используется оператор отката транзакции (отмены изменений): ROLLBACK [WORK]. Для фиксации или отката транзакции система создаёт неявные точки фиксации и отката. По команде rollback система откатит транзакцию на начало (на неявную точку отката). Для обеспечения целостности транзакции СУБД может откладывать запись изменений в БД до момента успешного выполнения всех операций, входящих в транзакцию, и получения команды подтверждения транзакции (commit). Но чаще используется другой подход: система записывает изменения в БД, не дожидаясь завершения транзакции, а старые значения данных сохраняет на время выполнения транзакции в сегментах отката. Сегмент отката (rollback segment, RBS) – это специальная область памяти на диске, в которую записывается информация обо всех текущих (незавершённых) изменениях. Обычно записывается "старое"и "новое"содержимое изменённых записей, чтобы можно было восстановить прежнее состояние БД при откате транзакции (по команде rollback) или при откате текущей операции (в случае возникновения ошибки). Данные в RBS хранятся до тех пор, пока транзакция, изменяющая эти данные, не будет завершена. Потом они могут быть перезаписаны данными более поздних транзакций. Команда savepoint запоминает промежуточную "текущую копию"состояния базы данных для того, чтобы при необходимости можно было вернуться к состоянию БД в точке сохранения: откатить работу от текущего момента до точки сохранения (rollback to <имя_точки>) или зафиксировать работу от начала транзакции до точки сохранения (commit to <имя_точки>). На одну транзакцию может быть несколько точек сохранения (ограничение на их количество зависит от СУБД). Для сохранения сведений о транзакциях СУБД ведёт журнал транзакций. Журнал транзакций — это часть БД, в которую поступают данные обо всех изменениях всех объектов БД. Журнал недоступен пользователям СУБД и поддерживается особо тщательно (иногда ведутся две копии журнала, хранимые на разных физических носителях). Форма записи в журнал изменений зависит от СУБД. Но обычно там фиксируется следующее:

- номер транзакции (номера присваиваются автоматически по возрастанию);
- состояние транзакции (завершена фиксацией или откатом, не завершена, находится в состоянии ожидания);
- точки сохранения (явные и неявные);
- команды, составляющие транзакцию, и проч.

Начало транзакции соответствует появлению первого исполняемого SQLоператора. При этом в журнале появляется запись об этой транзакции (Карпова 2009).

Транзакции как средство изолированности пользователей

Поддержание механизма транзакций —- показатель уровня развитости СУБД и основа обеспечения целостности базы данных. Транзакции также составляют основу изолированности в многопользовательских системах, где с одной базой данных параллельно могут работать несколько пользователей и (или) прикладных программ. Одна из основных задач СУБД —- обеспечение изолированности, т. е. создание такого режима функционирования, при котором каждому пользователю казалось бы, что база данных доступна только ему. Такую задачу СУБД принято называть параллелизмом транзакций. Большинство выполняемых действий производится в теле транзакций. По умолчанию каждая команда выполняется как самостоятельная транзакция. Как было показано ранее, при необходимости пользователь может явно указать ее начало и конец, чтобы иметь возможность включить в нее несколько команд. Решение проблемы параллельной обработки базы данных заключается в том, что строки таблиц блокируются, а последующие транзакции, модифицирующие эти строки, отвергаются и переводятся в режим ожидания. В связи со свойством сохранения целостности базы данных транзакции являются подходящими единицами изолированности пользователей. Действительно, если каждый сеанс взаимодействия с базой данных реализуется транзакцией, то пользователь начинает с того, что обращается к согласованному состоянию базы данных — состоянию, в котором она могла бы находиться, даже если бы пользователь работал с ней в одиночку. Если бы в СУБД не были реализованы механизмы блокирования, то при одновременном чтении и изменении одних и тех же данных несколькими пользователями могли бы возникнуть проблемы одновременного доступа.

Конфликты транзакций и уровни изоляции

Уровни изоляции определяют степень, в которой транзакция должна быть изолирована от изменений данных, сделанных любой другой транзакцией в системе базы данных. Уровни изоляции отличаются разрешениями следующих конфликтов:

- **Потерянное обновление (lost update)** — когда несколько транзакций что-то обновили в БД, но по итогам результат такой, будто отработала лишь часть транзакций. Самый опасный побочный эффект, по сути, полное отсутствие изоляции транзакций — две транзакции читают одну ячейку, записывают изменённое значение (одна вычитает стоимость мороженого, другая плату за квартиру). В итоге в ячейке значение от второй транзакции, а первой как будто и не было.

- **Грязное чтение (dirty read)** — ситуация, когда транзакция считывает данные, которые еще не были зафиксированы. Например, транзакция 1 обновляет строку и оставляет ее незафиксированной, в то время как транзакция 2 читает обновленную строку. Если транзакция 1 откатывает изменение, транзакция 2 будет считывать данные, которые считаются никогда не существовавшими.

- **Неповторяемое чтение (non-repeatable read)** — когда транзакция дважды считывает одну и ту же строку и каждый раз получает другое значение. Например, предположим, что транзакция T1 считывает данные. Из-за параллелизма другая транзакция T2 обновляет те

	Фантомное чтение	Неповторяющееся чтение	Грязное чтение	Потерянное обновлени
Serializable				
Repeatable Read				
Read Committed				
Read Uncommitted				

же данные и фиксирует их. Теперь, если транзакция T1 повторно считывает те же данные, она получит другое значение.

- **Фантомное чтение (phantom reads)** — когда выполняются два одинаковых запроса, но извлекаемые ими строки различаются. Например, транзакция T1 извлекает набор строк, удовлетворяющих некоторым критериям поиска. Далее транзакция T2 создает несколько новых строк, соответствующих критериям поиска для транзакции T1. Если транзакция T1 повторно выполняет инструкцию, которая считывает строки, на этот раз она получает другой набор строк. Отличие от предыдущего пункта в том, что в этом случае происходит агрегация строк, а в предыдущем происходит чтение лишь одной строки.

Основываясь на политиках разрешения перечисленных конфликтов, стандарт SQL определяет четыре уровня изоляции:

- **Read Uncommitted** — это самый низкий уровень изоляции. На этом уровне одна транзакция может считывать еще не зафиксированные изменения, сделанные другими транзакциями, тем самым допуская грязное чтение. На этом уровне транзакции не изолированы друг от друга.
- **Read Committed** — уровень изоляции гарантирует, что любые считанные данные фиксируются в момент их чтения. Таким образом, он не допускает грязного чтения. Транзакция удерживает блокировку чтения или записи для текущей строки и, таким образом, предотвращает ее чтение, обновление или удаление другими транзакциями.
- **Repeatable Read** — транзакция удерживает блокировки чтения для всех строк, на которые она ссылается, и записывает блокировки для указанных строк для действий обновления и удаления. Поскольку другие транзакции не могут читать, обновлять или удалять эти строки, следовательно, это позволяет избежать неповторяющегося чтения.
- **Serializable** — самый высокий уровень изоляции. Выполнение определяется как выполнение операций, в которых одновременно выполняемые транзакции кажутся последовательно выполняемыми(Paul Wilton 2021).

Методы сериализации транзакций

Делятся на три типа:

- **С блокирующим планировщиком (blocking sheduller)** — для каждого запроса планировщик запрашивает блокировку. Каждая блокировка запрашивается в определенном режиме (чтение или запись). Если запрашиваемый элемент данных еще не заблокирован в несовместимом режиме, блокировка предоставляется; в противном случае возникает конфликт блокировок и транзакция блокируется, пока текущий владелец блокировки не освободит блокировку. Например: двухфазные AL (Altruistic Locking), O2PL (Ordered Sharing of Locks), 2PL

(Two-Phase Locking), C2PL (Conservative Two-Phase Locking), S2PL (Strict Two-Phase Locking), SS2PL (Strong Strict Two-Phase Locking), не двухфазные WTL (Write-only Tree Locking), RWTL (Read-Write Tree Locking).

- **С неблокирующим планировщиком (non-blocking sheduller)** — запрос отрабатывает, не имея возможности заблокировать другой запрос, конфликты разрешаются постфактум. Например: TO (Timestamp Ordering), SGT (Serialization Graph Testing).
- **Смешанный** — использует разные типы для разных видов конфликтов (rw/wr, ww).(Gerhard Weikum 2021)

3.4 Блокировки

Блокировки, называемые также синхронизационными захватами объектов, могут быть применены к разному типу объектов. Наибольшим объектом блокировки может быть вся БД, однако этот вид блокировки сделает БД недоступной для всех приложений, которые работают с данной БД. Следующий тип объекта блокировки —- это таблицы. Транзакция, которая работает с таблицей, блокирует ее на все время выполнения транзакции. Этот вид блокировки предпочтительнее предыдущего, потому что позволяет параллельно выполнять транзакции, которые работают с другими таблицами.

В ряде СУБД реализована блокировка на уровне страниц. В этом случае СУБД блокирует только отдельные страницы на диске, когда транзакция обращается к ним. Этот вид блокировки еще более мягок и позволяет разным транзакциям работать даже с одной и той же таблицей, если они обращаются к разным страницам данных.

В некоторых СУБД возможна блокировка на уровне строк, однако такой механизм блокировки требует дополнительных затрат на поддержку этого вида блокировки.

В настоящее время проблема блокировок является предметом большого числа исследований (Intuit 2020a).

Режимы блокировок

Рассматривают два типа блокировок (синхронизационных захватов) (Intuit 2020a):

- совместный режим блокировки — нежесткая, или разделяемая, блокировка, обозначаемая как S (Shared). Этот режим обозначает разделяемый захват объекта и требуется для выполнения операции чтения объекта. Объекты, заблокированные таким образом, не изменяются в ходе выполнения транзакции и доступны другим транзакциям также, но только в режиме чтения;
- монопольный режим блокировки — жесткая, или эксклюзивная, блокировка, обозначаемая как X (eXclusive). Данный режим блокировки предполагает монопольный захват объекта и требуется для выполнения операций занесения, удаления и модификации. Объекты, заблокированные данным типом блокировки, фактически остаются в монопольном режиме обработки и недоступны для других транзакций до момента окончания работы данной транзакции.

Правила согласования блокировок

Захваты объектов несколькими транзакциями по чтению совместимы, то есть нескольким транзакциям допускается читать один и тот же объект, захват объекта одной транзакцией по чтению не совместим с захватом другой транзакцией того же объекта по записи, и захваты одного объекта разными транзакциями по записи не совместимы. В примере, представленном на рис. 1 считается, что первой блокирует объект транзакция A, а потом пытается получить к нему доступ транзакция B.

		Транзакция B		
		Разблокирована	Нежесткая блокировка	Жесткая блокировка
Транзакция A	Разблокирована	Да	Да	Да
	Нежесткая блокировка	Да	Да	Нет
	Жесткая блокировка	Да	Нет	Нет

Рис. 1: Правила применения жесткой и нежесткой блокировок транзакций

Двухфазный протокол синхронизационных блокировок

В базах данных и обработке транзакций двухфазная блокировка (2PL) — это метод управления параллелизмом, который гарантирует сериализуемость. Так же называют результирующий набор графиков транзакций базы данных (истории). Протокол использует блокировки, применяемые транзакцией к данным, которые могут блокировать (интерпретировать как сигналы для остановки) другие транзакции от доступа к тем же данным в течение жизни транзакции.

По протоколу 2PL блокировки (locks) применяются и удаляются в два этапа:

1. Фаза расширения: блокировки берутся и ни одна блокировка не освобождается.
2. Фаза сокращения: блокировки освобождаются и ни одна блокировка не берётся.

В базовом протоколе используются два типа блокировок: Shared и Exclusive locks. Уточнения базового протокола могут использовать больше типов блокировок. Используя блокировки, блокирующие процессы, 2PL могут подвергаться взаимоблокировкам, которые являются результатом взаимной блокировки двух или более транзакций (Wikipedia 2020a).

Тупиковые ситуации, их распознавание и разрушение

Одним из наиболее чувствительных недостатков метода сериализации транзакций на основе синхронизационных захватов является возможность возникновение тупиков (deadlocks) между транзакциями.

Вот простой пример возникновения тупика между транзакциями T1 и T2:

1. транзакции T1 и T2 установили монопольные захваты объектов r1 и r2 соответственно;

2. после этого T1 требуется совместный захват r2, а T2 - совместный захват r1;

3. ни одна из транзакций не может продолжаться, следовательно, монопольные захваты не будут сняты, а совместные - не будут удовлетворены.

Поскольку тупики возможны, и никакого естественного выхода из тупиковой ситуации не существует, то эти ситуации необходимо обнаруживать и искусственно устранять.

Основой обнаружения тупиковых ситуаций является построение (или постоянное поддержание) графа ожидания транзакций. Граф ожидания транзакций - это ориентированный двудольный граф, в котором существует два типа вершин - вершины, соответствующие транзакциям, и вершины, соответствующие объектам захвата. В этом графе существует дуга, ведущая из вершины-транзакции к вершине-объекту, если для этой транзакции существует удовлетворенный захват объекта. В графе существует дуга из вершины-объекта к вершине-транзакции, если транзакция ожидает удовлетворения захвата объекта.

Легко показать, что в системе существует ситуация тупика, если в графе ожидания транзакций имеется хотя бы один цикл.

Для распознавание тупика периодически производится построение графа ожидания транзакций (как уже отмечалось, иногда граф ожидания поддерживается постоянно), и в этом графе ищутся циклы. Традиционной техникой (для которой существует множество разновидностей) нахождения циклов в ориентированном графе является редукция графа.

Не вдаваясь в детали, редукция состоит в том, что прежде всего из графа ожидания удаляются все дуги, исходящие из вершин-транзакций, в которые не входят дуги из вершин-объектов. (Это как бы соответствует той ситуации, что транзакции, не ожидающие удовлетворения захватов, успешно завершились и освободили захваты). Для тех вершин-объектов, для которых не осталось входящих дуг, но существуют исходящие, ориентация исходящих дуг изменяется на противоположную (это моделирует удовлетворение захватов). После этого снова срабатывает первый шаг и так до тех пор, пока на первом шаге не прекратится удаление дуг. Если в графе остались дуги, то они обязательно образуют цикл.

Предположим, что нам удалось найти цикл в графе ожидания транзакций. Что делать теперь? Нужно каким-то образом обеспечить возможность продолжения работы хотя бы для части транзакций, попавших в тупик. Разрушение тупика начинается с выбора в цикле транзакций так называемой транзакции-жертвы, т.е. транзакции, которой решено пожертвовать, чтобы обеспечить возможность продолжения работы других транзакций.

Грубо говоря, критерием выбора является стоимость транзакции; жертвой выбирается самая дешевая транзакция. Стоимость транзакции определяется на основе многофакторная оценка, в которую с разными весами входят время выполнения, число накопленных захватов, приоритет.

После выбора транзакции-жертвы выполняется откат этой транзакции, который может носить полный или частичный характер. При этом, естественно, освобождаются захваты и может быть продолжено выполнение других транзакций.

Естественно, такое насильственное устранение тупиковых ситуаций является нарушением принципа изолированности пользователей, которого невозможно избежать.

Заметим, что в централизованных системах стоимость построения графа ожидания сравнительно невелика, но она становится слишком большой в по-настоящему распределенных СУБД, в которых транзакции могут выполняться в разных узлах сети. Поэтому в таких системах обычно

используются другие методы сериализации транзакций.

Еще одно замечание. Чтобы минимизировать число конфликтов между транзакциями, в некоторых СУБД (например, в Oracle) используется следующее развитие подхода. Монопольный захват объекта блокирует только изменяющие транзакции. После выполнении операции модификации предыдущая версия объекта остается доступной для чтения в других транзакциях. Кратковременная блокировка чтения требуется только на период фиксации изменяющей транзакции, когда обновленные объекты становятся текущими (Citforum 2020b)

3.5 Ссылочная целостность

Ссылочная целостность. Ссылочная целостность относится непосредственно к связям между таблицами. Если кратко, то ссылочная целостность должна отвечать на вопрос: что будет со строками одной таблицы, если в связанной таблице выполняется какая-либо операция модификации? Для того чтобы понять логику ссылочной целостности, будем считать таблицу главной в паре связанных таблиц, если она содержит первичный ключ, с помощью которого осуществляется связь. Вторую таблицу будем считать подчиненной таблицей. Теперь выделим три вида операции над связанными таблицами: удаление из главной таблицы, обновление строк главной таблицы, вставка в подчиненную таблицу.

1. Удаление строк из главной таблицы. Если удаляемая строка не связана со строками другой таблицы, то удалять можно без всяких последствий. Но если удаляемая строка связана со строками другой таблицы, то надо подумать, что же будет с этими строками. Просто оставить их без изменения нельзя, т. к. не понятно, как быть со значениями внешних ключей. В принципе, возможны следующие четыре сценария, поддерживаемые основными СУБД:

    - строки из подчиненной таблицы должны быть удалены вместе со связанными строками из подчиненной таблицы. Такой механизм называется каскадированием;
    - если удаляемая строка из главной таблицы связана со сроками из подчиненной таблицы, то такая операция удаления должна быть отвергнута. Данный механизм наиболее безопасен и предпочтителен при построении информационной системы;
    - если строка в подчиненной таблице связана с удаляемой строкой в главной таблице, то внешнему ключу следует присвоить значение NULL;
    - если строка в подчиненной таблице связана с удаляемой строкой в главной таблице, то внешнему ключу следует присвоить значение, принятое по умолчанию.

2. Обновление строк из главной таблицы. Если обновляемая строка не связана со строками другой таблицы или обновляются столбцы, не от- носящиеся к первичному ключу, то обновлять можно без всяких по- следствий. Но если обновляемая строка связана со строками другой таблицы и обновляется первичный ключ, то здесь, как и в предыдущем случае, возможны четыре сценария:

    - первичные ключи из главной таблицы обновляются вместе с внеш- ними ключами подчиненной таблицы. Как и в случае с подобной операцией удаления, этот механизм называется каскадированием;

- если обновляемая строка связана с какой-либо строкой подчиненной таблицы, то операция обновления должна быть отвергнута;
- если строка в подчиненной таблице связана с обновляемой строкой в главной таблице, то внешнему ключу следует присвоить значение NULL;
- если строка в подчиненной таблице связана с обновляемой строкой в главной таблице, то внешнему ключу следует присвоить значение, принятое по умолчанию.

3. Вставка строк в подчиненную таблицу. Здесь возможны следующие механизмы.
   - Строка в подчиненной таблице вставляется вместе со строкой в главной таблице. Здесь важно иметь в виду, что в главной таблице для всех столбцов должны быть определены значения по умолча- нию. Последовательность добавления такая: вначале добавляется строка в главную таблицу и определяется значение первичного клю- ча. Затем добавляется строка в подчиненную таблицу, в которой значению внешнего ключа присваивается значение первичного клю- ча в главной таблице.
   - Строка в подчиненную таблицу добавляется только при условии, что соответствующая ей строка в главной таблице уже существует.
   - При добавлении строки в подчиненную таблицу внешнему ключу присваивается значение NULL.
   - При добавлении строки в подчиненную таблицу внешнему ключу присваивается значение по умолчанию.

В некоторых простых СУБД отсутствует возможность устанавливать связи между таблицами и, таким образом, поддерживать ссылочную целостность. В этом случае поддержание ссылочной целостности полностью ложится на плечи программиста. Другими словами, связь между таблицами должна быть реализована на уровне прикладного программного обеспечения.

Декларативная и процедурная ссылочные целостности

Различают два способа реализации ограничений целостности:

- Декларативная поддержка ограничений целостности.
- Процедурная поддержка ограничений целостности.

> **Декларативная поддержка ограничений целостности** заключается в определении ограничений средствами языка определения данных (DDL - Data Definition Language). Обычно средства декларативной поддержки целостности (если они имеются в СУБД) определяют ограничения на значения доменов и атрибутов, целостность сущностей (потенциальные ключи отношений) и ссылочную целостность (целостность внешних ключей). Декларативные ограничения целостности можно использовать при создании и модификации таблиц средствами языка DDL или в виде отдельных утверждений (ASSERTION).

Например, следующий оператор создает таблицу PERSON и определяет для нее некоторые ограничения целостности:

```
CREATE TABLE PERSON
  (Pers_Id INTEGER PRIMARY KEY,
  Pers_Name CHAR(30) NOT NULL,
  Dept_Id REFERENCES DEPART(Dept_Id) ON UPDATE CASCADE ON DELETE CASCADE);
```

После выполнения оператора для таблицы PERSON будут объявлены следующие ограничения целостности:

- Поле Pers_Id образует потенциальный ключ отношения.
- Поле Pers_Name не может содержать null-значений.
- Поле Dept_Id является внешней ссылкой на родительскую таблицу DEPART, причем, при изменении или удалении строки в родительской таблице каскадно должны быть внесены соответствующие изменения в дочернюю таблицу.

> **Процедурная поддержка ограничений целостности** заключается в использовании триггеров и хранимых процедур.

Не все ограничения целостности можно реализовать декларативно. Примером такого ограничения может служить требование из примера 1, утверждающее, что поле Dept_Kol таблицы DEPART должно содержать количество сотрудников, реально числящихся в подразделении. Для реализации этого ограничения необходимо создать триггер, запускающийся при вставке, модификации и удалении записей в таблице PERSON, который корректно изменяет значение поля Dept_Kol. Например, при вставке в таблицу PERSON новой строки, триггер увеличивает на единицу значение поля Dept_Kol, а при удалении строки - уменьшает. Заметим, что при модификации записей в таблице PERSON могут потребоваться даже более сложные действия. Действительно, модификация записи в таблице PERSON может заключаться в том, что мы переводим сотрудника из одного отдела в другой, меняя значение в поле Dept_Id. При этом необходимо в старом подразделении уменьшить количество сотрудников, а в новом - увеличить (Citforum 2020a).

Внешний ключ

> **Внешний ключ** – это ограничение целостности, в соответствии с которым множество значений внешнего ключа является подмножеством значений первичного или уникального ключа родительской таблицы (Карпова 2009).

Ограничение целостности по внешнему ключу проверяется в двух случаях (Карпова 2009):

- при добавлении записи в подчинённую таблицу СУБД проверяет, что в родительской таблице есть запись с таким же значением первичного ключа;
- при удалении записи из родительской таблицы СУБД проверяет, что в подчинённой таблице нет записей с таким же значением внешнего ключа.

Способы поддержания ссылочной целостности

СУБД имеют механизм автоматического поддержания ссылочной целостности. Любая операция, изменяющая данные в таблице, вызывает автоматическую проверку ссылочной целостности. При этом (Wikipedia 2020b):

- При операции добавления записи автоматически проверяется, ссылаются ли внешние ключи в этой записи на существующие записи в заявленных при описании связанных таблицах. Если выясняется, что операция приведёт к появлению некорректных ссылок, она не выполняется — система возвращает ошибку.
- При операции редактирования записи проверяется:
  - если изменяется её первичный ключ и на данную запись имеются ссылки, то операция редактирования завершается с ошибкой;
  - если изменяется какой-то из внешних ключей, хранящихся в этой записи, и после изменения внешний ключ будет ссылаться на несуществующую запись, то операция редактирования завершается с ошибкой.
- При операции удаления записи проверяется, нет ли на неё ссылок. Если ссылки имеются, то возможно три варианта дальнейших действий (фактически выполняемый зависит от СУБД и от выбора программиста, который он должен сделать при описании связи):
  - Запрет — удаление блокируется и возвращается ошибка.
  - Каскадное удаление — в одной транзакции производится удаление данной записи и всех записей, ссылающихся на данную. Если на удаляемые записи также есть ссылки и настройки также требуют удаления, то каскадное удаление продолжается дальше. Таким образом, после удаления данной записи в базе не остаётся ни одной записи, прямо или косвенно ссылающейся на неё. Если хотя бы одну из ссылающихся записей удалить не получается (либо для неё настроен запрет, либо происходит какая-либо ещё ошибка), то все удаления запрещаются.
  - Присвоение NULL — во все внешние ключи записей, ссылающихся на данную, записывается маркер NULL. Если хотя бы для одной из ссылающихся записей это невозможно (например, если поле внешнего ключа описано как NOT NULL), то удаление запрещается.

3.6 Правила (триггеры)

Триггеры являются одной из разновидностей хранимых процедур. Их исполнение происходит при выполнении для таблицы какого-либо оператора языка манипулирования данными (DML). Триггеры используются для проверки целостности данных, а также для отката транзакций.

Триггер представляет собой специальный тип хранимых процедур, запускаемых сервером автоматически при попытке изменения данных в таблицах, с которыми триггеры связаны. Каждый триггер привязывается к конкретной таблице. Все производимые им модификации данных рассматриваются как одна транзакция. В случае обнаружения ошибки или нарушения целостности данных происходит откат этой транзакции. Тем самым внесение изменений запрещается. Отменяются также все изменения, уже сделанные триггером.

Триггер представляет собой весьма полезное и в то же время опасное средство. Так, при неправильной логике его работы можно легко уничтожить целую базу данных, поэтому триггеры необходимо очень тщательно отлаживать.

В отличие от обычной подпрограммы, триггер выполняется неявно в каждом случае возникновения триггерного события, к тому же он не имеет аргументов. Приведение его в действие иногда называют запуском триггера (Intuit 2020b)

Цели использования правил

С помощью триггеров достигаются следующие цели (Intuit 2020b):

- проверка корректности введенных данных и выполнение сложных ограничений целостности данных, которые трудно, если вообще возможно, поддерживать с помощью ограничений целостности, установленных для таблицы;
- выдача предупреждений, напоминающих о необходимости выполнения некоторых действий при обновлении таблицы, реализованном определенным образом;
- накопление аудиторской информации посредством фиксации сведений о внесенных изменениях и тех лицах, которые их выполнили;
- поддержка репликации.

Способы задания, моменты выполнения

Создает триггер только владелец базы данных. Это ограничение позволяет избежать случайного изменения структуры таблиц, способов связи с ними других объектов и т.п. Основной формат команды CREATE TRIGGER показан ниже:

```
<Определение_триггера>::=
  CREATE TRIGGER имя_триггера
  BEFORE | AFTER <триггерное_событие>
  ON <имя_таблицы>
  [REFERENCING
    <список_старых_или_новых_псевдонимов>]
  [FOR EACH { ROW | STATEMENT}]
  [WHEN(условие_триггера)]
  <тело_триггера>
```

Триггерные события состоят из вставки, удаления и обновления строк в таблице. В последнем случае для триггерного события можно указать конкретные имена столбцов таблицы.

Триггер – это процедура БД, которая привязана к конкретной таблице и вызывается автоматически при наступлении определённого события (добавления, удаления или модификации данных этой таблицы).

В отличие от обычной подпрограммы, триггер выполняется неявно в каждом случае возникновения триггерного события, к тому же он не имеет аргументов. Приведение его в действие иногда называют запуском триггера.

Время запуска триггера определяется с помощью ключевых слов BEFORE (триггер запускается до выполнения связанных с ним событий) или AFTER (после их выполнения).

Выполняемые триггером действия задаются для каждой строки (FOR EACH ROW), охваченной данным событием, или только один раз для каждого события (FOR EACH STATEMENT).

Обозначение <список_старых_или_новых_псевдонимов> относится к таким компонентам, как старая или новая строка (OLD / NEW) либо старая или новая таблица (OLD TABLE / NEW TABLE). Ясно, что старые значения не применимы для событий вставки, а новые – для событий удаления (Intuit 2020b).

3.7 События

Назначение механизма событий

Механизм событий в базе данных (database events) позволяет прикладным программам и серверу базы данных уведомлять другие программы о наступлении в базе данных определенного события и тем самым синхронизировать их работу (Jet Infosystems 1995).

Сигнализаторы событий. Типы уведомлений о происхождении события. Компоненты механизма событий

Операторы языка SQL, обеспечивающие уведомление, часто называют сигнализаторами событий в базе данных (database event alerters). Функции управления событиями целиком ложатся на сервер базы данных.

Механизм событий используется следующим образом. Вначале в базе данных для каждого события создается флажок, состояние которого будет оповещать прикладные программы о том, что некоторое событие имело место (оператор CREATE DBEVENT - СОЗДАТЬ СОБЫТИЕ). Далее во все прикладные программы, на ход выполнения которых может повлиять это событие, включается оператор REGISTER DBEVENT (ЗАРЕГИСТРИРОВАТЬ СОБЫТИЕ), который оповещает сервер базы данных, что данная программа заинтересована в получении сообщения о наступлении события. Теперь любая прикладная программа или процедура базы данных может вызвать событие оператором RAISE DBEVENT (ВЫЗВАТЬ СОБЫТИЕ). Как только событие произошло, каждая зарегистрированная программа может получить его, для чего должна запросить очередное сообщение из очереди событий (оператор GET DBEVENT - ПОЛУЧИТЬ СОБЫТИЕ) и запросить информацию о событии, в частности, его имя (оператор SQL INQUIRE_SQL) (Jet Infosystems 1995).

4 Механизмы обеспечения конфиденциальности в СУБД

4.1 Классификация угроз конфиденциальности СУБД

4.1.1 Причины, виды, основные методы нарушения конфиденциальности

Конфиденциальность – состояние информации, при котором доступ к ней осуществляют только субъекты, имеющие на него право.

Основными причинами нарушения конфиденциальности информации являются: - несоблюдение персоналом норм, требований, правил эксплуатации АС; - ошибки в проектировании АС и систем защиты АС; - ведение противостоящей стороной технической и агентурной разведок.

Источник: http://www.o-salo.narod.ru/page4.htm#_Toc287060195

Угрозы конфиденциальности информации направлены на несанкционированное перемещение информации от носителя-источника к носителю-получателю (угроза разглашения, утечки, несанкционированного доступа). Уязвимость информации к преднамеренным действиям злоумышленников или иных заинтересованных лиц является наиболее опасной. Как уже отмечалось выше, различные виды уязвимости обусловлены угрозами разглашения, НСД Уязвимость информации к разглашению – несанкционированному сообщению защищаемой информации лицам, не меющим

Уязвимость информации к НСД – противоправному преднамеренному овладению защищаемой информацией лицом, не имеющим права доступа к ней, – определяется наличием у информации, ее носителя или у самой системы защиты недостатков, которые создают возможность получения информации, ее модификации, уничтожения, блокирования доступа к ней.

Уязвимость информации к утечке – неконтролируемому распространению защищаемой информации за пределы круга лиц, которым эта информация была доверена, – возникает, как и в случае разглашения, из-за неправильной организации работы с информацией, а также в связи с "дырами"в системе защиты информации, неосторожными или умышленными действиями людей, допущенных к работе с защищаемыми сведениями. Утечка может происходить по акустическому, визуально-оптическому, материально-вещественному и электромагнитному каналам. Следовательно, уязвимость информации к утечке определяется слабыми сторонами носителя информации, среды, окружающей информацию, и технических средств, находящихся в непосредственной близости от носителей информации.

Источник: https://bit.ly/2XLDyia

Также, угрозы делят на умышленные и неумышленные(баги в ПО, сгнили провода и тп), умышленные в свою очередь делят на активные и пассивные:

Активные угрозы имеют целью нарушение нормального функционирования ИТ посредством целенаправленного воздействия на аппаратные, программные и информационные ресурсы.

Раскрытие конфиденциальной информации - это бесконтрольный выход конфиденциальной информации за пределы информационной технологии или круга лиц, которым она была доверена по службе или стала известна в процессе работы.

Умышленные угрозы подразделяются также на следующие виды:

Внутренние - возникают внутри управляемой организации. Они чаще всего сопровождаются социальной напряженностью и тяжелым моральным климатом на экономическом объекте, который провоцирует специалистов выполнять какие-либо правонарушения по отношению к информационным ресурсам

Внешние - Пассивные угрозы направлены на несанкционированное использование информационный ресурсов, не оказывая при этом влияния на функционирование ИТ

Далее приведены некоторые угрозы ИБ:

Раскрытие конфиденциальной информации - это бесконтрольный выход конфиденциальной информации за пределы информационной технологии или круга лиц, которым она была доверена по службе или стала известна в процессе работы.

Раскрытие конфиденциальной информации может быть следствием:

- разглашения конфиденциальной информации
- утечки информации по различным, главным образом техническим, каналам (по визуально-оптическим, акустическим, электромагнитным и др
- несанкционированного доступа к конфиденциальной информации различными способами

Несанкционированный доступ к информации выражается в противоправном преднамеренном овладении конфиденциальной информацией лицом, не имеющим права доступа к охраняемым сведениям.

Наиболее распространенными путями несанкционированного доступа к информации являются:

- перехват электронных излучений
- принудительное электромагнитное облучение (подсветка) линий связи с целью получения паразитной модуляции несущей
- применение подслушивающих устройств (закладок)
- дистанционное фотографирование
- перехват акустических излучений и восстановление текста принтера
- чтение остаточной информации в памяти системы после выполнения санкционированных запросов
- копирование носителей информации с преодолением мер защиты
- маскировка под зарегистрированного пользователя ("маскарад")
- использование недостатков языков программирования и операционных систем

Перечисленные пути несанкционированного доступа требуют достаточно больших технических знаний и соответствующих аппаратных или программных разработок со стороны взломщика. Например, используются технические каналы утечки - это физические пути от источника конфиденциальной информации к злоумышленнику, посредством которых возможно получение охраняемых сведений. Причиной возникновения каналов утечки являются конструктивные и технологические несовершенства схемных решений либо эксплуатационный износ элементов. Все это позволяет взломщикам создавать действующие на определенных физических принципах преобразователи, образующие присущий этим принципам канал передачи информации - канал несанкционированного доступа.

Несанкционированное использование информационных ресурсов, с одной стороны, является последствиями ее утечки и средством ее компрометации. С другой стороны, оно имеет самостоятельное значение, так как может нанести большой ущерб управляемой системе (вплоть до полного выхода информационной технологии из строя) или ее абонентам.

Незаконное использование привилегий. Любая защищенная технология содержит средства, используемые в чрезвычайных ситуациях, или средства, которые способны функционировать с нарушением существующей политики безопасности. Например, на случай внезапной проверки пользователь должен иметь возможность доступа ко всем наборам системы. Обычно эти средства

используются администраторами, операторами, системными программистами и другими пользователями, выполняющими специальные функции. Большинство систем защиты в таких случаях используют наборы привилегий, т. е. для выполнения определенной функции требуется определенная привилегия. Обычно пользователи имеют минимальный набор привилегий, администраторы - максимальный. Наборы привилегий охраняются системой защиты. Несанкционированный (незаконный) захват привилегий возможен при наличии ошибок в системе защиты, но чаще всего происходит в процессе управления системой защиты, в частности, при небрежном пользовании привилегиями.

"Взлом системы" - умышленное проникновение в информационную технологию, когда взломщик не имеет санкционированных параметров для входа. Способы взлома могут быть различными, и при некоторых из них происходит совпадение с ранее описанными угрозами. Например, использование пароля пользователя информационной технологии, который может быть вскрыт, например, путем перебора возможных паролей.

Источник: https://www.intuit.ru/studies/courses/3609/851/lecture/31660

4.1.2 Типы утечки конфиденциальной информации из СУБД, частичное разглашение

Утечки информации — неправомерная передача конфиденциальных сведений (материалов, важных для различных компаний или государства, персональных данных граждан), которая может быть умышленной или случайной. Утечка информации возможна по разным причинам.

Умышленные утечки:

Инсайдеры и избыточные права. К этому виду относятся случаи, основной причиной которых стали действия сотрудников, имеющих доступ к секретам легально, в силу своих служебных обязанностей. Все варианты инсайда условно можно разделить на две группы: в одном случае сотрудник, не имея доступа к информации, получает ее незаконно, а в другом он обладает официальным доступом к закрытым данным и умышленно выносит их за пределы компании.

Кража информации (извне). Проникновение в компьютер с помощью вредоносных программ и похищение информации с целью использования в корыстных интересах. На хакерские атаки приходится 15% от всего объема утечек информации. Вторжение в устройство извне и незаметная установка вредоносных программ позволяют хакерам полностью контролировать систему и получать доступ к закрытым сведениям, вплоть до паролей к банковским счетам и картам. Для этого могут применяться различные программы вроде «троянских коней». Главный атрибут данного вида утечки — активные действия внешних лиц с целью доступа к информации.

Взлом программного обеспечения. Приложения, используемые в рабочем процессе или на личном компьютере сотрудника, часто имеют незакрытые уязвимости, которые можно эксплуатировать и получать тем самым различные небезопасные возможности вроде исполнения произвольного кода или повышения привилегий.

Вредоносные программы (бекдоры, трояны) нацелены на причинение вреда владельцу устройства, позволяют незаметно проникать в систему и затем искажать или полностью удалять информацию, подменять ее другими, похожими данными.

Кражи носителей. Весьма распространенный вариант утечек, который случается в результате преднамеренного хищения устройств с информацией — ноутбуков, смартфонов, планшетов и других съемных носителей данных в виде флешек, жестких дисков.

Случайные утечки. К этому виду можно отнести инциденты, которые происходят из-за потери носителей данных (флешек, ноутбуков, смартфонов) или ошибочных действий сотрудников орга-

низации. Результатом может быть размещение конфиденциальной информации в интернете. Также не последнюю роль играет человеческий фактор, когда сотрудник по недосмотру открывает доступ к закрытым данным всем желающим.

Источник: https://www.anti-malware.ru/threats/leaks

4.1.3 Соотношение защищенности и доступности данных

??? В. Г. Проскурин. Защита в операционных системах 2014.pdf с 7 п4 мб об этом Суть такая, слишком жесткие политики безопасности могут повлечь нарушение доступности. Пример: запретить поль-лю SYSTEM в WINDOWS доступ к исполняемым файлам - тогда ОС нормально загрузиться не сможет.

4.1.4 Получение несанкционированного доступа к конфиденциальной информации путем логических выводов

Нередко путем логического вывода можно извлечь из базы данных информацию, на получение которой стандартными средствами у пользователя не хватает привилегий.

Если для реализации контроля доступа используются представления и эти представления допускают модификацию,

Основным средством борьбы с подобными угрозами, помимо тщательно проектирования модели данных, является механизм размножения строк. Суть его в том, что в состав первичного ключа, явно или неявно, включается метка безопасности, за счет чего появляется возможность хранить в таблице несколько экземпляров строк с одинаковыми значениями "содержательных"ключевых полей. Наиболее естественно размножение строк реализуется СУБД, поддерживающих метки безопасности (например, в INGRES/ Enhanced Security), однако и стандартными SQL-средствами можно получить удовлетворительное решение.

Рассмотрим базу данных, состоящую из одной таблицы с двумя столбцами - имя пациента и диагноз. Предполагается, что имя является первичным ключом. Каждая из строк таблицы относится к одному из двух уровней секретности - высокому (HIGH) и низкому (LOW). Соответственно, и пользователи подразделяются на два уровня благонадежности, которые мы также будем называть высоким и низким. К высокому уровню секретности относятся сведения о пациентах, находящихся под надзором правоохранительных органов или страдающих специфическими заболеваниями. На низком уровне располагаются данные о прочих пациентах, а также информация о некоторых "секретных"пациентах с "маскировочным"диагнозом. Части таблицы могут выглядеть примерно так:

Обратим внимание на то, что сведения о пациенте по фамилии Иванов присутствуют на обоих уровнях, но содержат разные диагнозы.

Мы хотим реализовать такую дисциплину доступа, чтобы пользователи с низким уровнем благонадежности могли манипулировать только данными на своем уровне и не имели возможности сделать какие-либо выводы о присутствии в секретной половине сведений о конкретных пациентах. Пользователи с высоким уровнем благонадежности должны иметь доступ к секретной половине таблицы, а также к информации о прочих пациентах. Дезинформирующих строк о секретных пациентах они видеть не должны:

Имя	Диагноз
Иванов	СПИД
Петров	Сифилис
Сидоров	Стреляная рана

Таблица 4.

Данные с высоким уровнем секретности.

Имя	Диагноз
Иванов	Пневмония
Ивлев	Рак легких
Ярцев	Ожог второй степени
Суворов	Микроинфаркт

Таблица 5.

Данные с низким уровнем секретности.

4.1.5 Методы противодействия. Особенности применения криптографических методов

В целях обеспечения конфиденциальности информации используются следующие криптографические примитивы:

- Симметричные криптосистемы
- Асимметричные криптосистемы

В симметричных криптосистемах для зашифрования и расшифрования информации используется один и тот же общий секретный ключ, которым взаимодействующие стороны предварительно обмениваются по некоторому защищённому каналу. Также к симметричным системам относят те, в которых получение ключа расшифрования из ключа шифрования является тривиальной операцией. В зависимости от объёма данных, обрабатываемых за одну операцию шифрования, симметричные шифры делятся на блочные, в которых за одну операцию шифрования происходит преобразование одного блока данных (32 бита, 64, 128 или больше), и потоковые, в которых работают с каждым

Имя	Диагноз
Иванов	СПИД
Ивлев	Рак легких
Петров	Сифилис
Сидоров	Стреляная рана
Ярцев	Ожог второй степени
Суворов	Микроинфаркт

Таблица 6.
Данные, доступные пользователю с высоким уровнем секретности (обратим внимание, что строка "Иванов Пневмания" здесь отсутствует.).

символом открытого текста по отдельности (например, с 1 битом или 1 байтом). Использование блочного шифра подразумевает разделение открытого текста на блоки одинаковой длины, к каждому из которых применяется функция шифрования. Кроме того, результат шифрования следующего блока может зависеть от предыдущего. Данная возможность регулируется режимом работы блочного шифра. В качестве примеров симметричных криптосистем можно привести международные стандарты DES и пришедший ему на смену AES, оба являются Блочными. Пример потокового - RC4, который также может выступать в качетве генератора псевдослучайной последовательности случайных чисел.

Асимметричные криптосистемы характерны тем, что в них используются различные ключи для зашифрования и расшифрования информации, при этом получение ключа расшифрования из ключа шифрования вычислительно сложно. Ключ для зашифрования (открытый ключ) можно сделать общедоступным, с тем чтобы любой желающий мог зашифровать сообщение для некоторого получателя. Получатель же, являясь единственным обладателем ключа для расшифрования (секретный ключ), будет единственным, кто сможет расшифровать зашифрованные для него сообщения. Все используемые на сегодняшний день асимметричные криптосистемы работают с открытым текстом, составляющим несколько сотен или тысяч бит, поэтому классификация таких систем по объёму обрабатываемых за одну операцию данных не производится. Примеры асимметричных криптосистем – RSA и схема Эль-Гамаля.

Симметричные и асимметричные криптосистемы, а также различные их комбинации используются в АС прежде всего для шифрования данных на различных носителях и для шифрования трафика.

Источник: Криптографические методы защиты информации. Учебное пособие. Владимиров С. М., Габидулин Э. М., Колыбельников А. И., Кшевецкий А. С.

4.1.6 Применение шифрования в базах данных

Шифрование базы данных — использование технологии шифрования для преобразования информации, хранящейся в базе данных (БД), в шифротекст, что делает её прочтение невозможным для лиц, не обладающих ключами шифрования.

Основные подходы можно классифицировать по тому, на каком уровне происходит шифрование:

- Шифрование на уровне хранилища.
- Шифрование на уровне базы данных.

- Шифрование на уровне приложения.

Шифрование на уровне хранилища:

Также называемое «прозрачным» (Transparent Database Encryption, TDE). Данная технология, применяется, например, в продуктах Microsoft и Oracle для шифрования и дешифрования ввода-вывода файлов БД. Данные шифруются перед записью на диск и дешифруются во время чтения в память, что решает проблему защиты «неактивных» данных, но не обеспечивает сохранность информации при передаче по каналам связи или во время использования. Преимуществом TDE является то, что шифрование и дешифрование выполняются прозрачно для приложений, то есть их модификация не требуется.

Реализация шифрование на уровне хранилища Microsoft:

TDE применяется для файлов БД и журнала транзакций на уровне страниц. Страницы шифруются с помощью специального симметричного ключа шифрования базы данных (англ. Database Encryption Key), защищённого сертификатом, который хранится в БД master и шифруется её главным ключом, или асимметричным ключом, защищённым модулем расширенного управления ключами (англ. Extensible Key Manager, EKM). Применение TDE не увеличивает размер зашифрованной БД, а влияние на производительность незначительно.

Реализация шифрование на уровне хранилища Oracle:

TDE применяется для файлов БД на уровне столбцов. Для таблицы, содержащей выбранные к шифрованию столбцы, создаётся симметричный ключ шифрования, защищённый главным ключом, который хранится в безопасном месте за пределами БД, называемом бумажником (англ. Wallet). Зашифрованные ключи таблиц содержатся в словаре данных (англ. Data Dictionary).

Шифрование на уровне базы данных:

Одним из примеров шифрования на уровне базы данных является шифрование на уровне столбцов (Column-Level Encryption), которое записывает в базу данных уже зашифрованные данные, а саму базу данных — без дальнейшего шифрования — в хранилище. Особенностью шифрования на уровне столбцов является использование единого ключа при обработке данных одного столбца. Ключи могут быть назначены пользователям и защищены паролем для предотвращения автоматической расшифровки, однако это усложняет администрирование БД. При использовании шифрования на уровне столбцов необходимо внесение изменений в клиентские приложения. Помимо этого уменьшается производительность БД.

Шифрование на уровне приложений:

В шифровании на уровне приложений процесс шифрования осуществляется приложением, которое создаёт или изменяет данные, то есть он происходит перед записью в базу данных. Этот подход является более гибким, так как приложению известны роли или права доступа пользователей, а также информация о том, какие данные являются конфиденциальными.

Приемущества шифрования на уровне приложений:

Одним из главных преимуществ шифрования, встроенного в приложение, является то, что нет необходимости использовать дополнительное решение для защиты данных при передаче по каналам связи, так как они отправляются уже зашифрованными. Ещё один плюс такого метода — это то, что кража конфиденциальной информации становится сложнее, так как злоумышленник должен иметь доступ к приложению для того, чтобы расшифровать данные, хранящиеся в БД.

Недостатки шифрования на уровне приложений:

Для реализации шифрования на уровне приложений необходимо внесение изменений не только в само приложение, но и в базу данных. Также могут возникнуть проблемы с производительностью БД, у которой, например, пропадает возможность индексирования и поиска. Ещё одним минусом является управление ключами в системе с таким шифрованием. Так как несколько приложений могут использовать БД, то ключи хранятся во многих местах, поэтому неправильное управление ими может привести к краже информации или невозможности её использования. В добавление к этому, если возникает необходимость изменения ключа, то для начала потребуется расшифровать все данные со старым ключом, и потом снова зашифровать, используя новый ключ.

4.2 Средства идентификации и аутентификации

4.2.1 Общие сведения

Начнем с понятий идентификации и аутентификации:

Идентификация объекта – это установление эквивалентности между объектом и его априорным обозначением (определением, представлением, образом, комплексом характеристик). Другими словами, идентификация это выделение обьекта и установление для него отличительного набора параметров или характеристик. Например для идентификации пользо- вателя это наличие идентифицирующей его информации, на основе которой можно днозначно

Аутентификация – это процедура проверки подлинности входящего в систему обьекта(пользователя), редьявившего

Аутентификация и идентификация являются взаимосвязанными процессами для определения и проверки подлинности пользователей системы. Именно от них зависит дальнейшее решение системы о том, какие действия следует предпринимать для предоставления ресурсов пользователю. Следует уточнить, что решение о предоставлении доступа зависит от используемой информационной системы.

Также стоит ввести понятие строгой аутентификации. Идея строгой аутентификации заключается в том, что проверяемая сторона доказывает свою подлинность проверяющей стороне на основе какого- либо секрета, который предварительно был размещен на обеих сторонах.Совместное применение средств идентификации и аутентификации, встроенных в СУБД и в ОС.

Основополагающим является тот факт, что проверяемая сторона не передает свой секрет, аутентификация обеспечивается ответами доказывающей стороны на сгенерированные вопросы проверяющей стороны. В соответствии с рекомендациями стандарта X.509 различают процедуры строгой аутентификации следующих типов: – односторонняя аутентификация – двусторонняя аутентификация – трехсторонняя аутентификация Односторонняя аутентификация предусматривает обмен информацией только в одном направлении. Двусторонняя аутентификация, по сравнению с односторонней, содержит дополнительный ответ проверяющей тороны Трехсторонняя аутентификация содержит дополнительную передачу данных от доказывающей стороны проверяющей.

Ещё до появления компьютеров использовались различные отличительные черты субъекта, его характеристики. Сейчас использование той или иной характеристики в системе зависит от требуемой надёжности, защищённости и стоимости внедрения. Выделяют три фактора аутентификации:

Фактор знания: что-то, что мы знаем — пароль. Это тайные сведения, которыми должен обладать только авторизованный субъект. Паролем может быть речевое слово, текстовое слово, комбинация для замка или личный идентификационный номер (PIN). Парольный механизм может быть

довольно легко реализован и имеет низкую стоимость. Однако он имеет существенные недостатки: сохранить пароль в тайне зачастую бывает сложно, злоумышленники постоянно придумывают новые способы кражи, взлома и подбора пароля (см. бандитский криптоанализ, метод грубой силы). Это делает парольный механизм слабозащищённым. Многие секретные вопросы, такие как «Где вы родились?», элементарные примеры фактора знаний, потому что они могут быть известны широкой группой людей или быть исследованы. Из-за малой энтропии пользовательских паролей во всех системах регистрации и аутентификации пользователей применяется специальная политика безопасности. Типичные минимальные требования: Длина пароля от 8 символов. Использование разных регитров и небуквенных символов в паролях. Запрет паролей изсловаря или часто используемых паролей. Ограниченное время действия пароля. Обязательная сменапароля по истечении срока действия. Блокирование возможности аутентификации после нескольких неудачных попыток.

Фактор владения: что-то, что мы имеем — устройство аутентификации. Здесь важно обстоятельство обладания субъектом каким-то неповторимым предметом. Это может быть личная печать, ключ от замка, для компьютера это файл данных, содержащих характеристику. Характеристика часто встраивается в особое устройство аутентификации, например пластиковую карту, смарт-карту. Для злоумышленника заполучить такое устройство более сложно, чем взломать пароль, а субъект может сразу же сообщить в случае кражи устройства. Это делает данный метод более защищённым, чем парольный механизм, однако стоимость такой системы более высокая.

Фактор свойства: что-то, что является частью нас — биометрия. Характеристикой является физическая особенность субъекта. Это может быть портрет, отпечаток пальца или ладони, голос или особенность глаза. С точки зрения субъекта, данный способ является наиболее простым: не надо ни запоминать пароль, ни переносить с собой устройство аутентификации. Однако биометрическая система должна обладать высокой чувствительностью, чтобы подтверждать авторизованного пользователя, но отвергать злоумышленника со схожими биометрическими параметрами. Также стоимость такой системы довольно велика. Но, несмотря на свои недостатки, биометрика остается довольно перспективным фактором.

Источник: https://bit.ly/3doPVr4

Источник: Криптографические методы защиты информации. Учебное пособие. Владимиров С. М., Габидулин Э. М., Колыбельников А. И., Кшевецкий А. С.

4.2.2 Совместное применение средств идентификации и аутентификации, встроенных в СУБД и в ОС

СУБД Oracle предоставляет следующие способы аутентификации пользователей:

1. Аутентификация средствами операционной системы. Некоторые ОС позволяют СУБД Oracle использовать информацию о пользователях, которыми управляет собственно ОС. В этом случае пользователь компьютера имеет доступ к ресурсам БД без дополнительного указания имени и пароля – используются его сетевые учетные данные. Данный вид аутентификации считается небезопасным и используется, в основном, для аутентификации администратора СУБД.

2. Аутентификация при помощи сетевых сервисов. Данный вид аутентификации обеспечивает опция сервера Oracle Advanced Security. Она предоставляет возможность SSL-аутентификации,

а так же аутентификацию с помощью служб третьих сторон, в роли которых могут выступать Kerberos, PKI, RADIUS или служба LDAP-каталога.

3. Аутентификация в многоуровневых приложениях. Приведенные выше методы аутентификации также могут быть применены и в многоуровневых приложениях. Как правило, для доступа к приложениям из сети Интернет используется аутентификация по имени и паролю (в том числе с использованием протокола RADIUS), либо по протоколу SSL. Прочие методы используются для работы пользователей в локальной сети.

Источник : http://www.iso27000.ru/chitalnyi -zai/ zaschita-personalnyh-dannyh/ obespechenie-zaschity-personaln%h-dannyh-v-subd-oracle

Аутентификация в Windows и хранение паролей:

ОС Windows, начиная с Vista, Server 2008, Windows 7, сохраняет пароли в виде NT-хэша, который вычисляется как 128-битовый хэш MD4 от пароля в Unicode кодировке. NT-хэш не использует «соль», поэтому применима словарная атака. На словарной атаке основаны программы поиска (взлома) паролей для Windows. Файл паролей называется SAM (англ. Security Account Manager) в случае локальной аутентификации. Если пароли хранятся на сетевом сервере, то они хранятся в специальном файле, доступ к которому ограничен. В последнем протоколе аутентификации NTLMv2 пользователь для входа в свой компьютер аутентифицируется либо локально на компьютере, либо удалённым сервером, если учётная запись пользователя хранится на сервере. Пользователь с именем user вводит пароль в программу-клиент, которая, взаимодействуя с программой-сервером (локальной или удалённой на сервере домена domain), аутентифицирует пользователя для входа в систему. Протокол аутентификации NTLMv2 обеспечивает одностороннюю аутентификацию пользователя серверу (или своему ПК). При удалённой аутентификации по сети последние версии Windows используют протокол Kerberos, который обеспечивает взаимную аутентификацию, и, только если аутентификация по Kerberos не поддерживается клиентом или сервером, она происходит по NTLMv2.

Источник: Криптографические методы защиты информации. Учебное пособие. Владимиров С. М., Габидулин Э. М., Колыбельников А. И., Кшевецкий А. С.

4.3 Средства управления доступом

4.3.1 Основные понятия: субъекты и объекты, группы пользователей, привилегии, роли и представления.

Сначала приведем формальные определения субьекта и обьекта доступа:

Субъект доступа – активная сущность АС (автоматизированной системы), которая может изменять состояние системы через порождение процессов над объектами, в том числе порождать новые объекты и инициализировать порождение новых субъектов.

Объект доступа – пассивная сущность АС, процессы над которой могут в определенных случаях быть источником порождения новых субъектов. Теперь "по пацански чисто для понимания (хотя и неверно):

- Объект – ресурс АС к которой обращается субьект (информация)
- Субъект – пользователь

Группа - это именованная совокупность пользователей

Роль – набор прав (полномочий, привилегий) или ролей

Привилегии – права доступа, предоставляемые субъекту (тут нужно прям уточнить в лоб - привилегия дается СУБЬЕКТУ)

Представления - это специфический образ таблицы или набора таблиц, определенный оператором SELECT.В отличие от обычных таблиц реляционных баз данных, представление не является самостоятельной частью набора данных, хранящегося в базе. Содержимое представления динамически вычисляется на основании данных, находящихся в реальных таблицах. Изменение данных в реальной таблице базы данных немедленно отражается в содержимом всех представлений, построенных на основании этой таблицы. Некоторые СУБД имеют расширенные представления для данных, доступных только для чтения. Так, СУБД Oracle реализует концепцию «материализованных представлений» — представлений, содержащих предварительно выбранные невиртуальные наборы данных, совместно используемых в распределённых БД. Эти данные извлекаются из различных удалённых источников (с разных серверов распределённой СУБД). Целостность данных в материализованных представлениях поддерживается за счёт периодических синхронизаций или с использованием триггеров. Аналогичный механизм предусмотрен в Microsoft SQL Server версии 2000.

По самой сути представления могут быть доступны только для чтения. Тем не менее, в некоторых СУБД (например, в Oracle) представления могут быть редактируемыми, как и обычные физические таблицы.

4.3.2 Виды привилегий: привилегии безопасности и доступа. Использование ролей и привилегий пользователей.

Привилегии в СУБД можно подразделить на две категории: привилегии безопасности и привилегии доступа. Привилегии безопасности позволяют выполнять административные действия. Привилегии доступа, в соответствии с названием, определяют права доступа субъектов к определенным объектам.

Привилегии безопасности всегда выделяются конкретному пользователю (а не группе, роли или всем) во время его создания (оператором CREATE USER) или изменения характеристик (оператором ALTER USER). Примеры таких привилегий в СУБД INGRES:

- security - право управлять безопасностью СУБД и отслеживать действия пользователей. Пользователь с этой привилегией может подключаться к любой базе данных, создавать, удалять и изменять характеристики пользователей, групп и ролей, передавать права на доступ к базам данным другим пользователям, управлять записью регистрационной информации, отслеживать запросы других пользователей.

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

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

- maintain_locations - право на управление расположением баз администратора сервера баз данных и операционной системы.

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

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

Для изменения состава группы служит оператор ALTER GROUP. Оператор DROP GROUP позволяет удалять группы, правда, только после того, как опустошен список членов группы. Оператор ALTER ROLE служит для изменения паролей ролей, а DROP ROLE - для удаления ролей.

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

4.3.3 Соотношение прав доступа СУБД и операционной системы

Данные, хранящиеся средствами СУБД, располагаются в файлах и/или логических томах операционной системы. Соответственно, и доступ к этим данным возможен как с помощью СУБД, так и посредством утилит ОС. Так называемая естественная защита баз данных, являющаяся следствием относительно сложного формата их хранения, едва ли способна остановить злоумышленника.

Чтобы средствами ОС нельзя было скомпрометировать базу данных и сопутствующую информацию, например журналы транзакций, необходимо установить максимально жесткий режим доступа к соответствующим файлам и томам. Для UNIX-систем целесообразно предоставить право непосредственного доступа только пользователям root, ingres и, быть может, администраторам баз данных. Все прочие субъекты должны осуществлять доступ к базам с помощью программ со взведенным битом переустановки действующего идентификатора пользователя. Данные из базы могут экспортироваться в файлы операционной системы или другие хранилища. Возможен и обратный процесс импорта данных. Необходимо следить за тем, чтобы подобные операции не понижали уровня защищенности информации. Сделать это, вообще говоря, непросто. Во-первых, операционная система (например, персонального компьютера) может не обеспечивать должной защиты. Во-вторых, даже для развитых ОС необходимо установить и поддерживать соответствие между механизмами защиты СУБД и операционных систем. При большом числе пользователей (порядка нескольких сотен) данная задача становится очень сложной. Видимо, наиболее практичное решение сводится к административному контролю за экспортом/импортом информации. Можно предположить, что исходные тексты сколько-нибудь сложных процедур баз данных будут храниться в файлах операционной системы, формально не имеющих отношения к СУБД. Потенциально возможно нелегальное изменение исходного текста, которое, как было показано ранее, способно привести к серьезным нарушениям информационной безопасности. Вероятно, и здесь лучшим средством является административный контроль за размещением процедур в базах данных и за передачей прав на их выполнение. СУБД и ОС предлагают во многом сходные средства защиты данных. Более того, многие СУБД просто полагаются на операционную систему в плане идентификации и проверки подлинности пользователей. Тем не менее даже для таких "дружественных"сервисов проведение в жизнь

согласованной политики безопасности является очень сложным делом. На практике приходится всячески ограничивать информационный обмен между СУБД и ОС.

4.3.4 Метки безопасности.

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

В СУБД INGRES/Enhanced Security к каждой реляционной таблице неявно добавляется столбец, содержащий метки безопасности строк таблицы. Метка безопасности состоит из трех компонентов: · уровень секретности. Смысл этого компонента зависит от приложения. В частности, возможен традиционный спектр уровней от "совершенно секретно"до "несекретно"; · категории. Понятие категории позволяет разделить данные на "отсеки"и тем самым повысить надежность системы безопасности. В коммерческих приложениях категориями могут служить "финансы "кадры "материальные ценности"и т.п. Ниже назначение категорий разъясняется более подробно; · области. Является дополнительным средством деления информации на отсеки. На практике компонент "область"может действительно иметь географический смысл, обозначая, например, страну, к которой относятся данные. Каждый пользователь СУБД INGRES/ Enhanced Security характеризуется степенью благонадежности, которая также определяется меткой безопасности, присвоенной данному пользователю. Пользователь может получить доступ к данным, если степень его благонадежности удовлетворяет требованиям соответствующей метки безопасности. Более точно: · уровень секретности пользователя должен быть не ниже уровня секретности данных; · набор категорий, заданных в метке безопасности данных, должен целиком содержаться в метке безопасности пользователя; · набор областей, заданных в метке безопасности пользователя, должен целиком содержаться в метке безопасности данных.

Рассмотрим ГИПОТЕТИЧЕСКИЙ пример И НИКОГО НЕ ИМЕЕМ ВВИДУ, ВСЕ ПЕРСОНАЖИ И СОБЫТИЯ ЯВЛЯЮТСЯ ФАНТАЗИЕЙ БОЛЬНОЙ ГОЛОВЫ АВТОРА. Пусть данные о том, ехал человек в поезде или нет имеют уровень секретности "для служебного пользования принадлежат категории "поезда"и "пассажиры"и относятся к областям "Москва"и "Татарстан". Далее, пусть степень благонадежности пользователя характеризуется меткой безопасности с уровнем секретности "для служебного пользования категориями "поезда"и "самолеты а также областью "Москва"и "Татарстан". Такой пользователь не получит доступ к данным, в метке пользователя была указана только категория "поезда"(без категории "пассажиры"), в доступе к данным ему было бы отказано.

Когда пользователь производит выборку данных из таблицы, он получает только те строки, меткам безопасности которых удовлетворяет степень его благонадежности. Для совместимости с обычными версиями СУБД, столбец с метками безопасности не включается в результирующую информацию.

Отметим, что механизм меток безопасности не отменяет, а дополняет произвольное управле-

ние доступом. Пользователи по-прежнему могут оперировать таблицами только в рамках своих привилегий, но даже при наличии привилегии SELECT им доступна, вообще говоря, только часть данных.

При добавлении или изменении строк они, как правило, наследуют метки безопасности пользователя, инициировавшего операцию. Таким образом, даже если авторизованный пользователь перепишет секретную информацию в общедоступную таблицу, менее благонадежные пользователи не смогут ее прочитать.

Специальная привилегия, DOWNGRADE, позволяет изменять метки безопасности, ассоциированные с данными. Подобная возможность необходима, например, для коррекции меток, по тем или иным причинам оказавшихся неправильными.

Представляется естественным, что СУБД INGRES/Enhanced Security допускает не только скрытое, но и явное включение меток безопасности в реляционные таблицы. Появился новый тип данных, security label, поддерживающий соответствующие операции сравнения.

4.3.5 Использование представлений для обеспечения конфиденциальности информации в СУБД.

СУБД предоставляют специфическое средство управления доступом - представления. Представления позволяют сделать видимыми для субъектов определенные столбцы базовых таблиц (реализовать проекцию) или отобрать определенные строки (реализовать селекцию). Не предоставляя субъектам прав доступа к базовым таблицам и сконструировав подходящие представления, администратор базы данных защитит таблицы от несанкционированного доступа и снабдит каждого пользователя своим видением базы данных, когда недоступные объекты как бы не существуют.

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

```
CREATE VIEW empview AS
SELECT name, dept
FROM employee
WHERE dept = "shoe";
```

Предоставим всем право на выборку из этого представления:

```
GRANT SELECT
ON empview
TO PUBLIC;
```

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

```
SELECT *
FROM empview
WHERE dept = "toy";
```

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

4.4 Обеспечение конфиденциальности путем тиражирования БД

Тиражирование (репликация) – технология, предусматривающая поддержку копий всей БД или ее фрагментов в нескольких узлах сети. Копия БД называется репликой. Копии БД обычно приближены к местам использования информации. В контексте информационной безопасности тиражирование можно рассматривать как средство повышения доступности данных. Стала легендой история про бакалейщика из Сан-Франциско, который после разрушительного землетрясения восстановил свою базу данных за 16 минут, перекачав из другого города предварительно протиражированную информацию.

Развитые возможности тиражирования предоставляет СУБД INGRES. Им посвящена статья [2]. Здесь мы рассмотрим возможности другого популярного сервера СУБД - Informix OnLine Dynamic Server (OnLine-DS) 7.1. В отличие от предыдущего раздела, речь пойдет об обычных (а не кластерных) конфигурациях. В Informix OnLine-DS 7.1 поддерживается модель тиражирования, состоящая в полном отображении данных с основного сервера на вторичные.

В конфигурации серверов Informix OnLine-DS с тиражированием выделяется один основной и ряд вторичных серверов. На основном сервере выполняется и чтение, и обновление данных, а все изменения передаются на вторичные серверы, доступные только на чтение. В случае отказа основного сервера вторичный автоматически или вручную переводится в режим доступа на чтение и запись. Прозрачное перенаправление клиентов при отказе основного сервера не поддерживается, но оно может быть реализовано в рамках приложений.

После восстановления основного сервера возможен сценарий, при котором этот сервер становится вторичным, а бывшему вторичному, который уже функционирует в режиме чтения-записи, придается статус основного; клиенты, которые подключены к нему, продолжают работу. Таким образом, обеспечивается непрерывная доступность данных. Тиражирование осуществляется путем передачи информации из журнала транзакций (логического журнала) в буфер тиражирования основного сервера, откуда она пересылается в буфер тиражирования вторичного сервера. Такая пересылка может происходить либо в синхронном, либо в асинхронном режиме. Синхронный режим гарантирует полную согласованность баз данных - ни одна транзакция, зафиксированная на основном сервере, не останется незафиксированной на вторичном, даже в случае сбоя основного сервера. Асинхронный режим не обеспечивает абсолютной согласованности, но улучшает рабочие характеристики системы.

Побочный положительный эффект тиражирования - возможность вынести преимущественно на вторичный сервер ресурсоемкие приложения поддержки принятия решений. В этом случае они могут выполняться с максимальным использованием средств параллельной обработки, не подавляя приложений оперативной обработки транзакций, сосредоточенных на основном сервере. Это также можно рассматривать как фактор повышения доступности данных.

Формальная модель для обеспечения конфиденциальности БД с помощью тиражирования. Вот не нашел источников под это. Архитектура и политика безопасности в модели SINTRA.

https://bit.ly/2U0iaoi

глава 4 на англуйцком было слажна и предлагается читателю перевести это самому

4.5 Аудит и подотчетность

Регистрация действий пользователей, так или иначе влияющих на информационную безопасность, является фактором, сдерживающим потенциальных злоумышленников и позволяющим расследовать уже случившиеся нарушения.

Более точно, журнал регистрационной информации может использоваться для следующих целей:

- обнаружение необычных или подозрительных действий пользователей и идентификации лиц, совершающих эти действия;
- обнаружение попыток несанкционированного доступа
- оценка возможных последствий состоявшегося нарушения информационной безопасности
- оказание помощи в расследовании случаев нарушения безопасности
- организация пассивной защиты от нелегальных действий. Пользователи, зная, что их действия фиксируются, могут не решиться на незаконные операции. Чтобы данная цель достигалась, необходимо довести до сведения каждого пользователя, что каждое его действие регистрируется и что за незаконные операции он может понести наказание.

4.5.1 Подотчетность действий пользователя и аудит связанных с безопасностью событий.

Протоколирование, или регистрация, представляет собой механизм подотчётности системы обеспечения информационной безопасности, фиксирующий все события, относящиеся к вопросам безопасности. В свою очередь, аудит – это анализ протоколируемой информации с целью оперативного выявления и предотвращения нарушений режима информационной безопасности.

Не менее важен вопрос о порядке хранения системных журналов. Поскольку файлы журналов хранятся на том или ином носителе, неизбежно возникает проблема переполнения максимально допустимого объёма системного журнала. При этом реакция системы может быть различной, например:

- система может быть заблокирована вплоть до решения проблемы с доступным дисковым пространством
- могут быть автоматически удалены самые старые записи системных журналов
- система может продолжить функционирование, временно приостановив протоколирование информации

Безусловно, последний вариант в большинстве случаев является неприемлемым, и порядок хранения системных журналов должен быть чётко регламентирован в политике безопасности организации.

В терминологии регистрационной службы любое действие, способное изменить состояние базы данных, называется событием и может регистрироваться. В СУБД Informix события обозначаются четырехбуквенными мнемониками. Приведем несколько событий безопасности:

- ACTB - доступ к таблице
- CLDB - закрытие таблицы
- DRTB - удаление таблицы

4.5.2 Регистрация действий пользователя

Регистрация всех событий для всех пользователей существенно снизит эффективность работы СУБД. Администратор СУБД (или лицо, отвечающее за информационную безопасность) должен выбрать приемлемый баланс между безопасностью и эффективностью. Рекомендуется регистрировать по крайней мере следующие события:

- передача привилегий доступа к базе данных (GRDB)
- лишение привилегий доступа к базе данных (RVDB)
- передача привилегий доступа к таблице (GRTB)
- лишение привилегий доступа к таблице (RVTB)
- открытие базы данных (OPDB)

Перечисленные события происходят нечасто, однако их фиксация позволяет составить представление о том, чем интересуется каждый из пользователей. Если какой-либо пользователь замечен в подозрительных действиях, для него можно включить режим регистрации всех событий. Для администратора СУБД и лица, отвечающего за безопасность, подобный режим должен быть включен постоянно. Отметим, что у пользователя нет возможности узнать, какие из его действий регистрируются. Целесообразно поддерживать в нем уверенность, что регистрируется все или почти все.

4.5.3 Управление набором регистрируемых событий

Для управления набором регистрируемых событий в СУБД Informix используются маски событий. Три стандартные маски с именами _default, _require и _exclude формируют стандартное регистрационное окружение. Кроме того, для отдельных пользователей могут быть заданы персональные маски с именами, совпадающими с входными именами этих пользователей.

Результирующее регистрационное окружение пользователя формируется следующим образом: Берется маска _default или маска пользователя, если таковая имеется.

В число регистрируемых дополнительно включаются события, заданные маской _require. Из числа регистрируемых исключаются события, заданные маской _exclude, при условии, что они не были упомянуты в маске _require.

Таким образом, в СУБД Informix выполнено одно из требований к системам класса безопасности C2, предписывающее возможность задания своего перечня регистрируемых событий для каждого пользователя.

Маски можно формировать с помощью утилиты onaudit. Приведем пример командной строки, позволяющей добавить к маске _default регистрацию событий, вызываемых операциями со строками таблиц: onaudit -m -u _default -e +DLRW,INRW,RDRW,UPRW

С помощью утилиты onaudit можно выдать состояние той или иной маски, создать, модифицировать или удалить ее. Та же утилита (и это важно отметить) позволяет включать и выключать регистрацию событий сервером СУБД Informix.

4.5.4 Анализ регистрационной информации

- ONLN - фиксированное поле, обозначающее события, фиксируемые сервером Informix-OnLine
- дата и время события
- имя клиентского компьютера, инициировавшего событие
- идентификатор клиентского процесса, инициировавшего событие
- имя серверного компьютера
- имя пользователя
- код завершения действия, вызвавшего событие
- мнемоника события
- дополнительные поля, идентифицирующие базы данных, таблицы и другие объекты, вовлеченные в событие

С помощью утилиты onshowaudit можно отобрать часть регистрационной информации и организовать ее просмотр или, воспользовавшись утилитой dbload, загрузить ее в базу данных и анализировать затем SQL-средствами. Регистрационная служба СУБД Informix является субъектно-ориентированной в том смысле, что можно задать свой набор отслеживаемых событий для каждого пользователя (субъекта), однако нет возможности указать имена объектов (таблиц, процедур и т.п.), операции с которыми отслеживались бы особым образом. Вместо этого предлагается полагаться на средства анализа регистрационной информации. Очевидно, после загрузки регистрационного журнала в базу данных можно отобрать сведения по сколь угодно сложному критерию и сгенерировать отчет, по существу, произвольного вида.

Регистрационные файлы и результат их загрузки в базу данных нуждаются в защите. В частности, для файлов рекомендуется установить в качестве владельца пользователя root, владеющей группой сделать informix, а режим доступа положить равным 0660 (доступ на чтение и запись только для владельца и группы).

Регистрационная информация нуждается в ежедневном анализе. В противном случае реакция на подозрительные действия или нарушения окажется запоздалой. В первом приближении подозрительными можно считать действия, завершившиеся ненормальным образом, то есть с ненулевым кодом. Более сложные эвристики могут опираться на статистический анализ спектра пользовательских действий. Как уже указывалось, после обнаружения подозрительной активности целесообразно включить режим регистрации всех действий пользователя и/или принять административные меры.

Источник на котором основана большая часть данной главы:

https://www.osp.ru/news /articles/1996/0131/13031467

5 Механизмы, поддерживающие высокую готовность

Определение: Высокая доступность (High Availability, HA) - это характеристика технической системы, разработанная для избежания невыполненного обслуживания путём уменьшения или управления сбоями и минимизации времени плановых простоев ((*High availability* 2022)). Доступность часто измеряется в «Nines» (девятки).

Высокая готовность для каждой конкретной системы зависит от цели, для которой предназначена эта система. Для некоторых компаний высокая готовность значит максимальное downtime за год в несколько минут. Для других HA может значит downtime несколько часов в месяц.

Availability %	Downtime per year[note 1]	Downtime per month	Downtime per week	Downtime per day
90% ("one nine")	**36.53 days**	**73.05 hours**	**16.80 hours**	**2.40 hours**
95% ("one and a half nines")	18.26 days	36.53 hours	8.40 hours	1.20 hours
97%	10.96 days	21.92 hours	5.04 hours	43.20 minutes
98%	7.31 days	14.61 hours	3.36 hours	28.80 minutes
99% ("two nines")	**3.65 days**	**7.31 hours**	**1.68 hours**	**14.40 minutes**
99.5% ("two and a half nines")	1.83 days	3.65 hours	50.40 minutes	7.20 minutes

Рис. 2: Пример «Nines»

HA включает в себя несколько важных аспектов:

1. Доступность данных
2. Защита данных
3. Производительность
4. Цена поддержки

5.1 Средства, поддерживающие высокую готовность

Аппаратная и программная поддержки Существует несколько возможных уровней обеспечения высокой готовности системы: аппаратный и программный уровни.
Аппаратный уровень включает в себя:

1. Репликация БД Репликация позволяет скопировать данные с одного сервера бд на другой. Существуют два подхода репликации баз данных, рассмотрим каждый из них ((*Репликация данных* 2022)):

   - Репликация Master-Slave
     В этом подходе выделяется один основной сервер базы данных, который называется Master. На нем происходят все изменения в данных (любые запросы INSERT/UPDATE/DELETE). Slave-сервер постоянно копирует все изменения с Master. С приложения на Slave-сервер могут отправляться запросы на чтение данных (запросы SELECT). Таким образом Master-сервер может отвечать за изменения данных, а Slave за чтение. Также Master сервер может отвечать за все операции с данными, а Slave являться бэкапом. При выходе из строя Slave, достаточно просто переключить все приложение на работу с Master. После этого восстановить репликацию на Slave и снова его запустить. Если выходит из строя Master,

нужно переключить все операции (и чтения, и записи) на Slave. Таким образом он станет новым Master. После восстановления старого Master, настроить на нем реплику, и он станет новым Slave. Выше был рассмотрен случай асинхронной репликации. При синхронной репликации результат сразу же записывается и в Master, и в Slave. Возвращение управления клиенту происходит только после записи в обе базы.

- Репликация Master-Master
  В этой схеме любой из серверов может использоваться как для чтения, так и для записи. При использовании такого типа репликации достаточно выбрать случайного Master для обработки соединения, но часто один из серверов выбирается основным(Active), а другой(Passive) используется, как бэкап, в который при необходимости(в случае выхода из строя основного) можно легко начать писать данные. Репликация типа Мастер-Мастер для баз данных увеличивает скорость доступа к данным и повышает избыточность для действующих сайтов. Но данная схема обладает недостатком из-за возможных рассинхронизаций при записи данных.

Помимо разбиения репликаций по способу организации системы (см. выше) стоит выделить различные подходы по способу внесения изменений в базу данных. Рассмотрим на примере PostgreSQL, который поддерживает эти реализации((*Как настроить репликацию в PostgreSQL* 2022)).

- Потоковая репликация
  Это репликация, при которой от основного сервера PostgreSQL на реплики передается WAL(журнал предзаписи транзакций). И каждая реплика затем по этому журналу изменяет свои данные. Для настройки такой репликации все серверы должны быть одной версии, работать на одной ОС и архитектуре. Потоковая репликация в Postgres бывает двух видов — асинхронная и синхронная.

  1.1 Асинхронная репликация
  В этом случае PostgreSQL сначала применит изменения на основном узле и только потом отправит записи из WAL на реплики. Преимущество такого способа — быстрое подтверждение транзакции, т.к. не нужно ждать пока все реплики применят изменения. Недостаток в том, что при падении основного сервера часть данных на репликах может потеряться, так как изменения не успели продублироваться.

  1.2 Синхронная репликация
  В этом случае изменения сначала записываются в WAL хотя бы одной реплики и только после этого фиксируются на основном сервере. Преимущество — более надежный способ, при котором сложнее потерять данные. Недостаток — операции выполняются медленнее, потому что прежде чем подтвердить транзакцию, нужно сначала продублировать ее на реплике.

- Логическая репликация
  Логическая репликация оперирует записями в таблицах PostgreSQL. Этим она отличается от потоковой репликации, которая оперирует физическим уровнем данных: биты, байты, и адреса блоков на диске. Возможность настройки логической репликации появилась в PostgreSQL 10. Этот вид репликации построен на механизме публикации/подписки: один

сервер публикует изменения, другой подписывается на них. При этом подписываться можно не на все изменения, а выборочно. Например, на Master-сервере 50 таблиц: 25 из них могут копироваться на один Slave-сервер, а 25 — на другой. Также есть несколько ограничений, главное из которых — нельзя реплицировать изменения структуры БД. То есть если на Master-сервере добавится новая таблица или столбец — эти изменения не попадут в Slave автоматически, их нужно применять отдельно. В отличие от потоковой репликации, логическая может работать между разными версиями PostgreSQL, ОС и архитектурами.

2. RAID - технология виртуализации данных для объединения нескольких физических дисковых устройств в логический модуль для повышения отказоустойчивости и производительности. Рассмотрим базовые уровни рейд массивов ((David A. Patterson 1988)):

   2.1 RAID 0 (striping — «чередование») — дисковый массив из двух или более жёстких дисков без резервирования. Информация разбивается на блоки данных фиксированной длины и записывается на оба/несколько дисков поочередно.

   2.2 RAID 1 (mirroring — «зеркалирование») — массив из двух (или более) дисков, являющихся полными копиями друг друга. Не следует путать с массивами RAID 1+0 (RAID 10), RAID 0+1 (RAID 01), в которых используются более сложные механизмы зеркалирования.

   2.3 RAID 2. Массивы такого типа основаны на использовании кода Хэмминга. Диски делятся на две группы: для данных и для кодов коррекции ошибок, причём если данные хранятся на $2^n - n - 1$ дисках, то для хранения кодов коррекции необходимо n дисков. Суммарное количество дисков при этом будет равняться $2^n - 1$. Данные распределяются по дискам, предназначенным для хранения информации, так же, как и в RAID 0, то есть они разбиваются на небольшие блоки по числу дисков.

   2.4 В массиве RAID 3 из n дисков данные разбиваются на куски размером меньше сектора (разбиваются на байты или блоки) и распределяются по $n - 1$ дискам. Ещё один диск используется для хранения блоков чётности. В RAID 2 для этой цели применялся $n - 1$ диск, но большая часть информации на контрольных дисках использовалась для коррекции ошибок «на лету», в то же время большинство пользователей устраивает простое восстановление информации в случае её повреждения, для чего хватает данных, умещающихся на одном выделенном жёстком диске.
   Отличия RAID 3 от RAID 2: невозможность коррекции ошибок на лету.

   2.5 RAID 4 похож на RAID 3, но отличается от него тем, что данные разбиваются на блоки, а не на байты. Таким образом, удалось отчасти «победить» проблему низкой скорости передачи данных небольшого объёма. Запись же производится медленно из-за того, что чётность для блока генерируется при записи и записывается на единственный диск.

   2.6 RAID 5. Основным недостатком уровней RAID от 2-го до 4-го является невозможность производить параллельные операции записи, так как для хранения информации о чётности используется отдельный контрольный диск. RAID 5 не имеет этого недостатка. Блоки данных и контрольные суммы циклически записываются на все диски массива, нет асимметричности конфигурации дисков. Под контрольными суммами подразумевается результат операции XOR (исключающее или). XOR обладает особенностью, которая

даёт возможность заменить любой операнд результатом, и, применив алгоритм XOR, получить в результате недостающий операнд.

2.7 RAID 6 — похож на RAID 5, но имеет более высокую степень надёжности — два (или более) диска данных и два диска контроля чётности. Основан на кодах Рида — Соломона и обеспечивает работоспособность после одновременного выхода из строя любых двух дисков. Обычно использование RAID 6 вызывает примерно 10-15 % падение производительности дисковой группы, относительно RAID 5, что вызвано бóльшим объёмом работы для контроллера (более сложный алгоритм расчёта контрольных сумм), а также необходимостью читать и перезаписывать больше дисковых блоков при записи каждого блока.

Также существуют различные комбинации данных подходов.

Программные подходы:

1. Автоматический перезапуск экземпляров БД, сетевых демонов и других ресурсов
2. Защита от повреждения данных(Data corruption protection)
   - Использование контрольных чек-сумм
   - Автоматическое восстановление из резервной копии или повторная передача
3. PITR (Point In Time Recovery) Данный механизм позволяет восстановить базу данных в том виде, в котором она была в каком-то моменте в прошлом.
4. Application continuity (Oracle) Маскирует от приложения и конечных пользователей падение базы данных и позволяет восстановить сеансы базы данных во время работы.
5. WAL Данный подход будет описан ниже.

Кластерная организация серверов баз данных

Кластеризация базы данных - это процесс объединения нескольких серверов или инстансов, соединяющих одну базу данных. Иногда одного сервера может быть недостаточно для управления объемом данных или количеством запросов, тогда возникает необходимость в кластере.

К общим требованиям, предъявляемым к кластерным системам, относятся:

1. Высокая готовность
2. Высокое быстродействие
3. Масштабирование
4. Удобство обслуживания

Кластеры баз данных являются распространенной технологией. Рассмотрим три типа архитектуры кластерных вычислений. Отказоустойчивые кластеры, высокопроизводительные кластеры и кластеры балансировки нагрузки.

1. Отказоустойчивые / высокодоступные кластеры. Кластер обеспечивает доступность сервиса путем репликации серверов и избыточной реконфигурации программного и аппаратного обеспечения. Таким образом, каждая система контролирует другую и работает на запросы, если какой-либо один из узлов выходит из строя.

2. Высокопроизводительные кластеры. Целью разработки высокопроизводительных кластеров баз данных является создание высокопроизводительных компьютерных систем. Основная цель - разумное распределение рабочей нагрузки.

3. Кластеры балансировки нагрузки. Эти кластеры базы данных служат для распределения нагрузки между различными серверами. Они стремятся обеспечить увеличенную пропускную способность сети, в конечном итоге увеличивая производительность. Системы в этой сети объединяют свои узлы, с помощью которых пользовательские запросы равномерно распределяются между участвующими узлами.

Несмотря на всю распределенную систему на заднем плане, пользователю это кажется единой системой. Использование кластеров варьируется от предприятия к предприятию, в зависимости от вида процессов и требуемого уровня производительности.

Параметры настройки СУБД
Непонятно о чем это, настройки чего

Сохранение и восстановление БД

Основным свойством транзакций СУБД является durability. Идея механизма обеспечения этого свойства является одинаковой для всех СУДБ ((*Как настроить репликацию в PostgreSQL* 2022)). СУБД использует специальны Следовательно, с точки зрения файловой системы INSERT и UPDATE не являются атомарными операциями: если кто-то внезапно выключит ваш сервер, данные окажутся испорчены. Если возникает сбой, база данных использует этот файл для того, чтобы восстановить данные на момент падения. WAL – это бинарный лог, так что для его чтения нужна специальная утилита. Например в PostreSQL данный файл называется WAL и лежит по пути $PGDATA/pg_xlog. PostgreSQL позволяет отключать WAL для отдельных таблиц, помечая их как UNLOGGED.

5.2 Оперативное администрирование

Задачи, средства и режимы администрирования Задачи администратора баз данных могут незначительно отличаться в зависимости от вида применяемой СУБД, но в основные задачи входит:

- Проектирование базы данных.
- Оптимизация производительности базы данных.
- Резервирование и восстановление базы данных.
- Обеспечение целостности баз данных.
- Обеспечение перехода на новую версию СУБД.

Существуют различные программы и способы администрирования БД, которые напрямую зависят от БД. Основным инструментом работы с базой данных является DSL и программы, например для Oracle можно использовать SQL Developer. Существуют программы, поддерживающие работу с любой БД, например продукты JetBrains. Также отдельно существуют средства мониторинга серверов.

Мониторинг серверов СУБД Мониторинг СУБД можно условно разделить на два типа:

1. Мониторинг средствами СУБД
2. Мониторинг хостов и сервисов, на которых запущена БД.

Мониторинг средствами СУБД
Средства мониторинга, предоставляемые СУБД зависят в первую очередь от конкретной реализации этой системы, однако можно выделить метрики, которые в большинстве случаев собираются встроенным мониторингом. В первую очередь это объем операций ввода/вывода, необходимых для исполнения транзакции, утилизация процессоров и временем отклика системы. Наиболее распространенной метрикой оценки производительности системы является ее время отклика, которое представляет собой интервал времени, в течении которого сервер возвращает первую строку результата исполнения запроса, т.е. пользователь получает визуальное подтверждение того, что его запрос исполняется. Пропускная способность обслуживаемых сервером процессов и пользователей определяет сколько запросов возможно исполнить в фиксированный интервал времени, и сколько строк и какого размера возвращается клиенту. При увеличении числа активных процессов и/или пользователей, возрастает и их конкуренция за системные ресурсы. Результатом такой чрезмерной нагрузки может стать увеличение времени отклика и снижение общей пропускной способности. Большое влияние на производительности базы данных оказывает также физическая и логическая целостность данных.
Мониторинг хостов и сервисов, на которых запущена БД
Рассмотрим две системы - немного устаревший, но еще использующийся Zabbix и более современный и активно набирающий популярность Prometheus((*Prometheus vs Zabbix* 2022)).

- Zabbix
  Многофункциональным средством является Zabbix — свободная система мониторинга и отслеживания статусов разнообразных сервисов компьютерной сети, серверов и сетевого оборудования. Он поддерживает несколько видов мониторинга. Simple checks — может проверять доступность и реакцию стандартных сервисов, таких как SMTP или HTTP без установки какого-либо программного обеспечения на наблюдаемом хосте. Zabbix agent — может быть установлен на UNIX-подобных или Windows хостах для получения данных о нагрузке процессора, использования сети, дисковом пространстве и так далее. External check — выполнение внешних программ. С точки зрения пользователя Zabbix разделен на две большие части: сервер и агенты. Сервер расположен на одной машине, которая собирает и хранит статистические данные, а агенты располагаются на тех машинах, с которых собираются данные. Агенты Zabbix поддерживают как пассивные (polling), так и активные проверки (trapping). Пассивные проверки означают, что сервер Zabbix запрашивает значение у агента Zabbix, а агент обрабатывает запрос и возвращает значение серверу Zabbix. Активные проверки означают, что

агент Zabbix запрашивает список активных проверок с сервера Zabbix, а затем периодически отправляет результаты. Важной особенностью Zabbix является способ хранения данных. Для своей работы данное средство при установке требует подключения внешней базы данных (MySQL, PostgreSQL, Oracle и т.д.). Также стоит отметить, что Zabbix не так гибок в запросах. Он использует ключи элементов для получения метрик.

- Prometheus
  Prometheus — это система мониторинга с открытым исходным кодом, предоставляющая своим пользователям мощный язык запросов, функции хранения и визуализации. Он собирает метрики в реальном времени и записывает их в базу данных временных рядов. Prometheus предоставляет многомерную модель данных, которая позволяет определять метрики по именам и/или тегам, чтобы идентифицировать их как часть уникального временного ряда. Благодаря большому сообществу многие сервисы могут отправлять метрики в формате Prometheus. Если какие-то сервисы не могут этого сделать, то есть множество библиотек, помогающих в экспорте существующих метрик из сторонних систем в виде метрик Prometheus. Prometheus для хранения данных использует свою базу данных временных рядов (TSDB). Используя собственную TSDB, Prometheus может получать и обрабатывать несравненно больше метрик, чем многие другие системы мониторинга. Данные могут быть записаны даже с отметками времени с миллисекундным разрешением. В отличие от Zabbix Prometheus является более гибким в плане выполнения запросов. Prometheus предоставляет собственный функциональный язык для запросов, который называется PromQL (Prometheus Query Language). PromQL невероятно гибкий, простой и мощный. Он может применять функции и операторы к вашим запросам метрик, фильтровать, группировать по меткам и использовать регулярные выражения для улучшения сопоставления и фильтрации.

5.3 Функциональная насыщенность СУБД

Формы избыточности Системы, обеспечивающие непрерывный доступ к данным (fault tolerant) или почти непрерывный (high availability) обычно опираются на различные формы избыточности. Как правило, это системы дублирования аппаратного обеспечения и контролируемой избыточности данных ((Г.Г. 1995)).

Избыточность данных Нормальная форма — свойство отношения в реляционной модели данных, характеризующее его с точки зрения избыточности, потенциально приводящей к логически ошибочным результатам выборки или изменения данных. Нормальная форма определяется как совокупность требований, которым должно удовлетворять отношение.

Процесс преобразования отношений базы данных к виду, отвечающему нормальным формам, называется нормализацией. Нормализация предназначена для приведения структуры БД к виду, обеспечивающему минимальную логическую избыточность, и не имеет целью уменьшение или увеличение производительности работы или же уменьшение или увеличение физического объёма базы данных. Конечной целью нормализации является уменьшение потенциальной противоречивости хранимой в базе данных информации. Как отмечает К. Дейт,общее назначение процесса нормализации заключается в следующем:

- исключение некоторых типов избыточности;

- устранение некоторых аномалий обновления;
- разработка проекта базы данных, который является достаточно «качественным» представлением реального мира, интуитивно понятен и может служить хорошей основой для последующего расширения;
- упрощение процедуры применения необходимых ограничений целостности.

Устранение избыточности производится, как правило, за счёт декомпозиции отношений таким образом, чтобы в каждом отношении хранились только первичные факты (то есть факты, не выводимые из других хранимых фактов).

Аппаратная избыточность Аппаратная избыточность может включать платформы с полным резервированием, поддерживающие (standby) процессоры, диски с двойным интерфейсом (dual-port), дисковые массивы и пр. Один из вариантов - зеркалирование дисков, когда один диск используется в качестве копии другого и может быть использован при сбое вместо него. Хотя аппаратная избыточность и важна для повышения общей надежности системы, ее реализация, как правило, не ориентирована на обработку транзакций СУБД и на связанные с этим специфические ограничения, например, обеспечение атомарности транзакции. В результате СУБД не может воспользоваться преимуществами чисто аппаратных решений резервирования системы для повышения своей производительности.

Программное зеркалирование Программное зеркалирование дисков, называемое также дуплексированием (duplexing) или мультиплексированием (multiplexing), может не только защитить от аппаратных сбоев, но и улучшить производительность. Поскольку зеркалирование базы данных (или ее частей - таблиц(ы), индексов, их фрагментов и пр.) производится на другом физическом устройстве, то операции чтения данных можно распределить между двумя устройствами и производить параллельно. Конечно, зеркалирование бесполезно с любой точки зрения, если оно организовано на одном диске ((Г.Г. 1995)). В случае повреждения зеркалируемого диска все операции автоматически переносятся на исправный диск, сбойный диск выводится в отключенное состояние, причем приложения не замечают каких-либо изменений в конфигурации системы. После замены неисправного диска параллельно с работой пользователей запускается процесс оперативной синхронизации зеркальных дисков (on-line remirroring), на физическом уровне копирующий рабочий диск.

Тиражирование данных Тиражирование в системах, требующих в первую очередь повышенной надежности, в целом подобно зеркалированию, но здесь копия данных может поддерживаться удаленно. Если происходит копирование всей базы данных, то обычно это делается с целью обеспечить горячий резерв (warm standby). Однако в некоторых реализациях есть возможность использовать копию для просмотра (без модификации) данных ((Г.Г. 1995)). Это способно обеспечить значительные преимущества для систем со смешанной загрузкой, поскольку приложения для принятия решений, генерации отчетов и т.п, могут обращаться к копии базы данных, в то время как приложения оперативной обработки транзакций используют первичную базу данных.

5.4 Системы, обладающие свойством высокой готовности

Casandra

Casandra - распределённая система управления базами данных, относящаяся к классу NoSQL-систем и рассчитанная на создание высокомасштабируемых и надёжных хранилищ огромных массивов данных, представленных в виде хэша ((2020)).

Хранилище само позаботится о проблемах наличия единой точки отказа (single point of failure), отказа серверов и о распределении данных между узлами кластера (cluster node). При чем, как в случае размещения серверов в одном центре обработки данных (data center), так и в конфигурации со многими центрами обработки данных, разделенных расстояниями и, соответственно, сетевыми задержками. Под надёжностью понимается итоговая согласованность (eventual consistency) данных с возможностью установки уровня согласования данных (tune consistency) каждого запроса.

Узлы кластера Casandra равноценны, и клиенты могут соединятся с любым из них, как для записи, так и для чтения. Запросы проходят стадию координации, во время которой, выяснив при помощи ключа и разметчика на каких узлах должны располагаться данные, сервер посылает запросы к этим узлам. Будем называть узел, который выполняет координацию — координатором (coordinator), а узлы, которые выбраны для сохранения записи с данным ключом — узлами-реплик (replica nodes). Физически координатором может быть один из узлов-реплик — это зависит только от ключа, разметчика и меток. Для каждого запроса, как на чтение, так и на запись, есть возможность задать уровень согласованности данных.

Когда данные приходят после координации на узел непосредственно для записи, то они попадают в две структуры данных: в таблицу в памяти (memtable) и в журнал закрепления (commit log). Таблица в памяти существует для каждого колоночного семейства и позволяет запомнить значение моментально. Технически это хеш-таблица (hashmap) с возможностью одновременного доступа (concurrent access) на основе структуры данных, называемой “списками с пропусками” (skip list). Журнал закрепления один на всё пространство ключей и сохраняется на диске. Журнал представляет собой последовательность операций модификации. Так же он разбивается на части при достижении определённого размера. Такая организация позволяет сделать скорость записи ограниченной скоростью последовательной записи на жесткий диск и при этом гарантировать долговечность данных (data durability). Журнал закрепления в случае аварийного останова узла читается при старте сервиса Casandra и восстанавливает все таблицы в памяти. Получается, что скорость упирается во время последовательной записи на диск, а у современных жёстких дисков это порядка 100МБ/с. По этой причине журнал закрепления советуют вынести на отдельный дисковый носитель.

PostreSQL + Patroni Patroni — это Python-приложение для создания высокодоступных PostgreSQL кластеров на основе потоковой репликации. С его помощью можно преобразовать систему из ведущего и ведомых узлов (primary — replica) в высокодоступный кластер с поддержкой автоматического контролируемого (switchover) и аварийного (failover) переключения. Patroni позволяет легко добавлять новые реплики в существующий кластер, поддерживает динамическое изменение конфигурации PostgreSQL одновременно на всех узлах кластера и множество других возможностей, таких как синхронная репликация, настраиваемые действия при переключении узлов, REST API, возможность запуска пользовательских команд для создания реплики вместо pg basebackup, взаимодействие с Kubernetes и т.д.((А.Клюкин 2022))

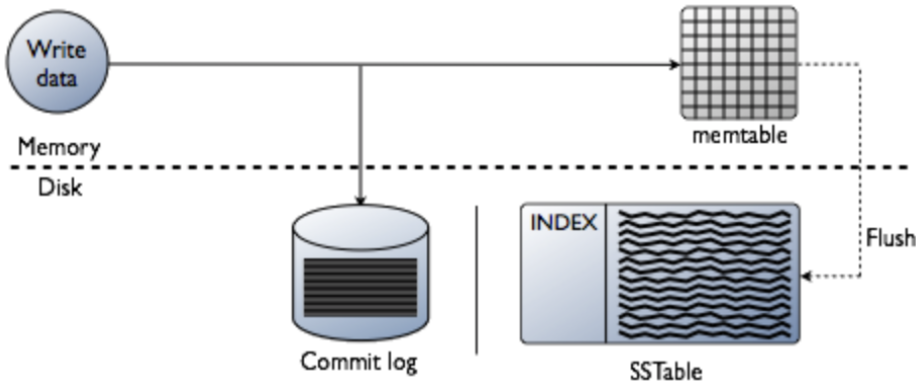


Рис. 3: Схема записи в Casandra

Опишем, как используется Patroni. Сама по себе потоковая репликация не является достаточным средством обеспечения высокой доступности. Потому что нет никакого встроенного решения, которое бы позволило перевести standby в режим нового мастера, если что-то произошло со старым мастером. Рассмотрим на примере ((Аристов Е.Н. 2021)).
Возьмем кластер из двух нод, и, допустим, у нас есть программа, запущенная на standby-сервере, мониторящая, жив ли основной сервер, и при его падении повышающая реплику до основного сервера:

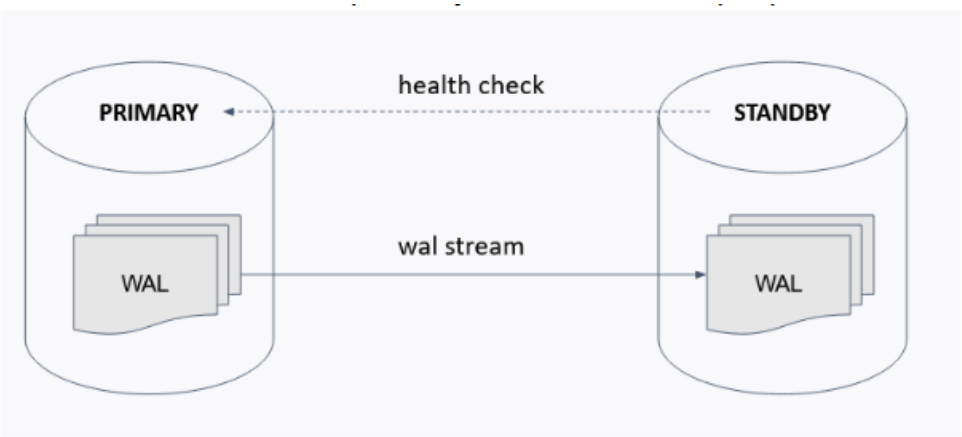


Рис. 4: Кластер из двух нод

Вроде всё хорошо, но что произойдёт, если у нас просто случится обрыв сетевого подключения между серверами?

Правильно! Получим два независимых основных кластера, каждый из них будет принимать запросы на запись, и мы получим такую ситуацию:

Она называется splitbrain. Это очень плохая ситуация, и в дальнейшем объединить изменённые данные с двух независимых серверов без потерь практически нереально. Казалось бы, есть простое решение - добавить стороннего наблюдателя. Что же может пойти не так в данной конфигурации:

В такой конфигурации могут возникнуть две проблемы: во-первых, может умереть наблюдатель и мониторить станет некому, во-вторых, может оборваться соединение с основной нодой и опять мы

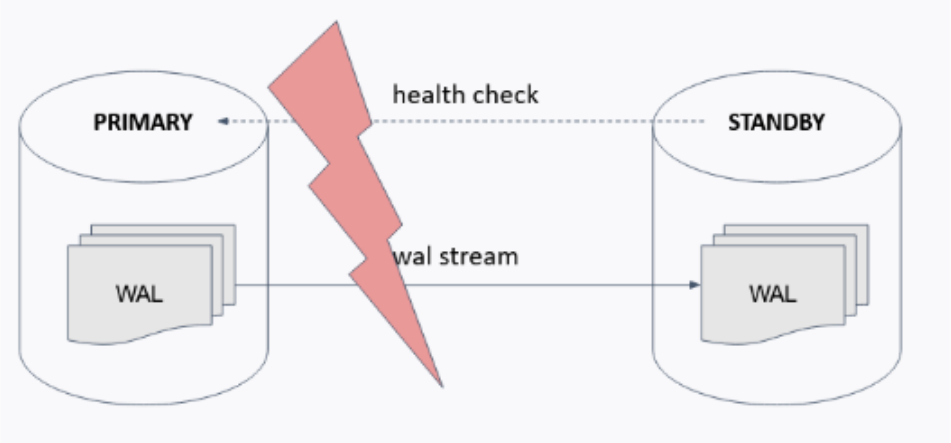


Рис. 5: Кластер из двух нод

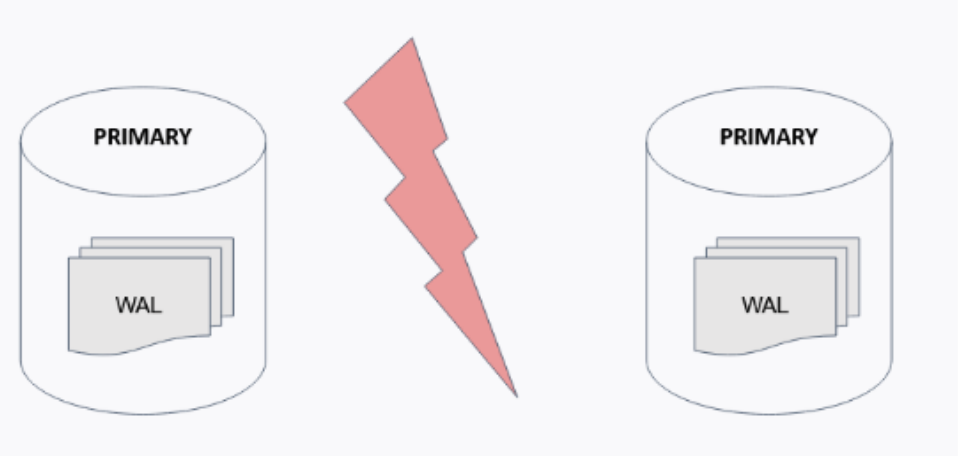


Рис. 6: Разрыв соединения в кластере из двух нод

получаем splitbrain.

И Patroni предлагает решение данной проблемы - это использование отказоустойчивого кластера для наблюдателя, в котором мы будем хранить статус активного сервера. То есть основной будет активно ходить и поддерживать статус основного, а остальные будут опрашивать жив ли основной сервер. При потере с ним соединения через некоторое время произойдут выборы и вторичный сервер станет основным и будет активно поддерживать свой статус в этом отказоустойчивом наблюдателе. При этом изначальный основной сервер при недоступности кластера наблюдения перейдёт в статус “только чтение".

Quorum позволяет нам решать сложные задачи разрешения партицирования сети. Когда какой-то сегмент сети недоступен, то несколько других сегментов могут принимать решения. А изолированный сегмент в этом случае должен остановить старого мастера.

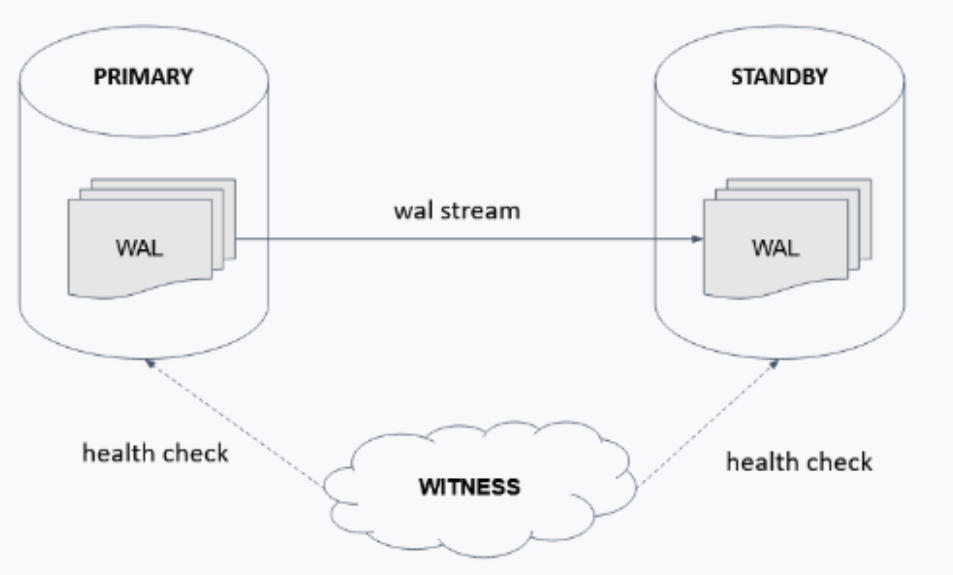


Рис. 7: Кластер из двух нод со сторонним наблюдателем

6 Защита данных в распределенных системах

6.1 Распределенные вычислительные среды

Распределённая обработка данных - методика выполнения прикладных программ группой систем. При этом пользователь получает возможность работать с сетевыми службами и прикладными процессами, расположенными в нескольких взаимосвязанных абонентских системах. (Т.Ю.Сергеева, М.Ю.Сергеев 2014)

Распределенная обработка информации в среде клиент-сервер. Концепция распределенной вычислительной среды Distributed Computing Environment (DCE). Распределенные базы данных в сетях ЭВМ

Компьютер (или программу), управляющий ресурсом, называют сервером этого ресурса (файл-сервер, сервер базы данных, вычислительный сервер...). Клиент и сервер какого-либо ресурса могут находиться как в рамках одной вычислительной системы, так и на различных компьютерах, связанных сетью. Основной принцип технологии "клиент-сервер"заключается в разделении функций приложения на три группы:

- ввод и отображение данных (взаимодействие с пользователем);
- прикладные функции, характерные для данной предметной области;
- функции управления ресурсами (файловой системой, базой данных и т.д.)

Поэтому, в любом приложении можно выделить следующие компоненты:

- компонент представления данных

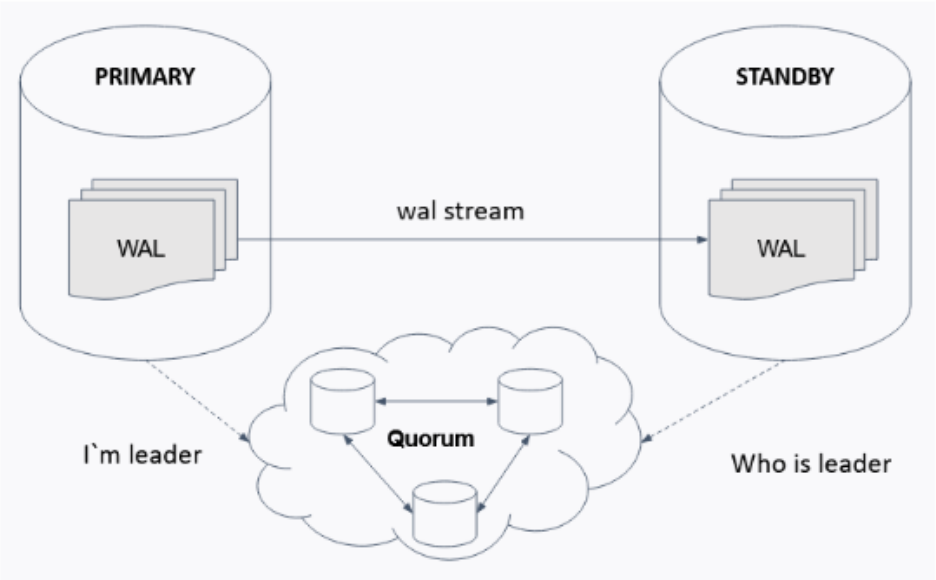


Рис. 8: Решение, используемое в Patroni

- прикладной компонент
- компонент управления ресурсом

Связь между компонентами осуществляется по определенным правилам, которые называют "протокол взаимодействия". Каждый из компонентов приложения при этом может работать на выделенном сервере (узле) или разделять ресурсы сервера с другими компонентами приложения. В связи с этим можно выделить следующие модели приложений:

- двухзвенная модель (модель «клиент-сервер»)
- трехзвенная модель (модель сервера приложений)
- многозвенная модель

Двухзвенная модель позволяет распределить различным образом три компонента приложения между двумя узлами. **Трехзвенная модель** предполагает выделение для каждого из трех компонентов приложения свой сервер. **Многозвенная модель** позволяет отдельным компонентам использовать ресурсы нескольких серверов, например, распределенные базы данных. Компанией Gartner Group, специализирующейся в области исследования информационных технологий, предложена следующая классификация двухзвенных моделей взаимодействия клиент-сервер (()bmstu)

DCE (Distributed Computing Environment) — это система программного обеспечения, предназначенная для разработки программ, использующих распределённые вычисления. (2022) Состоит из:

- Удалённый вызов процедур (RPC, remote procedure call) — DCE имеет собственную систему RPC (DCE/RPC). Как и любой RPC, используется для вызова процедур на удалённом компьютере.

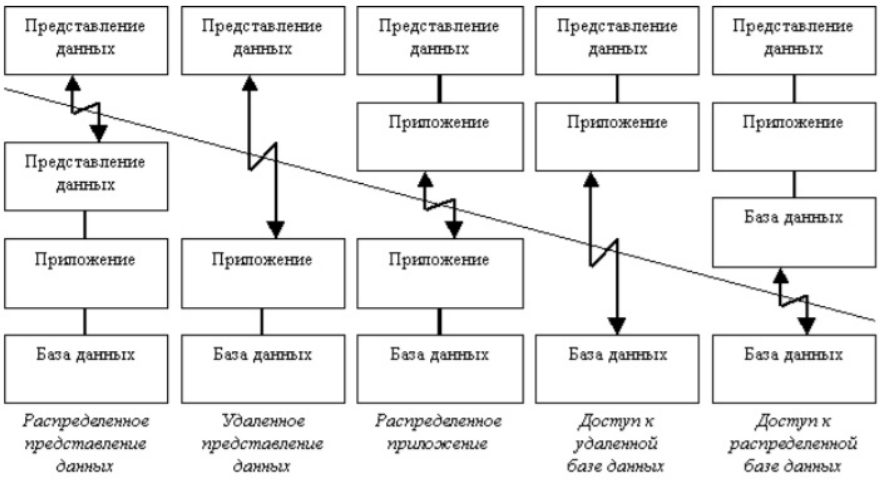


Рис. 9: Классификация двухзвенных моделей

- Служба каталогов (Directory Service) — центральное хранилище информации о ресурсах в распределённой системе. К ним относятся пользователи, компьютера и службы RPC, а также их атрибуты (свойства).
- Служба безопасности (Security Service) — служба, отвечающая за защиту данных при передаче по сети (с помощью шифрования, например), контроль доступа к вычислительным ресурсам, и т. д.
- Служба синхронизации времени (Time Service) — как понятно из названия, используются для синхронизации времени на узлах системы.
- Файловая служба (File Service) — позволяет пользователям получать доступ к файлам, хранящимся на файловом сервере.
- Threads — реализует многопоточность, если она не реализована на уровне ОС.

Появление сетей ЭВМ позволило наряду с централизованными создавать и распределенные базы данных. **Распределенная база данных** состоит из нескольких, возможно, пересекающихся или даже дублирующих друг друга частей, хранимых в различных ЭВМ вычислительной сети. Однако пользователь распределенной базы данных не обязан знать, каким образом ее компоненты размещены в узлах сети, и представляет себе эту базу данных как единое целое. Работа с такой базой данных осуществляется с помощью *системы управления распределенной базой данных* (СУРБД). Не следует путать распределённую базу данных с репликацией. Репликация — это поддержание синхронизированных копий одних и тех же данных, тогда как распределённая база данных — это распределение различных данных по различным серверам. Распределённая БД тоже может поддерживать репликацию.

Преимущества распределённых БД (2022):

- Для расширения нужно добавить новый компьютер, а не менять существующую систему.
- При отказе одного узла остальные продолжают работать.

- Если эффективно распределить данные, можно снизить время ожидания для пользователя и затраты на связь с БД.

Недостатки — более сложное управление всей системой и накладные расходы на передачу данных.

6.2 Угрозы безопасности распределенных СУБД

Угрозы доступности, целостности и конфиденциальности данных. Механизмы противодействия

6.2.1 Угрозы доступности, целостности, конфиденциальности данных

Современный подход к информационной безопасности

Информационная безопасность - это защита информации и инфраструктуры с помощью различных средств и методов. Комплексный подход включает использование защитных механизмов на всех этапах жизненного цикла системы. При проектировании системы безопасности учитываются передовые тенденции, включая интрасети (Intranet). Защита WEB-сервиса, центрального элемента интрасетей, имеет первостепенное значение.

Угрозы СУБД

Распределённые системы управления базами данных (СУБД) представляют собой сложную инфраструктуру, которая подвержена различным угрозам безопасности. Эти угрозы могут иметь серьёзные последствия, включая потерю или искажение данных, нарушение конфиденциальности и доступности информации. В этом разделе мы рассмотрим некоторые из ключевых угроз безопасности, связанных с распределёнными СУБД.

Проблемы синхронизации

В распределённых СУБД данные хранятся на разных серверах или узлах. Каждый узел обрабатывает свои запросы и обновляет данные. Если эти обновления не синхронизируются должным образом между всеми узлами, это может привести к несоответствию данных. Например, один и тот же запрос, выполненный на разных узлах, может дать разные результаты.

Угрозы сетевой безопасности

Поскольку данные передаются по сети между различными узлами, они подвержены угрозам, таким как перехват данных, атаки "человек посередине"и другие виды сетевых атак. Это может привести к утечке конфиденциальной информации или к нарушению целостности данных.

Проблемы доступа и аутентификации

Управление доступом и аутентификация пользователей в распределённой среде может быть сложной задачей. Если злоумышленник может обойти механизмы аутентификации или контроля доступа, он может получить несанкционированный доступ к данным.

Отказ оборудования или сети

В распределённых СУБД отказ одного из узлов или линии связи между узлами может привести к потере доступа к данным. Это может привести к простоям в работе и потере важной информации.

Угрозы целостности данных

В распределённых СУБД данные часто реплицируются на нескольких серверах для обеспечения отказоустойчивости. Если процесс репликации нарушается, это может повлиять на целостность данных. Например, обновление может быть применено на одном сервере, но не на другом.

6.2.2 Механизмы противодействия угрозам

СУБД отличаются от других компонентов ИС специфичными угрозами, и главным их источником является сама природа баз данных. Известно, что основным средство общения с СУБД выступает язык SQL, являющийся мощным инструментом манипулирования данными. С его помощью, используя механизм правил, могут быть созданы сложные, трудно поддающиеся анализу цепочки действий, позволяющие не явным образом передавать право на выполнение определенных процедур тем, кто не имеет на это полномочий.

В качестве примера можно привести несколько угроз, возникающих при использовании языка SQL: получение информации путем логических выводов, агрегирование данных, покушение на высокую доступность.

Методы борьбы против получения информации путем логических выводов состоят в тщательном проектировании модели данных, иерархии привилегий и видимых пользователям представлений.

Агрегирование данных состоит в получении новой информации путем комбинирования данных, полученным официальным путем. Причем информация, содержащаяся в скомбинированных данных, может иметь гриф более высокий, чем первичная информация.

Методом борьбы с агрегированием может быть тщательное проектирование модели данных и максимально допустимое ограничение доступа пользователей к информации.

Покушение на высокую доступность может быть реализовано, если пользователю-нарушителю доступны все возможности языка SQL. При этом он легко сможет заблокировать работу других пользователей. Поэтому, в целях борьбы с данным видом угроз, рекомендуется запрещать непосредственный SQL-доступ к базе данных, используя для этого серверы приложений.

В распределённых СУБД необходимо передавать данные между узлами. Поэтому, кроме перечисленных угроз, в них также реализуются угрозы, связанные с каналами связи. Защититься от них можно с помощью шифрования.

В распределённых СУБД пользователи могут обращаться к разным серверам. Например, первый пользователь обращается к первому серверу БД и просит удалить некоторую строку, которая расположена на втором сервере. А в это время второй пользователь обращается ко второму серверу и просит прочитать эту строку. В таком случае может возникнуть ситуация, когда строка уже удалена, но второй пользователь её получил. Защищаться от этого необходимо с помощью распределённых транзакций.

Средства резервного копирования

Резервное копирование программ и данных необходимо проводить с целью минимизации потерь в случае отказов оборудования, либо сбоев в программном обеспечении ИС. Данная задача наиболее сложна именно в интрасетях с их распределенными ресурсами и неоднородностью, в которых работают компьютеры под управлением различных операционных систем. Учитывая клиент/серверный характер интрасетей функцию резервного копирования целесообразно также выделить в виде отдельного сервера (сервера архива).

Распространение клиент/серверного подхода на процедуру резервного копирования информации и данных имеет ряд преимуществ по сравнению с традиционными методами. Они выражаются в следующем:

- Администраторы рабочих групп освобождаются от необходимости согласования действий и самой процедуры создания локальных резервных копий
- Единообразие процедуры создания резервных копий в ИС
- Возможность мониторинга процесса резервирования и диагностики возникших проблем

Одним из способов обеспечения высокой доступности информации является создание резервных копий с возможностью ее хранения в двух местах: один экземпляр хранится поблизости от оригинала, а другой в удаленном безопасном месте.

Обеспечение конфиденциальности данных

В СУБД, как правило, используется произвольное управление доступом, когда владелец объекта передает права доступа к нему (привилегии) по своему усмотрению. При этом привилегии в СУБД можно подразделить на две категории: привилегии безопасности и привилегии доступа.

Привилегии безопасности всегда выделяются конкретному пользователю и позволяют выполнять административные действия.

Привилегии доступа определяют права доступа субъектов к определенным объектам.

Специфическим механизмом управления доступом в СУБД являются представления. Они позволяют сделать видимыми для субъектов только те столбцы базовых таблиц, доступ к которым предоставлен субъектам администратором базы.

Поддержание целостности

Целостность данных не менее важна, чем конфиденциальность, ввиду того, что для баз данных, как и для ИС в целом, главными врагами являются не внешние нарушители, а ошибки оборудования, программ, администраторов и пользователей системы.

С точки зрения пользователей СУБД, основными средствами поддержания целостности данных являются ограничения и правила.

Ограничения могут относиться как к таблицам, так и к отдельным столбцам. Они накладываются владельцами таблицы и оказывают влияние на все операции с данными.

Правила позволяют вызывать выполнение заданных действий при определенных изменениях базы данных. В отличие от ограничений, являющихся лишь средствами контроля простых условий, правила позволяют создавать сколь угодно сложные соотношения между различными элементами базы данных.

Обеспечение доступности данных

Доступность данных подразумевает обеспечение информационной системы средствами поддержания высокой доступности. Поддержание высокой доступности позволяет свести к минимуму возможные сбои аппаратного обеспечения, в частности носителей информации, а также ошибки обслуживающего персонала и программного обеспечения. В качестве мер поддержания высокой доступности может быть названа кластеризация сервера баз данных (выделение нескольких компьютеров, выполняющих общее приложение), а также тиражирование данных (хранение базы данных в различных местах).

6.3 Распределенная обработка данных

Понятие распределенной транзакции

Если данные хранятся в одной базе данных, то транзакция к ней рассматривается как локальная. В распределенных базах транзакция, выполнение которой заключается в обновлении данных на нескольких узлах сети, называется глобальной или **распределенной транзакцией**.

Внешне выполнение распределенной транзакции выглядит как обработка транзакции к локальной базе данных. Тем не менее распределенная транзакция включает в себя несколько локальных транзакций, каждая из которых завершается двумя путями — фиксируется или прерывается. Распределенная транзакция фиксируется только в том случае, когда зафиксированы все локальные транзакции, ее составляющие.

Модель обработки транзакций

В стандарте ANSI/ISO SQL определены модель транзакций и функции операторов COMMIT и ROLLBACK. Стандарт определяет, что транзакция начинается с первого SQL-оператора, инициируемого пользователем или содержащегося в программе. Все последующие SQL-операторы составляют тело транзакции. Транзакция завершается одним из четырех возможных путей:

- оператор COMMIT означает успешное завершение транзакции; его использование делает постоянными изменения, внесенные в базу данных в рамках текущей транзакции
- оператор ROLLBACK прерывает транзакцию, отменяя изменения, сделанные в базе данных в рамках этой транзакции; новая транзакция начинается непосредственно после использования ROLLBACK
- успешное завершение программы, в которой была инициирована текущая транзакция, означает успешное завершение транзакции (как будто был использован оператор COMMIT)
- ошибочное завершение программы прерывает транзакцию (как будто был использован оператор ROLLBACK)

Точки сохранения применяются, как правило, в протяженных транзакциях и позволяют разделить транзакцию на несколько небольших осмысленных фрагментов. Пользователь может зафиксировать работу в любой точке транзакции с тем, чтобы выполнить ее откат к состоянию, соответствующему этой точке.

Откат и фиксация транзакций становятся возможными благодаря журналу транзакций. Он используется следующим образом. Известно, что все операции над реляционной базой данных суть операции над строками таблиц. Следовательно, для обеспечения отката таблиц к предыдущим состояниям достаточно хранить не состояния таблицы, а лишь те ее строки, которые подверглись изменениям.

При выполнении любого оператора SQL, который вносит изменения в базу данных, СУБД автоматически заносит очередную запись в журнал транзакций. Запись состоит из двух компонентов: первый - это состояние строки до внесения изменений, второй - ее же состояние после внесения изменений. Только после записи в журнал транзакций СУБД действительно вносит изменения в базу данных. Если после данного оператора SQL был выполнен оператор COMMIT, то в журнале транзакций делается отметка о завершении текущей транзакции. Если же после оператора SQL

следовал оператор ROLLBACK, то СУБД просматривает журнал транзакций и отыскивает записи, отражающие состояние измененных строк до внесения изменений. Используя их, СУБД восстанавливает те строки в таблицах базы данных, которые были изменены текущей транзакцией, - таким образом аннулируются все изменения в базе данных.

Мониторы обработки транзакций

Мониторы обработки транзакций (Transaction Processing Monitor - TPM), или мониторы транзакций - программные системы, предназначенные для поддержки выполнения распределённых транзакций. Они относятся к категории middleware, то есть к ПО, работающему между ОС и приложениями.

Это происходит следующим образом: клиент начинает транзакцию и обращается к TPM, который выдаёт ему идентификатор транзакции и контекст транзакции. Затем клиент с помощью RPC обращается к серверу, включая контекст в обращение. Сервер извлекает контекст, уведомляет TPM о том, что он (сервер) участвует в транзакции, и обрабатывает запрос как обычно. Так коиент обращается ко всем необходимым серверам. Когда клиент достиг окончания транзакции, он уведомляет об этом TPM, и тот выполняет протокол двухфазного коммита для всех серверов, участвовавших в транзакции. После завершения протокола TPM уведомляет об этом клиента (Рис. 10). (G. Alonso, F. Casati, H. Kuno, V. Machiraju 2004)

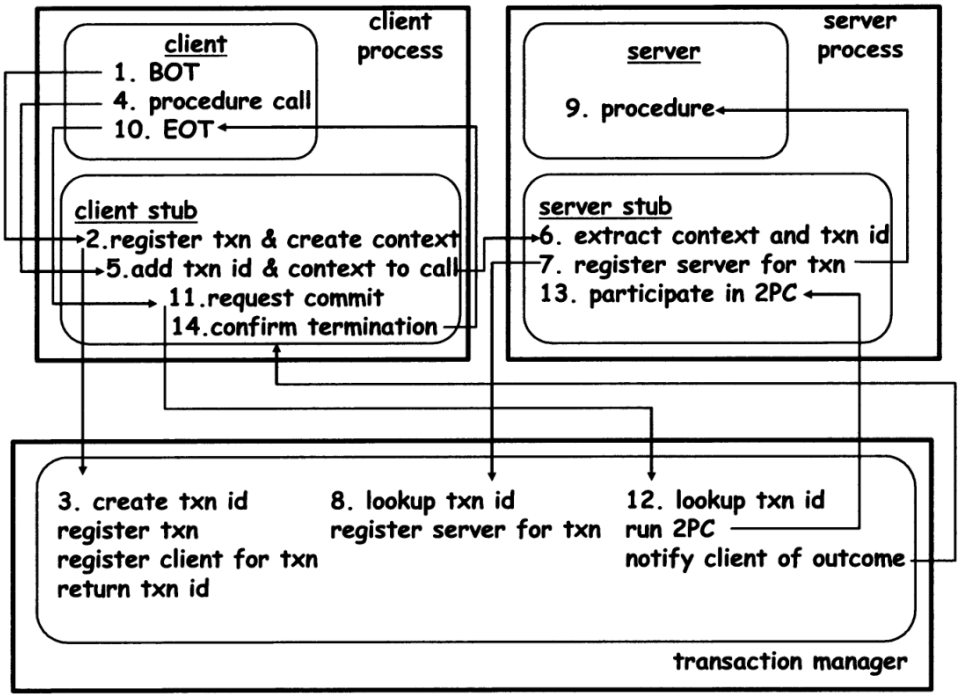


Рис. 10: Процесс взаимодействия с TPM

TPM опираются на трехзвенную модель "клиент-сервер"(модель сервера приложений или AS-модель), описанную в Разделе 2. Естественно, что все преимущества модели отражаются и на программных системах, построенных на ее основе.

На современном рынке мониторов транзакций основными "действующими лицами"являются такие системы, как ACMS (DEC), CICS (IBM), TOP END (NCR), PATHWAY (Tandem), ENCINA

(Transarc), TUXEDO System (USL). Несмотря на принципиальное сходство, конкретные ТРМ отличаются рядом характеристик, причем различия часто вытекают как из специфики операционной системы, в которой реализован и функционирует ТРМ.

Корпоративная среда обработки транзакций

ТРМ на базе UNIX опирается на фундаментальное понятие - корпоративную среду обработки транзакций (Enterprise Transaction Processing - ETP). Архитектура ETP - это три ряда компьютеров:

- Ряд 1: Персональные станции (Personal Workstations);
- Ряд 2: Компьютеры под управлением ОС UNIX (UNIX Transaction Processing Servers - UPTS);
- Ряд 3: Mainframe-системы (Proprietary Transaction Processing Servers - PTPS) или компьютеры под управлением UNIX с RISC-архитектурой процессоров;

Компьютеры **ряда 1**, функционирующие под управлением DOS, MS Windows, OS/2, UNIX, используются в качестве рабочих мест конечных пользователей. Характерная черта ETP - отсутствие ограничений на модели компьютеров, составляющих этот ряд. Однако, как правило, ряд 1 состоит из компьютеров на базе процессоров Intel 486/Pentium под управлением MS Windows (MS Windows фактически стала стандартом оконного графического интерфейса для большинства категорий пользователей и стандартом операционной среды для подавляющего числа прикладных программ и систем).

Ряд 2 составляют компьютеры среднего класса под управлением ОС UNIX, на которых функционирует ядро ТРМ и, как правило, реляционные СУБД (Oracle, Informix, Ingres), выступающие в качестве менеджера ресурсов. Кроме того, на них же может быть установлен шлюз к ТРМ в операционной среде мэйнфрейма (как правило, разработчики ТРМ на базе UNIX предусматривают в конфигурации своих систем шлюз к наиболее популярной такой системе - IBM CICS).

Ряд 3 представлен мэйнфреймами или RISC-компьютерами под управлением UNIX. О мэйнфреймах мы говорим в тех ситуациях, когда исторически сложилось так, что в организации они существуют уже долгое время, берут на себя большую часть всего объема обработки транзакций, концетрируют огромные вычислительные ресурсы и содержат большие массивы данных (то есть речь идет об унаследованных системах). Если этого "тяжелого наследия"нет, то можно смело использовать в качестве компьютеров ряда 3 RISC-серверы, сегодня приближающиеся по производительности к мэйнфреймам.

Таким образом, среда обработки транзакций формируется из набора разнородных компьютеров (и соответствующих ОС), ранжируемых от персональных компьютеров до мэйнфрейм-систем. ТРМ на базе UNIX представляет собой своего рода "клей который связывает вместе компьютеры трех рядов в открытую унифицированную среду обработки транзакций.

Ключом к интеграции систем, функционирующих на компьютерах различных рядов, является специализированный интерфейс прикладного программирования ATMI (Application Transaction Manager Interface), обеспечивающий:

- для ряда 1 - формирование и передачу запросов от клиентов к серверам, выполняющимся на компьютерах ряда 2

- для ряда 2 - обработку запросов, поступающих от компьютера ряда 1 (в том числе и с обращением к менеджеру ресурсов), и, по необходимости, формирование и направление запросов к серверам, выполняющимся на компьютерах ряда 3
- для ряда 3 - обработку запросов, поступающих от серверов ряда 2

Стоит отметить, что подобное представление о корпоративной среде обработки транзакций не является абстракцией. Сегодня многие организации приходят именно к такой, "трехуровневой"архитектуре информационных систем (с той оговоркой, что наличие ряда 3 вызвано историческими причинами - мэйнфреймы использовались первоначально и сразу от них отказаться невозможно).

6.4 Протоколы фиксации

Протоколы фиксации

Протоколы фиксации используются для обеспечения атомарности транзакций между сайтами. Транзакция, которая выполняется на нескольких сайтах, должна быть либо зафиксирована на всех сайтах, либо прервана на всех сайтах. Недопустимо, чтобы транзакция была зафиксирована на одном сайте и прервана на другом. В локальной системе базы данных для принятия транзакции диспетчер транзакций должен только передать решение о фиксации диспетчеру восстановления. Однако в распределенной системе диспетчер транзакций должен передать решение о фиксации всем серверам в различных сайтах, где выполняется транзакция, и обеспечить единообразное выполнение решения. Когда обработка завершается на каждом сайте, она достигает состояния частично подтвержденной транзакции и ожидает, пока все другие транзакции достигнут их частично подтвержденных состояний. Когда он получает сообщение о том, что все сайты готовы к фиксации, он начинает фиксировать. В распределенной системе либо все сайты фиксируют, либо ни один из них не делает.

Различные протоколы распределенной фиксации:

- Однофазный коммит
- Двухфазный коммит
- Трехфазный коммит

Широко используется протокол двухфазной фиксации (2PC).

Распределенная однофазная фиксация

Распределенная однофазная фиксация – это самый простой протокол фиксации. Давайте рассмотрим, что есть контролирующий сайт и несколько подчиненных сайтов, где выполняется транзакция. Шаги в распределенном коммите:

- После того, как каждое ведомое устройство локально завершило свою транзакцию, оно отправляет сообщение «ГОТОВО» на контролирующий сайт
- Подчиненные ожидают сообщения «Подтвердить» или «Прервать» с контролирующего сайта. Это время ожидания называется окном уязвимости

- Когда контролирующий сайт получает сообщение «СДЕЛАНО» от каждого ведомого, он принимает решение о фиксации или отмене. Это называется точкой фиксации. Затем он отправляет это сообщение всем рабам
- При получении этого сообщения ведомое устройство либо фиксирует, либо прерывает работу, а затем отправляет подтверждающее сообщение на контролирующий сайт

После того, как каждое ведомое устройство локально завершило свою транзакцию, оно отправляет сообщение «ГОТОВО» на контролирующий сайт.

Подчиненные ожидают сообщения «Подтвердить» или «Прервать» с контролирующего сайта. Это время ожидания называется окном уязвимости.

Когда контролирующий сайт получает сообщение «СДЕЛАНО» от каждого ведомого, он принимает решение о фиксации или отмене. Это называется точкой фиксации. Затем он отправляет это сообщение всем рабам.

При получении этого сообщения ведомое устройство либо фиксирует, либо прерывает работу, а затем отправляет подтверждающее сообщение на контролирующий сайт.

Распределенная двухфазная фиксация Распределенная двухфазная фиксация снижает уязвимость однофазных протоколов фиксации. Шаги, выполняемые на двух этапах:

Этап 1

- После того, как каждое ведомое устройство локально завершило свою транзакцию, оно отправляет сообщение «ГОТОВО» на контролирующий сайт. Когда контролирующий сайт получил сообщение «ГОТОВО» от всех подчиненных, он отправляет сообщение «Подготовить» подчиненным
- Рабы голосуют за то, хотят ли они по-прежнему совершать или нет. Если ведомый хочет зафиксировать, он отправляет сообщение «Готово»
- Раб, который не хочет коммитить, отправляет сообщение «Не готов». Это может произойти, если ведомое устройство имеет конфликтующие параллельные транзакции или истекло время ожидания

После того, как каждое ведомое устройство локально завершило свою транзакцию, оно отправляет сообщение «ГОТОВО» на контролирующий сайт. Когда контролирующий сайт получил сообщение «ГОТОВО» от всех подчиненных, он отправляет сообщение «Подготовить» подчиненным.

Рабы голосуют за то, хотят ли они по-прежнему совершать или нет. Если ведомый хочет зафиксировать, он отправляет сообщение «Готово».

Раб, который не хочет коммитить, отправляет сообщение «Не готов». Это может произойти, если ведомое устройство имеет конфликтующие параллельные транзакции или истекло время ожидания.

Этап 2

- После того, как контролирующий сайт получил сообщение «Готово» от всех ведомых –
  - Контролирующий сайт отправляет сообщение «Global Commit» подчиненным
  - Подчиненные устройства применяют транзакцию и отправляют сообщение «Подтвердить ACK» на контролирующий сайт

– Когда контролирующий сайт получает сообщение «Подтвердить ACK» от всех ведомых устройств, он считает транзакцию подтвержденной

- После того, как контролирующий сайт получил первое сообщение «Не готов» от любого ведомого –
  - Контролирующий сайт отправляет сообщение «Global Abort» подчиненным
  - Слэйвы отменяют транзакцию и отправляют сообщение «Abort ACK» на контролирующий сайт
  - Когда контролирующий сайт получает сообщение «Abort ACK» от всех ведомых устройств, он считает транзакцию отмененной

Обработка отказов

Введём следующие обозначения: T – транзакция, инициированная сайтом S_i, координатором которого выступает C_i.

- Отказ сайта

  Когда сайт восстанавливается, он просматривает свой журнал, чтобы определить судьбу транзакций, активных во время сбоя(**Fixations**):
  - Журнал содержит запись <commit T>: сайт выполняет переотправку (T)
  - Журнал содержит запись <abort T>: сайт выполняет отмену (T)
  - Журнал содержит запись <ready T>: сайт должен обратиться к C_i, чтобы определить судьбу T.
    - Если T зафиксировано, повторить (T)
    - Если T прервано, отмена (T).
  - Журнал не содержит никаких управляющих записей относительно ответов T что S_k потерпел неудачу, прежде чем ответил на сообщение prepare T от C_i
    - поскольку отказ S_k исключает отправку такого ответа C_1 должен прервать T
    - S_k должен выполнить отмену (T)

- Отказ координатора

  Если координатор вышел из строя во время выполнения протокола фиксации для T то участвующие сайты должны принять решение о дальнейшей судьбе T(**Fixations**):
  - Если активный сайт содержит запись <commit T> в своем журнале, то T должен быть зафиксирован.
  - Если активный сайт содержит в своем журнале запись <abort T>, то T должен быть прерван.
  - Если какой-либо активный участвующий сайт не содержит записи <ready T> в своем журнале, тогда отказавший координатор C_i не может принять решение о фиксации T. Поэтому можно прервать T.

– Если ни один из вышеперечисленных случаев не подходит, то все активные сайты должны иметь запись <ready T> в своих журналах, но никаких дополнительных управляющих записей (таких как <abort T> из <commit T>). В этом случае активные сайты должны ждать, пока C_i восстановится, чтобы найти решение.

– Проблема блокировки: активным сайтам, возможно, придется ждать, пока вышедший из строя координатора будет восстановлен.

- Разделение сети

  Если координатор и все его участники остаются в одном разделе, отказ не влияет на протокол фиксации.(**Fixations**) Если координатор и его участники принадлежат к нескольким разделам:

  – Сайты, которые не находятся в разделе, содержащем координатора считают, что координатор вышел из строя, и выполняют протокол, соответствующий случаю отказа координатора.

  Координатор и сайты, находящиеся в том же разделе, что и координатор считают, что сайты в другом разделе отказали, и следуют обычному протоколу фиксации.

Контролирование управления и параллелизма

В случае обработки in-doubt (т.е. таких, о которых в логах есть запись <ready T>, но нет ни <commit T>, ни <abort T>) транзакций восстанавливающий сайт должен определить её commit-abort статус контактируя с другими сайтами, что может замедлить и, возможно, заблокировать процесс востановления(**Fixations**).

Опишем основные принципы работы алгоритмов восстановления:

- Алгоритмы могут записывать информацию о фиксациях в лог.
- Вместо <ready T> вносится запись <ready T, L>, где L – список фиксаций, удерживаемых транзакцие T при записи лога, причём фиксации чтения могут быть опущены.
- Для каждой in-doubt транзакции T все фиксации, записанные в <ready T, L> запрашиваются снова.
- После повторного получения фиксации обработка транзакции может быть продолжена, при этом фиксация/откат in-doubt транзакции выполняется параллельно с исполнением новых транзакций.

Распределенная трехфазная фиксация Шаги в распределенной трехфазной фиксации следующие:

- Этап 1: Шаги такие же, как при распределенной двухфазной фиксации
- Этап 2: Подготовьтесь к фиксации

  – Контролирующий сайт выдает широковещательное сообщение «Ввод подготовленного состояния»

  – Ведомые сайты голосуют «ОК» в ответ

- Фиксация / Отмена

Этапы аналогичны двухфазной фиксации, за исключением того, что сообщение «Подтвердить ACK» / «Прервать ACK» не требуется

Защищенные протоколы фиксации

Обработка распределенных транзакций в базах данных с многоуровневой секретностью (MLS)
Известно, что в MLS/DBMS не ко всем данным, содержащимся в базе данных, доступ осуществляется одинаково. Однако современные СУБД, как правило, не имеют адекватных средств диагностики и механизма определения того, что пользователь имеет возможность доступа только к тем данным, которые являются релевантными. Таким образом, MLS/DBMS отличается от соответствующих DBMS, по крайней мере, следующими двумя особенностями:

- каждый элемент данных в базе данных связан с уровнем доступа
- доступ пользователя к данным должен контролироваться релевантностью для данного пользователя

Разработка сервиса MLS/DBMS в современных компьютерных системах представляет много проблем. До настоящего времени внедрение многоуровневого разграничения доступа в операционную систему представляет собой значительные трудности. Решение этой проблемы в виде аббревиатуры обозначается TCB. Хотя в разрешении вопросов TCB для удаленных пользователей в MLS/DBMS вводятся компромиссы, остается много проблем, которые требуется разрешать. Наиболее очевидная проблема состоит в том, что вопросы классификации в СУБД значительно сложнее, чем в файловых системах и могут быть сложнее реализованы. Другая проблема состоит в том, что для классификации данных, содержащих контекстные представления, временные параметры, их композицию, необходимы унифицированные базы данных.

6.5 Тиражирование данных

Тиражирование данных - это асинхронный перенос изменений объектов исходной базы данных (source database) в базы данных, принадлежащие к различным узлам распределенной системы. Тиражирование данных может происходит двумя различными способами: **шардингом** (или, иначе говоря, шардированием, фрагментацией) и **репликацией**.

6.5.1 Шардинг (sharding)

Шардинг — это метод проектирования базы данных, при котором данные разделяются на группы и хранятся на разных серверах. Это улучшает производительность, поскольку данные хранятся там, где они чаще всего используются, что уменьшает сетевой трафик.

Существуют два основных вида шардирования: **горизонтальное** и **вертикальное**. Они соответствуют операциям сокращения и проекции в реляционных базах данных. Важно, чтобы все фрагменты были независимыми, то есть ни один из фрагментов не может быть представлен как производный от других фрагментов.

Одной из ключевых особенностей шардинга является поддержка независимости от шардирования. Это означает, что пользователи могут работать так, как если бы данные в действительности были вовсе не шардированы. Это упрощает разработку пользовательских программ и выполнение терминальных операций.

Если обеспечивается независимость от шардирования, пользователи получают данные в виде представления, в котором шарды логически скомбинированы с помощью операций соединения и объединения. Задачей системного оптимизатора является определение шардов, к которым требуется доступ для выполнения запросов пользователя. (Дейт К. Дж. 2014)

Алгоритмы шардинга

Ранее уже упоминалось о двух основных вида шардинга: *горизонтальный* и *вертикальный*. Стоит их разобрать по подробнее и рассказать и про другие виды шардирования, такие как шадинг на основе каталогов и шардинг на основе ключей.

Горизонтальное шардирование

В этом методе мы разбиваем данные на основе диапазонов заданного значения, присущего каждой сущности. Допустим, у вас есть база данных имен ваших онлайн-клиентов и информации об электронной почте. Вы можете разделить эту информацию на два осколка. В одном осколке вы можете хранить информацию о клиентах, чье имя начинается с A-P, а в другом - информацию об остальных клиентах.

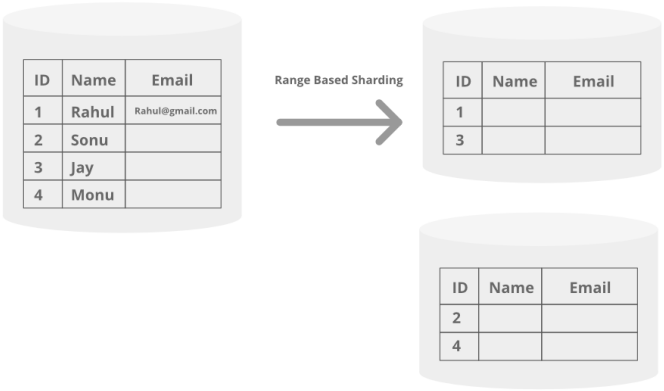


Рис. 11: Горизонтальное шардирование

Горизонтальное шардирование - самый простой метод сегментирования для реализации. Каждый осколок содержит другой набор данных, но все они имеют ту же схему, что и исходная база данных. В этом методе вам просто нужно определить, в какой диапазон попадают ваши данные, а затем вы можете сохранить запись в соответствующий шард. Этот метод лучше всего подходит для хранения нестатических данных (например, хранение контактной информации студентов колледжа).

Недостатком этого метода является то, что данные могут быть неравномерно распределены по шардам. В приведенном выше примере у вас может быть много клиентов, имена которых попадают

в категорию A-P. В таких случаях первый осколок должен будет взять на себя больше нагрузки, чем второй, и это может стать узким местом системы. (anuupadhyay 2021)

Вертикальное шардирование

В этом методе мы разделяем весь столбец из таблицы и помещаем эти столбцы в новые отдельные таблицы. Данные полностью независимы от одного раздела к другому. Кроме того, каждый раздел содержит как отдельные строки, так и столбцы. Возьмем, к примеру, функции Twitter. Мы можем разделить различные функции объекта на разные сегменты на разных машинах. В Твиттере у пользователя может быть профиль, количество подписчиков и некоторые твиты, опубликованные им самим. Мы можем разместить профили пользователей на одном осколке, подписчиков - на втором, а твиты - на третьем.

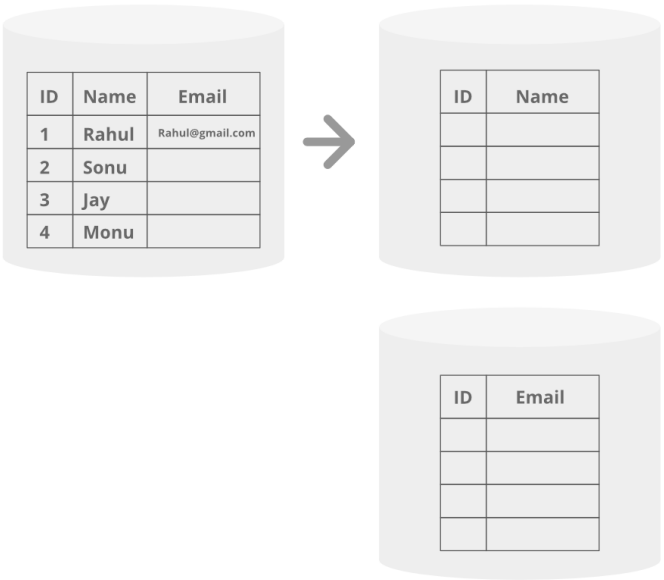


Рис. 12: Вертикальное шардирование

В этом методе вы можете отделить и обработать критическую часть (например, профили пользователей) от некритической части ваших данных (например, сообщения в блоге) по отдельности и построить вокруг нее различные модели репликации и согласованности. Это одно из главных преимуществ этого метода.

Основным недостатком этой схемы является то, что для ответа на некоторые запросы вам, возможно, придется комбинировать данные из разных шардов, что неоправданно увеличивает сложность разработки и эксплуатации системы. Кроме того, если ваше приложение будет расти позже, и вы добавите в него еще несколько функций, вам придется дополнительно сегментировать базу данных с конкретными функциями на нескольких серверах. (anuupadhyay 2021)

Шардинг на основе каталогов

В этом методе мы создаем и поддерживаем службу поиска или таблицу поиска для исходной базы данных. В основном мы используем ключ осколка для таблицы поиска и делаем сопоставление для

каждого объекта, существующего в базе данных. Таким образом, мы отслеживаем, какие осколки базы данных содержат какие данные.

Таблица поиска содержит статический набор информации о том, где можно найти конкретные данные. На приведенном выше изображении вы можете видеть, что мы использовали зону доставки в качестве ключа осколка. Во-первых, клиентское приложение запрашивает службу поиска, чтобы узнать осколок (раздел базы данных), на котором размещены данные. Когда служба поиска возвращает осколок, она запрашивает/обновляет этот осколок.

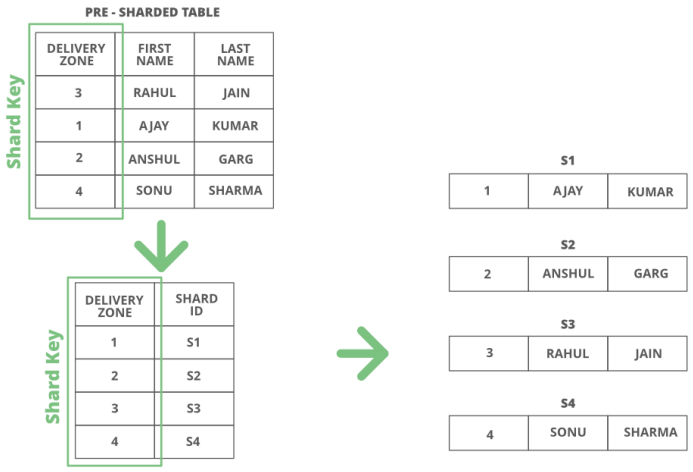


Рис. 13: Шардинг на основе каталогов

Сегментация на основе каталогов гораздо более гибкая, чем сегментация на основе диапазонов и ключей. В сегменте на основе диапазонов вы обязаны указать диапазоны значений. В key-based вы обязаны использовать фиксированную хэш-функцию, которую трудно изменить позже. При таком подходе вы можете использовать любой алгоритм, который хотите назначить для записей данных в сегменты. Кроме того, при таком подходе легко динамически добавлять шарды.

Основным недостатком этого подхода является единственная точка отказа таблицы поиска. Если он будет поврежден или не удался, это повлияет на запись новых данных или доступ к существующим данным из таблицы. (anuupadhyay 2021)

Шардинг на основе ключей

Этот метод также известен как сегментация на основе хэша. Здесь мы берем значение объекта, такого как идентификатор клиента, адрес электронной почты клиента, IP-адрес клиента, почтовый индекс и т. д., и мы используем это значение в качестве входных данных хэш-функции. Этот процесс генерирует хэш-значение, которое используется для определения того, какой шард нам нужно использовать для хранения данных. Мы должны иметь в виду, что значения, введенные в хэш-функцию, должны поступать из одного и того же столбца (ключ осколка), чтобы данные размещались в правильном порядке и согласованно. В принципе, ключи осколков действуют как первичный ключ или уникальный идентификатор для отдельных строк.

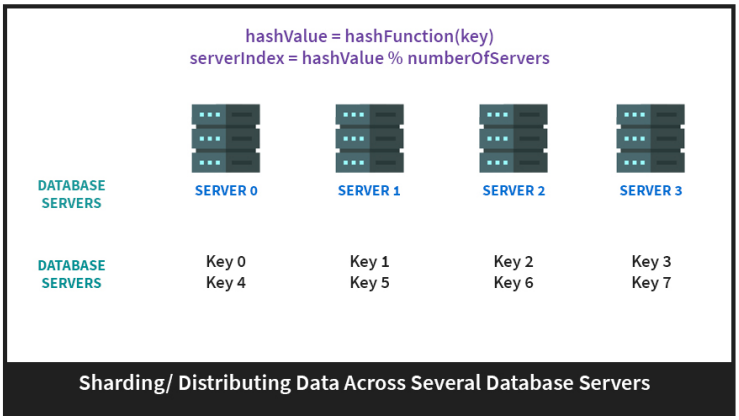


Рис. 14: Шардинг на основе ключей

Рассмотрим пример, что у вас есть 3 сервера баз данных, и каждый запрос имеет идентификатор приложения, который увеличивается на 1 каждый раз, когда регистрируется новое приложение. Чтобы определить, на каком сервере должны быть размещены данные, мы выполняем операцию по модулю для этих приложений id с номером 3. Затем остаток используется для идентификации сервера для хранения наших данных.

Недостатком этого метода является эластичная балансировка нагрузки, что означает, если вы попытаетесь динамически добавлять или удалять серверы баз данных, это будет сложный и дорогостоящий процесс. Например, в приведенном выше примере, если вы добавите еще 5 серверов, вам нужно добавить больше соответствующих хэш-значений для дополнительных записей. Кроме того, большинство существующих ключей необходимо переназначить на их новое, правильное хэш-значение, а затем перенести на новый сервер. Хэш-функция должна быть изменена с модуля 3 на модуль 8. В то время как миграция данных действует, как новые, так и старые хэш-функции не будут действительны. Во время миграции ваше приложение не сможет обслуживать большое количество запросов, и вы будете испытывать простои для своего приложения до завершения миграции. (anuupadhyay 2021)

Географический шардинг

Шардирование по географическому признаку позволяет хранить определенные данные вблизи своих потребителей и удовлетворять нормативным требованиям, когда данные должны находиться в определенной юрисдикции. Однако данный способ шардирования не является самодостаточным: шарды могут быть загружены не равномерно, и отличие может быть на порядки. На пример, экземпляр базы данных, хранящий данные пользователей Москвы и Московской области, будет намного превышать другой экземпляр базы данных, который хранит информацию о пользователях из Владимира. Таким образом, недостатком данного метода является необходимость дополнительного использования других методов шардирования.

6.5.2 Репликация

Репликация — это процесс изменения одного набора данных, называемого репликой, в ответ на

изменения другого набора данных, называемого основным. Репликация желательна по крайней мере по двум причинам. Во-первых, она способна обеспечить более высокую производительность, поскольку приложения смогут обрабатывать локальные копии вместо того, чтобы устанавливать связь с удаленными узлами. Во-вторых, наличие репликации может также обеспечивать более высокую степень доступности, поскольку любой реплицируемый объект остается доступным для обработки (по крайней мере, для выборки данных), пока хотя бы одна реплика в системе остается доступной. Главным недостатком репликации, безусловно, является то, что если реплицируемый объект обновляется, то и все его копии должны быть обновлены (проблема распространения обновлений).

Очевидно, что репликация, как и шардирование, теоретически должна быть "прозрачной для пользователя". Другими словами, система, которая поддерживает репликацию данных, должна также поддерживать независимость от репликации (иногда говорят "прозрачность репликации"). Для пользователей должна быть создана такая среда, чтобы они, по крайней мере, с логической точки зрения могли считать, что в действительности данные не дублируются. Независимость от репликации (как и независимость от шардирования) является весьма желательной, поскольку она упрощает создание пользовательских программ и выполнение терминальных операций. В частности, независимость от репликации позволяет создавать и уничтожать дубликаты в любой момент в соответствии с изменяющимися требованиями, не затрагивая при этом никакие из пользовательских программ или терминальных операций.

Из требования независимости от репликации следует, что к обязанностям системного оптимизатора также относится определение, какой именно из физических дубликатов будет применен для доступа к данным при выполнении каждого введенного пользователем запроса. (Дейт К. Дж. 2014)

Можно выделить три **подхода к репликации**:

- Блочная репликация на уровне системы хранения данных;
- Физическая репликация на уровне СУБД;
- Логическая репликация на уровне СУБД.

Блочная репликация

При блочной репликации каждая операция записи выполняется не только на основном диске, но и на резервном. Таким образом тому на одном массиве соответствует зеркальный том на другом массиве, с точностью до байта повторяющий основной том.

К достоинствам такой репликации можно отнести простоту настройки и надёжность. Записывать данные на удалённый диск может либо дисковый массив, либо нечто (устройство или программное обеспечение), стоящее между хостом и диском. Если дисковый массив не способен реплицировать данные, между хостом и массивом может быть установлен агент, осуществляющей запись на два массива сразу. Агент может быть как отдельным устройством, так и программным компонентом. В отличие от дискового массива, который может работать только с таким же массивом или, как минимум, с массивом того же производителя, агент может работать с совершенно разными дисковыми устройствами.

Главное назначение блочной репликации – обеспечение отказоустойчивости. Если база данных потеряна, то можно перезапустить её с использованием зеркального тома. Блочная репликация хороша своей универсальностью, но за универсальность приходится платить.

Во-первых, никакой сервер не может работать с зеркальным томом, поскольку его операционная система не может управлять записью на него; с точки зрения наблюдателя данные на зеркальном томе появляются сами собой. В случае аварии (отказ основного сервера или всего ЦОДа, где находится основной сервер) следует остановить репликацию, размонтировать основной том и смонтировать зеркальный том. Как только появится возможность, следует перезапустить репликацию в обратном направлении.

Во-вторых, сама СУБД на резервном сервере может быть запущена только после монтирования диска. В некоторых операционных системах, например, в Solaris, память под кеш при выделении размечается, и время разметки пропорционально объёму выделяемой памяти, то есть старт экземпляра будет отнюдь не мгновенным. Плюс ко всему кеш после рестарта будет пуст.

В-третьих, после запуска на резервном сервере СУБД обнаружит, что данные на диске неконсистентны, и нужно потратить значительное время на восстановление с применением журналов повторного выполнения: сначала повторить те транзакции, результаты которых сохранились в журнале, но не успели сохраниться в файлы данных, а потом откатить транзакции, которые к моменту сбоя не успели завершиться. (hard_sign 2020)

Физическая репликация (master-slave)

Журналы (redo log или write-ahead log) содержат все изменения, которые вносятся в файлы базы данных. Идея физической репликации состоит в том, что изменения из журналов повторно выполняются в другой базе (реплике), и таким образом данные в реплике повторяют данные в основной базе байт-в-байт.

Журналы СУБД не предназначены для использования вне этой платформы, их формат не документируется и может меняться без предупреждения. Отсюда совершенно естественное требование, что физическая репликация возможна только между экземплярами одной и той же версии одной той же СУБД. Отсюда же возможные ограничения на операционную систему и архитектуру процессора, которые тоже могут влиять на формат журнала.

Естественно, никаких ограничений на модели СХД физическая репликация не накладывает. Более того, файлы в базе-реплике могут располагаться совсем по-другому, чем на базе-источнике – надо лишь описать соответствие между томами, на которых лежат эти файлы.

Запись данных в реплику невозможна, поскольку изменения в неё приходят побайтно, и реплика не может обеспечить конкурентное исполнение своих запросов. В случае повреждения файла в основной базе можно просто скопировать соответствующий файл с реплики. Однако стоит учесть, что файл на реплике может быть не идентичен файлу в основной базе: когда файл расширяется, новые блоки в целях ускорения ничем не заполняются, и их содержимое случайно. База может использовать не всё пространство блока (например, в блоке может оставаться свободное место), но содержимое использованного пространства совпадает с точностью до байта.

Физическая репликация может быть как синхронной, так и асинхронной. При асинхронной репликации всегда есть некий набор транзакций, которые завершены на основной базе, но ещё не дошли до резервной, и в случае перехода на резервную базу при сбое основной эти транзакции будут потеряны. При синхронной репликации завершение операции commit означает, что все журнальные записи, относящиеся к данной транзакции, переданы на реплику. Важно понимать, что получение репликой журнала не означает применения изменений к данным. При потере основной базы транзакции не будут потеряны, но если приложение пишет данные в основную базу и считы-

вает их из реплики, то у него есть шанс получить старую версию этих данных.

Физическая репликация базы данных имеет множество преимуществ перед репликацией средствами СХД:

- объём передаваемых данных меньше за счёт того, что передаются только журналы, но не файлы с данными; эксперименты показывают уменьшение трафика в 5-7 раз;
- переключение на резервную базу происходит значительно быстрее: экземпляр-реплика уже поднят, поэтому при переключении ему нужно лишь откатить активные транзакции; более того, к моменту сбоя кеш реплики уже прогрет;
- на реплике можно выполнять запросы, сняв тем самым часть нагрузки с основной базы. В частности, реплику можно использовать для создания резервных копий.

Логическая репликация (active-active)
Все изменения в базе данных происходят в результате вызовов её API – например, в результате выполнения SQL-запросов. Очень заманчивой кажется идея выполнять одну и ту же последовательность запросов на двух разных базах. Для репликации необходимо придерживаться двух правил:

- нельзя начинать транзакцию, пока не завершены все транзакции, которые должны закончиться раньше; Так на рисунке ниже нельзя запускать транзакцию D, пока не завершены транзакции A и B;
- нельзя завершать транзакцию, пока не начаты все транзакции, которые должны закончиться до завершения текущей транзакции; Так на рисунке ниже даже если транзакция B выполнилась мгновенно, завершить её можно только после того, как начнётся транзакция C.

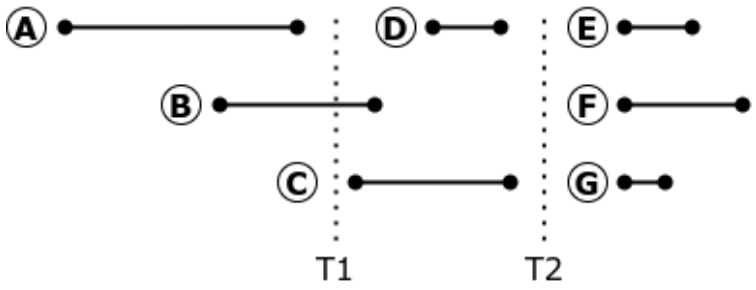


Рис. 15: Пример репликации

Репликация команд (statement-based replication) реализована, например, в MySQL. К сожалению, эта простая схема не приводит к появлению идентичных наборов данных – тому есть две причины.

- не все API детерминированы. Например, если в SQL-запросе встречается функция now() или sysdate(), возвращающая текущее время, то на разных серверах она вернёт разный результат – из-за того, что запросы выполняются не одновременно. Кроме того, к различиям могут привести разные состояния триггеров и хранимых функций, разные национальные настройки, влияющие на порядок сортировки, и многое другое.

- репликацию, основанную на параллельном исполнении команд, невозможно корректно приостановить и перезапустить. На рисунке выше если репликация остановлена в момент T1 транзакция B должна быть прервана и откачена. При перезапуске репликации исполнение транзакции B может привести реплику к состоянию, отличному от состояния базы-источника: на источнике транзакция B началась до того, как закончилась транзакция A, а значит, она не видела изменений, сделанных транзакцией A. Репликация запросов может быть остановлена и перезапущена только в момент T2, когда в базе нет ни одной активной транзакции. Разумеется, на сколько-нибудь нагруженной промышленной базе таких моментов не бывает.

Обычно для логической репликации используют детерминированные запросы. Детерминированность запроса обеспечивается двумя свойствами:

- запрос обновляет (или вставляет, или удаляет) единственную запись, идентифицируя её по первичному (или уникальному) ключу;
- все параметры запроса явно заданы в самом запросе.

В отличие от **репликации команд (statement-based replication)** такой подход называется **репликацией записей (row-based replication)**.

База-реплика открыта и доступна не только на чтение, но и на запись. Это позволяет использовать реплику для выполнения части запросов, в том числе для построения отчётов, требующих создания дополнительных таблиц или индексов. Важно понимать, что логическая реплика будет эквивалентна исходной базе только в том случае, если в неё не вносится никаких дополнительных изменений

Логическая репликация предоставляет ряд возможностей, отсутствующих в других видах репликации:

- настройка набора реплицируемых данных на уровне таблиц (при физической репликации – на уровне файлов и табличных пространств, при блочной репликации – на уровне томов);
- построение сложных топологий репликации – например, консолидация нескольких баз в одной или двунаправленная репликация;
- уменьшение объёма передаваемых данных;
- репликация между разными версиями СУБД или даже между СУБД разных производителей;
- обработка данных при репликации, в том числе изменение структуры, обогащение, сохранение истории.

Есть и недостатки, которые не позволяют логической репликации вытеснить физическую:

- все реплицируемые данные обязаны иметь первичные ключи;
- логическая репликация поддерживает не все типы данных;
- логическая репликация на практике не бывает полностью синхронной: время от получения изменений до их применения слишком велико, чтобы основная база могла ждать;
- логическая репликация создаёт большую нагрузку на реплику;

- при переключении приложение должно иметь возможность убедиться, что все изменения с основной базы, применены на реплике – СУБД зачастую сама не может этого определить, так как для неё режимы реплики и основной базы эквивалентны.

Два последних недостатка существенно ограничивают использование логической реплики как средства отказоустойчивости. Если один запрос в основной базе изменяет сразу много строк, реплика может существенно отставать. А возможность смены ролей требует недюжинных усилий как со стороны разработчиков, так и со стороны администраторов.

Есть несколько способов реализации логической репликации, и каждый из этих способов реализует одну часть возможностей и не реализует другую:

- репликация триггерами;
- использование журналов СУБД;
- использование программного обеспечения класса CDC (change data capture);
- прикладная репликация.

Репликация триггерами

Триггер – хранимая процедура, которая исполняется автоматически при каком-либо действии по модификации данных. Триггеру, который вызывается при изменении каждой записи, доступны ключ этой записи, а также старые и новые значения полей. При необходимости триггер может сохранять новые значения строк в специальную таблицу, откуда специальный процесс на стороне реплики будет их вычитывать

Преимущества:

- независимость от версий основной базы и реплики;
- широкие возможности преобразования данных.

Недостатки:

- нагрузка на основную базу;
- большая задержка при репликации.

Использование журналов СУБД

Сами СУБД также могут предоставлять возможности логической репликации. Источником данных, как и для физической репликации, являются журналы. К информации о побайтовом изменении добавляется также информация об изменённых полях, а также значение уникального ключа, даже если он не меняется. В результате объём журналов БД увеличивается – по разным оценкам от 10 до 15

К **недостаткам** данного подхода можно отнести увеличение объёма журналов и возможное увеличение трафика между узлами.

Использование CDC

Существует целый класс программного обеспечения, предназначенного для организации логической репликации. Это ПО называется CDC, change data capture. В задачу платформы входит чтение журналов базы данных, преобразование информации, передача информации на реплику и применение. Как и в случае репликации средствами самой СУБД, журнал должен содержать информацию об изменённых полях. Использование дополнительного приложения позволяет «на лету» выполнять сложные преобразования реплицируемых данных и строить достаточно сложные топологии репликации.

Преимущества:

- возможность репликации между разными СУБД, в том числе загрузка данных в отчётные системы;
- широчайшие возможности обработки и преобразования данных;
- минимальный трафик между узлами – платформа отсекает ненужные данные и может сжимать трафик;
- встроенные возможности мониторинга состояния репликации.

Недостатки:

- увеличение объёма журналов, как при логической репликации средствами СУБД;
- новое ПО – сложное в настройке и/или с дорогими лицензиями.

Прикладная репликация

Наконец, ещё один способ репликации – формирование векторов изменений непосредственно на стороне клиента. Клиент должен формировать детерминированные запросы, затрагивающие единственную запись. Добиться этого можно, используя специальную библиотеку работы с базой данных. Когда приложение завершает транзакцию, специально подключаемый модуль записывает вектор изменений в очередь и выполняет транзакцию в базе данных. Специальный процесс-репликатор вычитывает векторы из очереди и выполняет транзакции в базе-реплике.Этот механизм хорош для обновления отчётных систем. Может он использоваться и для обеспечения отказоустойчивости, но в этом случае в приложении должен быть реализован контроль состояния репликации

Преимущества:

- возможность репликации между разными СУБД, в том числе загрузка данных в отчётные системы;
- возможность обработки и преобразования данных, мониторинга состояния и т. д.;
- минимальный трафик между узлами – платформа отсекает ненужные данные и может сжимать трафик;
- полная независимость от базы данных – как от формата, так и от внутренних механизмов.

Недостатки:

- ограничения на архитектуру приложения;
- огромный объём собственного кода, обеспечивающего репликацию.

Сравнение подходов к репликации

Описав все подходы к репликации, можно установить следующее:

- **Блочная репликация** имеет смысл, когда других способов репликации нет; для баз данных её лучше не использовать.
- **Физическая репликация** хороша, когда требуется обеспечение отказоустойчивости инфраструктуры или перенос части читающих приложений на реплики.
- **Логическая репликация** подходит для обеспечения отказоустойчивости только в том случае, если приложение знает об этой репликации и умеет в случае аварии ждать синхронизации реплик.
- **Логическая репликация** идеальна для всевозможных отчётных баз.
- **Репликация триггерами** имеет смысл в том случае, если база сильно нагружена, а реплицировать нужно крайне ограниченное количество информации.
- **Платформы CDC** хороши, если у вас большое количество реплицируемых баз и/или есть необходимость сложных преобразований данных.
- Разработка **прикладной репликации** оправдана только в случае разработки собственной платформы или фреймворка.

(hard_sign 2020)

Greenplum

Ранее уже сравнивались различные подходы шардирования между собой. Сравнивались различные подходы репликации между собой. Осталось только сравнить способы тиражирования, то есть сравнить репликацию и шардирование.

Вообще говоря, шардинг и репликация не противоречат друг другу и могут сосуществовать. Они выполняют разные задачи, и сравнивать их не имеет смысла. Репликация используется для ускорения взаимодействия с бд с помощью использования копий основной базы данных. Кроме того, репликация осуществляет отказоустойчивость. В свою очередь шардинг осуществляет ускорение работы с базой данных с помощью разбиения исходной базы данных на несколько разных, хранящихся на разных серверах. Шардирование не реализует отказоустойчивость.

В принципе, на этом их сравнение можно закончить и перейти к примеру. Один из классических примеров в котором реализовано сочетание репликации и шардирования - **Greenplum**.

Greenplum (GP) – реляционная СУБД, имеющая массово-параллельную (massive parallel processing) архитектуру без разделения ресурсов (Shared Nothing). В общем случае кластер GP состоит из нескольких серверов-сегментов (именно сегменты непосредственно хранят данные, выполняют с ними операции и отдают результаты мастеру (в общем случае). По сути сегмент – самый обычный инстанс PostgreSQL 8.2.15 с настроенной логической репликацией в своё зеркало на другом сервере), одного сервера-мастера (сервер, на котором работает инстанс, являющийся одновременно координатором и входной точкой для пользователей в кластере), и одного сервера-секондари-мастера (инстанс, являющийся резервным мастером, включается в работу в случае недоступности основного мастера (переключение происходит вручную)), соединённых между собой одной или несколькими

быстрыми (10g, infiniband) сетями, обычно обособленными (interconnect). На рисунке ниже представлен состав кластера и сетевое взаимодействие элементов. Здесь — зелёная и красная линии — обособленные сети interconnect, синяя линия — внешняя, клиентская сеть.

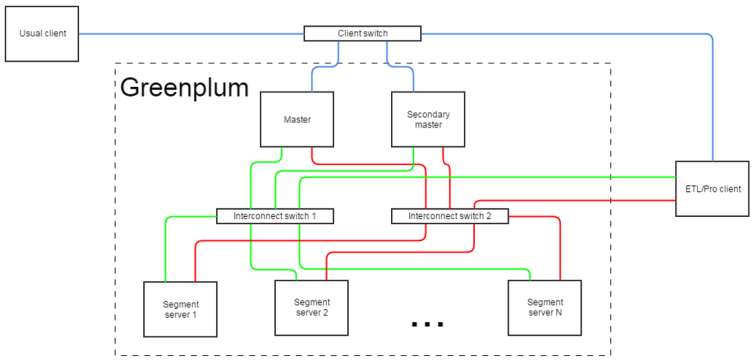


Рис. 16: Состав кластера и сетевое взаимодействие элементов Greenplum

При выборе числа серверов-сегментов важно правильно выбрать соотношение кластера «число процессоров/Тб данных» в зависимости от планируемого профиля нагрузки на БД — чем больше процессорных ядер приходится на единицу данных, тем быстрее кластер будет выполнять «тяжёлые» операции, а также работать со сжатыми таблицами.

При выборе числа сегментов в кластере (которое в общем случае к числу серверов никак не привязано) необходимо помнить следующее:

- все ресурсы сервера делятся между всеми сегментами на сервере (нагрузкой зеркал, в случае если они располагаются на этих же серверах, можно условно пренебречь);
- каждый запрос на одном сегменте не может потреблять процессорных ресурсов больше, чем одно ядро CPU. Это означает, например, что, если кластер состоит из 32-ядерных серверов с 4-я сегментами GP на борту и используется в среднем для обработки 3-4 одновременных тяжёлых, хорошо утилизирующих CPU, запросов, «в среднем по больнице» CPU не будет утилизироваться оптимально. В данной ситуации лучше увеличить число сегментов на сервере до 6-8;
- штатный процесс бекапа и рестора данных «из коробки» работает только на кластерах, имеющих одинаковое число сегментов. Восстановить данные, забекапленные на кластере из 96 сегментов, в кластер из 100 сегментов без напильника будет невозможно.

В Greenplum реализуется классическая схема шардирования данных. Каждая таблица представляет из себя N+1 таблиц на всех сегментах кластера, где N – число сегментов (+1 в этом случае — это таблица на мастере, данных в ней нет). На каждом сегменте хранится 1/N строк таблицы. Логика разбиения таблицы на сегменты задаётся ключом (полем) дистрибуции – таким полем, на основе данных которого любую строку можно отнести к одному из сегментов.

По подробнее почитать про Greenplum можно в источнике. В данной главе приведена только краткая информация об общей архитектуре и информация, относящаяся к тиражированию данных. (kapustor 2015)

6.6 Бесконфликтные реплицированные типы данных

Проблема репликации Предположим, реплики СУБД распределены по географическому признаку для ускорения работы приложения. А теперь пусть в разных репликах практически одновременно и независимо была изменена одна и та же строка. Какую строку считать правильно отражающей данные? Как восстановить согласованность между репликами? В общем случае, проблема одновременного обновления реплик не может быть разрешима. Поэтому большинство распределенных СУБД запрещают производить такие операции, например используя только один сервер для выполнения запросов. Однако существует некоторый класс структур данных, которые позволяют автоматически разрешать конфликты возникающие в процессе обновления нескольких реплик. Этот класс называется - бесконфликтные реплицированные типы данных или conflict-free replicated data types(CRDT).

CRDT. Опеределение и виды CRDT обладают следующими характеристиками:

- Приложение может обновлять реплики независимо, параллельно с другими репликами
- Алгоритм(как часть CRDT) автоматически разрешает возможные конфликты
- Данные в разных репликах могут быть в различных состояниях в один и тот же момент времени, но они гарантированно сойдутся спустя некоторое время

Просто используя специализированные структуры данных было бы сложно добиться бесконфликтности, поэтому для ее достижения используются также специальные алгоритмы обновления данных. Алгоритмы определяют требования к структурам данных, а также могут накладывать дополнительные требования на сети передачи данных. Рассмотрим два основных вида алгоритмов:

- Operation-based CRDTs. При обновлении данных, на реплике генерируется специальная функция обновления, которая затем рассылается всем другим нодам в сети. Каждая нода применяет эту функцию к своим данным и таким образом достигается консистентность. Важно, чтобы каждая функция была лишь раз применена на каждой ноде. Поэтому необходимо следить за тем, чтобы доставка гарантированно произошла и произошла только один раз. Связано это с тем, что функции могут быть коммутативными, но далеко не факт, что они будут идемпотентными(применение несколько раз подряд дает разные результаты).
- State-based CRDTs. При обновлении данных, репликам посылается полная локальная копия данных, после чего реплики локально у себя сливают эти изменения воедино, причем операция слияния (merge) обладает свойствами ассоциативности, коммутативности и идемпотентности, что позволяет уменьшить требования к каналам связи между репликами.
- Delta-state CRDTs - оптимизированный вариант state-based CRDTs, когда посылаются только недавно примененные изменения вместо полной копии состояния.

Помимо алгоритмов, важную роль играют процессы сходимости между нодами. Для сходимости в случае Operation-based CRDTs, каждая нода обязана получить ровно одно сообщение с фукнцией обновления и для этого необходим надежный протокол доставки. В случае с State-base CRDTs, наличие коммутативного и идемпотентного оператора слияния позволяет утверждать, что данные постепенно сойдутся в любом случае, что дает нам право не беспокоиться о протоколе доставки.

Примеры реализации На сегодняшний день известных CRDT не так много, одни из них: G-Counter (Grow only counter), PN-Counter (positive-negative counter), G-Set (grow only set) и несколько других типов. Основной особенностью всех этих типов данных является, то что каждая коллекция поддерживает узкий набор операций. А чтобы добавить например вычитание (некоммутативную операцию) в PN-Counter необходимо прибегать к хитрости и использовать два ворастающих счетчика типа G-Counter. Рассмотрим пример реалиции G-Counter-a.

```
payload integer[n] P
    initial [0,0,...,0]
update increment()
    let g = myId()
    P[g] := P[g] + 1
query value() : integer v
    let v = Σ_i P[i]
compare (X, Y) : boolean b
    let b = (∀i ∈ [0, n - 1] : X.P[i] ≤ Y.P[i])
merge (X, Y) : payload Z
    let ∀i ∈ [0, n - 1] : Z.P[i] = max(X.P[i], Y.P[i])
```

Рис. 17: Пример реализации Grow only Counter-a

Этот State-based счетчик сделан для кластера из n нод. Каждой ноде присвоен id, который можно получить вызвав функцию myid(). Таким образом, каждой ноде присваивается свой слот в массиве P, который нода может инкрементировать локально. Обновления расходятся по сети и при слиянии вычисляется максимум для каждого слота. При запросе значения счетчика все значения в слотах вектора суммируются и возвращаются в качестве ответа. Функция слияния является идемпотентной и коммутативной, поэтому данный счетчик является state-based. Использование CRDT позволяет существенно упростить жизнь для разработчиков СУБД и облегчить работу с репликами. Уже сейчас CRDT используется во многих крупных проектах. В их число входит: Redis, Riak, Apple, Facebook. В дальнейшем использование CRDT будет только увеличиваться.

6.7 Интеграция БД и Internet

Современные тенденции Заключаются в том, чтоб забыть о физических машинках и виртуалках, завернуть сервисы в контейнеры и перенести их в облако.

Обзор существующих технологий То немногое, из чего действительно полезно почитать хотя бы описание инструментов по диагонали.

Docker

Docker — средство виртуализации и менеджмента программ (сервисов), позволяющее гибко настраивать инфраструктуру проектов, и кратно облегчающее разработку, тестирование и масштабируемый деплой. Архитектура у Docker клиент-серверная, что означает наличие демона dockerd и клиентов docker, отправляющих демону команды (собери, скачай, запусти, пр.). Если вам нужно на коленке запустить оркестрацию (операции “возьми n нод, запусти на них k заданий (образов Docker)

и проследи, чтоб выполнились"), можно использовать docker swarm. (*Swarm mode key concepts* 2022) В проде такое лучше не использовать, потому что ноды приходится создавать ручками; иными словами, если у вас есть 2 машинки с разным количеством ресурсов каждая, придётся собственными силами подстраивать количество запускаемых на машинке нод, подстраиваясь под потребляемые ресурсы. Если требуется динамическое масштабирование, лучше посмотреть в сторону K8s.

Контейнеры

Основная идея — завернуть необходимый сервис (бинарники, скрипты, данные, конфиги) в легковесный **контейнер**, который далее будет запускаться на произвольном устройстве, способном запускать 64-битный Linux (Windows и macOS под капотом запускают сначала виртуалку с Linux, и лишь на ней крутят Docker). Такой подход с упаковкой всего необходимого в контейнер позволяет не волноваться о том, что на хосте (устройство, где контейнер запущен) будет недоставать пакетов, возникнут проблемы с совместимостью и тому подобное.

Изолированность

Контейнеры из коробки являются изолированными сущностями — процессы, запущенные в одном контейнере не смогут влиять и даже смотреть на то, что запущенно в другом контейнере или на хосте, а для получения таковой функциональности придётся приложить дополнительные усилия. Также для каждого контейнера заводится свой сетевой стек. Обеспечивается такая изоляция средствами ядра Linux, а именно — **пространствами имён** (namespaces) и **контрольными группами** (control group). Первые отвечают за то, чтоб котнейнеры не подглядывали друг за другом; а вторые распределяют и ограничивают используемые контейнерами ресурсы хоста, гарантируют не только то, что контейнеры не будут голодать по памяти, CPU, дисковым операциям, но и то, что контейнеры не отберут все ресурсы у других потребителей.

Атаки на dockerd

Демона dockerd по умолчанию (можно запустить и в rootless режиме) требует root привилегий. Например, это требуется для пробрасывания директории хоста в контейнер. В связи с чем следует предоставлять управление демоном только тем пользователям, которым доверяешь. В пример можно привести атаку, в ходе которой корень файловой системы хоста примонтируется в директорию внутри контейнера, к которой у постороннего пользователя будет доступ. Для защиты от этого Docker использует не TCP сокеты, а UNIX сокеты, на которые можно поставить стандартные проверки прав доступа UNIX. Также существует атака с подменой образа. Например, подменить образ, который позднее будет загружен через `docker load` (локально) или `docker pull` (по сети), однако современные версии докера сверяют хэш-суммы образов, что сильно усложняет эксплуатацию уязвимости. (*Docker security* 2022)

При использовании dockerd на устройстве рекомендуется все сервисы, запущенные на том же устройстве, тоже разнести по контейнерам.

Как можно доверять загружаемым образам

Часть Docker, позволяющая верифицировать целостность образов и их авторов путём проверки подписей называется Docker Content Trust (DCT). Сами образы при этом хранятся в Docker registry (DockerHub — пример общедоступного registry). В репозитории (например ubuntu, mongo), где хранятся образы, создаётся набор ключей, которыми автор может подписывать по желанию теги этого репозитория, при это можно даже выпустить две версии одного тэга: подписанную и неподписанную. Пользователь репозитория может фильтровать теги доступные к загрузке — можно запретить использование неподписанных тегов.

Доверие достигается следующим образом. У автора есть оффлайн-ключ (рутовый) и сертификат, с помощью которого создаются ключи и сертификаты тега (delegation keys) (обладание такими ключами позволяет вливать новые образы в репозиторий). (*Content trust in Docker* 2022) У ползователя же на машинке лежит CA сертификат, которым подписан registry сертификат, а также собственный сертификат пользователя и ключ (последние нужны для того, чтоб registry мог верифицировать пользователя). (*Verify repository client with certificates* 2022)

PKI в docker swarm

Ноды в swarm'е используют TLS для аутентификации, авторизации и шифрования соединения с другими нодами. Когда созадётся управляющая нода (manager), она генерирует CA сертификат и ключевую пару для дальнейшего общения с остальными. Также генерируются два токена — для добавляемых в swarm нод-работников и нод-управляющих. Токен является комбинацией открытого ключа сертификата и секрета. Открытый ключ используется подключаемой нодой для валидации сертификата управляющей ноды, а секрет используется управляющей нодой для валидации подключаемой ноды. Далее управляющая выдаёт подключённой ноде новый сертификат, подписанный CA сертификатом. Таким образом устанавливается PKI, и далее ноды могут проверять, действительно ли с ними общаются разрешённые ноды. (*Manage swarm security with public key infrastructure (PKI)* 2022)

Секреты в docker swarm

Когда в кластер добавляется секрет, он попадает в главную управляющую ноду с использованием TLS, далее реплицируется на остальные управляющие ноды, где хранится в зашифрованном логе Raft (используется для установления консенсуса между управляющими нодами, любознательные могут глянуть анимацию). Далее можно выдавать сервисам права на использование секретов, в таком случае секреты расшифровываются и монтируются в файловую систему контейнеров. Обновление/удаление/добавление секрета инициирует обновление сервиса, поэтому ротацию секретов придётся проводить в несколько шагов: добавить новый секрет, переключить сервис на его использование, удалить старый секрет. (*Manage sensitive data with Docker secrets* 2022)

Kubernetes

Kubernetes — инструмент, умеющий запускать контейнеры на множестве хостов, следить за их состоянием, а также обновлять запущенные версии контейнеров, используя заданную политику (например, задача "обнови реплики сервиса по очереди, переключая каждую из них лишь по завершении скриптов запуска и успешной проверки работоспособности"), откатывать сервис до одной из сохранённых версий. Но самое приятное — он умеет следить за количеством ресурсов на хостах и сам масштабирует сервис, понимая, можно ли добавить в него контейнер, или лучше удалить, чтоб остальные не голодали. Также K8s умеет производить обнаружение сервисов (хранить и выдавать при необходимости маршрут до сервиса), балансировку нагрузки, перезапуск контейнеров по триггеру, оркестрацию хранилищ, менеджмент секретов. В сравнении с Docker K8s более трудозатратно настраивается, но и возможностей у него побольше. Из явных отличий — уже упомянутые автомасштабирование и перезапуск сервисов в случае их выхода из строя. Да, можно и в swarm этого добиться, но зачем изобретать велосипед?

Из чего состоит

Сущность, получаемая после настройки K8s, называется кластером. Кластер состоит из набора нод-рабочих (worker node) (минимум одна на кластер), которые запускают у себя контейнеры. На этих

нодах запускается поды (pods). Управляют всем отдельные ноды (control plane). В проде обычно запущено несколько управляющих нод для отказоустойчивости. (*Kubernetes Components* 2022)

Архитектуры web-приложений

Как правило компьютеры и программы, входящие в состав информационной системы, не являются равноправными. Некоторые из них владеют ресурсами (файловая система, процессор, принтер, база данных и т.д.), другие имеют возможность обращаться к этим ресурсам. Компьютер (или программу), управляющий ресурсом, называют сервером этого ресурса (файл-сервер, сервер базы данных, вычислительный сервер...). Клиент и сервер какого-либо ресурса могут находится как на одном компьютере, так и на различных компьютерах, связанных сетью.

Архитектура информационной системы(приложения) - концепция, определяющая модель, структуру, выполняемые функции и взаимосвязь компонентов информационной системы(приложения). Компоненты информационной системы по выполняемым функциям можно разделить на три слоя: слой представления, слой бизнес-логики и слой доступа к данным.

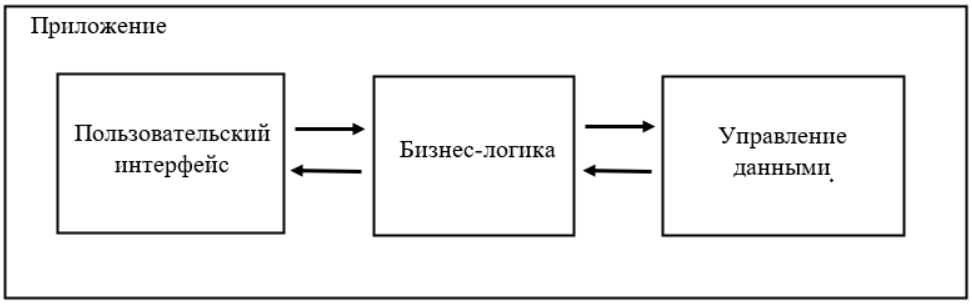


Рис. 18: Три основных компонента сетевого приложения

Слой представления - все, что связано с взаимодействием с пользователем: нажатие кнопок, движение мыши, отрисовка изображения, вывод результатов поиска и т.д.

Бизнес логика - правила, алгоритмы реакции приложения на действия пользователя или на внутренние события, правила обработки данных.

Слой доступа к данным - хранение, выборка, модификация и удаление данных, связанных с решаемой приложением прикладной задачей.

С точки зрения программно-аппаратной реализации можно выделить ряд типовых архитектур ИС.

Сравнение и описание архитектур представлено в соответствии с (*Различные архитектурные решения, используемые при реализации многопользовательских СУБД. Краткий обзор СУБД* 2012).

Двухзвенная структура

В любой сети, построенной на современных сетевых технологиях, присутствуют элементы клиент-серверного взаимодействия,часто на основе двухзвенной архитектуры.

Двухзвенная архитектура используется в клиент-серверных системах, где сервер отвечает на клиентские запросы напрямую и в полном объеме, при этом используя только собственные ресурсы. Т.е. сервер не вызывает сторонние сетевые приложения и не обращается к сторонним ресурсам для выполнения какой-либо части запроса.

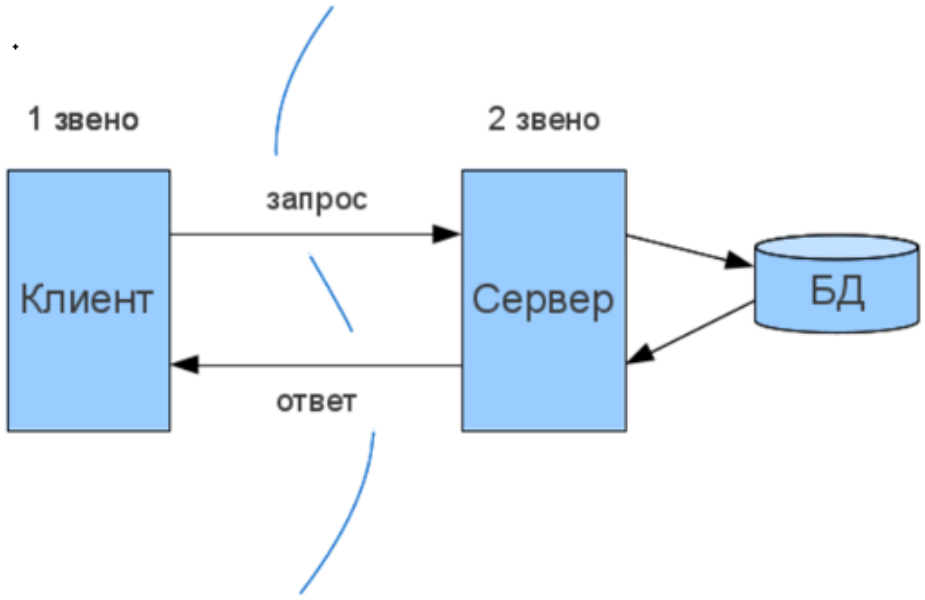


Рис. 19: Двухзвенная архитектура

Рис. 20: Типичный пример двухзвенной модели

Клиентская программа работает с данными через запросы к серверному ПО. Базовые функции приложения разделены между клиентом и сервером.

Плюсы:

- Отсутствие дублирования кода программы-сервера программами-клиентами.
- Так как все вычисления выполняются на сервере, то требования к компьютерам, на которых установлен клиент, снижаются.
- Все данные хранятся на сервере, который, как правило, защищён гораздо лучше большинства клиентов. На сервере проще обеспечить контроль полномочий, чтобы разрешать доступ к данным только клиентам с соответствующими правами доступа.
- Позволяет объединить различные клиенты. Использовать ресурсы одного сервера часто могут клиенты с разными аппаратными платформами, операционными системами и т. п.
- Позволяет разгрузить сети за счёт того, что между сервером и клиентом передаются небольшие порции данных.

Минусы:

- Бизнес логика приложений осталась в клиентском ПО. При любом изменении алгоритмов, надо обновлять пользовательское ПО на каждом клиенте.
- Высокие требования к пропускной способности коммуникационных каналов с сервером, что препятствует использование клиентских станций иначе как в локальной сети.
- Слабая защита данных от взлома, в особенности от недобросовестных пользователей системы.
- Высокая сложность администрирования и настройки рабочих мест пользователей системы.
- Необходимость использовать мощные ПК на клиентских местах.
- Высокая сложность разработки системы из-за необходимости выполнять бизнес-логику и обеспечивать пользовательский интерфейс в одной программе.

Трехзвенная архитектура

Еще одна тенденция в клиент-серверных технологиях связана со все большим использованием распределенных вычислений. Они реализуются на основе модели сервера приложений, где сетевое приложение разделено на две и более частей, каждая из которых может выполняться на отдельном компьютере. Выделенные части приложения взаимодействуют друг с другом, обмениваясь сообщениями в заранее согласованном формате. В этом случае двухзвенная клиент-серверная архитектура становится трехзвенной. На этой архитектуре построены большинство современных web-приложений.

Основным ее отличием от предыдущей архитектуры является физическое разделение программ, отвечающих за хранение данных (СУБД) от программ эти данные обрабатывающих. Такое разделение программных компонент позволяет оптимизировать нагрузки как на сетевое, так и на вычислительное оборудование комплекса.

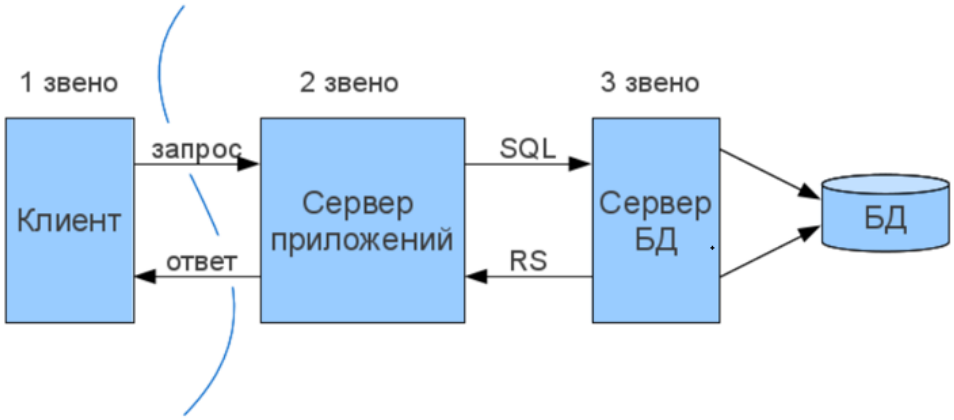


Рис. 21: Трехзвенная архитектура

Рис. 22: Типичный пример трехзвенной модели

Компоненты трехзвенной архитектуры, с точки зрения программного обеспечения реализуют определенные сервера БД, web-сервера и браузеры. Место любого из этих компонентов может занять программное обеспечение любого производителя.

Сервер приложений располагается на выделенном сервере приложений, выполняющем функции промежуточного ПО.

Плюсы:

- Тонкий клиент.
- Между клиентской программой и сервером приложения передается лишь минимально необходимый поток данных - аргументы вызываемых функций и возвращаемые от них значения.

- Сервер приложения может быть запущен в одном или нескольких экземплярах на одном или нескольких компьютерах.
- Дешевый трафик между сервером приложений и СУБД. Трафик между сервером приложений и СУБД может быть большим, однако это всегда трафик локальной сети, а их пропускная способность достаточно велика и дешева. В крайнем случае, всегда можно запустить СП и СУБД на одной машине, что автоматически сведет сетевой трафик к нулю.
- Дешевле наращивать функциональность и обновлять ПО.

Минусы:

- Более высокая сложность создания приложений.
- Сложнее в разворачивании и администрировании.
- Высокие требования к производительности серверов приложений и сервера базы данных, а, значит, и высокая стоимость серверного оборудования.
- Высокие требования к скорости канала (сети) между сервером базы данных и серверами приложений.

Отличия

По сравнению с двухзвенной клиент-серверной архитектурой трёхуровневая архитектура обеспечивает, как правило, большую масштабируемость (за счёт горизонтальной масштабируемости сервера приложений и мультиплексирования соединений), большую конфигурируемость (за счёт изолированности уровней друг от друга). Реализация приложений, доступных из веб-браузера или из тонкого клиента, как правило, подразумевает развёртывание программного комплекса в трёхуровневой архитектуре. При этом обычно разработка трёхзвенных программных комплексов сложнее, чем для двухзвенных, также наличие дополнительного связующего программного обеспечения может налагать дополнительные издержки в администрировании таких комплексов.

Авторизация

В распределенных системах регистрацией, идентификацией/аутентификацией занимается сервис, реализующий СУБД. Приложения не имеют прямого доступа к данным пользователя в БД, а обращаются к СУБД, которая возвращает данные, не позволяющие скомпрометировать пользователя. Для решения таких задач существуют стандарты идентификации. Самые распространенные из них это OAuth 2.0 (D. Hardt 2012a), OpenID Connect (Sakimura 2013).

С помощью OAuth 2.0 пользователь разрешает определенному сайту получить свои закрытые данные из соцсетей, но без передачи сайту своих логинов / паролей. Например, когда вы регистрируетесь на сайте через Facebook, то как раз и предоставляете этому сайту разрешение получить из Facebook ваше имя, e-mail адрес и другие закрытые данные.

Стандарт определяет следующие роли:

- Resource Owner — пользователь, который заходит на Сайт и дает ему разрешение использовать свои закрытые данные из Соцсети.
- Client (он же Сайт) — приложение или интернет сайт, которым пользуется пользователь и которое взаимодействует с Authorization Server и Resource Server для получения закрытых данных пользователя.
- Authorization Server — сервер который проверяет логин/пароль пользователя, он же Соцсеть.
- Resource Server — хранит закрытую пользовательскую информацию, которую можно получить с помощью API. Authorization Server и Resource Server могут быть совмещены в одну систему.(D. Hardt 2012b)

Теперь сам процесс. Детали конкретных реализаций могут различаться, но общая логика будет всегда следующая:

- Resource Owner заходит на Client, выбирает опцию “войти с помощью Соцсети”, сайт перенаправляет пользователя в Соцсеть на Authorization Server.
- Authorization Server проверяет есть ли у пользователя активная сессия и, если нет, то показывает форму для логина.
- Resource Owner вводит свои логин/пароль и подтверждает, что определенные закрытые данные могут быть использованы Сайт, например имя пользователя или e-mail адрес.
- Authorization Server проверяет пользователя и перенаправляет на адрес Callback с результатом аутентификации и “Authorization Code”
- В ответ Client посылает “Authorization Code”, Client ID и Client Secret.
- Authorization Server проверяет присланные данные и формирует “access token” в формате JWT (JSON Web Token), подписанный своим приватным ключом. В этом же JWT может содержаться и “refresh token”, с помощью которого возможно восстановление сессии после ее окончания.
- После этого Client может запросить закрытую информацию пользователя с помощью вызова API, в который передается “access token”.
- Resource Server проверяет “access token” (например, используя открытый ключ Authorization Server) и предоставляет доступ к данным пользователя.(D. Hardt 2012c)

OpenID Connect является надстройкой над OAuth 2.0:

- С помощью OAuth 2.0 выполняется только авторизация пользователя, т.е. о пользователе мы знаем только access token, с помощью которого можем получать определенные данные из Соцсети. Но access token ничего не говорит о личности пользователя и с помощью него мы не можем предоставить доступ к закрытым данным пользователя на нашем Сайте. OpenID Connect — добавляет сведения о логине и профиле пользователя (identity). Это как раз и позволяет реализовать его аутентификацию.

- OpenID Connect также добавляет возможность динамической регистрации и обнаружения сервисов “service discovery”. Это дает возможность строить системы SSO (Single Sign-On), в которых используется один логин для многих не связанных между собой сервисов.(Sakimura 2013)

OIDC расширяет OAuth 2.0 следующими основными возможностями:

- Authorization Server, помимо access token и refresh token, возвращает “identity token” (ID Token). Он содержится в том же JWT. Из ID Token можно извлечь следующую информацию: имя пользователя, время входа в учетную запись, срок окончания действия ID Token. ID Token можно передавать между участниками.
- OIDC предоставляет дополнительное API, которые позволяет запрашивать информацию о пользователе и его текущих сессиях.(D. Hardt 2013)

Диаграмма взаимодействия в OpenID Connect выглядит так же, как и в случае OAuth. Единственная разница в содержимом запросов:

- В первичном запросе на получение code добавляется дополнительный атрибут scope=openid.
- В результате работы алгоритма Client, помимо access и refresh token, получает ID Token.(D. Hardt 2013)

7 Безопасность в статистических БД

7.1 Определение статистической базы данных

Статистическая база данных — это специализированный тип базы данных, который разработан и оптимизирован для хранения и управления статистическими данными. Она представляет собой совокупность таблиц, где каждая таблица содержит переменные (столбцы), представляющие различные характеристики или атрибуты, и наблюдения (строки), представляющие отдельные единицы данных или объекты. Статистические базы данных часто содержат большие объемы данных, собранных из различных источников, таких как опросы, эксперименты, наблюдения и административные записи. Такие базы данных могут содержать разнообразные типы данных, включая числовые, категориальные, временные ряды и многие другие. Они могут быть созданы и использованы для различных целей, таких как проведение исследований, анализ данных, подготовка отчетов, прогнозирование и принятие решений.

Примеры статистических баз данных включают базы данных, содержащие данные о населении, экономические данные, данные о заболеваемости, результаты опросов и т.д. Эти базы данных могут быть созданы и поддерживаться различными организациями, такими как статистические агентства, исследовательские институты, университеты и государственные учреждения.

Статистические базы данных обычно имеют стандартизированную структуру и формат, чтобы обеспечить согласованность и удобство использования данных для статистического анализа. Они

также могут включать механизмы для обновления данных, контроля качества данных и обеспечения конфиденциальности и безопасности информации.

Основные характеристики статистических баз данных включают:

1. **Структуру данных:** статистические базы данных имеют четко определенную структуру, которая организует данные в таблицы, где каждая таблица представляет конкретную сущность данных или переменную. Структура включает поля (столбцы), представляющие различные атрибуты или характеристики, и записи (строки), представляющие отдельные экземпляры данных.

2. **Интеграцию данных:** статистические базы данных могут интегрировать данные из нескольких источников для создания комплексного набора данных для анализа. Этот процесс интеграции включает сопоставление и преобразование данных для обеспечения согласованности и совместимости.

3. **Качество данных:** обеспечение качества данных имеет важное значение в статистических базах данных. Они используют механизмы проверки данных, их очистки и стандартизации для минимизации ошибок, несогласованностей и пропущенных значений, которые могут повлиять на достоверность статистического анализа.

4. **Метаданные:** статистические базы данных часто включают метаданные, которые содержат информацию о данных, такую как определения переменных, источники данных, методы сбора данных и любые преобразования, примененные к данным.

5. **Политики безопасности и конфиденциальности:** статистические базы данных работают с конфиденциальными данными, такими как информация о персональных данных, банковские данные, бухгалтерские ведомости и подобными, и поэтому должны иметь меры безопасности для защиты конфиденциальности. Часто используются контроль доступа и методы анонимизации для обеспечения конфиденциальности данных.

Статистические базы данных служат ценным ресурсом для исследователей, аналитиков и принимающих решения. Они позволяют выполнять различные статистические операции, такие как агрегирование, корреляция, регрессия и проверка гипотез, для выявления паттернов, взаимосвязей и тенденций в данных.

Статистические базы данных позволяют проводить анализ данных, отвечать на исследовательские вопросы и принимать информированные решения на основе фактов. Они являются основой для множества статистических исследований, позволяя исследователям изучать различные явления, разрабатывать статистические модели и предсказывать будущие события.

Примеры статистических баз данных могут включать базы данных национальных статистических агентств, таких как Федеральное агентство по статистике (Росстат) или Бюро экономического анализа (BEA) в США. Эти базы данных содержат широкий спектр статистической информации о населении, экономике, здравоохранении, социальных показателях и других аспектах жизни.

Важно отметить, что статистические базы данных могут иметь различные спецификации и структуры, в зависимости от конкретной области знаний и целей использования данных.

7.2 Классификация статистических баз данных

Статистические базы данных можно классифицировать по различным критериям. Вот несколько основных классификаций статистических баз данных:

1. **По источнику данных:**
   - Государственные статистические базы данных: это базы данных, содержащие статистическую информацию, собранную и поддерживаемую государственными органами. Они могут включать данные о населении, экономике, здравоохранении, образовании и других социально-экономических аспектах.
   - Исследовательские базы данных: это базы данных, созданные исследователями для хранения и анализа статистических данных, полученных в результате научных исследований. Они могут содержать данные, собранные из различных источников, таких как опросы, эксперименты или наблюдения.
2. **По типу данных:**
   - Кросс-секционные базы данных: это базы данных, содержащие данные, собранные в определенный момент времени для различных наблюдаемых единиц. Например, база данных, содержащая информацию о доходах и образовании разных людей в определенном году.
   - Панельные базы данных: это базы данных, содержащие данные, собранные для одной и той же выборки наблюдаемых единиц в разные моменты времени. Например, база данных, содержащая информацию о доходах и образовании одной и той же группы людей в разные годы.
3. **По области применения:**
   - Социально-экономические базы данных: это базы данных, содержащие статистическую информацию о социальных и экономических аспектах общества, таких как занятость, безработица, инфляция, ВВП, образование и здравоохранение.
   - Медицинские базы данных: это базы данных, содержащие медицинскую статистическую информацию, такую как данные о заболеваемости, смертности, лекарственных препаратах и медицинских процедурах.
   - Экологические базы данных: это базы данных, содержащие статистическую информацию о состоянии окружающей среды, такую как данные о загрязнении воздуха, воды, климатических изменениях и биоразнообразии.
   - Демографические базы данных: это базы данных, содержащие статистическую информацию о населении, такую как данные о рождаемости, смертности, миграции, возрастной структуре и семейном составе.

- Банковские базы данных — это специальные базы данных, содержащие информацию о клиентах банка, включая их личные данные, счета, историю транзакций, кредитную историю клиентов, данных банковских рисков и аналитики.

4. **По типу взаимодействия:**

   - Онлайн-базы данных: в онлайн-режиме пользователь имеет прямое взаимодействие с базой данных в режиме реального времени. Он может выполнять запросы, получать результаты и обрабатывать данные непосредственно через терминал или интерфейс. Этот тип базы данных широко используется в банковской индустрии для онлайн-банкинга, систем платежей, интернет-магазинов и других сферах, где требуется мгновенное взаимодействие с данными.
   - Офлайн-базы данных: в отличие от онлайн-режима, в офлайн-режиме пользователь не контролирует обработку данных и не знает, когда выполняется его запрос данных. Вместо этого, пользователь предоставляет запросы или задания на обработку данных, а затем ожидает получения результатов. Такой режим часто используется в больших организациях, где запросы данных обрабатываются на серверах или вычислительных кластерах, и результаты предоставляются позднее. В этом режиме методы защиты, отслеживающие профили пользователей, становятся более громоздкими. Компромиссные методы, требующие большого количества запросов также усложняются при работе в автономном режиме.

5. **По возможности изменения базы данных**

   - Статическая база данных: статическая база данных не меняется после ее создания. Она остается неизменной со временем. Примером статической базы данных может служить база данных переписи, которая фиксирует данные о населении на определенный момент времени. Всякий раз, когда создается новая версия базы данных, эта новая версия рассматривается как отдельная статическая база данных. В статической базе данных изменения данных не происходят, она используется для анализа и исследования в определенный период времени.
   - Динамическая база данных: в отличие от статической базы данных, динамическая база данных может изменяться непрерывно. Она отражает изменения в данных со временем и обновляется в реальном времени. Примером динамической базы данных может служить база данных банка, которая содержит информацию о транзакциях клиентов. Новые записи добавляются, а существующие данные могут изменяться или удаляться в зависимости от действий пользователей. Динамические базы данных обеспечивают актуальность данных и позволяют оперативно реагировать на изменения, но также требуют более сложных методов безопасности.

6. **По типу централизованности**

   - Централизованная база данных: В централизованной статистической базе данных существует одна единственная база данных, которая хранит все данные. Эта база данных может располагаться на одном сервере или в одном месте, и все пользователи получают доступ к данным через эту централизованную точку доступа. Централизованная база

данных обеспечивает удобство управления и контроля над данными, поскольку все данные находятся в одном месте.

- Децентрализованная база данных: В децентрализованной статистической базе данных подмножества данных хранятся на различных узлах, которые связаны между собой сетью. Это может быть полностью реплицированная, частично реплицированная или секционированная база данных. Каждый узел содержит только определенную часть данных и обслуживает определенную группу пользователей или задач. Децентрализованная база данных позволяет более эффективное использование ресурсов и повышает отказоустойчивость системы.

7.3 Безопасность статистических баз данных

В безопасности статистических баз данных могут возникать различные проблемы и уязвимости. Некоторые из распространенных проблем:

1. **Несанкционированный доступ:** это одна из основных угроз безопасности баз данных. Несанкционированные пользователи или злоумышленники могут получить доступ к базе данных, что может привести к утечке или несанкционированному использованию данных.
2. **Утечка персональных данных:** если в статистической базе данных содержатся персональные данные, то утечка этих данных может нарушить конфиденциальность и привести к нарушению приватности субъектов данных.
3. **Недостаточные уровни защиты:** если механизмы защиты недостаточно реализованы, это может создать уязвимости и позволить злоумышленникам получить несанкционированный доступ к базе данных.
4. **Недостатки в управлении доступом:** некорректно настроенные или управляемые механизмы управления доступом могут привести к неправильному назначению привилегий или возможности несанкционированного доступа к данным.
5. **Недостатки в обработке и фильтрации данных:** некорректная обработка или фильтрация входных данных может привести к возникновению уязвимостей в базе данных, которые могут быть использованы злоумышленниками для выполнения атак, таких как внедрение SQL-кода или внедрение вредоносного программного обеспечения.
6. **Недостатки в резервном копировании и восстановлении:** если не регулярно создаются резервные копии базы данных или не проверяется их целостность, это может создать проблемы при восстановлении данных в случае сбоя или утраты данных.
7. **Недостаточное обновление и патчи:** не обновленное программное обеспечение и отсутствие установленных патчей безопасности могут оставить систему открытой для известных уязвимостей и атак.

Для обеспечения безопасности статистических баз данных необходимо принимать меры по предотвращению и устранению указанных проблем. Некоторые из подходов к обеспечению безопасности

статистических баз данных включают:

1. **Реализация аутентификационных механизмов:** использование надежных методов аутентификации, таких как пароли, многофакторная аутентификация или биометрическая аутентификация, помогает предотвратить несанкционированный доступ к базе данных.

2. **Применение принципа наименьших привилегий:** управление доступом к базе данных должно быть настроено таким образом, чтобы пользователи имели доступ только к необходимым данным и функциональности.

3. **Шифрование данных:** применение шифрования для защиты данных в базе данных может помочь предотвратить несанкционированный доступ в случае утечки данных или физической кражи носителя данных.

4. **Регулярное обновление и патчи:** важно регулярно обновлять программное обеспечение базы данных и устанавливать патчи безопасности, чтобы устранить известные уязвимости и защитить базу данных от известных атак.

5. **Защита от SQL-инъекций:** применение строгих проверок и фильтрации входных данных может помочь предотвратить атаки, связанные с внедрением SQL-кода.

6. **Установка механизмов мониторинга и аудита:** реализация механизмов мониторинга и аудита позволяет отслеживать подозрительную активность или попытки несанкционированного доступа.

7.4 Безопасность персональных данных в статистических БД

Вопросы защиты статистических баз данных безусловно связаны со многими вопросами защиты баз данных в целом, тем не менее специфика статистических баз данных обязывает нас обратить особое внимание на проблему обработки персональных данных. Для защиты персональных данных в статистических базах данных существует несколько математических методов и подходов. Вот некоторые из них:

1. **Анонимизация данных:**

   - Обобщение (Generalization): при обобщении конкретные значения заменяются на более общие категории или диапазоны. Например, возрастные данные могут быть обобщены до групп возрастов. Обобщение помогает сохранить общую информацию о данных, но уменьшает точность идентификации отдельных лиц.
   - Подавление (Suppression): при подавлении определенные значения удаляются из данных. Например, можно удалить точные географические координаты, чтобы сохранить только общую информацию о местоположении. Подавление может помочь снизить риск идентификации отдельных лиц, но может также привести к потере информации.

- Синтез данных (Data Synthesis): синтез данных предполагает создание синтетических данных, которые сохраняют статистические свойства и распределения оригинальных данных, но не содержат исходных значений. Синтез данных может быть осуществлен с использованием алгоритмов генерации синтетических данных, таких как алгоритмы генерации синтетических популяций. Синтетические данные могут использоваться для анализа и обмена без раскрытия реальных персональных данных.
- Добавление шума (Noise Addition): добавление случайного шума к данным является еще одним методом анонимизации. Шум может быть добавлен к числовым данным или текстовым данным, чтобы затруднить идентификацию отдельных лиц. При этом сохраняется общая статистическая информация, но точные значения скрываются.

2. **Криптографические протоколы:**

- Симметричное шифрование: это метод шифрования, при котором один и тот же ключ используется для шифрования и расшифрования данных. Симметричное шифрование обеспечивает конфиденциальность данных, но требует, чтобы отправитель и получатель имели доступ к общему секретному ключу.
- Асимметричное шифрование: асимметричное шифрование использует пару ключей - публичный и приватный. Публичный ключ используется для шифрования данных, а приватный ключ используется для их расшифровки. Асимметричное шифрование позволяет безопасно обмениваться данными без необходимости общего секретного ключа.
- Цифровые подписи: цифровая подпись представляет собой математическую конструкцию, которая связывает определенные данные с приватным ключом отправителя. Это обеспечивает подтверждение авторства и целостности данных. Получатель может использовать публичный ключ отправителя для проверки цифровой подписи и убедиться, что данные не были изменены в процессе передачи и что они были созданы конкретным отправителем.

3. **Протоколы нулевого разглашения:**

- Доказательство знания (Proof of Knowledge): этот протокол позволяет одной стороне доказать, что она знает определенную информацию, не раскрывая саму информацию. Например, можно доказать знание пароля, не раскрывая сам пароль.
- Защита от проверки: протоколы нулевого разглашения могут использоваться для защиты от проверки, при которой одна сторона пытается проверить определенное условие, а другая сторона хочет сохранить конфиденциальность своих данных.

4. **Гомоморфное шифрование:**

- Полностью гомоморфное шифрование (Fully Homomorphic Encryption): Этот метод позволяет выполнить любые операции на зашифрованных данных, включая сложение, умножение и другие операции, сохраняя конфиденциальность данных. Результат операции будет зашифрован и может быть расшифрован только владельцем приватного ключа.
- Частично гомоморфное шифрование (Partially Homomorphic Encryption): В отличие от полностью гомоморфного шифрования, частично гомоморфное шифрование позволяет

выполнять только определенные операции на зашифрованных данных, например, сложение или умножение, но не все операции.

5. **Техники приватного подсчета:**

   - Приватный подсчет суммы (Private Summation): Этот метод позволяет вычислять сумму значений без раскрытия исходных данных. Различные протоколы, такие как протоколы шумового добавления или протоколы Secure Multiparty Computation, могут быть использованы для выполнения приватного подсчета суммы.
   - Приватный подсчет среднего значения (Private Averaging): Этот метод позволяет вычислить среднее значение данных из различных источников без раскрытия исходных данных. Он может использовать те же протоколы, что и приватный подсчет суммы.

Каждый из этих методов и подходов представляет собой различные техники, которые могут быть применены для обеспечения безопасности персональных данных в статистических базах данных. Выбор конкретного метода зависит от требований безопасности, типа данных и контекста применения. Комбинация этих методов и подходов может обеспечить эффективную защиту персональных данных в статистических базах данных, минимизируя риски несанкционированного доступа, утечки информации и нарушения конфиденциальности.

7.5 Проблемы безопасности персональных данных в статистических базах данных

Статистической (в приведенном в этом разделе контексте) считается база данных, в которой допускаются запросы с обобщением данных (суммированием, вычислением среднего значения и т.д.), но не допускаются запросы по отношению к элементарным данным. Например, в статистической базе данных разрешается выдача запроса "Какова средняя зарплата программистов? тогда как выдача запроса "Какова зарплата программиста Мэри?"запрещена. Проблема безопасности в статистических баз данных заключается в том, что иногда с помощью логических заключений на основе выполнения разрешенных запросов можно вывести ответ, который прямо может быть получен только с помощью запрещенного запроса. "Обобщенные значения содержат следы исходной информации, и она может быть восстановлена злоумышленником после соответствующей обработки этих обобщенных значений. Такой процесс называется логическим выводом конфиденциальной информации". Практически для любой статистической базы данных всегда может быть определен общий трекер (в отличие от множества индивидуальных трекеров). Общий трекер (general tracker) — это логическое выражение, которое может быть использовано для поиска ответа на любой запрещенный запрос, т.е. запрос, включающий недопустимое логическое выражение. (В противоположность этому индивидуальный трекер работает только на основе запросов, включающих конкретные запрещенные выражения.) Требуется поддерживать баланс между репрезентативностью данных и конфиденциальностью отдельных записей.

Согласно источнику Graham G. S., Denning P. J. Protection - Principles and Practice:

"...методы нарушения защиты данных просты и не связаны с большими расходами. Поэтому требование обеспечения полной секретности конфиденциальной информации несовместимо с

требованием возможности вычисления точных статистических показателей для произвольных подмножеств данных в базе. По крайней мере одно из этих требований должно быть снято прежде, чем можно будет поверить в гарантии обеспечения секретности.

Существует точка зрения, что обеспечение полной безопасности статистических баз данных (СБД) может быть проблематичным. Одной из основных проблем является проблема вывода. В общих чертах, проблема вывода для СБД заключается в том, что с помощью характеристической функции C можно определить подмножество записей в базе данных. Запрос, использующий C, предоставляет статистику по выбранному подмножеству. Если подмножество достаточно маленькое, даже состоящее из одной записи, пользователь запроса может сделать выводы о характеристиках отдельного человека или небольшой группы. Даже для больших подмножеств, структура и характер данных могут быть такими, что несанкционированная информация может быть раскрыта. Для злоумышленника, пытающегося извлечь индивидуальные данные, задача заключается в создании общего трекера. Необходимо разработать последовательность запросов и операций, чтобы вывести индивидуальную информацию. Суть заключается в том, что с помощью определенной комбинации запросов и операций можно деанонимизировать записи и получить доступ к конфиденциальным данным.

Иными словами, даже при применении методов безопасности, существует возможность извлечения конфиденциальной информации из статистических баз данных путем тщательно спланированных запросов и операций. Это вызвано особенностями структуры данных и возможностью вывести индивидуальные данные из больших объемов информации. Поэтому обеспечение полной безопасности СБД представляет сложную задачу, требующую учета различных факторов и использования эффективных методов защиты данных.

Для статической базы данных можно привести следующий пример: предположим, у нас есть запрос1, который запрашивает значения al + a2 + a3, и запрос2, который запрашивает значения a1 + a2. Если мы вычтем результат запроса2 из запроса1, то получим значение a3. Таким образом, используя комбинацию запросов, мы можем извлечь конкретное значение a3 из статической базы данных.

В случае динамической онлайн базы данных, к примеру, мы можем рассмотреть ситуацию, когда мы хотим получить информацию о зарплате Мэри, зная, что ей 20 лет. Мы добавляем в базу данных множество ложных записей с возрастом 20 лет и нулевой зарплатой. Затем мы делаем запрос с минимальной агрегацией, чтобы получить информацию о людях с возрастом 20 лет. Таким образом, мы можем получить возможное значение зарплаты Мэри. Чем больше у нас исходных знаний о Мэри, тем точнее будет полученная информация. Более того необходимо помнить, что мы работаем с базой данных, поэтому статистическая база данных наследует все угрозы, присущие обычным базам данных, некоторые из которых были перечислены ранее.

Несмотря на большую сложность обеспечения безопасности статистических баз данных, они находят широкое применение в социально значимых сферах, вследствие чего вопрос безопасности стоит крайне остро. Рассмотрим общие подходы защиты СБД, существующих на данный момент:

1. **Концептуальный**

   Основная проблема заключается в том, что реляционная алгебра позволяет выводить индивидуальные данные. Концептуальный подход предлагает заменить классическую систему реляционных БД, работающую с индивидуальными данными. Одним из основных методов этого подхода является разделение базы данных на агрегированные и обезличенные записи. Например, можно использовать метод микроагрегации, в котором исходные данные разбиваются на несколько записей, для которых рассчитываются средние значения, а затем заменяются исходные значения на полученные средние. Также можно применить сеточную модель агрегации данных по различным признакам на разных уровнях детализации.

2. **Ограничение запросов**

   Основные методы обеспечения безопасности, которые также используются в СБД, включают ограничение доступа, ограничение запросов и возмущение данных. Ограничение запросов "Query Set Restriction"является механизмом, который устанавливает минимальный размер выборки, выдаваемой в результате запроса.

   Метод "Limiting intersection of query sets"представляет собой механизм защиты, который блокирует запросы, приводящие к выводу данных через пересечение множеств запросов. Это достигается путем сохранения исторических данных о выполнении запросов и отклонения любых запросов, использующих значительное количество исходных данных, которые были обработаны предыдущим запросом.

   Аудит (Auditing) представляет собой процесс ведения журнала, который позволяет обнаруживать и регистрировать подозрительную или недобросовестную активность.

3. **Возмущение данных**

   Возмущение данных предполагает добавление небольшого случайного шума или введение дополнительных строк данных при обработке запросов. Это может быть реализовано, например, путем случайного изменения результатов каждого запроса или использования случайных наборов исходных данных для ответов на запросы. Возмущение данных "Data Perturbation"включает добавление небольшого случайного шума к данным. Такой подход предполагает, что точные значения данных будут уничтожены, но при этом может возникнуть проблема искажения репрезентативности данных. Метод также предлагает случайное добавление дополнительных строк к основному набору данных, обрабатываемому запросом. Также предлагается использование "обмена данными"("data swapping"), что означает обмен значениями атрибутов между кортежами таким образом, чтобы сохранить статистическую точность. Даже если злоумышленнику удастся идентифицировать отдельное значение, например зарплату, у него не будет способа узнать, какому конкретному кортежу, например сотруднику, это значение принадлежит. Однако данному подходу присущи сложности в поиске множества записей, между

которыми можно организовать обмен значениями соответствующим образом. Аналогичные затруднения возникают и при использовании большинства других методов.

4. **Возмущение данных**

Подход "возмущение вывода"(Output Perturbation) представляет собой метод, аналогичный предыдущему подходу, однако в данном случае результат каждого запроса преобразуется. Для этого применяется метод случайного выборочного исследования (Random Sampling), который обеспечивает различные наборы записей при повторном выполнении одного и того же запроса. В данном подходе для ответов на запросы используются только случайные наборы исходных данных.

На рисунке 23 изображена схема 3 подходов.

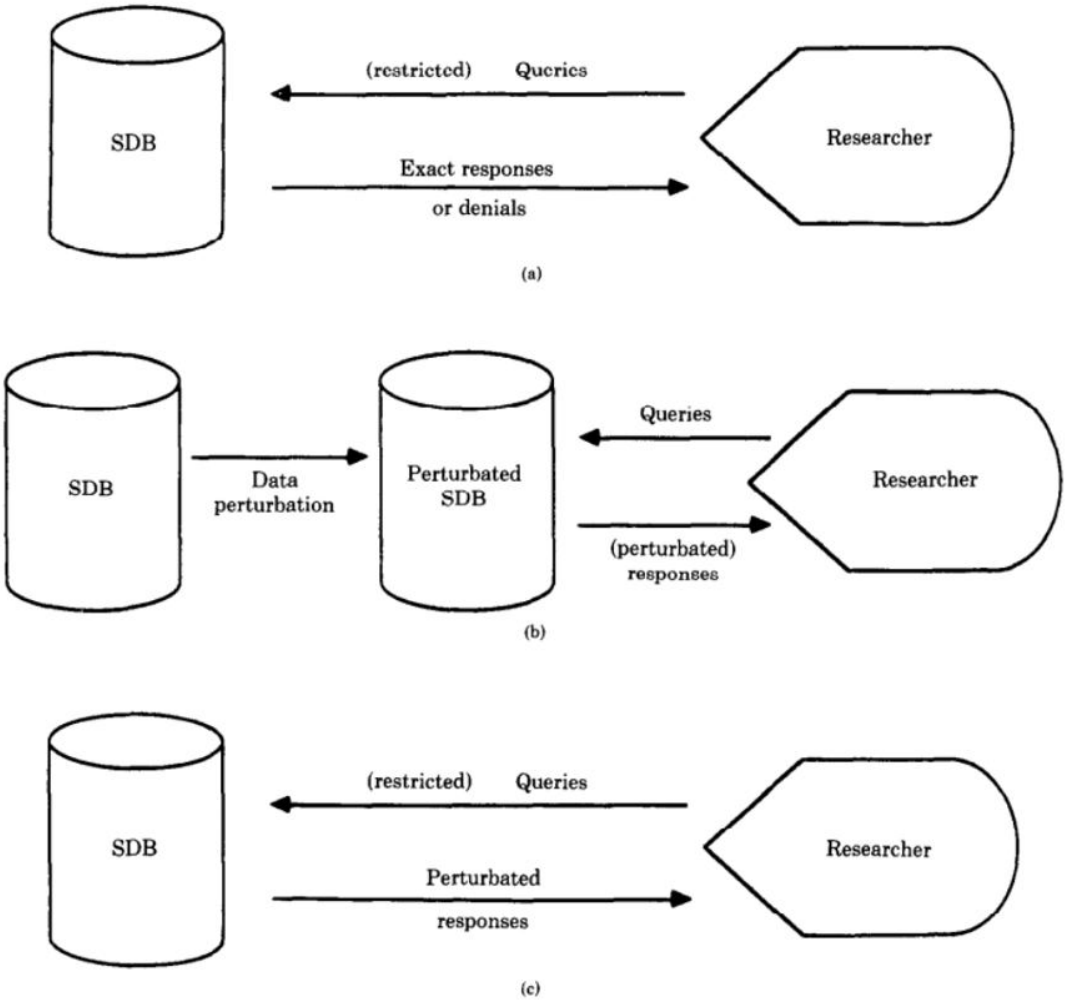


Рис. 23: Схемы безопасности СБД

7.6 Критерии безопасности статистических баз данных

Критерии безопасности включают оценку вероятности раскрытия записей в СБД, оценку консистентности данных при использовании методов возмущения данных и оценку зависимости между возмущением и конкретными записями. Также учитываются затраты, связанные с выполнением запросов, и назначается конкретная цена каждому запросу. Пользователям предоставляется начальная сумма, которую они могут использовать для запросов, чтобы предотвратить выведение данных.

Когда речь идет о критериях безопасности, необходимо отметить, что полной гарантии безопасности в статистических базах данных (СБД) достичь невозможно. Вместо этого проводится оценка важности деанонимизированных данных и статистической точности, учитывая предпринятые меры по защите данных.

Критерий безопасности "Security"оценивает вероятность раскрытия (включая частичное раскрытие) записи в СБД. Для каждого разрешенного агрегирующего запроса проводится оценка критерия безопасности. Основываясь на количестве информации в базе данных, можно определить, сколько запросов требуется для деанонимизации данных. На основе этой оценки определяется минимальный размер выборки для таких запросов.

Критерий безопасности "Consistency"оценивает консистентность данных для методов возмущения данных. Оценивается степень возмущения данных путем создания агрегирующего запроса и измерения его изменений после внесения возмущений.

Критерий безопасности "Robustness"оценивает зависимость между возмущением и конкретными записями. Желательно, чтобы возмущение не зависело от данных, однако необходимо сохранить статистические свойства данных.

Критерий безопасности "Costs"определяет конкретную стоимость каждого запроса. Пользователю предоставляется начальная сумма, которую он может использовать для запросов, чтобы предотвратить вывод данных. Этот критерий скорее представляет собой метод управления, а не непосредственный критерий безопасности.

Таким образом, применение критериев безопасности в СБД позволяет оценивать уровень безопасности, учитывая важность данных, статистическую точность, консистентность, робастность и стоимость запросов.

Исходя из вышеизложенных глав, можем убедиться, что безопасность статистических баз данных является сложной задачей, но существуют различные подходы и методы для обеспечения безопасности данных. Каждый подход имеет свои преимущества и недостатки, и выбор методов безопасности должен основываться на анализе конкретных требований и контекста использования СБД. Оценка важности данных, статистической точности и затрат помогает принять решение о применении конкретных методов обеспечения безопасности.

8 Распознавание вторжений в БД

8.1 Основные понятия

Обнаружение вторжений (применительно к БД) – процесс выявления действий, которые способны нарушить конфиденциальность, целостность и доступность информации, хранимой в базе данных (БД).

Система обнаружения вторжений (СОВ) [согласно ФСТЭК] – программное или программно-техническое средство, реализующие функции автоматизированного обнаружения (блокирования) действий в информационной системе, направленных на преднамеренный доступ к информации, специальные воздействия на информацию (носители информации) в целях ее добывания, уничтожения, искажения и блокирования доступа к ней.

Применение систем обнаружения вторжения относится к реактивным мерам (то есть к таким мерам, которые используются уже после того, как произошел инцидент) для противодействия активности злоумышленника в тех случаях, когда злоумышленник преодолел все проактивные меры. Поэтому обычно системы обнаружения вторжений являются *вторичным фактором защиты в общей системе защиты* и предназначены для обнаружения и регистрации уже произошедних событий, а также оповещение персонала при срабатывании определенных правил.

Если классифицировать методы обнаружения вторжения, используемые в системах обнаружения вторжений (СОВ), то можно выделить три типа (согласно классификации, предложенной Стефаном Аксельсоном [Axelsson 2000]):

- синтаксические методы (или методы поиска злоупотреблений, англ. misuse detection). К таким методам обычно относятся методы, которые основываются на обнаружения вторжений путём сравнения SQL-запросов с шаблонами недопустимых синтаксических конструкций.
- методы обнаружения аномалий (англ. anomaly detection). Такие методы, наоборот, в отличие от первого типа, подрузамевают создание шаблонов нормального поведения пользователя и последующее сравнение этих шаблонов с действиями, выполняемыми пользователями во время работы с БД.
- смешанные методы, представляющие собой композицию первых двух

8.2 Системы распознавания вторжений

Системы обнаружения вторжения (англ. Intrusion Detection System, IDS) и системы предотвращения вторжений (англ. Intrusion Preventation System, IPS) обычно рассматриваются вместе, причем часто под термином система предотвращение вторжений подрузамевается расширение систем обнаружения вторжений, так как IPS системы также должны обладать возможностью обнаружения вторжений. Иначе говоря, IPS системы являются активными IDS системами.

В IDS системах, система также ответственна за реализацию ответных мер на нарушения: блокировка соединения, настройки межсетевого экрана и прочее. Таким образом,

Система обнаружения и предотвращения вторжений (IPS/IDS) (применительно к БД) – это программное или программно-техническое средство, предназначенное для выявления фактов неавторизованного доступа и предотвращания попыток несанкционированного доступа к БД.

8.2.1 Типы моделей систем распознавания вторжений (ID-систем)

Существует множество способов классификации СОВ, которые не являются однозначными или обязательными. Следует рассмотреть наиболее известные и используемые классификации, которые характеризуют систему по:

- **По способу мониторинга**
- **По способу анализа**
- **По скорости реакции**
- **По классу защиты**

По способу мониторинга.

- **Сетевая СОВ (англ. Network-based IDS, NIDS)** – система, которая занимается проверкой сетевого трафика с концентратора или коммутатора и анализируя сетевые пакеты.
- **Узловая СОВ (англ. Host-based IDS, HIDS)** – система, отслеживающая вторжения, используя анализ системных вызовов, логов приложений, модификаций файлов (исполняемых, файлов паролей, системных баз данных), состояния хоста и прочих источников.
- **Основанная на прикладных протоколах СОВ (англ. Application-based IDS, APIDS)** – система, в которой ведется наблюдение за специализированными прикладными протоколами и анализ соответствующих данных. Например, на веб-сервере с SQL базой данных СОВ будет отслеживать содержимое SQL команд, передаваемых на сервер.
- **Гибридная СОВ** – система, которая является композицией нескольких видов систем обнаружения вторжений.

По способу анализа. О данной классификации уже говорилось ранее. Она строится по методу анализа событий, полученных из источника информации, и методу принятия решения, что происходит проникновение. Способами анализа являются **обнаружение злоупотреблений** и **обнаружение аномалий**.

По скорости реакции. Определяют два типа СОВ по времени между получением информации из источника и ее анализом и принятием решения. В зависимости от задержки во времени, СОВ разделятся на:

- **с пакетным режимом (англ. interval-based)**. В таких системах реакция происходит через определенные интервалы времени, а информационный поток от точек мониторинга до инструментов анализа не является непрерывным.
- **непрерывные (англ. real-time)**. В таких системах обрабатывается непрерывный поток информации от источников. При этом таких тип является преобладающей типом в сетевых СОВ, которые получают информацию из потока сетевого трафика.

По классу защиты. Для каждой сертифицированной в России СОВ присваивается некоторый класс защиты согласно классификации ФСТЭК и выполненным требованиям для определенного профиля защиты (ПЗ). Всего существует 6 классов, где самый низкий класс - шестой, а самый высокий - первый.

Согласно ФСТЭК СОВ разделяются на [*ИНФОРМАЦИОННОЕ ПИСЬМО ОБ УТВЕРЖДЕНИИ ТРЕБОВАНИЙ К СИСТЕМАМ ОБНАРУЖЕНИЯ ВТОРЖЕНИЙ* 2012]:

- **СОВ с 6 классом защиты**: применяются в информационных системах персональных данных 3 и 4 классов.
- **СОВ с 5 классом защиты**: применяются в информационных системах персональных данных 2 класса.
- **СОВ с 4 классом защиты**: применяются в государственных информационных системах, в которых обрабатывается информация ограниченного доступа, не содержащая сведения, составляющие государственную тайну, в информационных системах персональных данных 1 класса, а также в информационных системах общего пользования II класса.
- **СОВ с 3, 2 или 1 классом защиты**: применяются в информационных системах, в которых обрабатывается информация, содержащая сведения, составляющие государственную тайну.

8.2.2 Общая структура ID-систем

Архитектура СОВ. Основнымы архитектурными компонентами СОВ являются:

1. **Host** – система, на которой выполняется ПО СОВ.
2. **Target** – система, за которой наблюдает СОВ.

Первоначально многие СОВ выполнялись на тех же системах, которые они защищали. Основная причина этого была в том, что большинство систем было mainframe, и стоимость выполнения СОВ на отдельном компьютере была очень большой. Это создавало проблему с точки зрения безопасности, так как любой атакующий, который успешно атаковал целевую систему, мог в качестве одной из компонент атаки просто запретить функционирование СОВ.

Но с появлением рабочих станций и персональных компьютеров в большинстве архитектур СОВ предполагается выполнение СОВ на отдельной системе, тем самым разделяя системы Host и Target. Это улучшает безопасность функционирования СОВ, так как в этом случае проще спрятать существование СОВ от атакующих.

Компоненты современных СОВ:

- сенсор, который отслеживает события в сети или системе;
- анализатор событий, обнаруженных сенсорами;
- компонента принятия решения.

Способы управления СОВ:

- **Централизованное управление**. При централизованных стратегиях управления весь мониторинг, обнаружение и отчетность управляются непосредственно с единого "поста". В этом случае существует единственная консоль СОВ, которая связана со всеми сенсорами, расположенными в сети.
- **Частично распределенное управление**. Мониторинг и определение управляются с локально управляемого узла, с иерархической отчетностью в одно или более центральных расположений.
- **Полностью распределенное управление**. Мониторинг и определение выполняются с использованием подхода, основанного на агентах, когда решения об ответе делаются в точке анализа.

При этом в сети должны поддерживаться следующие связи:

- связи для передачи отчетов СОВ. Эти связи создаются между сенсорами как сетевого мониторинга, так и мониторинга хоста, и центральной консолью СОВ;
- связи для мониторинга хостов и сетей;
- связи для выполнения ответов СОВ.

8.2.3 Определение злоупотреблений

Детекторы злоупотреблений анализируют деятельность системы, анализируя событие или множество событий на соответствие заранее определенному образцу, который описывает известную атаку. Соответствие образца известной атаке называется сигнатурой, определение злоупотребления иногда называют “сигнатурным определением”. Наиболее общая форма определения злоупотреблений, используемая в коммерческих продуктах, специфицирует каждый образец событий, соответствующий атаке, как отдельную сигнатуру. (*Лекции по системному администрированию МГУ им. Ломоносова, Лекция 4 Intrusion Detection Systems* 2022)

Тем не менее существует несколько более сложных подходов для выполнения определения злоупотреблений (называемых state-based технологиями анализа), которые могут использовать единственную сигнатуру для определения группы атак.

Обнаружение злоупотреблений позволяет идентифицировать несанкционированные действия, если имеется их точное представление в виде шаблонов атак. Здесь под шаблоном атаки понимается некоторая совокупность явно описывающих конкретную атаку действий (правил сопоставления, вывода), применяя которые к признакам и полям идентифицируемого объекта можно получить однозначный ответ о его принадлежности к этой атаке. (А.А. Браницкий 2016)

Преимущества определения злоупотреблений:

- Детекторы злоупотреблений являются очень эффективными для определения атак и не создают при этом огромного числа ложных сообщений.

- Детекторы злоупотреблений могут быстро и надежно диагностировать использование конкретного инструментального средства или технологии атаки. Это может помочь администратору скорректировать меры обеспечения безопасности.

- Детекторы злоупотреблений позволяют администраторам, независимо от уровня их квалификации в области безопасности, начать процедуры обработки инцидента.

Недостатки определения злоупотреблений:

- Детекторы злоупотреблений могут определить только те атаки, о которых они знают, следовательно, надо постоянно обновлять их базы данных для получения сигнатур новых атак.

- Многие детекторы злоупотреблений разработаны таким образом, что могут использовать только строго определенные сигнатуры, а это не допускает определения вариантов общих атак.

Подытоживая сказанное, отметим, что методы обнаружения злоупотреблений являются эффективным инструментом для выявления известных типов атак, но их применимость по отношению к новым атакам, а также к модификациям известных атак является безрезультативной.

8.2.4 Определение аномалий и шаблоны классов пользователей

Рассматривать шаблоны классов пользователей имеет смысл только в контексте СОВ, применяющих методы обнаружения аномалий, так как именно они строят и используют такие шаблоны поведения пользователей. Таким образом, детекторы аномалий определяют необычное поведение на хосте или в сети. Они предполагают, что атаки отличаются от некоторой нормальной деятельности и могут, следовательно, быть определены системой, которая умеет отслеживать эти отличия. Детекторы аномалий создают профили, представляющие собой нормальное поведение пользователей, хостов или сетевых соединений. Эти профили создаются, исходя из данных истории, собранных в период нормального функционирования. Затем детекторы собирают данные о событиях и используют различные метрики для определения того, что анализируемая деятельность отклоняется от нормальной.

Метрики и технологии, используемые при определении аномалий, включают:

- определение допустимого порога. В этом случае основные атрибуты поведения пользователя и системы выражаются в количественных терминах. Для каждого атрибута определяется некоторый уровень, который устанавливается как допустимый. Такие атрибуты поведения могут определять число файлов, доступных пользователю в данный период времени, число неудачных попыток входа в систему, количество времени ЦП, используемое процессом и т.п. Данный уровень может быть статическим или эвристическим — например, может определяться изменением анализируемых значений.

- статистические метрики: параметрические, при которых предполагается, что распределение атрибутов профиля соответствует конкретному образцу, и непараметрические, при которых распределение атрибутов профиля является "обучаемым"исходя из набора значений истории, которые наблюдались за определенный период времени.

- метрики, основанные на правилах, которые аналогичны непараметрическим статистическим метрикам в том, что наблюдаемые данные определяют допустимые используемые образцы, но отличаются от них в том, что эти образцы специфицированы как правила, а не как численные характеристики.
- другие метрики, включая нейросети, генетические алгоритмы и модели иммунных систем.

Преимущества определения аномалий:

- IDS, основанные на определении аномалий, обнаруживают неожиданное поведение и, таким образом, имеют возможность определить симптомы атак без знания конкретных деталей атаки.
- Детекторы аномалий могут создавать информацию, которая в дальнейшем будет использоваться для определения сигнатур для детекторов злоупотреблений.

Недостатки определения аномалий:

- Подходы определения аномалий обычно создают большое количество ложных сигналов при непредсказуемом поведении пользователей и непредсказуемой сетевой активности.
- Подходы определения аномалий часто требуют некоторого этапа обучения системы, во время которого определяются характеристики нормального поведения.

8.2.5 Модели известных атак

Далее будут рассмотрены основные типы SQL-инъекций (*SQL injections* 2022),(Yunus 2018) на простом примере критически уязвимой страницы

Рис. 24: Пример уязвимой страницы

- **Атаки комментированием**
  Использование однострочных комментариев позволяет игнорировать часть запроса, идущую после вашей инъекции. Например, ввод в уязвимое поле Username запроса admin'– позволит зайти на ресурс под администратором, потому что поверка пароля будет закомментирована.

```
SELECT * FROM members WHERE username = 'admin'–' AND password =
'password'
```

Многострочные комментарии могут справится с проверкой или определить тип базы данных. Например, подобные запросы обойдут примитивный текстовый анализ:

```
DROP/*some comment*/sampletable
DR/**/OP/*random comment to cheat*/sampletable
```

- **Манипуляции со строками**
  При помощи конкатенации строк можно обходить фильтр кавычек.

```
SELECT CONCAT(login, password) FROM members
```

Можно представлять строки в шеснадцатиричном виде, с помощью функции HEX() или вводить их посимвольно.

```
//0x633A5C626F6F742E696E69 == c:\boot.ini
SELECT CONCAT('0x','633A5C626F6F742E696E69'))
SELECT CONCAT(CHAR(75),CHAR(76),CHAR(77))
```

- **Обход аутентификации**
  При помощи OR и сравнения констант можно обойти форму аутентификации. Существуют даже словари, содержащие основные запросы для обхода уязвимой формы

  Примеры запросов:

```
' or 1=1
' or 1=1–
' or 1=1#
admin' –
admin' or '1'='1
```

- **Union injection**
  При помощи UNION комбинировать данные из разных таблиц в одну. Это одна из самых популярных и опасных классических инъекций.

  Допустим, на сайте есть список товаров с уязвимой строкой поиска. Тогда, подобрав правильное количество колонок и определив их название, через UNION можно вывести практически любые данные

```
SELECT name, price FROM products UNION ALL SELECT name, pass FROM
members
#Такой запрос позволит получить данные о таблицах и найти таблицу поль-
```

```
зователей
UNION(SELECT TABLE_NAME,
TABLE_SCHEMA FROM information_schema.tables)
```

- **Последовательные запросы**

  В некоторых СУБД можно использовать простой знак ';' для последовательного вызова вредоносных запросов

```
#Удаление таблицы
SELECT * FROM products WHERE productName = ; DROP users–
#Выключение SQL Server
SELECT * FROM products WHERE productName = ; shutdown –
```

- **Error-based injection**

  Инъекции, основанные на том, что злоумышленник может видеть вывод ошибки. Уязвимость устраняется просто отключением этого вывода

  Последовательное выполнение следующих запросов может помочь определить в тексте ошибки названия столбцов:

```
' HAVING 1=1 –
' GROUP BY table.columnfromerror1 HAVING 1=1 –
' GROUP BY table.columnfromerror1, columnfromerror2 HAVING 1=1 –
.....
' GROUP BY table.columnfromerror1, columnfromerror2, columnfromerror(n)
HAVING 1=1 –
Если ошибки перестали появляться, значит столбцы закончились
```

- **Error-based injection**

  Инъекции, основанные на том, что злоумышленник может видеть вывод ошибки. Уязвимость устраняется просто отключением этого вывода

  Последовательное выполнение следующих запросов может помочь определить в тексте ошибки названия столбцов:

```
' HAVING 1=1 –
' GROUP BY table.columnfromerror1 HAVING 1=1 –
' GROUP BY table.columnfromerror1, columnfromerror2 HAVING 1=1 –
.....
' GROUP BY table.columnfromerror1, columnfromerror2, columnfromerror(n)
HAVING 1=1 –
Если ошибки перестали появляться, значит столбцы закончились
```

- **Boolean-based blind injection**

  Если атакующий все же может получить информацию о наличии или отсутствии ошибки из HTTP-статуса, в сервисе имеется уязвимость к обычной слепой атаке. Рассмотрим запрос, который позволит нам при помощи алгоритма бинарного поиска посимвольно определить название первой таблицы и в дальнейшем всех данных

  ```
  TRUE : SELECT ID, Username, Email FROM [User]WHERE ID = 1 AND
  ISNULL(ASCII(SUBSTRING((SELECT TOP 1 name FROM sysObjects
  WHERE xtYpe=0x55 AND
  name NOT IN(SELECT TOP 0 name FROM sysObjects WHERE
  xtYpe=0x55)),1,1)),0)>78–
  #Этот запрос говорит нам, что ASCII-значение первого символа больше 78
  #дальнейший перебор определит точное значение
  ```

- **Time-based blind injection**

  Если атакующий не наблюдает никаких отличий в ответах сервера, можно попробовать SLEEP или WAIT FOR DALAY

  ```
  SELECT * FROM products WHERE id=1; WAIT FOR DELAY '00:00:15'
  ```

  Конечно, реальные примеры будут выглядеть примерно как boolean-based, только true и false атакующий будет отличать по времени отклика

8.3 Экспертные ID-системы

Технологию построения экспертных систем часто называют инженерией имплекационных правил. Как правило, этот процесс требует специфической формы взаимодействия создателя имплекационных правил и одного или нескольких экспертов в некоторой предметной области. Инженер имплекационных правил «встраивает» процедуры, стратегии, эмпирические правила в экспертную систему. В результате появляется компьютерная программа, которая решает задачи во многом так же, как эксперты – люди. (*Системы и методы обнаружения вторжений* б.г.)

Главное преимущество использования продукционных систем заключается в возможности разделения причин и решений возникающих проблем.

В экспертных системах могут использоваться импликационные правила (**Если** *условие* то **действие**).

Например:

- ЕСЛИ с одного узла за время T поступает N пакетов, ТО записать в лог факт: происходит DoS атака (факт A)
- Если наблюдается более чем N фактов A, ТО записать в лог факт: происходит DDoS атака

Основные проблемы, которые обычно возникают при их практическом применении:

- Недостаточная эффективность при работе с большими объемами данных.
- Трудно учесть зависимую природу данных параметров оценки.

При использовании продукционных систем для обнаружения вторжений можно установить символическое проявление вторжения при помощи имеющихся данных.

Трудности:

- Отсутствие встроенной или естественной обработки порядка последовательностей в анализируемых данных. База фактов, соответствующая левой части «продукции», используется для определения правой части. В левой части продукционного правила все элементы объединяются при помощи связи «и».
- Встроенная экспертиза хороша только в том случае, если моделируемые навыки администратора безопасности не противоречивы. Это практическое рассуждение, возможно, касается недостаточной централизованности усилий экспертов безопасности в направлении создания исчерпывающих множеств правил. Обнаруживаются только известные уязвимости.
- Существуют определенный программный инжиниринг, связанный с установкой (поддержанием) баз знаний. При добавлении или удалении какого-либо из правил должно изменяться остальное множество правил.
- Обнаруживаются только известные уязвимости.
- Объединение различных измерений вторжений и создание связанной картины вторжения приводит к тому, что частные причины становятся неопределенными. Ограничения продукционных систем, в которых используется неопределенная причина, довольно хорошо известны.

(P. Beynon-Davies 1991)

8.3.1 Метрики

- **Показатель активности** – величина, при превышении которой активность подсистемы оценивается как быстро прогрессирующая. В общем случае используется для обнаружения аномалий, связанных с резким ускорением в работе. Пример: среднее число записей аудита, обрабатываемых для элемента защищаемой системы в единицу времени.
- **Распределение активности в записях аудита** – распределение во всех типах активности в актуальных записях аудита. Здесь под активностью понимается любое действие в системе, например, доступ к файлам, операции ввода-вывода.
- **Измерение категорий** – распределение определенной активности в категории [13]. Например, относительная частота количества регистраций в системе (логинов) из каждого физического места нахождения. Предпочтения в использовании программного обеспечения системы (почтовые службы, компиляторы, командные интерпретаторы, редакторы и т.д.)
- **Порядковые измерения** – используется для оценки активности, поступающей в виде цифровых значений. Например, количество операция ввода-вывода, инициируемых каждым пользователем. Порядковые изменения вычисляют общую числовую статистику значений определенной активности, в то время как измерение категорий подсчитывает количество активностей.

(P. Beynon-Davies 1991)

[13] Здесь под *категорией* понимается группа подсистем, объединенных по некоему общему принципу

8.3.2 Статистические модели

При обнаружении аномалий с использованием профайла в основном применяют статистические методы оценки. Процесс обнаружения происходит следующим образом: текущие значения измерений профайла сравнивают с сохраненными значениями. Результат сравнения - показатель аномальности в измерении. Общий показатель аномальности в простейшем случае может вычисляться при помощи некоторой общей функции от значений показателя аномалии в каждом измерении профайла.

Например, пусть $M_1, M_2, \ldots, M_n$ – измерения профайла, а $S_1, S_2, \ldots, S_n$ – соответственно представляют собой значения аномалии каждого из измерений. Чем больше число S_i, тем больше аномалии в i-ом показателе. Объединяющая функция может быть взвешенной суммой их квадратов:

$$a_1 s_1^2 + a_2 s_2^2 + \cdots + a_n s_n^2 > 0, \tag{1}$$

где a_i – отражает относительный вес метрики M_i.

Параметры $M_1, M_2, \ldots, M_n$ могут быть зависимыми друг от друга. В таком случае, объединяющая функция будет более сложной.(*Системы и методы обнаружения вторжений* б.г.)

Основное преимущество заключается в том, что применяются хорошо известные статистические методы.

Недостатки:

- Нечувствительность к последовательности возникновения событий. То есть статистическое обнаружение может упустить вторжение, которое проявляется в виде последовательности сходных событий.
- Система может быть последовательно обучена таким образом, что аномальное поведение будет считаться нормальным. Злоумышленники, которые знают, что за ними наблюдают при помощи таких систем, могут обучить их для использования в своих целях. Именно поэтому в большинстве существующих схем обнаружения вторжения используется комбинация подсистем обнаружения аномалий и злоупотреблений.
- Трудно определить порог, выше которого аномалии можно рассматривать как вторжение. Занижение порога приводит к ложному срабатыванию (false positive), а завышение – к пропуску вторжений (false negative).
- Существуют ограничения к типам поведения, которые могут быть смоделированы, используя чистые статистические методы. Применение статистических технологий для обнаружения аномалий требует предположения, что данные поступают от квазистатического процесса.

(T.D. Garvey, T.F. Lunt 1991)

8.3.3 Профили

Одним из способов формирования «образа» нормального поведения системы состоит в накоплении измерений значения параметров оценки в специальной структуре данных. Эта структура данных называется *профайлом.*(*Системы и методы обнаружения вторжений* б.г.)

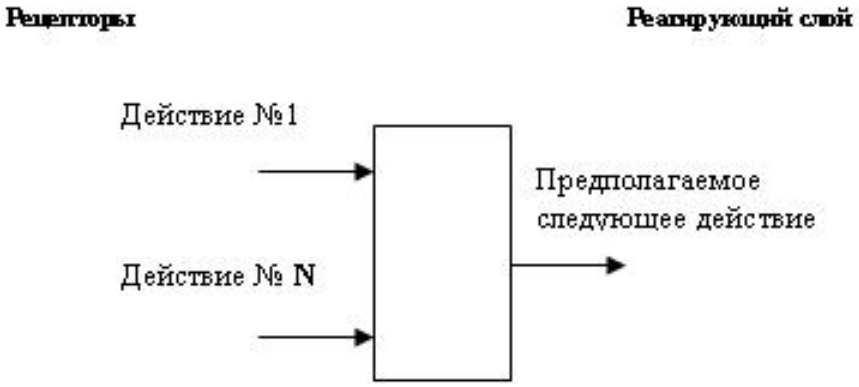
Основные требования, предъявляемые к структуре профайла:

- Минимальный конечный размер
- Быстрое выполнение операции обновления

8.3.4 Нейронные сети для представления профиля

Другой способов представления «образа» нормального поведения системы – обучение нейронной сети значениями параметров оценки.

Обучение нейронной сети осуществляется последовательностью информационных единиц (далее команд), каждая из которых может находиться на более абстрактном уровне по сравнению с используемыми параметрами оценки. Входные данные сети состоят из текущих команд и прошлых **W** команд, которые обрабатываются нейронной сетью с целью предсказания последующих команд; **W** также называют размером окна. После того как нейронная сеть обучена множеством последовательных команд защищаемой системы или одной из ее подсистем, сеть представляет собой «образ» нормального поведения. Процесс обнаружения аномалий представляет собой определение показателя неправильно предсказанных команд, то есть фактически обнаруживается отличие в поведение объекта. На уровне рецептора стрелки показывают входные данные последних **W** команд, выполненных пользователем. Входной параметр задает несколько значений или уровней, каждый из которых уникально определяет команду. Выходной реагирующий слой состоит из одного многоуровневого, который предсказывает следующую возможную команду пользователя. (P. Beynon-Davies 1991)



Недостатки:

- Топология сети и веса узлов определяются только после огромного числа проб и ошибок.
- Размер окна – еще одна величина, которая имеет огромное значение при разработке. Если сделать окно маленьким то сеть будет не достаточно производительной, слишком большим – будет страдать от неуместных данных.

Преимущества:

- Успех данного подхода не зависит от природы исходных данных.
- Нейронные сети легко справляются с зашумленными данными.

- Автоматически учитываются связи между различными измерениями, которые, несомненно, влияют на результат оценки.

(P. Beynon-Davies 1991)

8.3.5 Нейронные сети для представления профиля

Представление «образа» в данном случае основывается на предположении о том, что текущие значения параметров оценки можно связать с текущим состоянием системы. После этого функционирование представляется в виде последовательности событий или состояний.

Были предложены временные правила, которые характеризуют совокупности значений параметров оценки (далее паттерна) нормальной (не аномальной) работы. Эти правила формируются индуктивно и заменяются более «хорошими» правилами динамически во время обучения. Под «хорошими правилами» понимаются правила с большей вероятностью их появления и с большим уровнем уникальности для защищаемой системы. Для примера рассмотрим следующее правило:

$$E1 \rightarrow E2 \rightarrow E3 \Rightarrow E4 = 0.95, E5 = 0.05 \quad (2)$$

где E1...E5 - события безопасности.

Это утверждение, основанное на ранее наблюдавшихся данных, говорит о том, что для последовательности паттернов установилась следующая зависимость: если имеет место **E1** и далее **E2** и **E3**, то после этого вероятность проявления **E4** - 0.95 и **E5** – 0.05.

Именно множество правил, создаваемых индуктивно во время наблюдения работы пользователя, составляет «образ». Аномалия регистрируется в том случае, если наблюдаемая последовательность событий соответствует левой части правила выведенного ранее, а события, которые имели место в системе после этого, значительно отличаются от тех, которые должны были наступить по правилу.

Основной недостаток данного подхода заключается в том, что неузнаваемые паттерны поведения могут быть не приняты за аномальные из-за того, что они не соответствуют ни одной из левых частей всех правил.(*Системы и методы обнаружения вторжений* б.г.)

Данный метод довольно эффективно определяет вторжения, так как принимаются во внимание:

- Зависимости между событиями.
- Последовательность появления событий.

Достоинства метода:

- Лучшая обработка пользователей с большим колебанием поведения, но с четкой последовательностью паттернов.
- Возможность обратить внимание на некоторые важные события безопасности, а не на всю сессию, которая помечена как подозрительная.
- Лучшая чувствительность к обнаружению нарушений: правила содержат в себе семантику процессов, что позволяет гораздо проще заметить злоумышленников, которые пытаются обучить систему в своих целях.

(T.D. Garvey, T.F. Lunt 1991)

8.3.6 Примеры ID-систем

GreenSQL (APIDS система) – межсетевой экран, функционирующий как прокси-сервер между веб-приложением и SQL сервером. То есть приложение устанавливает соединения к БД не напрямую, а через сервер GreenSQL. GreenSQL анализирует SQL запросы на предмет аномальных запросов, и в случае нормального запроса (то есть если степень риска запроса мала) перенаправляет его на внутренний сервер БД.

GreenSQL поддерживает следующие режимы работы:

- **Режим симуляции (англ. Simulation Mode)** – пассивная система обнаружения атак (IDS). Протоколирует SQL запросы, выдает предупреждения на консоль управления.
- **Блокировка подозрительных команд (англ. Blocking Suspicious Commands/Risk Based)** – активная СОВ. Атаки не только обнаруживаются, но и блокируются (IPS) в соответствии с установленными правилами, указывающими на аномальность запроса.
- **Активная защита от неизвестных запросов (англ. Database Firewall)** – блокирование всех неизвестных запросов.
- **Режим обучения (англ. Learning mode)** – предназначен для прогона и настройки правил в «чистой» среде, что позволяет сформировать белый список и предотвратить в последствии ложные срабатывания анализатора запросов.

Особенности GreenSQL:

- Поддержка ряда СУБД: Microsoft SQL 2000/2005/2008, MySQL 4.x/5.x, PostgreSQL 7.x/8.x.
- Является кросс-платформенной. Среди официально поддерживаемых платформ Microsoft Windows Server 2003/2008, Ubuntu, CentOS. Поддерживаются 32-х и 64-х разрядные системы.
- GreenSQL находит подозрительные запросы, используя ряд методов:
  - Путем определения административных и чувствительных команд SQL. Например: SHOW TABLES, CREATE TABLE, ALTER TABLE.
  - Путем подсчета риска запроса. На это может влиять пустая строка пароля, "or"внутри запроса или выражения SQL, которые всегда возвращают истину (1=1).

Snort (NIDS система) – свободная сетевая система предотвращения вторжений (IPS) и обнаружения вторжений (IDS) с открытым исходным кодом, способная выполнять регистрацию пакетов и в реальном времени осуществлять анализ трафика в IP-сетях.

Snort поддерживает следующие режимы работы:

- **Sniffer mode**. В таком режиме программа только считывает сетевые пакеты и выводит их на консоль.
- **Packet Logger Mode**. В режиме логирования пакетов, программа будет регистрировать/логировать пакеты на диске.

- **Network Intrusion Detection System Mode**. В режиме обнаружения вторжений программа будет отслеживать сетевой трафик и анализировать его в соответствии с набором правил, определенным пользователем. Затем программа выполнит определенное действие, основанное на том, что было идентифицировано.

Особенности Snort:

- Возможность написания собственных правил.
- Расширение функциональности с помощью подключения дополнительных модулей.
- Гибкая система оповещения об атаках: Log-файлы, устройства вывода, БД и прочие.

Suricata (NIDS система) – свободная сетевая система предотвращения вторжений (IPS) и обнаружения вторжений (IDS) с открытым исходным кодом. Основана разработчиками, которые трудились над IPS версией Snort, поэтому имеет схожий функционал и обладает полной поддержкой формата правил Snort.

Особенности Suricata:

- Многозадачность.
- Автоматическое определение протокола.
- Высокая производительность, позволяющая обрабатывать трафик до 10Gbit на обычном оборудовании.

Sagan (HIDS система) – многопоточная, высокопроизводительная система анализа журналов и мониторинга появления в логах событий, связанных с безопасностью, и реагирования на эти события в режиме реального времени. Система работает в операционных системах Unix.

Sagan также относится к категории систем управления инцидентами и событиями информационной безопасности (англ. Security Information and Log Management, SEIM).

8.3.7 Системы анализа и оценки уязвимостей

Инструментальные средства анализа уязвимостей (известные также как оценка уязвимостей) тестируют сеть или хост для определения наличия уязвимостей для известных атак. Анализ уязвимостей предоставляет дополнительную информацию для систем обнаружения проникновения. Используемыми источниками информации являются атрибуты состояния системы и выходные данные осуществленных атак. Источники информации являются частью механизма оценки. Интервал времени анализа либо является фиксированным, либо определяется параметром в пакетном режиме, типом анализа является определение злоупотреблений. Это означает, что системы оценки уязвимостей являются пакетным режимом детекторов злоупотреблений, которые оперируют с информацией о состоянии системы, и результатом становятся специальные тестовые подпрограммы. (*Лекции по системному администрированию МГУ им. Ломоносова, Лекция 4 Intrusion Detection Systems* 2022)

Анализ уязвимостей — очень сильная технология управления безопасностью, но эта технология является лишь дополнением к использованию IDS, а отнюдь не ее заменой. Если организация

полагается исключительно на инструментальные средства анализа уязвимостей для слежения за системой, осведомленный атакующий может просмотреть систему анализа уязвимостей, заметить, когда выполняется анализ, и осуществить атаку во время интервала между этими проверками.

Тем не менее системы анализа уязвимостей могут создавать "моментальный снимок" состояния безопасности системы в конкретное время. Более того, так как они являются исключительно тестовыми системами поиска уязвимостей для большого числа известных атак, системы анализа уязвимостей могут позволять менеджеру безопасности контролировать ошибки человека или выполнять аудит системы для анализа согласованности с конкретной политикой безопасности системы.

Преимущества систем анализа уязвимостей:

- Анализ уязвимостей имеет важное значение как часть системы мониторинга безопасности, позволяя определять проблемы в системе, которые не может определить IDS.
- Системы анализа уязвимостей имеют возможность выполнять относящееся к безопасности тестирование, которое может использоваться для документирования состояния безопасности систем в момент начала функционирования программы безопасности или для переустановки базовых функций безопасности всякий раз, когда происходили существенные изменения.
- Когда системы анализа уязвимостей используются регулярно, они могут надежно обнаруживать изменения в состоянии безопасности системы, оповещая администраторов безопасности о проблемах, которые требуют решения.

Недостатки и проблемы систем анализа уязвимостей:

- Некоторые network-based проверки, особенно касающиеся DoS-атак, могут разрушить систему, которую они тестируют.
- При выполнении оценки уязвимостей в сетях, в которых работают системы обнаружения проникновений, IDS могут блокировать проведение последующих оценок. Хуже того, регулярные network-based оценки могут "обучить" некоторые IDS, основанные на обнаружении аномалий, игнорировать реальные атаки.

8.4 Развитие систем распознавания вторжений

8.4.1 Развитие практических аспектов СОВ

Коммерческое использование СОВ находится в стадии формирования. Некоторые коммерческие СОВ получили негативную публичную оценку за большое число ложных срабатываний, неудобные интерфейсы управления и получения отчетов, огромное количество отчетов об атаках, плохое масштабирование и плохую интеграцию с системами сетевого управления. Тем не менее потребность в хороших СОВ возрастает, поэтому с большой вероятностью эти проблемы будут успешно решаться в ближайшее время.

Ожидается, что улучшение качества функционирования СОВ будет осуществляться аналогично антивирусному ПО. Раннее антивирусное ПО создавало большое число ложных тревог при нормальных действиях пользователя и не определяло все известные вирусы. Однако сейчас положение

существенно улучшилось, антивирусное ПО стало прозрачным для пользователей и достаточно эффективным.

Более того – очень вероятно, что основные возможности СОВ скоро станут ключевыми в сетевой инфраструктуре (такой как роутеры, мосты и коммутаторы) и в операционных системах. При этом, скорее всего, разработчики СОВ сфокусируют свое внимание на решении проблем, связанных с масштабируемостью и управляемостью СОВ.

Имеются также и другие тенденции, которые, скорее всего, будут влиять на функциональности СОВ следующего поколения. Существует заметное движение в сторону аппаратно-программных (appliance) решений для СОВ. Также вероятно, что в будущем некоторые функции определения соответствия шаблону могут быть реализованы в аппаратуре, что увеличит скорость обработки.

Наконец, механизмы, связанные с управлением рисками в области сетевой безопасности, будут оказывать влияние на требования к СОВ.

8.4.2 Развитие теоретических аспектов СОВ

Дальнейшие направления совершенствования связаны с внедрением в теорию и практику СОВ общей теории систем, методов синтеза и анализа информационных систем, конкретного аппарата теории распознавания образов. Эти разделы теории предполагают получение конкретных методов исследования для области систем СОВ.

До настоящего времени не описана СОВ как подсистема информационной системы в терминах общей теории систем. Необходимо обосновать показатель качества СОВ, элементный состав СОВ, ее структуру и взаимосвязи с информационной системой.

В связи с наличием значительного количества факторов различной природы, слаженная работа информационной системы и СОВ имеет вероятностный характер. Вследствие этого актуальным является обоснование вида вероятностных законов конкретных параметров функционирования. Особо следует выделить задачу обоснования функции потерь информационной системы, задаваемую в соответствии с ее целевой функцией на области параметров функционирования системы. При этом целевая функция должна быть определена не только на экспертном уровне, но и в соответствии с совокупностью параметров функционирования всей информационной системы и задачами, возложенными на нее. В таком случае показатель качества СОВ будет определяться как один из параметров, влияющих на целевую функцию, а его допустимые значения – допустимыми значениями функции потерь.

После обоснования законов и функций, реальной задачей является получение оптимальной структуры СОВ в виде совокупности математических операций с помощью формализованных методов. Таким образом, может быть решена задача синтеза структуры СОВ. На основе полученных математических операций можно будет рассчитать зависимости показателей качества функционирования СОВ от параметров ее функционирования, а также от параметров функционирования информационной системы, то есть, будет возможен реальный анализ качества функционирования СОВ.

Сложность применения формализованного аппарата анализа и синтеза информационных систем к СОВ заключается в том, что конкретные реализации информационного комплекса и его подсистемы - СОВ состоят из разнородных элементов, которые могут описываться различными разделами теории (системами массового обслуживания, конечными автоматами, теорией вероятности, теорией распознавания образов и т.д.), то есть, рассматриваемый объект исследования является составным.

В результате, математические модели, по-видимому, можно получить только для отдельных составных частей СОВ, что затрудняет анализ и синтез СОВ в целом. Однако, дальнейшая конкретизация применения формализованного аппарата анализа и синтеза позволит оптимизировать СОВ.

На основе изложенного можно сделать вывод о наличии в практической среде значительного опыта решения проблем обнаружения вторжений. Применяемые СОВ в значительной степени основаны на эмпирических схемах процесса обнаружения вторжений. Дальнейшее совершенствование СОВ связано с конкретизацией методов синтеза и анализа сложных систем, теории распознавания образов в применении к СОВ.

9 Проектирование безопасности БД

9.1 Основные понятия

Безопасное программное обеспечение Безопасное программное обеспечение - программное обеспечение, разработанное с использованием совокупности мер, направленных на предотвращение появления и устранение уязвимостей программы.

Правила безопасности Политика безопасности — это совокупность норм и правил, определяющих принятые в организации меры по обеспечению безопасности информации, связанной с деятельностью организации. Только человек, четко осознающий цели организации и условия ее функционирования, может определить, какую информацию необходимо защищать и насколько существенными могут стать потери от несанкционированного распространения, искажения или разрушения информации. После того как политика безопасности определена, должен решаться вопрос о технологии ее реализации в автоматизированном контуре. Для реализации сформулированных в терминах естественного языка правил и норм политики безопасности необходимо использовать (или разработать) некоторую формальную модель, которая допускает эффективное программирование на каком-либо формальном языке. Наибольшее распространение в настоящее время получили две базовые модели безопасности данных: дискреционная и мандатная.

9.2 Методология проектирования

9.2.1 Отличия в проектировании безопасных ОС и СУБД

Системы управления базами данных (СУБД), как и операционные системы, содержат комбинацию сервисов безопасности, однако, в отличие от ОС, не являются самодостаточными. СУБД используют механизмы и функции ОС. Такая двухуровневость ведет к появлению специфических угроз и требует привлечения соответствующих средств противодействия. Например, базы данных располагаются в файлах или на дисках, управляемых ОС; следовательно, к объектам БД можно обратиться как штатными средствами СУБД, так и с помощью механизмов ОС, получив доступ к файлу или устройству. Подобные возможности должны учитываться в профиле защиты для СУБД. Необходимо реализовать хотя бы один из двух механизмов аутентификации: внешний (аутентификация средствами ОС) или внутренний (аутентификация средствами СУБД). Еще одно проявление упомянутой выше двухуровневости - предположение безопасности базовой конфигурации, состоящее в том, что базовая система (операционная система, и/или сетевые сервисы безопасно-

сти, и/или специальное программное обеспечение) установлены, сконфигурированы и управляются безопасным образом. Аналогичную направленность имеют цели безопасности для среды, предусматривающие, что базовая система должна обеспечить механизмы управления доступом, которые позволят защитить от несанкционированного доступа все связанные с СУБД файлы; кроме того, ОС предоставит средства для изоляции функций безопасности и защиты процессов СУБД. Однако в распределенной информационной системе уровень безопасности может поддерживаться не только средствами базовой ОС, но и архитектурно, путем разнесения компонентов СУБД по узлам сети и использования межсетевых экранов.

9.2.2 Основные требования к безопасности СУБД

Можно выделить меры обеспечения безопасности СУБД зависящие и независящие от данных. **Не зависящими** от данных можно назвать следующие требования:

- **Функционирование в доверенной среде.** Под доверенной средой следует понимать инфраструктуру предприятия и ее защитные механизмы, обусловленные политиками безопасности. Таким образом, речь идет о функционировании СУБД в соответствии с правилами безопасности, применяемыми и ко всем прочим системам предприятия.

- **Организация физической безопасности файлов данных.** Требования к физической безопасности файлов данных СУБД в целом не отличаются от требований, применяемых к любым другим файлам пользователей и приложений.

- **Организация безопасной и актуальной настройки СУБД.** Данное требование включает в себя общие задачи обеспечения безопасности, такие как своевременная установка обновлений, отключение неиспользуемых функций или применение эффективной политики паролей.

Следующие требования можно назвать **зависящими** от данных:

- **Безопасность пользовательского ПО.** Сюда можно отнести задачи построения безопасных интерфейсов и механизмов доступа к данным.

- **Безопасная организация и работа с данными.** Вопрос организации данных и управления ими является ключевым в системах хранения информации. В эту область входят задачи организации данных с контролем целостности и другие, специфичные для СУБД проблемы безопасности. Фактически эта задача включает в себя основной объем зависящих от данных уязвимостей и защиты от них.

9.2.3 Независимые принципы целостности данных

В работах Кларка и Вильсона определены девять абстрактных теоретических принципов, выполнение которых позволит обеспечить целостность данных:

- **корректность транзакций;** По первому принципу данные могут изменяться только посредством «корректных» транзакций. Прямое (произвольным образом) изменение данных не допускается. В свою очередь корректность транзакций должна быть некоторым способом доказана.

- **авторизация пользователей;** Второй принцип гласит, что изменение данных может осуществляться только авторизованными пользователями, имеющими определенные привилегии.
- **минимизация привилегий;** Минимальность привилегий подразумевает, что пользователи (в конечном счете, субъекты) должны быть наделены теми и только теми привилегиями, которые минимально необходимы для выполнения тех или иных действий.
- **разграничение функциональных обязанностей;** Разграничение функциональных обязанностей подразумевает организацию работы с данными таким образом, что в каждой из ключевых стадий, составляющих единый критически важный с точки зрения целостности процесс, необходимо участие различных пользователей. Этим гарантируется, что один пользователь не может выполнить весь процесс целиком (или даже две его стадии) с тем, чтобы нарушить целостность данных.
- **аудит произошедших событий;** Аудит произошедших событий (включая возможность восстановления полной картины происшедшего) является превентивной мерой в отношении потенциальных нарушителей и позволяет восстановить данные в случае их повреждения.
- **объективный контроль;** Принцип объективного контроля также является одним из краеугольных камней политики контроля целостности. Суть данного принципа заключается в том, что контроль целостности данных имеет смысл лишь тогда, когда эти данные отражают реальное положение вещей. Очевидно, что нет смысла заботиться о целостности данных, связанных с размещением боевого арсенала, который уже отправлен на переплавку. В связи с этим Кларк и Вильсон указывают на необходимость регулярных проверок, целью которых является выявление возможных несоответствий между защищаемыми данными и объективной реальностью, которую они отражают.
- **управление передачей привилегий;** Управление передачей привилегий необходимо для эффективной работы всей политики безопасности. Если схема назначения привилегий неадекватно отражает организационную структуру предприятия или не позволяет администраторам безопасности гибко манипулировать ею для обеспечения эффективности производственной деятельности, защита становится тяжким бременем и провоцирует попытки обойти ее там, где она мешает «нормальной» работе.
- **эффективное применение механизмов защиты;** В основу восьмого принципа контроля целостности заложен ряд идей, призванных обеспечить эффективное применение имеющихся механизмов обеспечения безопасности. На практике зачастую оказывается, что предусмотренные в системе механизмы безопасности или некорректно используются, или полностью игнорируются системными администраторами.
- **простота использования защитных механизмов;** Простота использования защитных механизмов подразумевает, что самый безопасный путь эксплуатации системы будет также наиболее простым, и наоборот, самый простой - наиболее защищенным.

9.2.4 Модель авторизации в System R

Подход к авторизации в System R основан на списках доступа, с возможностью отмены доступа. В System R нет администратора базы данных - суперпользователя в обычном смысле. Любой

пользователь может создавать таблицы. Когда пользователь создал таблицу он становится ее суперпользователем. Если он хочет поделиться своей таблицей с другими пользователями, он может использовать команду GRANT для предоставления различных привелегий в этой таблице различным пользователям, таким образом составляя Access List для пользователей, имеющих доступ к таблице, состоящий из UID пользователей.

9.2.5 Архитектура безопасной СУБД

Многоуровневые защищенные архитектуры РСУБД можно разделить на два общих типа, в зависимости от того, осуществляется ли принудительный контроль доступа самой РСУБД или делегируется доверенной операционной системе. Эти два основных типа - это архитектура Вудс-Холла и архитектура TrustedSubjects.

Архитектура Вудс-Холла Архитектура Вудс-Холла предполагает, что для доступа к данным используется ненадежная стандартная РСУБД и что доверенный код разрабатывается вокруг этой РСУБД для обеспечения общей безопасной РСУБД. Их можно разделить на две категории: ядерные (kernel) архитектуры и распределенные архитектуры.

Ядерная(kernel) архитектура Ядерная архитектура использует надежную операционную систему и несколько копий готовой СУБД, где каждая копия связана с некоторым доверенным интерфейсом. Каждая пара (доверенный интерфейс, СУБД) связана с определенным уровнем безопасности. Доверенная операционная система применяет свою политику полного контроля доступа ко всем доступам СУБД к объектам СУБД. Это гарантирует, что данные с разными уровнями безопасности хранятся отдельно, и что каждая копия СУБД получает доступ к данным, авторизованным для соответствующего уровня безопасности. Последнее возможно, потому что многоуровневая база данных разбита на несколько одноуровневых баз данных, каждая из которых представляет собой фрагмент концептуальной многоуровневой базы данных. Каждый фрагмент хранится в одноуровневом объекте операционной системы (например, в файле), который помечен операционной системой на соответствующем уровне безопасности, и, таким образом, к нему можно получить доступ только в соответствии с политикой MAC операционной системы.

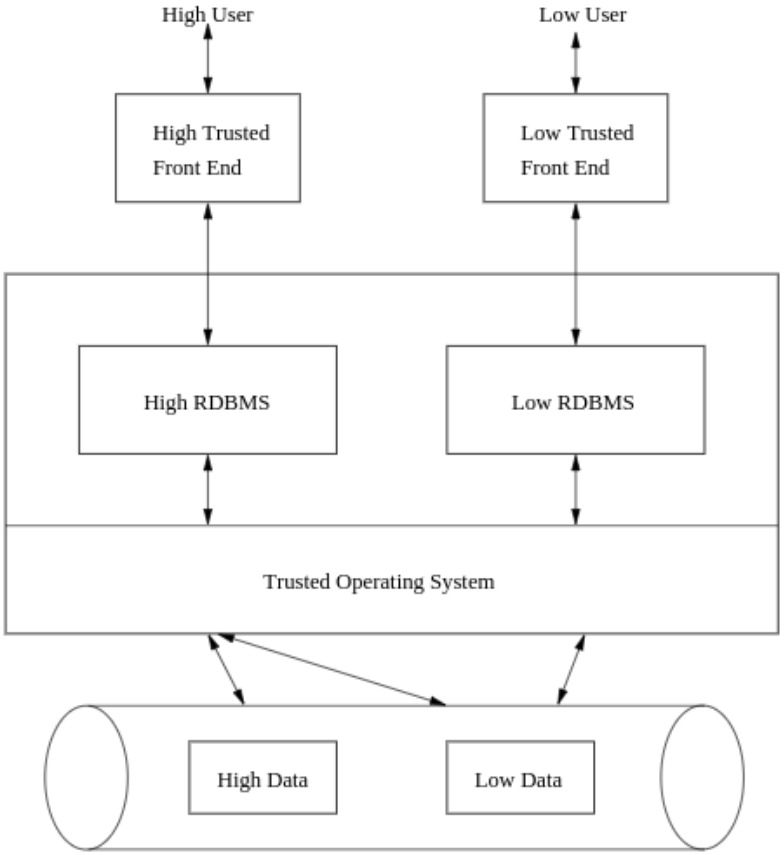


Figure 1: Multilevel secure kernelized RDBMS architecture.

Рис. 25: Ядерная архитектура

На картинке изображена Ядерная архитектура, в которой одна РСУБД связана с уровнем безопасности «Высокий», а другая РСУБД связана с уровнем безопасности «Низкий». РСУБД, связанная с уровнем безопасности «Высокий», имеет доступ как к фрагменту базы данных с высоким уровнем безопасности, так и к фрагменту базы данных с низким уровнем безопасности. А РСУБД, ассоциированная с уровнем безопасности «Низкий», имеет доступ только к фрагменту базы данных с низким уровнем безопасности. Преимущество этой архитектуры заключается в том, что данные на разных уровнях безопасности изолированы. Еще одно преимущество состоит в том, что при условии, что надежная операционная система уже готова, эта архитектура должна минимизировать количество времени и усилий для установки СУБД. Однако эта архитектура приводит к дополнительным накладным расходам, поскольку доверенной операционной системе необходимо разделять данные на разных уровнях безопасности при добавлении в базу данных, а также может потребоваться совершать трудоемкие операции объединения данных с разных уровней безопасности.

Распределенные архитектуры Распределенная (или реплицированная) архитектура - это вариант ядерной архитектуры. Он использует несколько копий доверенного интерфейса и СУБД, каждая из которых связана со своим собственным хранилищем базы данных. В этой архитектурной схеме СУБД на уровне безопасности содержит реплику каждого элемента данных, к которому субъект на уровне может получить доступ. Таким образом, когда данные извлекаются повторно, СУБД извлекает их только из своей собственной базы данных. Еще одно преимущество этой архитектуры состоит в том, что данные физически разделены на отдельные аппаратные базы данных. Однако эта схема приводит к дополнительным накладным расходам, когда данные обновляются, поскольку различные реплики должны быть синхронизированы.

Архитектура TrustedSubjects Архитектура TrustedSubjects - это схема, которая содержит надежную СУБД и надежную операционную систему. Согласно этой архитектуре, политика обязательного контроля доступа обеспечивается самой СУБД. Объекты базы данных (например, таблица) хранятся в объектах операционной системы (например, в файле) с наивысшим уровнем безопасности. Таблица базы данных может содержать хранить строки с разными уровнями безопасности. Такие строки различаются на основе их уровня безопасности, который явно сохраняется с каждой строкой. Эта архитектура называется TrustedSubjects, потому что РСУБД имеет привилегию нарушать политику MAC операционной системы при доступе к объектам базы данных. Например, когда пользователь с низким уровнем безопасности запрашивает таблицу базы данных, к объекту операционной системы, в котором хранится эта таблица, оказывается, что это является нарушением MAC-политики операционной системы. Но РСУБД способна возвращать пользователям только те строки, для которых он или она авторизованы в соответствии с политикой MAC. Преимущество этой архитектуры состоит в том, что СУБД имеет доступ ко всем уровням данных в одно и то же время, что сводит к минимуму извлечение и обработку обновлений. Однако эта архитектура приводит к созданию специальной РСУБД, которая требует разработки и проверки большого количества доверенного кода наряду с обычными функциями РСУБД. Недостатком можно считать недостаточный уровень гарантии аппаратной изоляции объектов. Также сложно доказать, что доверенное программное обеспечение, используемое для изоляции объектов (например, потоков данных с разными уровнями безопасности), работает правильно, не допуская потока данных с высоким уровнем безопасности для пользователей с низким уровнем безопасности.

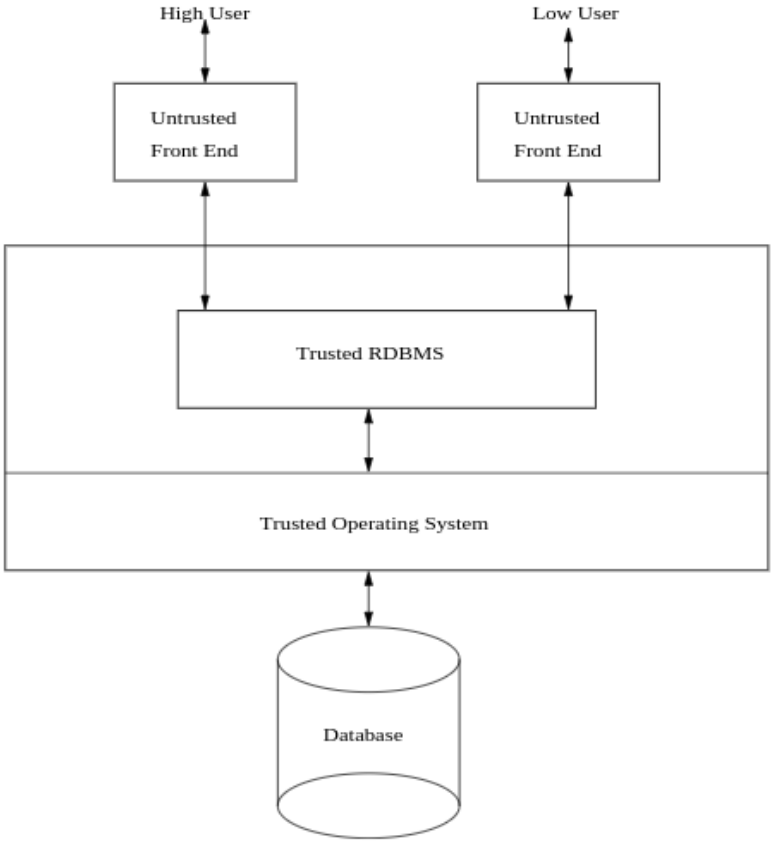


Figure 2: Multilevel secure trusted subject RDBMS architecture.

Рис. 26: архитектура TrustedSubjects

9.3 Проектирование безопасных БД

Разработать универсальную защищенную систему баз данных скорее всего нереально. При любом разумном методе измерения уровня защищенности этот уровень является неубывающей функцией от затрат на построение системы защиты. В любом практическом случае, когда существуют ограничения на бюджет системы защиты, существует и некоторый предельный уровень информационной безопасности, который теоретически может быть достигнут.

В настоящее время отсутствует общепринятая методология разработки защищенных автоматизированных информационных систем и, в частности, систем баз данных. В подобных случаях традиционно используется подход, основанный на анализе лучшего мирового опыта решения некоторого класса проблем и формулировании руководящих принципов построения соответствующих систем, концентрирующих накопленный опыт. Именно таким образом для проблемы обеспечения безопасности информационных систем разрабатывался британский стандарт BS 7799 и созданный на его основе международный стандарт ISO 17799.

Анализ наиболее успешных решений в области обеспечения информационной безопасности баз данных позволил сформулировать несколько полезных принципов, которыми можно руководствоваться при проектировании систем защиты:

- **Экономическая оправданность механизмов защиты.** Предписывает использование простейшего из всевозможных вариантов проекта, который обеспечивает достижение желаемой цели. Хотя этот принцип относится ко многим аспектам проектирования систем, он наиболее пригоден при разработке механизмов защиты, так как ошибки проектирования и реализации, которые ведут к неконтролируемым способам доступа к данным, могут быть не замечены в ходе нормального использования системы. Строгое соблюдение этого принципа приводит к применению на практике таких методов, как проверка «строка за строкой» программных средств и физическая проверка аппаратных средств, реализующих механизмы защиты.

- **Открытое проектирование.** Технология систем защиты не должна базироваться на «секретных» алгоритмах. Этот принцип широко используется при проектировании безопасных систем и сетей связи. Высокое качество систем защиты обеспечивается не недостатком знаний у возможных нарушителей, а использованием широко опробованных (как правило, открытых) стандартов и правильной организацией управления ключевой информацией. Использование алгоритмов, основанных на открытых стандартах в области информационной безопасности, повышает степень доверия пользователей к системе защиты и формирует правильную психологическую установку на необходимость внимательности и аккуратности при работе с ключевой информацией.

- **Распределение полномочий между различными субъектами в соответствии с правилами организации.** Состоит в том, что для критически важных приложений целесообразно использовать многокомпонентные схемы доступа к данным. То есть для выполнения соответствующей операции необходимо провести аутентификацию нескольких ее обязательных участников. Физический аналог этого принципа можно наблюдать в конструкциях банковских сейфов, когда для того, чтобы открыть сейф, необходимо наличие двух ключей, которые обычно хранятся у различных людей. Ясно, что механизмы защите, требующие двух ключей для доступа к информации, являются более устойчивыми, чем механизмы, которые разрешают доступ на основе предъявления единственного ключа. В то же время подобные многокомпонентные процедуры требуют больших затрат и, как правило, более сложных процедур управления ключами (включая хранение резервной копии). При проектировании многокомпонентных схем доступа за прообраз берется существующая в организации практика. Действительно, переход в автоматизированном контуре на более сложные, чем использовались «в доавтоматизированную эру», технологии может вызвать психологический дискомфорт и различные формы скрытого саботажа системы.

- **Минимально возможные привилегии для пользователей и администраторов.** Предписывает, чтобы каждый пользователь (процесс) системы оперировали с данными, используя наименьший из возможных набор привилегий, необходимых для выполнения конкретной функции. Применение данного принципа нацелено на минимизацию ущерба, который может быть нанесен в случае сбоя, ошибки программного обеспечения или компрометации элементов системы защиты данных. Принцип минимальных привилегий, используемый при созда-

нии пользователей Oracle, — пример реализации этого принципа. Использование точек входа SYSTEM и, особенно, SYS должно быть предметом особого регламента. Хорошей практикой является использование администратором безопасности нескольких точек входа: обычной, с набором привилегий, достаточным для выполнения основных работ, и особой (типа SYSTEM), используемой только при возникновении необходимости выполнения действий, требующих высоких привилегий.

- **Управляемость системы при возникновении отказов и сбоев.** Проектирование информационной системы, реализованной на базе СУБД, должно осуществляться в предположении, что ошибки операционной системы и СУБД, а также сбои аппаратуры неизбежны. При создании системы возможность реализации таких событий должна быть учтена: при проектировании процедур и функций должны быть обработаны все исключительные ситуации, при обработке данных, содержащих конфиденциальную информацию, должны быть минимизированы риски восстановления этих данных по дампам оперативной памяти и содержимому временных файлов и т. п. Также должны быть разработаны документы, регламентирующие действия участников процесса обработки данных (как пользователей, так и обслуживающего персонала) при возникновении нештатных ситуаций. Персонал должен проходить регулярные инструктажи и тренинги по обучению действиям в нештатных ситуациях, с иерархией передачи данных.

- **Психологическая приемлемость работы средств защиты данных.** Взаимодействие людей с системой (и подсистемой защиты) не должно быть сложным. Пользователи должны шаблонно и автоматически применять имеющиеся механизмы защиты. Чрезмерное усложнение механизмов защиты может вызывать их внутреннее неприятие и побуждать к использованию различных форм скрытого саботажа. Осознанное принятие используемых средств и методов обеспечения информационной безопасности и оценка комплекса применяемых мер как необходимых приводит к уменьшению числа ошибок пользователей. В этом случае аномальное поведение потенциального нарушителя становится более заметным и проще устанавливается. Принцип психологической приемлемости является важным при выборе процедур аутентификации и модели управления доступом.

9.3.1 Фазы проектирования безопасных БД (по DoD)

Министерство обороны (Department of Defence) предложило методологию проектирования, использования и вывода системы из эксплуатации безопасных информационных систем, в том числе безопасных БД. Стадии жизненного цикла информационной системы, наиболее часто используемые Министерством обороны, показаны на рис. 6.

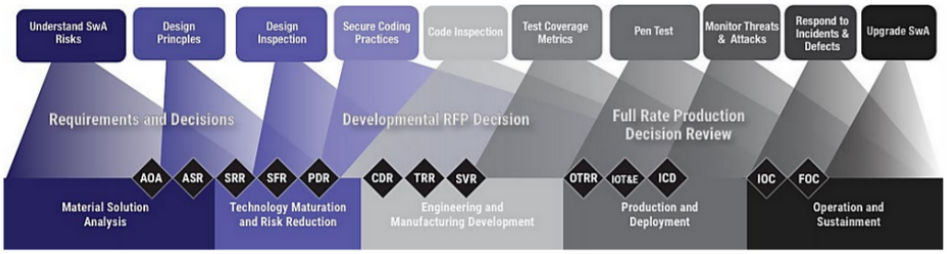


Рис. 27: Стадии жизненного цикла по DoD

Как показано на рис. 6, основная часть жизненного цикла информационной системы включает пять этапов: работа с требованиями, проектирование архитектуры информационной системы, реализация информационной системы, тестирование и введение в эксплуатацию. На этапе требований происходит анализ требований заказчика, формулирование политик безопасности и требований к разрабатываемой информационной системе. На этапе проектирования архитектуры на основе набора требований, представленного в функциональной форме, составляется концептуальная модель системы, на основе концептуальной – логическая, на основе логической – физическая.

9.3.2 Предварительный анализ

Анализ требований является одним из первых этапов процесса системного проектирования и в некоторой степени представляет собой интерфейс между внутренними действиями и внешними источниками, предоставляющими входные данные. В процессе анализа требований исследуются, оцениваются и преобразуются внешние входные данные в набор функциональных требований, политик безопасности и требований к производительности, которые будут являться основой для последующего концептуального, логического и физического проектирований. Цель анализа требований – определение требований к системе.

Анализ миссии системы, анализ среды использования системы определяют потребности клиента и формулируют их в терминах, которые могут быть использованы для определения функций системы, требований к производительности или проектных ограничений. Такой анализ определяет функциональные требования к системе, уточняет требования к характеристикам или проектированию. По мере продвижения этой деятельности исходные предположения и выводы сверяются с меняющимися деталями. Обычно это приводит к некоторому изменению исходного мышления заказчика, может даже отражаться на его потребностях, некоторые из которых могут оказаться непрактичными или чрезмерно дорогостоящими.

Результатом анализа требований является набор функциональных определений верхнего уровня и сопутствующих требований к производительности и архитектуре, которые становятся отправной точкой концептуального проектирования. Анализ требований проводится итеративно, цикл анализа требований служит для уточнения требований и инициирования повторной оценки, чтобы определить, насколько жесткими являются требования к элементам системы. Позже подробные характеристики системы сравниваются с установленными требованиями, чтобы убедиться, что они выполняются. На этом этапе обычно мало изменений в требованиях из-за обратной связи проверки, но иногда некоторые незначительные изменения рассматриваются, когда отдача значительна.

9.3.3 Требования и политики безопасности

Политика безопасности — это совокупность норм и правил, определяющих принятые в организации меры по обеспечению безопасности информации, связанной с деятельностью организации. Только человек, четко осознающий цели организации и условия ее функционирования, может определить, какую информацию необходимо защищать и насколько существенными могут стать потери от несанкционированного распространения, искажения или разрушения информации. После того как политика безопасности определена, должен решаться вопрос о технологии ее реализации в автоматизированном контуре. Для реализации сформулированных в терминах естественного языка правил и норм политики безопасности необходимо использовать (или разработать) некоторую формальную модель, которая допускает эффективное программирование на каком-либо формальном языке. Наибольшее распространение в настоящее время получили две базовые модели безопасности данных: дискреционная и мандатная.

Цель формализации политики безопасности для информационной системы — ясное изложение взглядов руководства организации на существо угроз информационной безопасности организации и технологий обеспечения безопасности ее информационных ресурсов. Политика безопасности обычно состоит из двух частей: общих принципов и конкретных правил работы с информационными ресурсами, то есть требований, и, в частности, с базами данных для различных категорий пользователей. Политика безопасности – это всегда некоторый компромисс между желаемым уровнем защищенности ресурсов информационной системы, удобством работы с системой и затратами средств, выделяемых на ее эксплуатацию.

Политика безопасности должна быть оформлена документально на нескольких уровнях управления. На уровне управляющего высшего звена руководства должен быть подготовлен и утвержден документ, в котором определены цели политики безопасности, структура и перечень решаемых задач и ответственные за реализацию политики. Основной документ должен быть детализирован администраторами безопасности информационных систем (управляющими среднего звена) с учетом принципов деятельности организации, соотношения важности целей, и наличия ресурсов. Детальные решения должны включать ясные определения методов защиты технических и информационных ресурсов, а также инструкции, определяющие поведение сотрудников в конкретных ситуациях.

В руководстве по компьютерной безопасности, разработанном национальным институтом стандартов и технологий США (National Institute of Standards and Technology — NIST), рекомендовано включать в описание политики безопасности следующие разделы:

- **Предмет политики.** В разделе должны быть определены цели и причины разработки политики, область ее применения в конкретном фрагменте системы документооборота организации. Должны быть ясно сформулированы задачи, решаемые с использованием информационных систем, которые затрагивает данная политика. При необходимости могут быть сформулированы термины и определения, используемые в остальных разделах.

- **Описание позиции организации.** В этом разделе необходимо ясно описать характер информационных ресурсов организации, перечень допущенных к информационным ресурсам лиц и процессов и порядок получения доступа к информационным ресурсам организации.

- **Применимость.** В разделе может быть уточнен порядок доступа к данным ИС, определены ограничения или технологические цепочки, применяемые при реализации политики безопас-

ности.

- **Роли и обязанности.** В разделе определяются ответственные должностные лица и их обязанности в отношении разработки и внедрения различных элементов политики. Обычно определяются обязанности администратора безопасности данных (отвечает за содержательную сторону предоставления доступа к информационным ресурсам организации), администратора баз данных (определяет техническую реализацию механизмов разграничения доступа), администратора локальной сети, операторов.

- **Соблюдение политики.** В разделе описываются права и обязанности пользователей ИС. Необходимо явное описание и документированное знакомство пользователей с перечнем недопустимых действий при осуществлении доступа к информационным ресурсам организации и наказания за нарушения режимных требований. Должна быть ясно определена технология фиксации фактов нарушения политики безопасности и применения административных мер воздействия к нарушителям.

Для эффективной реализации политика безопасности должна быть понятной всем пользователям информационных систем организации. Возможна подготовка презентаций и проведение семинаров с разъяснением основных положений и практических технологий реализации политики безопасности. Новые сотрудники организации должны быть ознакомлены или обучены конкретным правилам и технологиям доступа к ресурсам ИС, реализованным в соответствии с принятой политикой безопасности. Целесообразно проводить контрольные проверки действий сотрудников с обсуждением результатов.

Эффективное проведение политики безопасности возможно только, если она согласована с существующими приказами и общими задачами организации. Основным способом координации политики безопасности с действующими нормами организации является ее согласование с заинтересованными подразделениями в ходе разработки.

Комплект документов, представляющий основные решения организации по реализации политики безопасности, должен включать:

- документацию, определяющую используемые подходы к оцениванию и управлению рисками для организации в целом и при необходимости конкретных подразделений;

- обоснование принятых решений по выбору средств защиты для рассматриваемой информационной системы;

- формальное описание процедуры определения допустимого уровня остаточного риска;

- директиву, определяющую процедуру проверки режима информационной безопасности и журналов, в которых фиксируются результаты проверки (документы необходимы для осуществления проверки эффективности реализации средств обеспечения информационной безопасности, осуществления их контроля качества и правильности использования);

- документацию, регламентирующую процессы обслуживания и администрирования информационных систем;

- документацию по подготовке периодических проверок по оцениванию и управлению рисками;

- документ «Ведомость соответствия», включающий сведения по организации системы управления информационной безопасностью и регистрации средств управления безопасностью;
- контрмеры для противодействия выявленным рискам.

Успех проведения в жизнь политики безопасности больше зависит от усилий и опытности людей, реализующих политику, чем от сложных программно-технических средств контроля.

9.3.4 Концептуальное проектирование

Цель этапа концептуального проектирования – создание концептуальной модели данных исходя из представлений пользователей о предметной области. Для ее достижения выполняется ряд последовательных процедур.

1. **Определение сущностей и их документирование.** Для идентификации сущностей определяются объекты, которые существуют независимо от других. Такие объекты являются сущностями. Каждой сущности присваивается осмысленное имя, понятное пользователям. Имена и описания сущностей заносятся в словарь данных. Если возможно, то устанавливается ожидаемое количество экземпляров каждой сущности.

2. **Определение связей между сущностями и их документирование.** Определяются только те связи между сущностями, которые необходимы для удовлетворения требований к проекту базы данных. Устанавливается тип каждой из них. Выявляется класс принадлежности сущностей. Связям присваиваются осмысленные имена, выраженные глаголами. Развернутое описание каждой связи с указанием ее типа и класса принадлежности сущностей, участвующих в связи, заносится в словарь данных.

3. **Создание ER-модели предметной области.** Для представления сущностей и связей между ними используются ER-диаграммы. На их основе создается единый наглядный образ моделируемой предметной области – ER-модель предметной области.

4. **Определение атрибутов и их документирование.** Выявляются все атрибуты, описывающие сущности созданной ER-модели. Каждому атрибуту присваивается осмысленное имя, понятное пользователям. О каждом атрибуте в словарь данных помещаются следующие сведения:

   - имя атрибута и его описание;
   - тип и размерность значений;
   - значение, принимаемое для атрибута по умолчанию (если такое имеется);
   - может ли атрибут иметь NULL-значения;
   - является ли атрибут составным, и если это так, то из каких простых атрибутов он состоит. Например, атрибут "Ф.И.О. клиента"может состоять из простых атрибутов "Фамилия "Имя "Отчество а может быть простым, содержащим единые значения, как-то "Сидорский Евгений Михайлович". Если пользователь не нуждается в доступе к отдельным элементам "Ф.И.О. то атрибут представляется как простой;
   - является ли атрибут расчетным, и если это так, то как вычисляются его значения.

5. **Определение значений атрибутов и их документирование.** Для каждого атрибута сущности, участвующей в ER-модели, определяется набор допустимых значений и ему присваивается имя. Например, атрибут "Тип счета"может иметь только значения "депозитный "текущий "до востребования "карт-счет". Обновляются записи словаря данных, относящиеся к атрибутам, – в них заносятся имена наборов значений атрибутов.

6. **Определение первичных ключей для сущностей и их документирование.** На этом шаге руководствуются определением первичного ключа – как атрибута или набора атрибутов сущности, позволяющего уникальным образом идентифицировать ее экземпляры. Сведения о первичных ключах помещаются в словарь данных.

7. **Обсуждение концептуальной модели данных с конечными пользователями.** Концептуальная модель данных представляется ER-моделью с сопроводительной документацией, содержащей описание разработанной модели данных. Если будут обнаружены несоответствия предметной области, то в модель вносятся изменения до тех пор, пока пользователи не подтвердят, что предложенная им модель адекватно отображает их личные представления.

9.3.5 Логическое проектирование

Цель этапа логического проектирования – преобразование концептуальной модели на основе выбранной модели данных в логическую модель, не зависимую от особенностей используемой в дальнейшем СУБД для физической реализации базы данных. Для ее достижения выполняются следующие процедуры:

1. **Выбор модели данных.** Чаще всего выбирается реляционная модель данных в связи с наглядностью табличного представления данных и удобства работы с ними.

2. **Определение набора таблиц исходя из ER-модели и их документирование.** Для каждой сущности ER-модели создается таблица. Имя сущности – имя таблицы. Причем каждому атрибуту сущности соответствует столбец таблицы. Правила генерации таблиц из ER-диаграмм опираются на два основных фактора – тип связи и класс принадлежности сущности. Устанавливаются связи между таблицами посредством механизма первичных и внешних ключей. Структуры таблиц и установленные связи между ними документируются.

   Изложим правила генерации таблиц из ER-диаграмм, используя пример ER-модели предметной области БАНК, представленной на рис. 7, со следующими наборами атрибутов сущностей предметной области БАНК, представленными на рис. 8.

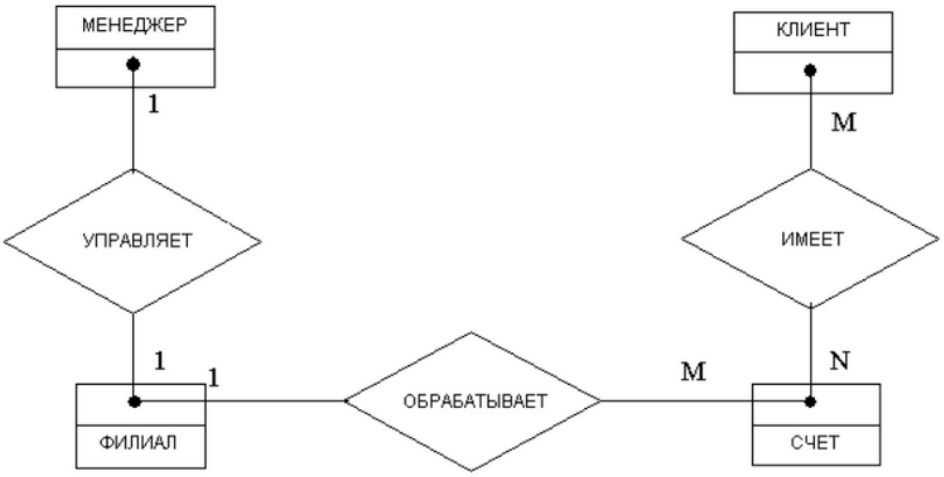


Рис. 28: Пример ER-модели предметной области БАНК

МЕНЕДЖЕР
Номер менеджера (НМ)
Стаж работы (СТАЖ)
Специальность (СПЕЦ)

ФИЛИАЛ
Номер филиала (НФ)
Адрес филиала (АДР_Ф)

СЧЕТ
Номер счета (НС)
Тип счета (ТИП)
Остаток на счете (ОСТ)

КЛИЕНТ
Номер клиента (НК)
Ф.И.О. клиента (ФИО_К)
Социальное положение (СОЦ)
Адрес клиента (АДР_К)

Рис. 29: Наборы атрибутов сущностей предметной области БАНК

Для связи типа 1:1 существуют три отдельных правила формирования предварительных таблиц из ER-диаграмм.

Правило 1

Если связь типа 1:1 и класс принадлежности обеих сущностей является обязательным, то необходима только одна таблица. Первичным ключом этой таблицы может быть первичный ключ любой из двух сущностей.

На ER-диаграмме связи 1:1, класс принадлежности сущностей МЕНЕДЖЕР, ФИЛИАЛ является обязательным. Тогда согласно правилу 1 должна быть сгенерирована одна таблица

следующей структуры:

МЕНЕДЖЕР–ФИЛИАЛ

НМ	СТАЖ	СПЕЦ	**НФ**	АДР_Ф

Первичным ключом этой таблицы может быть и первичный ключ сущности МЕНЕДЖЕР – НМ.

Правило 2

Если связь типа 1:1 и класс принадлежности одной сущности является обязательным, а другой – необязательным, то необходимо построить таблицу для каждой сущности. Первичный ключ сущности должен быть первичным ключом соответствующей таблицы. Первичный ключ сущности, для которой класс принадлежности является необязательным, добавляется как атрибут в таблицу для сущности с обязательным классом принадлежности.

Представим, что на ER-диаграмме связи 1:1, изображенной на рис. 7, класс принадлежности сущности МЕНЕДЖЕР будет обязательный, а сущности ФИЛИАЛ – необязательный. Тогда согласно правилу 2 должны быть сгенерированы две таблицы следующей структуры:

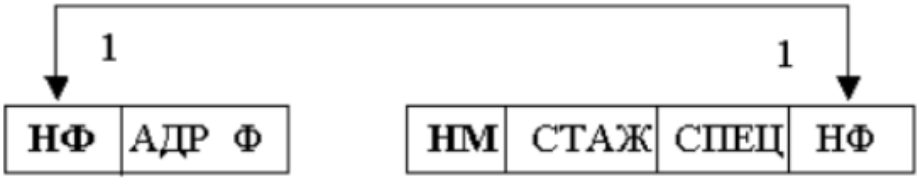
МЕНЕДЖЕР – ФИЛИАЛ

НМ	СТАЖ	СПЕЦ	НФ

ФИЛИАЛ

НФ	АДР_Ф

Сущность с необязательным классом принадлежности (ФИЛИАЛ) именуется родительской, а с обязательным (МЕНЕДЖЕР) – дочерней. Первичный ключ родительской сущности (НФ), помещаемый в таблицу, представляющую дочернюю сущность, называется внешним ключом родительской сущности. Связь между указанными таблицами устанавливается путем связи первичного и внешнего ключа и имеет вид:



Примечание. Если внешний ключ представляет связь 1:1, то должны быть запрещены его дублирующие значения.

Правило 3

Если связь типа 1:1 и класс принадлежности обеих сущностей является необязательным, то необходимо построить три таблицы – по одной для каждой сущности и одну для связи. Первичный ключ сущности должен быть первичным ключом соответствующей таблицы. Таблица для связи среди своих атрибутов должна иметь ключи обеих сущностей.

Представим, что на ER-диаграмме связи 1:1, изображенной на рис. 7, класс принадлежности сущностей МЕНЕДЖЕР, ФИЛИАЛ будет необязательный. Тогда согласно правилу 3 должны быть сгенерированы три таблицы следующей структуры:

МЕНЕДЖЕР

НМ	СТАЖ	СПЕЦ

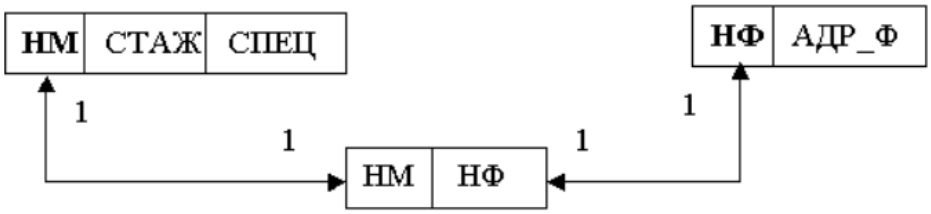
ФИЛИАЛ

НФ	АДР_Ф

МЕНЕДЖЕР–ФИЛИАЛ

НМ	НФ

При этом осуществляется декомпозиция связи 1:1 на две связи 1:1 следующим образом:



Для связи типа 1:M существуют только два правила. Выбор одного из них зависит от класса принадлежности сущности на стороне M. Класс принадлежности сущности на стороне 1 не влияет на выбор.

Правило 4

Если связь типа 1:M и класс принадлежности сущности на стороне M является обязательным, то необходимо построить таблицу для каждой сущности. Первичный ключ сущности должен быть первичным ключом соответствующей таблицы. Первичный ключ сущности на стороне 1 добавляется как атрибут в таблицу для сущности на стороне M.

На ER-диаграмме связи 1:M, представленной на рис. 7, класс принадлежности сущности СЧЕТ является обязательным. Тогда согласно правилу 4 должны быть сгенерированы две таблицы следующей структуры:

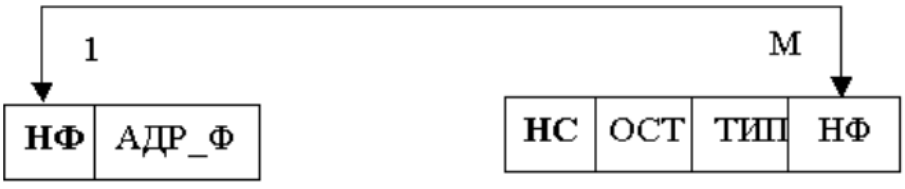
ФИЛИАЛ

НФ	АДР_Ф

СЧЕТ– ФИЛИАЛ

НС	ОСТ	ТИП	НФ

Связь между указанными таблицами будет иметь вид:



Примечание. Если внешний ключ представляет связь 1:M, то должны быть разрешены его дублирующие значения.

Правило 5

Если связь типа 1:M и класс принадлежности сущности на стороне M является необязательным, то необходимо построить три таблицы – по одной для каждой сущности и одну для связи. Первичный ключ сущности должен быть первичным ключом соответствующей таблицы. Таблица для связи среди своих атрибутов должна иметь ключи обеих сущностей.

Представим, что на ER-диаграмме связи 1:M, изображенной на рис. 7, класс принадлежности сущности СЧЕТ является необязательным. Тогда согласно правилу 5 должны быть сгенерированы три таблицы следующей структуры:

ФИЛИАЛ

НФ	АДР_Ф

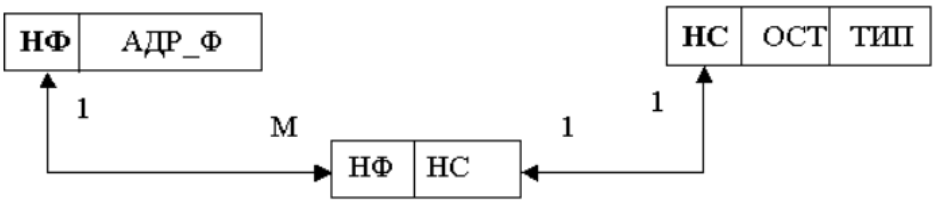
СЧЕТ

НС	ОСТ	ТИП

ФИЛИАЛ – СЧЕТ

НФ	НС

При этом осуществляется декомпозиция связи 1:M на две связи – 1:M и 1:1 – следующим образом



Для связи типа M:N класс принадлежности сущности не имеет значения.

Правило 6

Если связь типа M:N, то необходимо построить три таблицы – по одной для каждой сущности и одну для связи. Первичный ключ сущности должен быть первичным ключом соответствующей таблицы. Таблица для связи среди своих атрибутов должна иметь ключи обеих сущностей.

ER-диаграмма связи M:N имеется на рис. 7. Согласно правилу 6 на основе этой ER-диаграммы должны быть сгенерированы три таблицы следующей структуры:

КЛИЕНТ

НК	ФИО_К	СОЦ_П	АДР_К

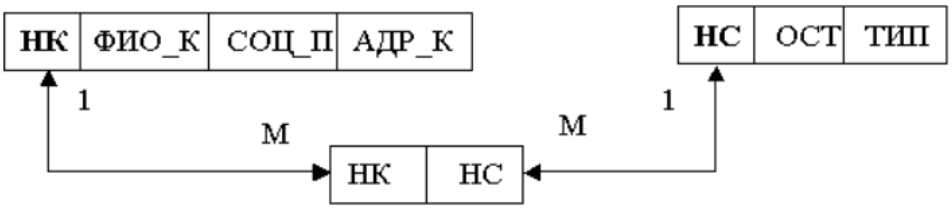
СЧЕТ

НС	ОСТ	ТИП

КЛИЕНТ– СЧЕТ

НК	НС

При этом осуществляется декомпозиция связи M:N на две связи 1:M следующим образом:

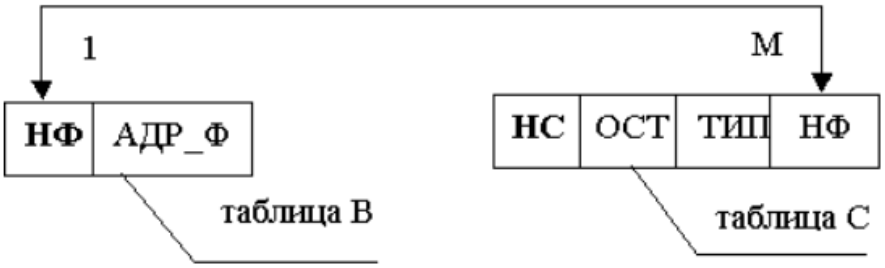


В таблице КЛИЕНТ–СЧЕТ клиенту, имеющему, например, три счета будут соответствовать три строки с одним и тем же номером клиента. А счет, у которого, например, два владельца, представляется двумя строками с различными номерами клиентов, владеющими этим счетом.

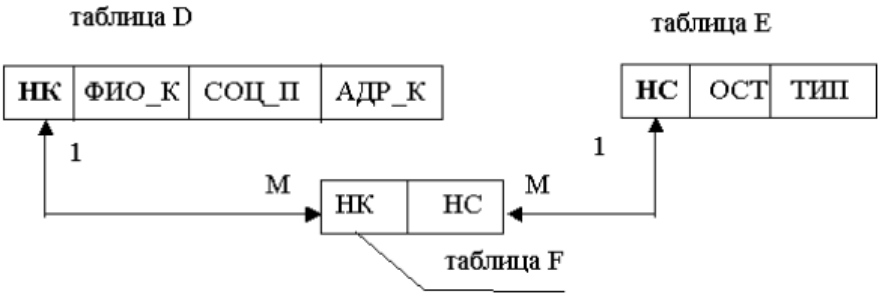
К ER-модели предметной области БАНК, представленной на рис. 7, применимы правила 1, 4, 6. Связь МЕНЕДЖЕР – ФИЛИАЛ представляется (согласно правилу 1) одной таблицей А:

НМ	СТАЖ	СПЕЦ	**НФ**	АДР_Ф

Связь ФИЛИАЛ – СЧЕТ представляется (согласно правилу 4) связью:



Связь КЛИЕНТ – СЧЕТ представляется (согласно правилу 6) связью:



Анализ состава атрибутов полученных таблиц A, B, C, D, E, F показывает, что таблица B является составной частью таблицы A, таблица E – составной частью таблицы C. Поэтому таблицы B, E можно исключить из рассмотрения. Оставшиеся таблицы A, C, D, F можно связать посредством связи первичных и внешних ключей. В результате получим реляционную модель для ER-модели предметной области БАНК.

3. **Нормализация таблиц.** Для правильного выполнения нормализации проектировщик должен глубоко изучить семантику и особенности использования данных. На этом шаге он проверяет корректность структуры таблиц, созданных на предыдущем шаге, посредством применения к ним процедуры нормализации. Она заключается в приведении каждой из таблиц, по крайней мере, к 3НФ. В результате нормализации получается очень гибкий проект базы данных, позволяющий легко вносить в нее нужные расширения.

   Нормализация таблиц - процесс, позволяющий минимизировать избыточность данных. Чтобы пояснить этот процесс, будем исходить из описания предметной области БАНК, представленного на рис. 7, и предположения, что на его основе была разработана база данных, состоящая из следующих двух таблиц:

ФИЛИАЛ

НФ	АДР_Ф	НМ	**НС**	ОСТ	ТИП
511	Ванеева, 6	7	1111 2222 3333	200 350 1000	Д Т Т
513	Солтыса, 3	9	5555 6666	800 14	Т Д

КЛИЕНТ

НК	ФИО_К	СОЦ_П	АДР_К	**НС**
23	Сокол С.С.	Служащий	Садовая, 1	1111 3333
34	Брас Б.Б.	Рабочий	Гая, 9	5555
45	Лань Л.Л.	Служащий	Лесная, 4	2222 6666 1111

1 нормальная форма (1НФ)

Таблица находится в 1НФ, если все ее поля содержат только простые неделимые значения.

Таблицы ФИЛИАЛ и КЛИЕНТ не удовлетворяют требованиям 1НФ. Для приведения их к 1НФ в них надо вставить новые записи следующим образом:

ФИЛИАЛ

НФ	АДР_Ф	НМ	**НС**	ОСТ	ТИП
511	Ванеева, 6	7	1111	200	Д
511	Ванеева, 6	7	2222	350	Т
511	Ванеева, 6	7	3333	1000	Т
513	Солтыса, 3	9	5555	800	Т
513	Солтыса, 3	9	6666	14	Д

КЛИЕНТ

НК	ФИО_К	СОЦ_П	АДР_К	**НС**
23	Сокол С.С.	Служащий	Садовая, 1	1111
23	Сокол С.С.	Служащий	Садовая, 1	3333
34	Брас Б.Б.	Рабочий	Гая, 9	5555
45	Лань Л.Л.	Служащий	Лесная, 4	2222
45	Лань Л.Л.	Служащий	Лесная, 4	6666
45	Лань Л.Л.	Служащий	Лесная, 4	1111

Но полученные таблицы неэффективны, так как содержат много избыточной информации. Необходимо их привести к 2НФ.

2 нормальная форма (2НФ)

Таблица находится в 2НФ, если она удовлетворяет требованиям 1НФ и неключевые поля функционально полно зависят от первичного ключа.

Функциональная зависимость – это понятие, отображающее определенную семантическую связь между полями таблицы. Неключевое поле А функционально полно зависит от первичного ключа, если:

3.1 оно функционально зависит от первичного ключа, т.е. каждой комбинации значений полей первичного ключа соответствует одно и только одно значение поля А;

3.2 не существует функциональной зависимости А ни от какого подмножества полей первичного ключа (в противном случае А находится в частичной функциональной зависимости от первичного ключа).

В таблице КЛИЕНТ неключевые поля ФИО_К, СОЦ_П, АДР_К функционально зависят от ключа (НК, НС). Кроме того, они функционально зависят от подмножества ключа – НК.

Следовательно, неключевые поля ФИО_К, СОЦ_П, АДР_К находятся в частичной функциональной зависимости от первичного ключа (НК, НС) и нарушаются требования 2НФ. Эти поля надо из таблицы КЛИЕНТ удалить. Полученную в результате этого таблицу назовем КЛИЕНТ–СЧЕТ, которая имеет вид:

КЛИЕНТ–СЧЕТ

НК	НС
23	1111
23	3333
34	5555
45	2222
45	6666
45	1111

Эта таблица удовлетворяет требованиям 2НФ.

Удаленные неключевые поля помещаются в новую таблицу совместно с подмножеством НК, от которого они зависят. И это подмножество будет первичным ключом новой таблицы КЛИЕНТ вида:

КЛИЕНТ

НК	ФИО_К	СОЦ_П	АДР_К
23	Сокол С.С.	Служащий	Садовая, 1
34	Брас Б.Б.	Рабочий	Гая, 9
45	Лань Л.Л.	Служащий	Лесная, 4

Новая таблица КЛИЕНТ также удовлетворяет требованиям 2НФ. Ее неключевые поля функционально полно зависят от первичного ключа.

Полученные таблицы не содержат избыточной информации, и нет основания приводить их к 3НФ. Таблица ФИЛИАЛ удовлетворяет требованиям 2НФ, так как ее неключевые поля НФ, АДР_Ф, НМ, ОСТ, ТИП функционально полно зависят от первичного ключа. Но в таблице ФИЛИАЛ повторяется информация о филиале для всех счетов, обрабатываемых им. Поэтому ее надо привести к 3НФ.

3 нормальная форма (3НФ)

Таблица находится в 3НФ, если она удовлетворяет требованиям 2НФ и не содержит транзитивных зависимостей.

Транзитивной зависимостью называется функциональная зависимость между неключевыми полями. В таблице ФИЛИАЛ она наблюдается. Следовательно, нарушаются требования 3НФ. Из таблицы ФИЛИАЛ надо удалить поля, участвующие в этой транзитивной зависимости, – АДР_Ф, НМ. Получится таблица, характеризующая счет, вида:

СЧЕТ

НС	ОСТ	ТИП	НФ
1111	200	Д	511
2222	350	Т	511
3333	1000	Т	511
5555	800	Т	513
6666	14	Д	513

Затем создается новая таблица, в которую помещаются удаленные поля и поле, от которого они зависят. Она имеет вид:

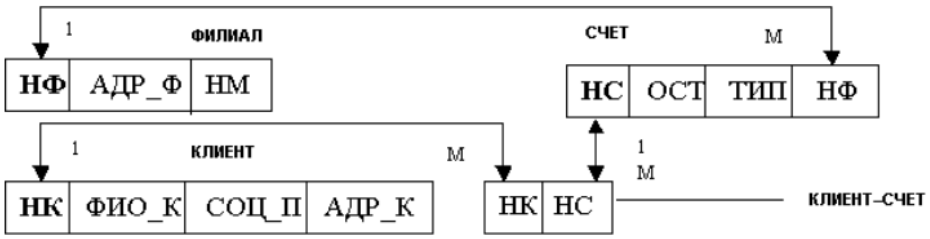
ФИЛИАЛ

НФ	АДР_Ф	НМ
511	Ванеева, 6	7
513	Солтыса, 3	9

Полученные таблицы приведены к 3НФ. В них каждая запись есть отдельное независимое утверждение. Повторяются только значения внешнего ключа НФ в таблице СЧЕТ, что неизбежно, так как одним филиалом могут обрабатываться несколько счетов.

Как видим, нормализация приводит к фрагментации исходных таблиц. В нашем примере таблица КЛИЕНТ разбивается на таблицы 1, 2, а таблица ФИЛИАЛ – на таблицы 3, 4. Осуще-

ствив связь этих таблиц посредством связи первичных и внешних ключей, получим реляционную модель данных предметной области БАНК, в которой минимизирована избыточность данных:



4. **Проверка логической модели данных на предмет возможности выполнения всех транзакций, предусмотренных пользователями.** Транзакция – это набор действий, выполняемых отдельным пользователем или прикладной программой с целью изменения содержимого базы данных. Так, примером транзакции в проекте БАНК может быть передача права распоряжаться счетами некоторого клиента другому клиенту. В этом случае в базу данных потребуется внести сразу несколько изменений. Если во время выполнения транзакции произойдет сбой в работе компьютера, то база данных окажется в противоречивом состоянии, так как некоторые изменения уже будут внесены, а остальные еще нет. Поэтому все частичные изменения должны быть отменены для возвращения базы данных в прежнее непротиворечивое состояние.

   Перечень транзакций определяется действиями пользователей в предметной области. Используя ER-модель, словарь данных и установленные связи между первичными и внешними ключами, производится попытка выполнить все необходимые операции доступа к данным вручную. Если какую-либо операцию выполнить вручную не удается, то составленная логическая модель данных является неадекватной и содержит ошибки, которые надо устранить. Возможно, они связаны с пропуском в модели сущности, связи или атрибута.

5. **Определение требований поддержки целостности данных и их документирование.** Эти требования представляют собой ограничения, которые вводятся с целью предотвратить помещение в базу данных противоречивых данных. На этом шаге вопросы целостности данных освещаются безотносительно к конкретным аспектам ее реализации. Должны быть рассмотрены следующие типы ограничений:

   - обязательные данные. Выясняется, есть ли атрибуты, которые не могут иметь Null-значений;
   - ограничения для значений атрибутов. Определяются допустимые значения для атрибутов;
   - целостность сущностей. Она достигается, если первичный ключ сущности не содержит Null-значений;
   - ссылочная целостность. Она понимается так, что значение внешнего ключа должно обязательно присутствовать в первичном ключе одной из строк таблицы для родительской сущности;

- ограничения, накладываемые бизнес-правилами. Например, в случае с проектом БАНК может быть принято правило, запрещающее клиенту распоряжаться, скажем, более чем тремя счетами.

Сведения обо всех установленных ограничениях целостности данных помещаются в словарь данных.

6. **Создание окончательного варианта логической модели данных и обсуждение его с пользователями.** На этом шаге подготавливается окончательный вариант ER-модели, представляющей логическую модель данных. Сама модель и обновленная документация, включая словарь данных и реляционную схему связи таблиц, представляется для просмотра и анализа пользователям, которые должны убедиться, что она точно отражает предметную область.

9.3.6 Физическое проектирование

Цель этапа физического проектирования – описание конкретной реализации базы данных, размещаемой во внешней памяти компьютера. Это описание структуры хранения данных и эффективных методов доступа к данным базы. При логическом проектировании отвечают на вопрос – что надо сделать, а при физическом – выбирается способ, как это сделать. Процедуры физического проектирования следующие:

1. **Проектирование таблиц базы данных средствами выбранной СУБД.** Осуществляется выбор реляционной СУБД, которая будет использоваться для создания базы данных, размещаемой на машинных носителях. Глубоко изучаются ее функциональные возможности по проектированию таблиц. Затем выполняется проектирование таблиц и схемы их связи в среде СУБД. Подготовленный проект базы данных описывается в сопроводительной документации.

2. **Реализация бизнес-правил в среде выбранной СУБД.** Обновление информации в таблицах может быть ограничено бизнес-правилами. Способ их реализации зависит от выбранной СУБД. Одни системы для реализации требований предметной области предлагают больше возможностей, другие – меньше. В некоторых системах вообще отсутствует поддержка реализации бизнес-правил. В таком случае разрабатываются приложения для реализации их ограничений. Все решения, принятые в связи с реализацией бизнес-правил предметной области, подробно описываются в сопроводительной документации.

3. **Проектирование физической организации базы данных.** На этом шаге выбирается наилучшая файловая организация для таблиц. Выявляются транзакции, которые будут выполняться в проектируемой базе данных, и выделяются наиболее важные из них. Анализируется пропускная способность транзакций – количество транзакций, которые могут быть обработаны за заданный интервал времени, и время ответа – промежуток времени, необходимый для выполнения одной транзакции. Стремятся к повышению пропускной способности транзакций и уменьшению времени ответа. На основании указанных показателей принимаются решения об оптимизации производительности базы данных путем определения индексов в таблицах, ускоряющих выборку данных из базы, или снижения требований к уровню нормализации таблиц. Проводится оценка дискового объема памяти, необходимого для размещения создаваемой базы данных. Стремятся к его минимизации. Принятые решения по изложенным вопросам документируются.

4. **Разработка стратегии защиты базы данных.** База данных представляет собой ценный корпоративный ресурс, и организации ее защиты уделяется большое внимание. Для этого проектировщики должны иметь полное и ясное представление обо всех средствах защиты, предоставляемых выбранной СУБД.

5. **Организация мониторинга функционирования базы данных и ее настройка.** После создания физического проекта базы данных организуется непрерывное слежение за ее функционированием. Полученные сведения об уровне производительности базы данных используются для ее настройки. Для этого привлекаются и средства выбранной СУБД.

Решения о внесении любых изменений в функционирующую базу данных должны быть обдуманными и всесторонне взвешенными.

9.4 Формальные верификации и спецификации

Оценивание качества программного обеспечения информационных систем неразрывно связано с процессом верификации, т. е. с подтверждением того, что функциональные возможности ПО соответствуют их описаниям в программной документации (т.е. их спецификации). Функциональные возможности, которые не имеют описания в документации или не соответствуют описанию, называются недекларированными возможностями. Они могут являться как следствием ошибок разработчика, так и результатом выполнения умышленно внедрённого в программу кода. Как случайные, так и преднамеренные НДВ (последние называются также программными закладками) представляют угрозу информационной безопасности программных систем. Цель верификации программ – выявление и регистрация НДВ для последующего их устранения.

Любое программное средство обладает определённой корректностью. Понятие корректность является более узким, чем понятие качество, так как последнее включает в себя такие характеристики, как надёжность, мобильность, понятность, сопровождаемость программной системы. Корректность характеризует функциональные возможности программы, т. е. одну из составляющих качества. Корректность программы наиболее полно определяется степенью её соответствия программной спецификации.

В настоящее время существует множество методов и средств автоматизированной верификации ПО, часто применяемых совместно, взаимно дополняющих друг друга. Но не менее важным, чем ПО, компонентом ИС является база данных. Она также оказывает влияние на качество ИС и нуждается в верификации. Для проведения полноценной верификации требуется:

- правильное понимание сущности БД;
- система количественных и качественных показателей корректности БД;
- наличие автоматизированных систем верификации БД.

На современном этапе БД должны рассматриваться не только как хранилища данных, но и как полноценные программные компоненты. Обоснованием данного положения служит практика использования промышленных СУБД и ИС, основанная в значительной степени на использовании клиент-серверной архитектуры и технологии активного сервера. Суть последней заключается в том, что функции ИС, выполняющие обработку данных в БД, реализуются не в клиентских приложениях, а на стороне сервера СУБД в виде хранимых подпрограмм БД. Слово «хранимый» указывает

на то, что коды этих подпрограмм располагаются вместе с данными и являются, таким образом, объектами БД.

Известно три типа хранимых подпрограмм: функции, процедуры и триггеры.

Функции и процедуры запускаются на выполнение путём явного вызова. Клиентскому приложению, чтобы вызвать процедуру, необходимо предварительно соединиться с БД, где расположен ее откомпилированный код. Обращение к процедуре производится по имени с передачей входных данных и, возможно, с заданием выходных параметров. Функция отличается от процедуры тем, что она всегда возвращает атомарное значение определённого типа и вызывается путем использования в арифметических и логических выражениях на месте операнда.

Триггер – это специальный тип хранимых подпрограмм. Он выполняется автоматически в ответ на вставку, обновление или удаление записей в таблице и служит для поддержания корректности и согласованности (целостности) данных. Реализация хранимых подпрограмм возможна как на языке баз данных, так и на языках программирования высокого уровня C, C++, Pascal, Java.

Из сказанного следует, что БД, поддерживаемые промышленными СУБД имеют двойственную природу. Оставаясь хранилищами данных, они одновременно играют роль программного обеспечения в ИС или, точнее, становятся полноценными программными компонентами, напоминающими динамически подключаемые библиотеки операционной системы Windows или COM-объекты. Как и любая другая часть ИС, БД нуждаются в верификации и тестировании.

Учитывая то, что верификация есть подтверждение корректности, необходимо заметить, что корректность БД должна оцениваться при помощи системы математических показателей. К числу наиболее часто встречающихся сегодня на практике функциональных показателей корректности БД относятся следующие:

- **полнота накопленных описаний объектов** – относительное число объектов и документов, имеющихся в БД, к общему числу объектов в аналогичной БД;
- **достоверность** – степень соответствия записей БД реальным объектам в данный момент времени;
- **идентичность данных** – относительное число описаний объектов, не содержащих ошибки, к общему числу документов об объектах в БД;
- **актуальность данных** – относительное число устаревших данных к общему числу накопленных и обрабатываемых записей.

Помимо функциональных показателей, существуют конструктивные показатели корректности, отличающиеся большей универсальностью, независимостью от сферы применения базы данных:

- **объем БД** – число описаний объектов или документов в БД, доступных для хранения и обработки;
- **оперативность** – степень соответствия динамики изменения данных в процессе сбора и обработки состояниям реальных объектов, или величина запаздывания между появлением (изменением) характеристик реального объекта и его отражением в БД;
- **периодичность** – промежуток времени между поставками двух последовательных, достаточно различающихся информацией версий БД;

- **глубина ретроспективы** – интервал времени от даты выпуска (записи в БД) самого раннего документа до настоящего времени;

- **динамичность** – относительное число изменяемых описаний объектов к общему числу записей в БД за некоторый интервал времени, определяемый периодичностью издания версий БД.

Оценивание корректности БД предполагает анализ ее содержимого, логической структуры и программной составляющей (хранимых подпрограмм). В связи с этим необходимо различать три вида корректности БД: корректность содержимого, корректность схемы данных и программная корректность. Рассмотрим каждый из перечисленных видов по отдельности.

Корректность содержимого. Любая запись в БД должна адекватно отражать характеристики реального существующего объекта, экземпляра сущности предметной области; если же в записи фиксируется информация о взаимодействии объектов, то речь должна идти о взаимодействии, имеющем место в реальной действительности.

Очевидно, что корректность содержимого определяется функциональными и конструктивными показателями, рассмотренными выше.

Корректность схемы данных. Традиционно под реляционной схемой понимается набор взаимосвязанных схем отношений или, с точки зрения обычного пользователя, весь набор таблиц БД. Можно выделить следующие подвиды корректности схемы данных:

1. **Точность отображения концептуальной схемы (ER-схемы) на реляционную.**

   Концептуальная модель реляционной БД обычно представляется в виде диаграмм сущность–связь, или ER-диаграмм, на которых показываются объекты предметной области и связи (способы взаимодействия) между ними. При концептуальном проектировании не оперируют понятиями «таблица», «внешний ключ» и т. п. Но ER-диаграммы легко преобразуются в схему реляционной БД по набору типовых правил. В итоге сущностям, представленным на ER-схеме, соответствуют таблицы реляционной БД, а связям – вспомогательные таблицы и внешние ключи. Типовые правила отображения дают возможность «обратного проектирования» – получения ER-схемы по имеющейся реляционной схеме БД. Однако не всегда «обратное проектирование» даёт ER-схему, эквивалентную исходной концептуальной модели. Причиной этого в ряде случаев является неточность прямого преобразования, в результате чего схема БД получается некорректной по отношению к исходной ER-модели.

   Точность отображения, очевидно, может быть определена как соответствие ER-схемы, полученной в результате «обратного проектирования», исходной концептуальной схеме. Точность отображения характеризуется множеством показателей, учитывающих детализацию сравнения на уровне атрибутов, сущностей, связей и участков ER-диаграмм, образованных двумя отдельно взятыми сущностями и одной ассоциацией между ними.

   В каждом случае необходимо рассчитывать относительное число правильно отображенных на реляционную схему сущностей, атрибутов, связей и участков, к общему числу соответствующих элементов, определенных на этапе концептуального проектирования. С другой стороны, в процессе создания реляционной схемы могут появиться недекларированные атрибуты, сущности и связи. Процесс верификации БД должен включать в себя оценивание их процентного

содержания в общем числе отраженных на реляционной схеме сущностей, атрибутов и связей соответственно. Чем ниже процентное содержание недекларированных элементов схемы данных, тем корректнее БД по отношению к концептуальной модели.

2. **Нормализованность таблиц.**

   Основная цель проектирования БД – группирование атрибутов по таблицам так, чтобы минимизировать избыточность данных и по возможности сократить объём памяти, необходимый для физического хранения таблиц. Теория нормальных форм и нормализации описывает один из формальных методов проектирования БД, в котором идентификация таблиц основана на выявлении первичных ключей и функциональных зависимостей между атрибутами.

   Высшей нормальной формой на практике оказывается, как правило, нормальная форма Бойса–Кодда. Согласно её требованиям, все неключевые атрибуты таблицы должны полностью функционально зависеть от потенциального ключа, а функциональных зависимостей между различными неключевыми атрибутами быть не должно. Нарушение этих требований приводит к избыточности данных и, как следствие, к аномалиям вставки, обновления и удаления записей.

   Современные методы проектирования (в том числе упомянутый ранее метод ER-моделирования) позволяют автоматически приводить таблицы БД к нормальной форме Бойса–Кодда. Однако ошибки могут возникать, особенно если разработчиками принимается нешаблонное решение о частичной денормализации таблиц.

   Проще всего оценивать нормализованность БД при помощи относительного числа полностью нормализованных таблиц к общему числу таблиц. Более сложной задачей является оценивание степени допустимости денормализации для данного проекта, если таковая имеет место.

3. **Логическая целостность.**

   Важной составляющей корректности схемы данных является логическая целостность. С каждой предметной областью связаны определённые требования целостности, которые тем или иным образом ограничивают диапазоны возможных значений атрибутов сущностей и связей, говорят о допустимости либо недопустимости комбинаций некоторых значений. Реализация требований целостности в БД позволяет избавиться от появления отрицательных возрастов и масс, от повторения одного и того же ИНН у различных лиц, от некорректной последовательности дат начала и завершения работы (в случае, когда они в результате ошибки пользователя меняются местами) – словом, избежать появления логически противоречивой, несогласованной, «невозможной» с позиции здравого смысла информации.

   Простые требования целостности реализуются при помощи специальных объектов БД, называемых ограничениями. Большинством реляционных СУБД сегодня поддерживаются такие ограничения целостности, как первичный ключ, внешний ключ, уникальность значений, запрет неопределённых значений и ограничение на основе логического условия. Более сложные требования целостности реализуются процедурно с помощью специальных подпрограмм – триггеров. Следует заметить, что целостность в данном случае пересекается с программной корректностью, которая будет рассматриваться далее. И в первом, и во втором случае речь идёт об автоматическом поддержании целостности. Во время выполнения операций вставки,

удаления и обновления неверные данные либо не принимаются совсем, либо автоматически корректируются.

Целесообразно оценивать логическую целостность БД при помощи следующих показателей.

- **Полнота реализации требований целостности** – отношение реализованных в БД ограничений и триггеров к общему числу заявленных требований. Ограничения, которые не были заявлены в документации, не рассматриваются.
- **Относительное число не заявленных в документации ограничений целостности к общему числу реализованных.** Если недекларированных ограничений нет, показатель принимает нулевое значение, что говорит о корректности схемы данных.
- **Согласованность ограничений с содержимым таблиц.** Возможны ситуации, когда дополнительное ограничение целостности или новый триггер добавляются в уже заполненную БД. При этом нельзя допустить, чтобы новое ограничение вошло в противоречие с уже добавленными в таблицы записями. Для обычных ограничений целостности эта проблема решена на уровне СУБД. Система запрещает вводить ограничение, если в БД уже имеются строки таблиц, которые ему не удовлетворяют. Но СУБД не в состоянии выполнить подобную проверку для триггера. Поэтому возможны противоречия: с одной стороны, имеется сложное ограничение целостности, реализуемое триггером; с другой стороны, записи, появившиеся в БД до создания триггера, могут заведомо не удовлетворять этому ограничению.

  Показатель согласованности ограничений и содержимого таблиц рассчитывается как отношение записей, удовлетворяющих всем ограничениям, к общему числу записей. Расчет этого показателя неразрывно связан с анализом программного кода триггеров, цель которого – выяснить декларативный смысл каждого триггера. Задача эта сама по себе нетривиальна и требует отдельного рассмотрения.
- **Непротиворечивость ограничений.** В результате ошибок проектирования в БД могут возникать конфликты между ограничениями и триггерами. Когда разные триггеры и ограничения предъявляют к данным противоречивые требования, вплоть до взаимоисключающих, таблицы БД могут оказаться недоступными для вставки и обновления строк.

  Свойство непротиворечивости ограничений затруднительно оценить количественно. Вероятно, речь должна идти о качественном показателе или комплексе количественных и качественных показателей, позволяющих оценить степень изолированности самих ограничений друг от друга, степень доступности данных, с которыми связано множество ограничений и триггеров, а также долю противоречивых ограничений во всей БД.
- **Сложность реализации требования целостности.** Даже простые ограничения могут быть реализованы с помощью триггеров. Теоретически это допустимо, но на практике подобные решения снижают эффективность системы. Кроме того, очевидно, что они затрудняют последующую верификацию БД, увеличивают время анализа, так как приходится восстанавливать декларативный смысл каждого триггера. Декларативный смысл обычного ограничения целостности всегда ясен из его описания.

  Сложность реализации требования целостности – качественный показатель, оцениваемый для каждого триггера. Если данное требование может быть полностью реализовано

при помощи стандартных ограничений, сложность является очень высокой. В противном случае вычисление показателя производится на основе семантического анализа команд, выполняемых в теле триггера. Возможные значения показателя сложности реализации: «очень низкая», «низкая», «средняя», «высокая», «очень высокая».

4. **Программная корректность баз данных.**

   Корректность программных процедур существенно зависит от их топологической и информационной сложности: чем выше сложность, тем больше вероятность появления неумышленных НДВ, с одной стороны, и тем сильнее усложняется обнаружение любых НДВ, с другой стороны.

   К настоящему времени предложено множество метрик оценивания сложности программ. Оценивание сложности хранимых подпрограмм БД может производиться с использованием метрик размера программ и метрик сложности потока управления.

   Исходя из сказанного, можно определить следующие характеристики программной сложности БД:

   - **Суммарная сложность подпрограмм БД.** Вычисляется как алгебраическая сумма сложностей (весов) всех подпрограмм. С тем, какой именно показатель (из перечисленных выше) будет характеризовать сложность отдельно взятой подпрограммы, необходимо определиться заранее, до выполнения вычислений.
   - **Группа показателей, отражающая наличие сцепления между подпрограммами.** Сцепление подпрограмм имеет место в том случае, если они используют общие атрибуты таблиц. Малое сцепление в программном классе или компоненте желательно, поскольку оно увеличивает инкапсуляцию и снижает вероятность возникновения ошибок в поведении компонента. Наиболее простым показателем сцепления является разность между числом пар несцепленных и числом пар сцепленных подпрограмм. Отрицательный показатель сцепления всегда приравнивается к нулю.

Список литературы

David A. Patterson Garth Gibson, Randy H. Katz (1988). *A Case of Redundant Arrays of Inexpensive Disks*. Department of Electrical Engineering и Computer Sciences, University of California. URL: https://www2.eecs.berkeley.edu/Pubs/TechRpts/1987/CSD-87-391.pdf.

P. Beynon-Davies (1991). *Expert database systems*. McGraw-Hill.

T.D. Garvey, T.F. Lunt (1991). *Model-based Intrusion Detection*. Baltimore, MD.

Jet Infosystems (1995). *Системы управления базами данных - коротко о главном*. URL: https://www.osp.ru/news/articles/1995/0402/13031414#part_4 (дата обр. 30.05.2020).

Г.Г., Барон (1995). «Параллельные архитектуры серверов баз данных». В: *Системы управления базами данных* 2. URL: https://www.osp.ru/dbms/1995/02/13031421#part_4.

Axelsson, Stefan (2000). «Intrusion detection systems: a survey and taxonomy». В: *Technical Report 99–15*.

Зегжда Д.П., Ивашко А.М. (2000). *Основы безопасности информационных систем*. Горячая линия - Телеком. ISBN: 5-93517-018-3. URL: https://www.studmed.ru/zegzhda-d-p-osnovy-bezopasnosti-informacionnyh-sistem_dcc012e982a.html.

G. Alonso, F. Casati, H. Kuno, V. Machiraju (2004). *Web Services. Concepts, Architectures and Applications*. ISBN: 978-3-662-10876-5.

Дейт (2005). *Введение в системы баз данных*. Вильямс. ISBN: 5-8459-0788-8. URL: http://tc.kpi.ua/content/lib/vvedenie_v_sistemy_baz_dannyh_8izdanie.pdf.

Смирнов (2007). *Безопасность систем баз данных*. Гелиос АРВ. ISBN: 9785854381635. URL: https://www.twirpx.com/file/1706071/.

МЕТРОЛОГИИ, ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ТЕХНИЧЕСКОМУ РЕГУЛИРОВАНИЮ И (2008). *ГОСТ Р 50922—2006. Защита информации: ОСНОВНЫЕ ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ*. URL: https://files.stroyinf.ru/Data2/1/4293836/4293836037.pdf.

Утебов Д. Р., Белов С. В. (2008). «Классификация угроз в системах управления базами данных». В: *Вестник Астраханского государственного технического университета* 1, с. 87—92. URL: https://cyberleninka.ru/article/n/klassifikatsiya-ugroz-v-sistemah-upravleniya-bazami-dannyh.

Утебов Данияр Рашидович, Белов Сергей Валерьевич (2008). «Классификация угроз в системах управления базами данных». В: *Вестник Астраханского государственного технического университета* 1, с. 87—92. URL: https://cyberleninka.ru/article/n/klassifikatsiya-ugroz-v-sistemah-upravleniya-bazami-dannyh/viewer.

Карпова (2009). *Базы данных. Учебное пособие*. Питер. ISBN: 978-5-94157-770-5. URL: https://rucont.ru/file.ashx?guid=a46c217d-4dbd-496b-a229-2ac562197d70.

Кирилов (2009). *Введение в реляционные базы данных*. БХВ-Петербург. ISBN: 978-5-496-00546-3. URL: https://mipt.ru/dnbic/content/db.pdf.

Пирогов (2009). *Информационные системы и базы данных: организация и проектирование*. БХВ-Петербург. ISBN: 978-5-9775-0399-0. URL: https://litmy.ru/knigi/os_bd/107950-informacionnye-sistemy-i-bazy-dannyh-organizaciya-i-proektirovanie.html.

Лихоносов А. Г. (2011). *Интернет-курс по дисциплине Безопасность баз данных*. URL: http://www.e-biblio.ru/book/bib/01_informatika/b_baz_dan/sg.html (дата обр. 01.03.2020).

D. Hardt, Ed. (2012a). *The OAuth 2.0 Authorization Framework*. RFC, с. 1—75. URL: https://datatracker.ietf.org/doc/html/rfc6749#section-1.1.

— (2012b). *The OAuth 2.0 Authorization Framework*. RFC, с. 5. URL: https://datatracker.ietf.org/doc/html/rfc6749#section-1.1.

— (2012c). *The OAuth 2.0 Authorization Framework*. RFC, с. 9—11. URL: https://datatracker.ietf.org/doc/html/rfc6749#section-1.1.

ИНФОРМАЦИОННОЕ ПИСЬМО ОБ УТВЕРЖДЕНИИ ТРЕБОВАНИЙ К СИСТЕМАМ ОБНАРУЖЕНИЯ ВТОРЖЕНИЙ (2012). Тех. отч. ФЕДЕРАЛЬНАЯ СЛУЖБА ПО ТЕХНИЧЕСКОМУ И ЭКСПОРТНОМУ КОНТРОЛЮ (ФСТЭК РОССИИ).

Различные архитектурные решения, используемые при реализации многопользовательских СУБД. Краткий обзор СУБД (2012). Тех. отч., с. 1—3. URL: https://intuit.ru/studies/courses/508/364/lecture/8643?page.

D. Hardt, Ed. (2013). *OpenID Connect Standard 1.0*. RFC, с. 7—8. URL: https://openid.net/specs/openid-connect-standard-1_0-21.html#rfc.authors.

Sakimura, N. (2013). *OpenID Connect Standard 1.0*. RFC, с. 1—54. URL: https://openid.net/specs/openid-connect-standard-1_0-21.html#rfc.authors.

Дейт К. Дж. (2014). *Введение в системы баз данных [7 издание]*. Уфимский Государственный Авиационный Технический Университет.

Т.Ю.Сергеева, М.Ю.Сергеев (2014). *Распределенная обработка данных*. URL: https://cchgeu.ru/upload/iblock/cf5/metod_roi_ivt_ras_24.06.2016.pdf (дата обр. 20.04.2020).

Faragallah, Osama S. (2015). *Multilevel Security for Relational Databases*. CRC Press. URL: http://www.ittoday.info/Excerpts/Multilevel_Database_Security.pdf.

kapustor (2015). *Greenplum DB*. URL: https://habr.com/ru/company/tinkoff/blog/267733/ (дата обр. 13.10.2015).

А.А. Браницкий, И.В. Котенко (2016). «Анализ и классификация методов обнаружения сетевых атак». В: *Труды СПИИРАН, выпуск 45*, с. 207—244. URL: http://www.mathnet.ru/links/348a035b6c0c5f4351d9a14db6e991fe/trspy873.pdf.

В.В., Скакун (2017). *Защита информации в базах данных и экспертных системах*. БГУ. URL: https://elib.bsu.by/bitstream/123456789/48524/5/%D0%97%D0%B0%D1%89%D0%B8%D1%82%D0%B0%20%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D0%B8%20%D0%B2%20%D0%91%D0%94.pdf.

Yunus, Mohd Amin Bin Mohd (2018). «Review of SQL Injection : Problems and Prevention». В: *International journal on informatics visualization*. URL: https://www.researchgate.net/publication/325940419_Review_of_SQL_Injection_Problems_and_Prevention.

Мысев Алексей Эдуардович, Морозов Николай Владимирович (2019). «Правовое регулирование информационной безопасности в Российской Федерации». В: *Отечественная юриспруденция* 3, с. 51—55. URL: https://cyberleninka.ru/article/n/pravovoe-regulirovanie-informatsionnoy-bezopasnosti-v-rossiyskoy-federatsii.

Citforum (2020a). *Двухфазная блокировка*. URL: https://ru.wikipedia.org/wiki/%D0%94%D0%B2%D1%83%D1%85%D1%84%D0%B0%D0%B7%D0%BD%D0%B0%D1%8F_%D0%B1%D0%BB%D0%BE%D0%BA%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0 (дата обр. 30.05.2020).

— (2020b). *Транзакции и целостность баз данных*. URL: http://citforum.ru/database/dblearn/dblearn09.shtml (дата обр. 30.05.2020).

hard_sign (2020). *Путеводитель по репликации баз данных*. URL: https://habr.com/ru/post/514500/ (дата обр. 10.08.2020).

Intuit (2020a). *Лекция 11: Модели транзакций*. URL: https://www.intuit.ru/studies/professional_retraining/953/courses/297/lecture/7419?page=4 (дата обр. 30.05.2020).

— (2020b). *Лекция 14: Триггеры: создание и применение*. URL: https://www.intuit.ru/studies/courses/5/5/lecture/148 (дата обр. 30.05.2020).

Wikipedia (2020a). *Двухфазная блокировка*. URL: https://ru.wikipedia.org/wiki/%D0%94%D0%B2%D1%83%D1%85%D1%84%D0%B0%D0%B7%D0%BD%D0%B0%D1%8F_%D0%B1%D0%BB%D0%BE%D0%BA%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0 (дата обр. 30.05.2020).

— (2020b). *Ссылочная целостность*. URL: https://ru.wikipedia.org/wiki/%D0%A1%D1%81%D1%8B%D0%BB%D0%BE%D1%87%D0%BD%D0%B0%D1%8F_%D1%86%D0%B5%D0%BB%D0%BE%D1%81%D1%82%D0%BD%D0%BE%D1%81%D1%82%D1%8C (дата обр. 30.05.2020).

anuupadhyay (2021). *Database Sharding – System Design Interview Concept*. URL: https://www.geeksforgeeks.org/database-sharding-a-system-design-concept/ (дата обр. 02.09.2021).

Jalili (2021). *Database Security, 2nd Semester, CE, SUT*. URL: http://ce.sharif.ac.ir/courses/84-85/2/ce925/resources/root/Course%20Materials/3_Acten_Wood_Slides.pdf (дата обр. 12.04.2021).

Lattice-based access control - Wikipedia (2021). URL: https://en.wikipedia.org/wiki/Lattice-based_access_control (дата обр. 13.04.2021).

Security models (2021). URL: https://slideplayer.com/slide/4007727/ (дата обр. 12.04.2021).

Аристов Е.Н. (2021). *PostgreSQL 13. Тюнинг, Kubernetes, облака*. ООО "Сам Полиграфист".

Модель Take-Grant — Википедия (2021). URL: https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_Take-Grant (дата обр. 10.04.2021).

Модель Харрисона-Руззо-Ульмана — Википедия (2021). URL: https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%A5%D0%B0%D1%80%D1%80%D0%B8%D1%81%D0%BE%D0%BD%D0%B0-%D0%A0%D1%83%D0%B7%D0%B7%D0%BE-%D0%A3%D0%BB%D1%8C%D0%BC%D0%B0%D0%BD%D0%B0 (дата обр. 09.04.2021).

High availability (2022). URL: https://en.wikipedia.org/wiki/High_availability (дата обр. 10.06.2022).

Prometheus vs Zabbix (2022). URL: https://www.metricfire.com/blog/prometheus-vs-zabbix/ (дата обр. 12.06.2022).

А.Клюкин, А.Кукушкин (2022). *Управление высокодоступными PostgreSQL кластерами с помощью Patroni*. URL: https://habr.com/ru/post/504044/ (дата обр. 12.06.2022).

Как настроить репликацию в PostgreSQL (2022). URL: https://selectel.ru/blog/tutorials/how-to-set-up-replication-in-postgresql/ (дата обр. 12.06.2022).

Репликация данных (2022). URL: https://highload.today/replikatsiya-dannykh/ (дата обр. 12.06.2022).

(2020). URL: https://cassandra.apache.org (дата обр. 22.05.2020).

(2022). URL: https://www.ibm.com/docs/en/cics-ts/5.3?topic=SSGMCP_5.3.0/com.ibm.cics.ts.doc/dfhtm/topics/dfhtm0a.html (дата обр. 14.06.2022).

(2022). URL: https://www.tutorialspoint.com/distributed_dbms/distributed_dbms_databases.htm (дата обр. 13.06.2022).

Chai, Wesley (2021). *data dictionary*. URL: https://www.techtarget.com/searchapparchitecture/definition/data-dictionary (дата обр. 19.03.2021).

Content trust in Docker (2022). URL: https://docs.docker.com/engine/security/trust/ (дата обр. 20.04.2022).

Docker security (2022). URL: https://docs.docker.com/engine/security/ (дата обр. 20.04.2022).

Gerhard Weikum, Gottfried Vossen (2021). *Transactional Information Systems: Theory, Algorithms, and the Practice of Concurrency Control and Recovery*. URL: https://db.in.tum.de/teaching/ss20/transactions/pdf/chapter4.pdf (дата обр. 16.04.2021).

Kubernetes Components (2022). URL: https://kubernetes.io/docs/concepts/overview/components/ (дата обр. 20.04.2022).

Manage sensitive data with Docker secrets (2022). URL: https://docs.docker.com/engine/swarm/secrets/ (дата обр. 20.04.2022).

Manage swarm security with public key infrastructure (PKI) (2022). URL: https://docs.docker.com/engine/swarm/how-swarm-mode-works/pki/ (дата обр. 20.04.2022).

MongoDB, Inc. (2021). *data dictionary*. URL: https://docs.mongodb.com/manual/reference/command/ (дата обр. 19.03.2021).

Paul Wilton, John Colby (2021). *Beginning SQL*. URL: https://books.google.com/books?id=9eqbXSnji84C (дата обр. 19.03.2021).

SeaView. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=55088.

SQL injections (2022). URL: https://proglib.io/p/vzlamyvaem-sayty-shpargalka-po-sql-inekciyam-2019-12-21 (дата обр. 14.06.2022).

Sql-oracle.

Swarm mode key concepts (2022). URL: https://docs.docker.com/engine/swarm/key-concepts/ (дата обр. 20.04.2022).

Verify repository client with certificates (2022). URL: https://docs.docker.com/engine/security/certificates/ (дата обр. 20.04.2022).

Лекции по системному администрированию МГУ им. Ломоносова, Лекция 4 Intrusion Detection Systems (2022). URL: https://intuit.ru/studies/professional_retraining/943/courses/20/lecture/631 (дата обр. 19.06.2022).

Лилия Козленко (2021). *Информационная безопасность в современных системах управления базами данных*. URL: http://citforum.ru/security/articles/safe_db/ (дата обр. 09.04.2021).

Системы и методы обнаружения вторжений. URL: http://citforum.ru/security/internet/ids_overview/.

Уральский федеральный университет (2021). *Лекция 10. Политика и модели безопасности*. URL: https://learn.urfu.ru/resource/index/data/resource_id/40980/revision_id/0 (дата обр. 07.04.2021).