Основы построения защищенных баз данных

Ваша команда по спасению компьютерной безопасности

1 июня 2020 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] Нагло скопипизженно с вечно загнивающей
[2] Мне стыдно указывать ресурс, где я взял этот кусок, так что пусть будет pornhub

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 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

[3] А эти определения я взял с прошлого года

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 формального 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

[4]Нагло взято отсюда

- 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
- 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

[5]За оригиналом сюда

- 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.

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 (кон-

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

тролируемого) 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;

[7] А это взято с лекций по администрированию баз данных

- 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
- 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.

[8]За оригиналом сюда

1.0.5 Защита от вывода

Мы уже говорили об этом в угрозах безопасности 1.0.2 на 8 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, из-

> меняющего 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 9.

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.

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

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.

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 однозначную 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

Политика безопаности – это набор законов, правил, процедур и норм поведения, определяющих, 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

https://web.archive.org/web/20160303230445/http://ftp.fas.org/irp/nsa/rainbow/tg021.htm

2.1.9 Критерии безопасности ГТК

https://fstec.ru/component/attachments/download/293

2.2 Модели безопасности в СУБД

2.2.1 Дискреционная модель безопасности

Матричная или дискреционная модель является наиболее простой и распространённой. Она строится по следующим принципам:

- В системе определяются субъекты и объекты
- Задаются права доступа для каждого сочетания «субъект+объект»

Отношения между субъектами объектами можно представить в виде матрицы доступа (access matrix), в строках которой перечислены субъекты, в заголовках столбцов перечислены объекты, а в ячейках (на пересечении строк и столбцов) указываются разрешенные виды доступа.

Виды доступа могут быть определены в каждом случае индивидуально.

Недостатками матричной модели являются ее статичность и чрезмерно детализированный способ указания отношений между субъектами и объектами.

При большом количестве пользователей администрирование такой системы усложняется, что приводит к возникновению ошибок.

Но основной недостаток матричной модели заключается в том, что субъект, имеющий право на чтение информации может передать эту информацию другим субъектам, без уведомления владельца объекта. Кроме того, не всегда можно назначить владельца каждому объекту (объекты часто принадлежат не отдельным субъектам, а всей системе).

Статичность модели препятствует быстрому внесению изменений в систему, например, при увольнении/найме новых сотрудников или при смене их должности/переводу в другое подразделение. Кроме того, для получения полного доступа к системе злоумышленнику достаточно узнать данные учетной записи пользователя.

2.2.2 Мандатная модель безопасности

Многоуровневые или мандатные модели доступа были разработаны с целью устранения недостатков матричных моделей. В многоуровневой модели задается порядок следования уровней доступа и уровней секретности, а затем устанавливаются связи между уровнем доступа и уровнем секретности. Эти связи, устанавливающие разрешение на доступ, называются мандатными. При создании системы безопасности с использованием мандатной модели необходимо:

- Определить все объекты в системе, к которым необходимо предоставить доступ;
- Составить список субъектов, получающих доступ к объектам;
- Разбить все объекты на группы по уровню конфиденциальности (секретности);
- Создать группы для субъектов, различающиеся по уровню доступа (установить формы допуска);
- Установить для каждого субъекта уровень доступа (выдать им формы допуска);
- Определить все возможные виды разрешений на доступ;
- Установить связи (в виде разрешений на доступ) между группами субъектов и группами объектов.

Вторая задача, решаемая с помощью мандатной модели – предотвращение утечки информации от субъектов и объектов с высоким уровнем доступа к субъектам и объектам с более низким уровнем доступа.

Очевидно, что субъект с низким уровнем доступа не должен получать доступ к объектам с более высоким уровнем секретности.

Менее очевидно то, что субъект с высоким уровнем доступа не должен иметь полного доступа к объектам с меньшим уровнем секретности.

Проблема заключается в том, что информация, полученная субъектом с высоким уровнем доступа, может быть (случайно или намеренно) передана им объекту с более низким уровнем доступа.

Например, субъект, получивший легальный доступ к объекту (документу) с уровнем «Совершенно секретно», не должен иметь возможность скопировать текст, содержащий совершенно секретную информацию, и записать его в объект (документ) с более низким уровнем «Секретно», так как таким образом субъекты с уровнем доступа «Секретно» получат доступ к совершенно секретным данным.

2.2.3 Классификация моделей безопасности

- Модели систем дискреционного разграничения доступа;
- Модели систем мандатного разграничения доступа;
- Модели безопасности информационных потоков;
- Модели ролевого разграничения доступа;
- Субъектно-ориентированная модель изолированной программной среды.

2.2.4 Аспекты исследования моделей безопасности

2.2.5 Особенности применения моделей безопасности в СУБД

2.2.6 Дискреционные модели

HRU

Take-Grant

Action-Entity

Wood

2.2.7 Мандатные модели

Bell-LaPadula

Biba

Dion

Sea View

Jajodia&Sandhu

Smith&Winslett

Решеточная

2.2.8 БД с многоуровневой секретностью (MLS)

2.2.9 Многозначность

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)

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 входят *базовые таблицы* и *доступные пользователю представления*. Основу словаря данных составляет совокупность базовых таблиц, хранящих информацию о базе данных. Эти таблицы читаются и пишутся на системном уровне; они редко используются непосредственно пользователями. Словарь данных содержит 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, и используют соединения и фразы WHERE, чтобы упростить информацию. Большинство пользователей имеют доступ к этим представлениям вместо базовых таблиц словаря. 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.

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

Представления словаря

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 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 этом подразумевается некая форма хранения информации в постоянной памяти как часть транзакции. 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). Важно только то, что по окончании выполнения транзакции (фиксация транзакции) все правила целостности базы данных должны быть 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 следующие проблемы одновременного доступа:

- проблема последнего изменения возникает, когда несколько пользователей изменяют одну и ту же строку, основываясь на ее начальном значении; тогда часть данных будет потеряна, т. к. каждая последующая тран- закция перезапишет изменения, сделанные предыдущей. Выход из этой ситуации заключается в последовательном внесении изменений;
- проблема "грязного"чтения возможна в том случае, если пользователь выполняет сложные операции обработки данных, требующие множественного изменения данных перед тем, как они обретут логически верное состояние. Если во время изменения данных другой пользователь будет считывать их, то может оказаться, что он получит логически неверную информацию. Для исключения подобных проблем необходимо производить считывание данных после окончания всех изменений;
- проблема неповторяемого чтения является следствием неоднократного считывания транзакцией одних и тех же данных. Во время выполнения первой транзакции другая может внести в данные изменения, поэтому при повторном чтении первая транзакция получит уже иной набор данных, что приведет к нарушению их целостности или логической несогласованности;

- проблема чтения фантомов появляется после того, как одна транзакция выбирает данные из таблицы, а другая вставляет или удаляет строки до завершения первой. Выбранные из таблицы значения будут некорректны.

Сериализация транзакций

Для того, чтобы добиться изолированности транзакций, в СУБД должны использоваться какие-либо методы регулирования совместного выполнения транзакций.

План (способ) выполнения набора транзакций называется сериальным, если результат совместного выполнения транзакций эквивалентен результату некоторого последовательного выполнения этих же транзакций.

Сериализация транзакций – это механизм их выполнения по некоторому сериальному плану. Обеспечение такого механизма является основной функцией компонента СУБД, ответственного за управление транзакциями. Система, в которой поддерживается сериализация транзакций, обеспечивает реальную изолированность пользователей.

Основная реализационная проблема состоит в выборе метода сериализации набора транзакций, который не слишком ограничивал бы их параллельность. Приходящим на ум тривиальным решением является действительно последовательное выполнение транзакций. Но существуют ситуации, в которых можно выполнять операторы разных транзакций в любом порядке с сохранением сериальности. Примерами могут служить только читающие транзакции, а также транзакции, не конфликтующие по доступу к объектам базы данных.

Между транзакциями могут существовать следующие виды конфликтов:

- W-W - транзакция 2 пытается изменять объект, измененный не закончившейся транзакцией 1;
- R-W - транзакция 2 пытается изменять объект, прочитанный не закончившейся транзакцией 1;
- W-R - транзакция 2 пытается читать объект, измененный не закончившейся транзакцией 1.

Практические методы сериализации транзакций основываются на учете этих конфликтов (Citforum 2020b).

Методы сериализации транзакций

Существуют два базовых подхода к сериализации транзакций – основанный на синхронизационных захватах объектов базы данных и на использовании временных меток. Суть обоих подходов состоит в обнаружении конфликтов транзакций и их устранении. Предварительно заметим, что для каждого из подходов имеются две разновидности – пессимистическая и оптимистическая. При применении пессимистических методов, ориентированных на ситуации, когда конфликты возникают часто, конфликты распознаются и разрешаются немедленно при их возникновении. Оптимистические методы основываются на том, что результаты всех операций модификации базы данных сохраняются в рабочей памяти транзакций. Реальная модификация базы данных производится только на стадии фиксации транзакции. Тогда же проверяется, не возникают ли конфликты с другими транзакциями.

Пессимистические методы сравнительно просто трансформируются в свои оптимистические варианты (Citforum 2020b).

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. Это также имя результирующего набора 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 модифика-

ции 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, поддерживающие высокую готовность

Определение: Высокая готовность (High Availability, HA) - это характеристика технической системы, разработанная для избежания невыполненного обслуживания путём уменьшения или управления сбоями и минимизации времени плановых простоев.

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. Доступность(готовность) часто измеряется в «Nines»

HA включает в себя несколько важных аспектов:

1. Доступность данных
2. Защита данных

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. Взято из википедии

3. Производительность
4. Цена поддержки

4.1 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 Master-Slave
     В этом подходе выделяется один основной сервер базы данных, который называется Мастером. На нем происходят все изменения в данных (любые запросы INSERT/UPDATE/DELETE). Слейв сервер постоянно копирует все изменения с Мастера. С приложения на Слейв сервер могут отправляться запросы на чтение данных (запросы SELECT). Таким образом Мастер сервер может отвечать за изменения данных, а Слейв за чтение. Также Мастер сервер может отвечать за все операции с данными, а Слейв являться бэкапом. При выходе из строя Слейва, достаточно просто переключить все приложение на работу с Мастером. После этого восстановить репликацию на Слейве и снова его запустить. Если выходит из строя Мастер, нужно переключить все операции (и чтения и записи) на Слейв. Таким образом он станет новым Мастером. После восстановления старого Мастера, настроить на нем реплику, и он станет новым Слейвом. Рассмотрена асинхронная репликация. При синхронной репликации результат сразу же записывается и в Мастер и в Слейв. 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 выбирать случайное соединение из доступных Мастеров. Вероятные поломки делают Master-Master репликацию непривлекательной. Выход из строя одного из серверов практически всегда приводит к потере каких-то данных. Последующее восстановление также сильно затрудняется необходимостью ручного анализа данных, которые успели либо не успели скопироваться.

   Как уже говорилось ранее, репликацию данных можно разделить на синхронную и асинхронную. При асинхронной репликации изменения в slave или 2 master могут появляться с

задержкой. При сихронной репликации может существовать только одна версия данных, что накладывает ограничения на работу с данными в процессе репликации.

Существует логическая репликация. Логическая репликация — это метод репликации объектов данных и изменений в них, использующий репликационные идентификаторы (обычно это первичный ключ). Мы называем такую репликацию «логической», в отличие от физической, которая построена на точных адресах блоков и побайтовом копировании. PostgreSQL поддерживает оба механизма одновременно; Логическая репликация позволяет более детально управлять репликацией данных и аспектами безопасности.

В логической репликации используется модель публикаций/подписок с одним или несколькими подписчиками, которые подписываются на одну или несколько публикаций на публикующем узле. Подписчики получают данные из публикаций, на которые они подписаны, и могут затем повторно опубликовать данные для организации каскадной репликации или более сложных конфигураций.

Логическая репликация таблицы обычно начинается с создания снимка данных в публикуемой базе данных и копирования её подписчику. После этого изменения на стороне публикации передаются подписчику в реальном времени, когда они происходят. Подписчик применяет изменения в том же порядке, что и узел публикации, так что для публикаций в рамках одной подписки гарантируется транзакционная целостность. Этот метод репликации данных иногда называется транзакционной репликацией.

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

   (a) RAID 0 (striping — «чередование») — дисковый массив из двух или более жёстких дисков без резервирования.

   (b) 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.

   (c) 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.

   (d) 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.

(e) 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.

(f) 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.

(g) 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 свойствая является одинаковой для всех СУДБ. СУБД использует специальный лог(Write-Ahead Lock - WAL), в который записываются все транзакции, результат которых не попал на физический диск. WAL — это журнал опережающей записи, технология, которую часто используют для улучшения надежности базы данных. Когда вы заносите информацию в базу данных, база выполняет несколько операций, манипулируя блоками. 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.

4.2 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, 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, на которых запущена БД.
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. Zabbix также поддерживает мониторинг через SNMP. Благодаря расширяемости данная система мониторинга позволяет контролировать любые метрики системы. К недостаткам можно отнести некоторую трудоемкость при необходимости слежения за нестандартными параметрами системы. Для мониторинга систем чаще всего используется протокол SNMP, и можно сказать, что он являлся долгое время стандартом де факто. Протокол SNMP работает на базе протокола UDP и предназначен для использования сетевыми управляющими станциями. Он позволяет управляющим станциям собирать информацию, изменять, посылать «trap» на сервер мониторинга. Протокол определяет формат данных, их обработка и интерпретация остаются на усмотрение системного администратора и системы мониторинга. SNMP-сообщения не имеют фиксированного формата и фиксированных полей. Как следствие протокол SNMP универсален, но его главным недостатком является дополнительный трафик и необходимость загружать резидентные продукты (агенты) на управляемых объектах.

4.3 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.

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.

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).
Существуют 6 нормальных форм.

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 (рис. 2). Конечно, зеркалирование бесполезно с любой точки зрения, если оно организовано на одном диске. 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. 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.

4.4 Системы, обладающие свойством высокой готовности

Описание, назначение, примеры
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.

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 кассандры 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 кассандры 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.

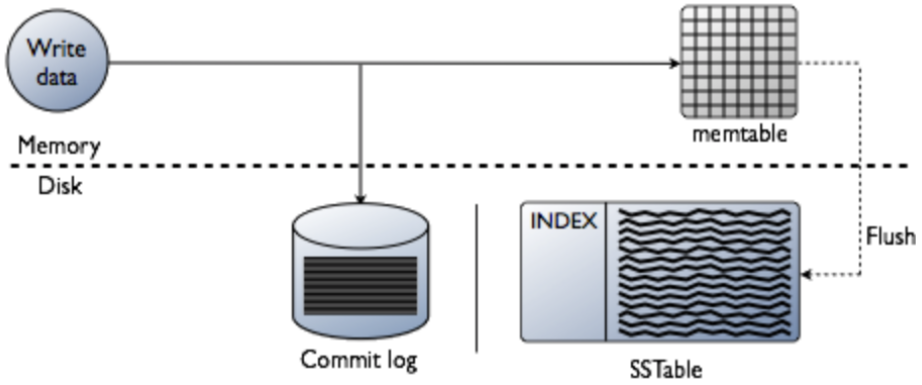

Рис. 3: Схема записи в Casandra

5 Защита данных в распределенных системах

5.1 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
- 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 (Бомонка 2019)

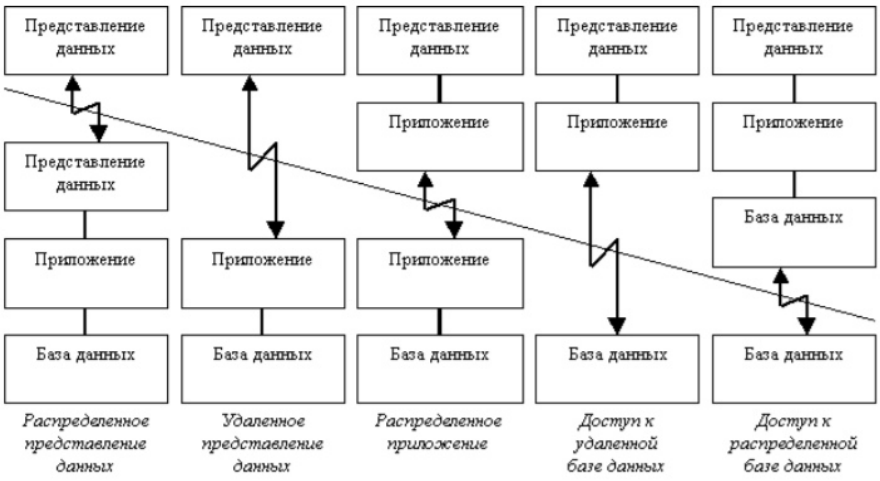

Рис. 4: Классификация двухзвенных моделей

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).

5.2 Угрозы безопасности распределенных СУБД

Угрозы доступности, целостности и конфиденциальности данных. Механизмы противодействия

Современный подход к информационной безопасности

Под информационной безопасностью понимается защита информации и поддерживающей ее инфраструктуры с помощью совокупности программных, аппаратно-программных средств и методов, а также организационных мер, с целью недопущения причинения вреда владельцам этой информации или поддерживающей ее инфраструктуре.

При построении системы защиты должен учитываться комплексный подход в обеспечении безопасности информации. Он подразумевает использование защитных механизмов на всех этапах жизненного цикла системы, от ее проектирования и до вывода из эксплуатации, и совместное решение целого спектра вопросов, начиная от физической защиты объектов ИС, с применением интеллектуальной системы контроля доступа, и заканчивая вопросами поддержки нормального функционирования ИС в критических ситуациях.

Проектирование системы безопасности информации (СБИ) осуществляется совместно с проектированием самой информационной системы. При внесении любых изменений в структуру ИС, это должно найти адекватное отражение и в системе защиты.

При разработке систем информационной безопасности необходимо учитывать передовые тен-

денции развития информационных технологий, к каким в настоящее момент относятся интрасети (Интранет - Intranet). Характерными чертами данных сетей является то, что они основываются на технологии клиент/сервер, имеют в своем составе разнородные корпоративные информационные системы и пользуются внешними сервисами, базовым для которых является протокол TCP/IP, а также предоставляет аналогичные сервисы вовне. Центральным элементом интрасетей является WEB-сервис, поэтому чрезвычайно важным является вопрос обеспечения защищенности этого сервиса.

Разработка политики безопасности

Первым шагом на пути построения СБИ является разработка политики безопасности ИС. Под политикой безопасности следует понимать утвержденный высшим должностным лицом организации, в интересах которой разрабатывается ИС и система защиты, документ, с указанием организационных и технических мероприятий, направленных на защиту информации, и поддерживающей ее инфраструктуры.

В данном документе должны найти обязательное отражение основные принципы безопасности информации, такие как многоуровневая защита и разнообразие средств и методов защиты, минимизация привилегий, разделение полномочий, невозможность обхода средств защиты и т.п.

Построение модели нарушителя

После разработки политики безопасности производится анализ угроз ИС, возможных каналов утечки информации и построение на этой основе модели потенциального нарушителя.

Согласно предлагаемой модели, в качестве нарушителя может выступать как отдельное физическое лицо, так и специальные программы, содержащие в себе некие деструктивные элементы - разрушающие программные компоненты (РПК).

Под объектом защиты в данной модели понимается материальный носитель защищаемой информации, а также сама информация, доступ к которым определен режимом разграничения доступа, реализуемым системой защиты, а также поддерживающая инфраструктура.

Предполагается, что по отношению к ИС нарушитель может быть двух типов: внутренним или внешним. При этом внутренний нарушитель выступает в качестве законного пользователя системы или лица из числа обслуживающего персонала или администрации ИС, а внешний является лицом, не подпадающим ни под одну из указанных категорий. РПК, в свою очередь, может быть отнесено к категории внутреннего нарушителя.

Возможные угрозы нарушителя напрямую зависят от его типа. Возможности внутреннего нарушителя ограничены установленными правилами разграничения доступа, внутриобъектовым режимом и его функциональными обязанностями. Возможности РПК в ИС намного шире и определяются работоспособностью ПО, в котором они заложены, а также наличием в системе средств активного аудита и мониторинга.

Проектирование СБИ

Следующим шагом после разработки модели нарушителя является проектирование СБИ. Согласно принципу многоуровневой защиты и разнообразия средств и методов защиты, можно выделить следующие самостоятельные системы в составе СБИ (при этом необходимо учитывать, что все они интегрированы и находятся в тесной взаимосвязи):

- Система контроля доступа на объекты и в помещения ИС

- Система защиты информации в ИС от НСД
- Средства защиты от РПК
- Средства поддержания доступности информации в ИС

Система защиты информации в ИС от НСД

Данная система может быть реализована как встроенными средствами защиты сетевых операционных систем и систем управления базами данных (СУБД), так и дополнительными средствами. Стоит лишь отметить, что при построении системы защиты от НСД необходимо, по возможности, избегать дублирования многих функций, с тем, чтобы СБИ не страдала избыточностью, что в конечном итоге скажется как на удобстве работы пользователей и их желании выполнять все предписания системы защиты, так и на управляемости самой СБИ.

В состав указанной системы также должны входить:

- средства контроля, управления и идентификации при удаленном доступе
- средства управления доступом и идентификации в рамках ИС
- средства экранирования ИС от открытых сетей, а также разноуровневых сетей внутри данной ИС друг от друга
- средства управления, анализа и аудита СБИ в рамках сетевых конфигураций

Средства контроля, управления и идентификации при удаленном доступе к ИС

Указанная категория средств предназначена для осуществления процедуры контроля подключения к ИС удаленных пользователей, а также для управления их доступом и осуществления идентификации. Данная проблема возникает в связи со структурой интрасетей, когда удаленные пользователи получают доступ к ресурсам корпоративной ИС по выделенным, либо коммутируемым каналам связи. В этом случае существует реальная вероятность несанкционированного подключения нарушителя к линии связи, с активным или пассивным прослушиванием сети и выдачей себя за регистрированного пользователя ИС. В целях недопущения подобного целесообразно применение методов с использованием одноразовых паролей и единого входа в сеть

Принцип систем с одноразовыми паролями основан на однократном использовании пароля для процедуры идентификации в ИС, в результате чего перехват его нарушителем становится бессмысленным. Идентификация и аутентификация пользователя осуществляется сервером безопасности (аутентификации), входящим в состав ИС.

Концепция единого входа в сеть предоставляет существенное удобство для пользователей, так как при подключении к ИС им достаточно только один раз доказать свою подлинность. Кроме этого, применение этой концепции способствует усилению информационной безопасности, ввиду того, что в сети отсутствует открытая передача аутентификационной информации.

Средства управления доступом и идентификации в рамках ИС

Учитывая территориальную разнесенность современных ИС и использование технологии клиент/сервер, целесообразно выделение функции проверки подлинности в виде отдельного сервера безопасности (аутентификации). Услугами данного сервера должны пользоваться все другие серверы и пользователи ИС.

Сервер безопасности может быть реализован в виде одного или нескольких серверов, функционирующих на физически защищенных компьютерах. Серверы должны содержать репозиторий субъектов ИС и их секретные ключи.

При построении сервера безопасности возможны два пути:

- Сервер безопасности с центральным хранением данных репозитория
- Сервер безопасности с децентрализованным хранением данных репозитория

Однако централизованное хранение является нецелесообразным по следующим причинам:

- сильная зависимость от готовности сервера
- возможность расширения или комбинирования сетей в процессе развития ИС

Дополнительной причиной децентрализации сервера безопасности является объединение сетей при развитии ИС. Возможна ситуация, когда две до того независимые сети, имеющие каждая в своем составе отдельный сервер безопасности, объединяются. В этом случае один из серверов должен быть уничтожен, а все определенные в нем данные по авторизации переданы на другой сервер. Если же связь между двумя сетями устанавливается временно, то необходимо наличие нескольких серверов безопасности. В противном случае, один сервер должен будет производить постоянные, двойные определения то для одной, то для другой сети. Для того, чтобы избежать этого, необходимо наличие нескольких серверов безопасности.

Средства экранирования ИС

Средства экранирования используются для подключения ИС к открытым сетям или развязки разноуровневых сетей, т.е. сетей, обрабатывающих информацию с различным грифом секретности. Концепция систем типа Firewall (брандмауер, межсетевой экран) была разработана для снижения риска нелегального доступа к закрытой информации при подключении частных сетей (в том числе ЛВС) к сетям общего пользования. Межсетевой экран представляет собой программно-аппаратный комплекс, размещенный на стыке двух сетей и реализующий следующие три функции:

- обеспечение обмена данными между сетями только через указанную систему
- фильтрация трафика обмена
- предотвращение возможности проникновения в сам экран

В этом случае обеспечивается эффективная блокировка внешнего трафика частной сети и жесткий контроль за ним. Кроме того экраны могут осуществлять разграничение доступа между различными сегментами одной корпоративной сети, а также контроль за информационными потоками, направленными во вне, обеспечивая тем самым необходимый режим конфиденциальности.

Применение экранов также позволяет существенно уменьшить уязвимость внутренних сервисов безопасности, так как нарушителю необходимо вначале преодолеть защитные механизмы самого экрана, где они сконфигурированы особенно тщательно.

Существующие в настоящее время экраны могут быть условно разделены на следующие четыре типа:

- экраны с фильтрацией пакетов (packet-filtering firewall)

- шлюзы сеансового уровня (circuit-level gateway)
- шлюзы прикладного уровня (application-level gateway)
- экраны экспертного уровня (stateful inspection firewall)

Однако лишь некоторые экраны могут быть отнесены только к одной из указанный категорий. При этом необходимо отметить, что экраны экспертного уровня обеспечивают один из самых высоких на сегодняшний день уровней безопасности интрасетей.

Средства управления, анализа и аудита

Средства аудита занимают свое особое положение в ряду средств обеспечения безопасности информации, заключающееся в том, что все действия нарушителя по преодолению средств защиты фиксируются, позволяя тем самым вовремя обнаружить попытку несанкционированного входа в ИС. Причем, учитывая принцип многоуровневой защиты, нарушителю придется преодолевать несколько защитных рубежей, что будет обязательно отмечено в регистрационном протоколе. В случае если указанная процедура выполняется в режиме реального времени, администратором безопасности могут быть своевременно предприняты соответствующие меры по предотвращению незаконного вторжения на одном из следующих уровнях защиты.

Кроме средств аудита часть программных продуктов также позволяет осуществлять оценку системы безопасности сети, имитируя все известные способы, применяемые нарушителями для проникновения в интрасети, и тем самым, обнаруживая в системе защиты слабые места. Данные программные продукты не только выявляют уязвимые места, но и определяют действия, которые необходимо предпринять для ликвидации пробелов в сетевой системе безопасности. Администратору остается лишь выбрать способы их устранения.

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.

Безопасность систем управления базами данных

Составной частью информационной безопасности ИС является безопасность систем управления базами данных (СУБД). Учитывая, что СУБД является ключевым элементом современной ИС, можно отметить, что для них важны все три аспекта информационной безопасности: конфиденциальность, целостность и доступность.

В СУБД для идентификации и проверки подлинности применяются либо соответствующие механизмы операционной системы, либо специальный SQL-оператор.

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).

Угрозы СУБД

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.

5.3 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, проще, мониторы транзакций - программные системы (которые относят к категории middleware, то есть к посредническому или промежуточному программному обеспечению), обеспечивающие эффективное управление информационно-вычислительными ресурсами в распределенной системе. Они представляют собой гибкую, открытую среду для разработки и управления мобильными приложениями, ориентированными на оперативную обработку распределенных транзакций. В числе важнейших характеристик TPM - масштабируемость, поддержка функциональной полноты и целостности приложений, достижение максимальной производительности обработки данных при невысоких стоимостных показателях, поддержка целостности данных в гетерогенной среде. 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 TPM отличаются рядом характеристик, причем различия часто вытекают как из специфики операционной системы, в которой реализован и функционирует TPM.

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).

5.4 Протоколы фиксации

Протоколы фиксации

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

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

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 «OK» 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.

5.5 Тиражирование данных

Тиражирование данных - это асинхронный перенос изменений объектов исходной базы данных (source database) в базы данных, принадлежащие к различным узлам распределенной системы. Функции тиражирования данных возложены на специальный компонент СУБД - сервер тиражирования данных, называемый репликатором (replicator), задача которого состоит в обеспечении идентичности данных в принимающих базах данных (target database) данным в исходной БД.

Обзор средств тиражирования данных

На практике установка тиражируемой системы сводится к чисто административным действиям, для выполнения которых необходимо получить ответы на четыре вопроса: **Что** тиражировать? **Где** тиражировать? **Когда** тиражировать? **Как разрешать** предполагаемые конфликты?

Что тиражировать? Как и любая другая методология, тиражирование имеет собственную терминологию. Краеугольным камнем тиражирования является понятие согласованного распределенного набора данных (consistent distributed data set - CDDS). Фактически, это тот самый набор данных в базе (или база данных целиком), идентичность которого на всех узлах, вовлеченных в процесс тиражирования, и поддерживает репликатор. Здесь и далее мы будем говорить об узлах распределенной системы, хотя участвовать в тиражировании могут и базы данных, расположенные на одном узле.

CDDS может быть представлен следующими конфигурациями данных:

- Вся база данных
- Избранные объекты базы данных: таблицы или представления
- Вертикальные проекции объектов БД: избранные столбцы таблиц и/или представлений
- Горизонтальные подмножества объектов: избранные записи из таблиц и/или представлений
- Сочетание наборов 2-4

Где тиражировать?

Следующий основополагающий элемент схемы тиражирования - это путь переноса изменений (data propagation path - DPP) из каждой тиражируемой базы данных в другие БД. Гибкость тиражирования в значительной степени обеспечивается богатством выбора способа передачи данных между узлами распределенной системы.

Практика эксплуатации распределенных систем подсказала следующие схемы тиражирования данных, реализуемые репликатором:

- Центр-филиалы изменения в БД филиалов переносятся в центральную БД, и /или наоборот
- Равноправное, несколько БД разделяют общий набор изменяемых и тиражируемых данных
- Каскадное, изменения в одной БД переносятся в другую БД, откуда в свою очередь в третью БД и т. д., эта схема позволяет оптимизировать баланс загрузки серверов баз данных, расположенных на различных узлах
- Через шлюзы изменения в базе данных могут переносится в БД другой СУБД
- Различные комбинации всех перечисленных выше схем

Когда тиражировать?

После определения тиражируемого набора данных и маршрута переноса изменений остается определить лишь момент инициации репликационного сервера. Как уже говорилось раньше, элементарным изменением, вызывающим реакцию репликатора, является транзакция, но тиражировать

каждую транзакцию по одиночке было бы не всегда удобным. Например, было бы трудно зафиксировать состояние в принимающей базе данных на определенный час. Стремясь быть максимально гибким, репликатор предоставляет следующие возможности:

- тиражирование начинается после завершения определенного числа транзакций, в том числе и после каждой транзакции
- тиражирование происходит через равные промежутки или к определенному моменту времени
- процесс тиражирования контролируется вручную администратором системы или созданным пользователем монитором тиражирования

Как разрешать конфликты?

Конфликты, возникающие в некоторых ситуациях, например, при встречном тиражировании или при восстановлении базы данных с помощью репликатора из реплицированной копии, можно отнести к разряду планируемых проблем. Репликатор при необходимости самостоятельно обнаруживает противоречия в тиражируемых данных и предоставляет разрешение конфликта администратору (противоречивые данные обязательно регистрируются в журнале), либо делает это автоматически. Возможны следующие варианты:

- разрешение конфликта в пользу более раннего или более позднего изменения
- разрешение конфликта в пользу наивысшего приоритета тиражируемой записи

Эффективные алгоритмы тиражирования

Алгоритмы тираживания:

- Алгоритм восстановления данных при тиражировании
- Алгоритм создания копии при тиражировании данных

Более подробно об алгоритмах тиражирования можно почитать тут (https://www.hse.ru/data/2010/08/04/121
Lyadova_Strelkov.pdf)

Сравнение подходов к тиражированию БД

5.6 Интеграция БД и Internet

Современные тенденции

Обзор существующих технологий

Вопросы безопасности: угрозы и методы противодействия

Перспективы развития

6 Безопасность в статистических БД

Инфы про статистические базы данных на русском языке мало и большая часть бредовая. Гуглите statistical database. Все источники будут на английском, переведено специально для вас. Источников несколько Lawrie Brown 2008 Дейт К. Дж. 2014, под конец была найдена статья, NABIL R. ADAM 1989 в которой есть вся собранная инфа.

6.1 Общие сведения

. Определение статистической БД. Классификация статистических БД. Характеристики статистических БД.

Статистической (в приведенном здесь контексте) называется 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 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.

6.2 Классификация

Классифицирует по следующим признакам:

- Чисто статистическая база данных - Обычная база данных со статистическим доступом

  этот тип базы данных хранит только статистические данные. Примером может служить база данных переписи населения. этот тип базы данных содержит отдельные записи. Кроме того, база данных поддерживает набор статистических пользователей, которым разрешены только статистические запросы. Для этих последних пользователей совокупная статистика, основанная на базовых необработанных данных, генерируется в ответ на запрос пользователя или может быть предварительно вычислена и сохранена как часть базы данных.

- Оффлайн-Онлайн

  В онлайн-SDB существует прямое взаимодействие пользователя с данными в режиме реального времени через терминал. В автономном SDB пользователь не контролирует обработку данных и не знает, когда выполняется его запрос данных. В этом режиме методы защиты,

отслеживающие профили пользователей, становятся более громоздкими. Компромиссные методы, требующие большого количества запросов (например, метод компромисса на основе регрессии, см. Раздел 7), также усложняются при работе в автономном режиме.

- Статическая-Динамический

  Статическая база данных-это такая база данных, которая никогда не меняется после ее создания. Большинство баз данных переписи являются статическими. Всякий раз, когда создается новая версия базы данных, эта новая версия считается другой статической базой данных. В отличие от этого, динамические базы данных могут изменяться непрерывно. Эта особенность может значительно усложнить проблему безопасности, поскольку частые выпуски новых версий могут позволить ищейкам использовать различия между версиями способами, которые трудно предвидеть. Методы возмущения данных могут не подходить для динамических СДБ, так как усилия по преобразованию исходного СДБ в возмущенный могут стать непомерными.

- Централизованный-Децентрализованный

  В централизованном SDB есть одна база данных. В децентрализованном (распределенном) SDB перегруженные подмножества базы данных хранятся на различных узлах, Соединенных коммуникационной сетью. Распределенная база данных может быть полностью реплицирована, частично реплицирована или секционирована. Проблема безопасности распределенного SDB является более сложной, чем проблема централизованного SDB из-за необходимости дублировать на каждом узле накладные расходы по контролю безопасности, а также трудности интеграции профилей пользователей. Выделенная-общая компьютерная система в выделенной SDB компьютерной системе используется исключительно для обслуживания приложений SDB. В общей системе приложения SDB работают на одной аппаратной системе с другими приложениями (возможно, с использованием различных баз данных). Общую среду защитить сложнее, так как другие приложения могут вмешиваться в защищаемые данные непосредственно через операционную систему, минуя механизм безопасности SDB.

6.3 Угрозы статистических БД

Об безопасности в статистических БД Деннинг [16.6]

> "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.

То есть существует мнение, что нельзя нормально обезопасить СБД.

Основная проблема СБД – проблема вывода. В общих чертах, проблема вывода для SDB может быть сформулирована следующим образом. Характеристическая функция C определяет подмножество записей (строк) в базе данных. Запрос, использующий C, предоставляет статистику по выбранному подмножеству. Если подмножество достаточно мало, возможно, даже одна запись, спрашивающий может быть в состоянии сделать вывод о характеристиках одного человека или небольшой

группы. Даже для больших подмножеств характер или структура данных могут быть такими, что несанкционированная информация может быть выпущена. Для злоумышленика, который пытается достать индивидуальные данные задача состоит в том, чтобы сделать общий трекер. Нужно придумать такую последовательность запросов, чтобы вывести индививидуальную информацию. Суть в том, что можно набрать несколько запросов и проделать операции над ними там, чтобы деанонить записи.

Например для статичной СБД можно сделать такое

запрос1: выдать a1 + a2 + a3, запрос2: выдать a1 + a2. запрос1 - запрос2 = a3.

Для динамической онлайн СБД Мы хотим достать зарплату Мэри. Знаем, что ей 20 лет. Записываем в базу много левых записей с возрастом = 20 лет и нулевой зарплатой. Делаем запрос с минимальной агрегацией людей с возврастом 20 лет. Таким образом можно узнать возможную зп Мэри. Чем больше исходных знаний есть о Мэри, тем точнее получим данные. Также не стоит забывать что речь идет о БД. Поэтому статистическая БД наследует все опасности обычных БД.

6.4 Защита в статистических БД

Разделяют четыре общих подхода защиты статистических БД:

- концептуальный;
- ограничение запросов;
- возмущение данных;
- возмущение вывода;

На рисунке 5 изображена схема 3 подходов.

Концептуальный Основная проблема в том, что вся реляционная алгебра позволяет выводить индивидуальные данные. Концептуальный подход предлагает заменить классическую систему реляционной БД работающую с индивидуальными данными. Основной подход: Partitioning of the data base – Записи хранятся не отдельно, а агрегировано и обезличино. Судя по-всему похоже на microagregation. Microaggregation – есть исходные данные, выделим несколько записей, посчитаем для них средние значения, заменим значения записей полученным средним, повторить для остальных записей в исходных данных

Еще один способ – решетчатая модель. Агрегация данных по разным признакам на разном уровне детализации.

Ну и естественно не забываем про классические способы. Access Restriction - пусть вы доктор, тогда вам можно всё. А исследователям и прочим анонам даём только некоторые записи или даже готовые стат.результаты по данным (агрегация)

Ограничение запросов Query Set Restriction – Ограничение на минимальный размер выдаваемой выборки.

Limiting intersection of query sets – Метод защиты блокирует такие запросы которые приводят к выводу данных через пересечение множеств запросов. Сохранение исторических сведений о результатах выполнения запросов и отклонение любых запросов, использующих значительное количество

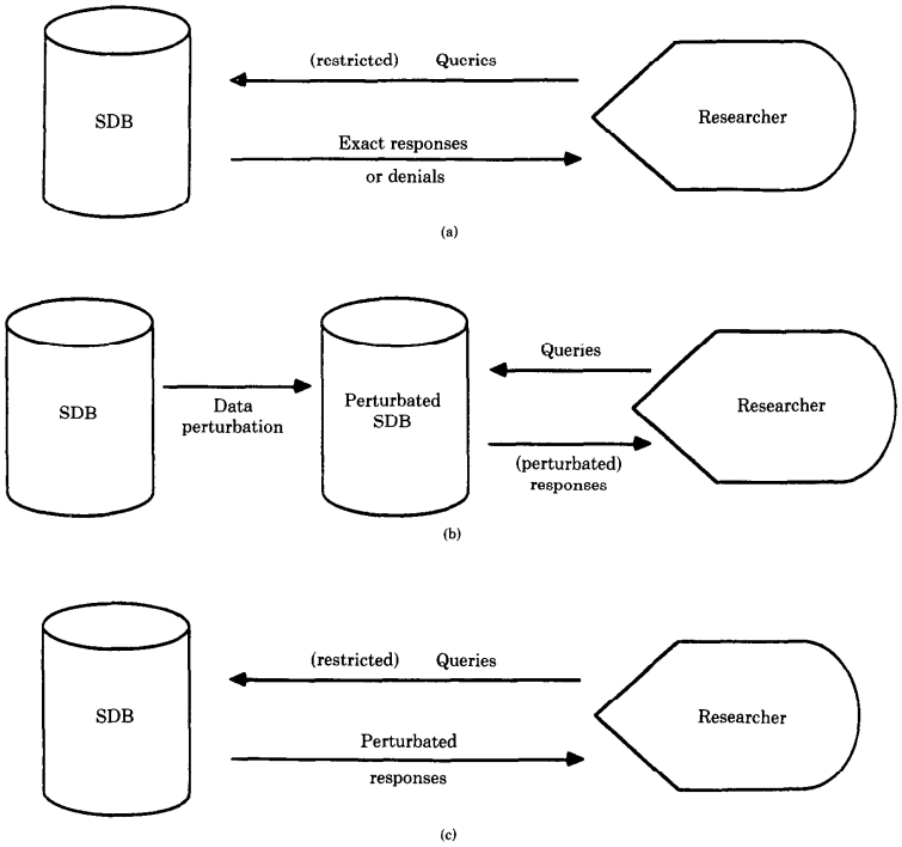

Рис. 5: Схемы безопасности СДБ

исходных данных, обработанных предыдущим запросом. Auditing - ведём логи, ловим негодяев и всяких подозрительных

Возмущение данных Data Perturbation – добавим небольшой, случайный шум. Тогда точные данные вроде как и будут уничтожены, но можно заиграться и исказить репрезентативность

Случайное добавление дополнительных строк к основному набору данных, обрабатываемому запросом. Также предлагается организовать "обмен данными"("data swapping"), т.е. обмен значениями атрибутов между кортежами, осуществляемый таким образом, чтобы поддерживалась лишь статистическая точность. При этом даже если злоумышленнику удастся идентифицировать отдельное значение (например, некоторое значение зарплаты), то у него не будет способа узнать, какому именно кортежу (в нашем примере — сотруднику) оно принадлежит. Сложность этого подхода заключается в необходимости отыскать множество тех записей, между которыми можно будет организовать обмен значениями соответствующим образом. Подобные затруднения имеют место и при использовании большинства других методов.

Возмущение вывода Output Perturbation – как предыдущее, но преобразовывать будем результат каждого запроса Random Sampling – сделать так, чтобы один и тот же запрос давал каждый раз разные наборы записей Использование для ответов на запросы только случайных наборов исходных данных

Критерии безопасности Как было сказано гарантированной безопасности в СДБ не добиться. Поэтому делают оценку важности деанонимизированных данных и статистической точности (из-за наших всех действий для защиты данных)

Security – сферическая вероятность раскрыть (в том числе частично) запись в сбд Для каждого разрешенного агрегирующего запроса оценивается критерий безопасности. На основе количества инфы в базе можно подсчитать сколько тех или иных запросов нужно сделать для деанонимизации. На основе этого расчитывается минимальная выборка для таких запросов.

Consistency – оценивается консистентность данных для методов возмущения данных. Оценивается степень возмущения данных. Создаем агрегирующий запрос и замеряем его изменения после внесенных возмущений. Robustness – оценка зависимости между возмущением и конкретной записью. Было бы круто чтобы возмущение не было зависимо от данных, но нужно каким-то образом сохранить ее статистические свойства.

Costs – Назначаем каждому запросу конкретную цену. Пользователю даем начальную сумму на которую он может делать запросы, такую чтобы он не смог вывести данные. Выглядит больше как метод, а не критерий.

7 Распознавание вторжений в БД

7.1 Основные понятия

Обнаружение вторжений – это сбор и анализ информации из различных точек защищаемой компьютерной системы (вычислительной сети) для выявления как попыток нарушения, так и реальных нарушений защиты (вторжений).

Среди методов, используемых в подсистемах анализа современных систем обнаружения вторжений (СОВ), можно выделить два направления:

- обнаружение аномалий в защищаемой системе
- поиск злоупотреблений

Цели выявления злоупотреблений – поиск шаблонов известных атак. В качестве искомого шаблона может выступать последовательность событий, паттерны сетевого трафика различных протоколов, определенные ассемблерные команды и т.д.

Место процедуры распознавания вторжений в общей системе защиты – обнаружение и регистрация атак, а также оповещение при срабатывании определенного правила.

7.2 Системы распознавания вторжений

Системы обнаружения и предотвращения вторжений (IPS/IDS) – это комплекс программных или аппаратных средств, которые выявляют факты и предотвращают попытки несанкционированного доступа в корпоративную систему.

Типы моделей систем распознавания вторжений (ID-систем)

Системы обнаружения и предотвращения разделяют на:

- **Сетевые (NIDS)** – проверке подвергается сетевой трафик с концентратора или коммутатора

- **Основанные на протоколах (PIDS)** – предполагает наблюдение за HTTP- и HTTPS-протоколами
- **Основанные на прикладных протоколах (APIDS)** – в таких системах проверяются специализированные прикладные протоколы. Например, на веб-сервере с SQL базой данных СОВ будет отслеживать содержимое SQL команд, передаваемых на сервер.
- **Узловые Host-Based (HIDS)** – подвергают анализу журналы приложений, состояния хостов, а также системные вызовы.
- **Гибридные** – являются композицией нескольких видов систем обнаружения вторжений.

Общая структура ID-систем

Концептуальная схема систем обнаружения вторжений включает в себя:

- Подсистему сбора событий (сенсорную)
- Подсистему анализа данных, полученных от сенсорной подсистемы
- Подсистему хранения событий
- Консоль администрирования

Шаблоны классов пользователей

Модели известных атак

- Необычные запросы и команды администрирования SQL сервера, часто употребляемые взломщиками (DROP, CREATE и т.д.)
- Незаконные операции DELETE, UPDATE и INSERT
- Запросы, содержащие id администратора
- Обращение к служебным таблицам и скрытым данным
- Запросы, всегда возвращающие TRUE
- Попытка обнулить поля с паролем
- Внедрение в запросе OR, сравнения констант и другое

7.3 Экспертные ID-системы

Технологию построения экспертных систем часто называют инженерией знаний. Как правило, этот процесс требует специфической формы взаимодействия создателя экспертной системы, которого называют инженером знаний, и одного или нескольких экспертов в некоторой предметной области. Инженер знаний «извлекает» из экспертов процедуры, стратегии, эмпирические правила, которые они используют при решении задач, и встраивает эти знания в экспертную систему. В

результате появляется компьютерная программа, которая решает задачи во многом так же, как эксперты – люди (Никулин А.Н. 2015).

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

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[10]. 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.

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.

[10]Здесь под *категорией* понимается группа подсистем, объединенных по некоему общему принципу

Параметры $M_1, M_2, \ldots, M_n$ могут быть зависимыми друг от друга. В таком случае, объединяющая функция будет более сложной.

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*.

Основные требования, предъявляемые к структуре профайла:

- Минимальный конечный размер
- Быстрое выполнение операции обновления

Примеры ID-систем

1. **GreenSQL-FW**. Работает как прокси-сервер между веб-приложением и SQL сервером. Анализирует SQL команды на предмет аномальных запросов.

   GreenSQL поддерживает несколько режимов работы:

   - **Simulation Mode** – пассивная система обнаружения атак (IDS). Протоколирует SQL запросы, выдает предупреждения на консоль управления.
   - **Blocking Suspicious Commands** – активная СОА. Атаки не только обнаруживаются, но и блокируются (IPS) в соответствии с установленными правилами, указывающими на аномальность запроса.
   - **Active protection from unknown queries** – блокирование всех неизвестных запросов (db firewall)
   - **Learning mode** – предназначен для прогона и настройки правил в «чистой» среде, что позволяет сформировать белый список и предотвратить в последствии ложные срабатывания анализатора запросов.

2. **Snort** – 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. Способна выявлять атаки на SQL базы данных.

   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

7.4 Развитие систем распознавания вторжений

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.

Список литературы

NABIL R. ADAM, JOHN C. WORTMANN (1989). «Security-Control Methods for Statistical Databases: A Comparative Study». В: *ACM Computing Surveys, Vol. 21,* 4. URL: https://dl.acm.org/doi/pdf/10.1145/76894.76895.

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).

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/.

Lawrie Brown, William Stallings (2008). *Computer Security Principles and Practice*. Pearson. ISBN: 9780134794105.

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).

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).

Никулин А.Н. (2015). *Экспертные системы*. Ульяновск, УлГТУ. ISBN: 978-5-9795-1489-5. URL: http://venec.ulstu.ru/lib/disk/2016/75.pdf.

Бомонка (2019). *Распределенная обработка данных*. URL: https://ru.bmstu.wiki/%D0%A0%D0%B0%D1%81%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_%D0%BE%D0%B1%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85 (дата обр. 20.04.2020).

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).

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).

Sql-oracle.