Глава 1

Информационная безопасность и аудит ИС

Современные требования бизнеса, предъявляемые к определению уровня обеспечения информационной безопасности, и существенный рост рисков потерь (материальных, финансовых, моральных, информационных) от нарушения информационной безопасности во всех сферах жизнедеятельности общества и государства, диктуют настоятельную необходимость использовать в своей работе обоснованные технико-экономические методы и средства, позволяющие количественно и качественно измерять уровень защищенности организаций и систем информационной технологий, а также оценивать экономическую эффективность затрат на информационную безопасность. Одним из направлений, позволяющих оценить уровень обеспечения информационной безопасности, является аудит информационной безопасности, цель которого - установление степени выполнения требований по обеспечению состояния защищенности системы информационных технологий. На сегодняшний день СУБД играют ключевую роль в обеспечении эффективного выполнения процессов предприятий. Вместе с тем повсеместное использование СУБД для хранения, обработки и передачи информации приводит к повышению актуальности проблем, связанных с их защитой. Именно для решения этих проблем и применяется аудит безопасности системы. В качестве объекта аудита может выступать как СУБД в целом, так и её отдельные сегменты, в которых проводится обработка информации, подлежащей защите. Существует возможность использовать стандартные средства аудита таких СУБД как: Oracle, MS SQL Server и т.д. Но, как правило, данные средства есть только у платных СУБД. Для бесплатных же СУБД, таких, к примеру, как PostreSQL, подобных решений пока не нет. Основной задачей является разработка решения аудита информационной системы, использующей СУБД PostgreSQL. Данную задачу можно решить путём создания подсистемы аудита, которая позволит обнаружить действия, нарушающие целостность основной систем, другими словами, с помощью данной подсистемы можно выявить как мошеннически введенные данные, так и несанкционированные запросы.

В настоящее время организации все больше зависят от информации, которую они используют. Если подвергать риску эту информацию с точки зрения потерь или несанкционированного доступа конкурентов могут последовать разрушительные последствия для организации. Таким образом, управление информационной безопасностью стало серьезной проблемой для всех организаций. Управление информационной безопасностью основывается на множестве политик и внутреннем контроле, с помощью которых организация и управляет своей информационной безопасностью. Информация и системы, которые обрабатывают ее, имеют решающее значение в работе практически всех организаций. Информация становится все более уязвимой для большого количества рисков, которые могут поставить под угрозу само существование предприятия. Это вынуждает принимать сложные решения о том, как сделать информационную безопасность эффективней. Цели информационной безопасности, как правило, считается выполненным, если:

1) Информационные системы имеются в наличии и готовы к использованию при необходимости;

2) Данные и информация раскрывается только тем, кто имеет право их знать (конфиденциальность);

3) Данные и информация защищена от несанкционированного изменения (целостность).

Организации должны реализовать эти цели, чтобы удостовериться, что их ценная информация защищается от возможных потерь, недоступности, изменения или неправомерного раскрытия. Стоит рассмотреть принципы аудита средств управления информационной безопасности и как они могут помочь организациям обеспечить, чтобы эти цели были удовлетворены

и что никаких недостатков системы существуют. Однако, во-первых, важна причина, почему аудит так важен. Цель аудита состоит в том, чтобы оценить производительность управления. Из-за распространенного использования систем информационных технологий (ИТ), важно, чтобы средства управления существовали. Средства управления ИТ - определенные процессы ИТ, разработанные, чтобы поддерживать бизнес-процесс. Средства управления ИТ могут быть категоризированы или как общие средства управления или как средства управления приложением. Общие средства управления - те средства управления, которые широко распространены на все компоненты систем, процессы и данные для данной организации или системной среды. Они включают средства управления такими областями как центр обработки данных и сетевые операции, системный сбор программного обеспечения и обслуживание, система обеспечения безопасности доступа и сбор прикладной системы, разработка и обслуживание. Средства управления приложением - те средства управления, которые являются подходящими для индивидуальных подсистем учета, такими как платежная ведомость или кредиторская задолженность. Они относятся к обработке отдельных приложений и помогают гарантировать, что транзакции произошли, авторизованы, и полностью и точно зарегистрированы, обработаны и сообщены. Это означает, что организации должны заняться расследованиями, достигают ли средства управления своих целей, выполняя аудит. Целью аудита является:

1) Обеспечение управления с достаточной гарантией того, что цели управления будут достигнуты;

2) Обоснование риска, где есть существенные слабые места управления;

3) Консультирование руководства по корректирующим действиям.

Общепринятая структура процесса аудита заключается в следующем:

1) Получение понимания связанных рисков бизнес-требований и соответствующих мер контроля;

2) Оценка целесообразности установленных средств управления;

3) Оценка соответствия путем тестирования работают ли средства управления как предписано, последовательно и непрерывно;

4) Обоснование риска целей управления, не встречаемых при помощи аналитических методов и/или консультационных альтернативных источников.

ИБ и аудит СУБД

Поскольку на сегодняшний день СУБД играют ключевую роль в обеспечении эффективного выполнения процессов организаций, то рассмотрим более подробно непосредственно обеспечение безопасности и аудит СУБД.

В современных СУБД поддерживается один из двух наиболее общих подходов к вопросу обеспечения безопасности данных: избирательный подход и обязательный подход. В обоих подходах единицей данных или "объектом данных для которых должна быть создана система безопасности, может быть как вся база данных целиком, так и любой объект внутри базы данных. Эти два подхода отличаются следующими свойствами:

• В случае избирательного управления некоторый пользователь обладает различными правами (привилегиями или полномочиями) при работе с данными объектами. Разные пользователи могут обладать разными правами доступа к одному и тому же объекту. Избирательные права характеризуются значительной гибкостью.

• В случае неизбирательного управления, наоборот, каждому объекту данных присваивается некоторый классификационный уровень, а каждый пользователь обладает некоторым уровнем допуска. При таком подходе доступом к определенному объекту данных обладают

только пользователи с соответствующим уровнем допуска.

• Для реализации избирательного принципа предусмотрены следующие методы. В базу данных вводится новый тип объектов БД — это пользователи. Каждому пользователю в БД присваивается уникальный идентификатор. Для дополнительной защиты каждый пользователь кроме уникального идентификатора снабжается уникальным паролем, причем если идентификаторы пользователей в системе доступны системному администратору, то пароли пользователей хранятся чаще всего в специальном кодированном виде и известны только самим пользователям.

• Пользователи могут быть объединены в специальные группы пользователей. Один пользователь может входить в несколько групп. В стандарте вводится понятие группы PUBLIC, для которой должен быть определен минимальный стандартный набор прав. По умолчанию предполагается, что каждый вновь создаваемый пользователь, если специально не указано иное, относится к группе PUBLIC.

• Привилегии или полномочия пользователей или групп — это набор действий (операций), которые они могут выполнять над объектами БД.

• В последних версиях ряда коммерческих СУБД появилось понятие "роли". Роль — это поименованный набор полномочий. Существует ряд стандартных ролей, которые определены в момент установки сервера баз данных. И имеется возможность создавать новые роли, группируя в них произвольные полномочия. Введение ролей позволяет упростить управление привилегиями пользователей, структурировать этот процесс. Кроме того, введение ролей не связано с конкретными пользователями, поэтому роли могут быть определены и сконфигурированы до того, как определены пользователи системы.

• Пользователю может быть назначена одна или несколько ролей.

• Объектами БД, которые подлежат защите, являются все объекты, хранимые в БД: таблицы, представления, хранимые процедуры и триггеры. Для каждого типа объектов есть свои действия, поэтому для каждого типа объектов могут быть определены разные права доступа.

На самом элементарном уровне концепции обеспечения безопасности баз данных исключительно просты. Необходимо поддерживать два фундаментальных принципа: проверку полномочий и проверку подлинности (аутентификацию). Проверка полномочий основана на том, что каждому пользователю или процессу информационной системы соответствует набор действий, которые он может выполнять по отношению к определенным объектам. Проверка подлинности означает достоверное подтверждение того, что пользователь или процесс, пытающийся выполнить санкционированное действие, действительно тот, за кого он себя выдает. Система назначения полномочий имеет в некотором роде иерархический характер. Самыми высокими правами и полномочиями обладает системный администратор или администратор сервера БД. Традиционно только этот тип пользователей может создавать других пользователей и наделять их определенными полномочиями. СУБД в своих системных каталогах хранит как описание самих пользователей, так и описание их привилегий по отношению ко всем объектам. Далее схема предоставления полномочий строится по следующему принципу. Каждый объект в БД имеет владельца — пользователя, который создал данный объект. Владелец объекта обладает всеми правами-полномочиями на данный объект, в том числе он имеет право предоставлять другим пользователям полномочия по работе с данным объектом или забирать у пользователей ранее предоставленные полномочия. В ряде СУБД вводится следующий уровень иерархии пользователей — это администратор БД. В этих СУБД один сервер может управлять множеством СУБД (например, MS SQL Server, Sybase). В СУБД Oracle применяется однобазовая архитектура, поэтому там вводится понятие подсхемы — части общей схемы БД и вводится пользователь, имеющий доступ к подсхеме. В стандарте

SQL не определена команда создания пользователя, но практически во всех коммерческих СУБД создать пользователя можно не только в интерактивном режиме, но и программно с использованием специальных хранимых процедур. Однако для выполнения этой операции пользователь должен иметь право на запуск соответствующей системной процедуры. В стандарте SQL определены два оператора: GRANT и REVOKE соответственно предоставления и отмены привилегий. Различают три вида привилегий:

• Объектные (Object privileges) – это разрешения на объекты схемы, такие как таблицы, представления, последовательности, пакеты. Для использования объектов схемы принадлежащих другому пользователю, необходимы привилегии на этот объект.

• Системные (System privileges) – это разрешения на операции уровня базы данных, например подключение к базе данных, создание пользователей, внесение изменений в конфигурацию базы данных.

• Ролевые (Role privileges) – это объектные и системные привилегии, которые пользователь получает как роль. Роли – это возможность для администрирования групп или привилегий. Ролью называется именованный набор привилегий. Объединение привилегий в роли значительно упрощает процесс назначения и снятия привилегий. Если СУБД поддерживает управление ролями, то в SQL-операторах GRANT и REVOKE вместо имени пользователя можно указывать имя роли.

Для управления привилегиями определены следующие правила:

• объект принадлежит пользователю, его создавшему (если синтаксисом не указано создание объекта другого пользователя, конечно, при соответствующих полномочиях);

• владелец объекта, согласно стандарту, может изменять привилегии своего;

• объектная привилегия всегда соотносится с конкретным объектом, а системная - с объектами вообще.

Также одним из направлений, позволяющих обеспечить информационную безопасность, является аудит. В двух словах, аудит — это действия, позволяющие определить, кто что делал в системе и имел ли он право на эти действия. Обеспечение безопасности имеет целью, прежде всего, помешать пользователям делать недопустимые вещи. Под термином "аудит"часто подразумевают ведение журнала аудита. Журнал аудита — это специальный набор записей, создаваемых системой, который надежно защищён от несанкционированного доступа. Иногда эти понятия используют как синонимы. Основное назначение журнала аудита — выступать в роли средства, позволяющего обнаружить действия, которые могут нарушить целостность системы. Иначе говоря, с его помощью мы можем выявить как мошеннически введенные данные, так и несанкционированные запросы. Теоретически кандидатом на регистрацию является любое происходящее системе событие. Особенно важны для аудита вход в систему и выход из системы, а также обновление данных. Аудит запросов несколько затруднен, если запросы производятся не через хранимые процедуры, поскольку средства аудита можно встроить в сами процедуры. Существует возможность использовать стандартные средства аудита таких СУБД как: Oracle, MS SQL Server и т.д. Можно привести несколько примеров. Обозначим несколько задач, которые должен решать аудит. Аудит доступа к базе данных. Контроль доступа к БД является фундаментальной задачей для того, что бы определить кто, когда и откуда имеет доступ к информации. Неудачные попытки, так же как и попытки входа в аномальное время в течение дня должны быть отслежены. Аудит изменений в структуре базы данных. В производственной базе данных никому из пользователей никогда не следует изменять структуру схемы. Администраторам баз данных следует вносить изменения в специально отведенное для этого время. Какие-либо другие изменения следует рассматривать как подозрительные. Наблюдение за структурными изменениями может включить индикаторы некорректного использования базы данных. Третья задача, которую можно было бы здесь

привести, это аудит использования любых системных привилегий. Заключительная группа команд аудита, которая может быть задействована это организация контроля за любыми изменениями данных, при помощи самих объектов.

Аудит различных СУБД

Существует возможность использовать стандартные средства аудита таких СУБД как: Oracle, MS SQL Server и т.д. Можно привести несколько примеров.

Использование средств аудита Oracle. Некоторые встроенные возможности аудита в Oracle могут быть довольно полезными. Если аудит включен на уровне экземпляров, то команда:

1) AUDIT SESSION; регистрирует все подключения к базе данных и отключения от нее, как успешные, так и неудачные.

2) AUDIT ALL ON LIVE.Таблица1 BY ACCESS; Эта команда заставляет Oracle регистрировать "детали"всех операций над таблицей Таблица1. Фраза BY ACCESS обеспечивает наличие элемента для каждого случая доступа, а не только одного элемента на тип доступа для данного сеанса. Oracle заносит все аудиторские действия в одну таблицу SYS.AUD$, и, если включено много журналов аудита, эта таблица становится очень загроможденной. Но Oracle обеспечивает ряд представлений, позволяющих просмотреть содержимое AUD$ в виде структурированных наборов.

Аудиторские представления позволяют запрашивать доступы к таблице Таблица1 по пользователям, по типам доступа, по дате и времени доступа. Однако мы не можем установить, что было изменено в таблице. Задачу аудита базы данных Oracle не следует ограничивать только лишь использованием команд аудита; так же успешно могут быть применены и другие технологии. Приведем некоторые основные методы, которые могут быть использованы для аудита базы данных Oracle:

1) Аудит Oracle. Все привилегии, которые могут быть предоставлены пользователю или роли базы данных могут быть проконтролированы. Сюда включено доступ на чтение, запись и удаление объектов на табличном уровне.

2) Системные триггеры. Эта возможность была представлена начиная с Oracle 8 и разрешает выполнение операций триггера, когда имеет место системное событие. Сюда включены запуск и останов базы данных, попытки входа и выхода, создание, изменение и удаление объектов схемы. С помощью автономных транзакций, можно записывать в журнал упомянутые системные события.

3) Update, delete и insert триггеры Для того, что бы отслеживать изменения в базе на уровне столбца и строки, можно написать триггеры, которые позволят полностью сохранять данные, до или после выполненного действия. Использование этого типа контроля очень ресурсоемко, так как создается и хранится много дополнительных записей. Кроме того, что существует еще один недостаток, связанный с этим методом - доступ на чтение нельзя отследить с помощью обычных триггеров базы данных.

4) Детализированный (Fine-grained) аудит Детализированный аудит решает проблему отслеживания доступа на чтение. Данная возможность основана на внутренних триггерах, срабатывающих, при разборе какой-нибудь части SQL-предложени я. Это очень эффективно, так как SQL-предложени е разбирается единожды для аудита и выполнения. Эта возможность использует предикаты, которые определены и проверяются каждый раз, когда происходит доступ к соответствующим объектам. Этот метод позволяет контролировать не только DML-операции на уровне строк и столбцов, но и предложения чтения.

5) Системные журналы СУБД Oracle генерирует много журнальных файлов, и многие из них могут содержать полезную информацию для проведения аудита. Например, alert log

используется для записи информации о запуске и останове базы, а также о вносимых структурных изменениях, таких как добавление файла данных в базу.

Рассмотрим подсистему аудита SQL Server (Database Engine)

Аудит экземпляра среды Компонент SQL Server Database Engine или отдельной базы данных включает в себя отслеживание и протоколирование событий, происходящих в компоненте Компонент Database Engine. Аудит среды SQL Server позволяет проводить аудит сервера, который может включать в себя спецификации аудита сервера для событий на уровне сервера, а также спецификации аудита базы данных для событий на уровне базы данных. События аудита могут записываться в журналы событий или файлы аудита. В SQL Server доступно несколько уровней аудита, применение которых зависит от существующих требований или стандартов установки. Подсистема аудита SQL Server предоставляет средства и процессы, необходимые для включения, хранения и просмотра аудитов на различных объектах серверов и баз данных. Группы действий аудита сервера можно записывать для всего экземпляра, а также группы действий аудита базы данных либо действия аудита базы данных для каждой базы данных. Событие аудита будет происходить каждый раз при обнаружении действия, подлежащего аудиту. Аудит на уровне сервера поддерживается во всех выпусках SQL Server. Аудит на уровне базы банных доступен только в выпусках Enterprise Edition, Developer Edition и Evaluation Edition.

Компоненты подсистемы аудита SQL Server

Аудит — это сочетание в едином пакете нескольких элементов для определенной группы действий сервера или базы данных. Компоненты подсистемы аудита SQL Server совместно формируют выходные данные, называемые аудитом, аналогично тому, как определение отчета в сочетании с элементами графики и данных формирует отчет. Подсистема аудита SQL Server использует расширенные события для создания аудита. события. Подсистема аудита SQL Server

Объект Подсистема аудита SQL Server объединяет отдельные экземпляры действий или групп действий уровня сервера или базы данных, за которыми нужно проводить наблюдение. Аудит работает на уровне экземпляра SQL Server. В одном экземпляре SQL Server может существовать несколько аудитов. При определении аудита задается место для вывода результатов. Оно называется назначением аудита. Аудит создается в отключенном состоянии и не выполняет автоматический аудит никаких действий. После включения аудита назначение аудита начинает получать от него данные. Спецификация аудита сервера Объект Спецификация аудита сервера принадлежит аудиту. На каждый аудит можно создать один объект спецификации аудита сервера, поскольку они оба создаются в области экземпляра SQL Server. Спецификация аудита сервера собирает множество групп действий уровня сервера, вызываемых компонентом расширенных событий. В спецификацию аудита сервера можно включить группы действий аудита. Группы действий аудита — это стандартные группы действий, являющиеся атомарными событиями, происходящими в компоненте Компонент Database Engine. Эти действия передаются аудиту, который регистрирует их в целевом объекте.

Назначение

Результаты аудита отправляются цели, которая может быть файлом, журналом событий безопасности Windows или журналом событий приложений Windows. Журналы необходимо периодически просматривать и архивировать, чтобы у цели оставалось достаточно места для создания дополнительных записей. Для записи в журнал событий безопасности Windows необходимо добавить в политику Создание аудитов безопасности учетную запись службы SQL Server. Если данные аудита сохраняются в файл, то для предотвращения подмены можно ограничить доступ к файлу следующим образом.

• Учетная запись службы SQL Server должна обладать разрешением на чтение и запись.

• Администраторам аудита обычно требуется разрешение на чтение и запись. Здесь подразумевается, что администраторы аудита — это учетные записи Windows, предназначенные для администрирования файлов аудита, в том числе копирования их в другие общие папки, резервного копирования и других операций.

• Агенты чтения аудита должны иметь разрешение только для чтения файлов аудита.

Даже если запись в файл выполняется компонентом Database Engine, другие пользователи Windows могут прочитать файл аудита, если имеют нужное разрешение. Компонент Компонент Database Engine не получает монопольную блокировку, запрещающую операции чтения. Поскольку компонент Database Engine может получать доступ к файлу, то имена входа SQL Server, имеющие разрешение CONTROL SERVER, могут использовать компонент Database Engine для доступа к файлам аудита. Чтобы зарегистрировать пользователей, читающих файлы аудита, надо определить аудит в представлении master.sys.fn_get_audit_file. В результате будут записаны имена входа с разрешением CONTROL SERVER, которые получали доступ к файлу аудита через SQL Server. Если администратор аудита скопирует файл в другое место (в целях архивирования или по другой причине), то список управления доступом к новому месту следует сократить до следующего набора разрешений:

• администратор аудита — чтение и запись;

• агент чтения аудита — только чтение.

Можно обеспечить дополнительную защиту от несанкционированного доступа путем шифрования папки, в которой хранится файл аудита, с применением шифрования диска Windows BitLocker или шифрованной файловой системы Windows (EFS). Для определения аудита можно использовать среду SQL Server Management Studio или Transact-SQL. Просматривать журналы событий Windows можно с помощью программы Средство просмотра событий в Windows. Для чтения целевых файлов можно использовать Средство просмотра журнала, среду SQL Server Management Studio или функцию fn_read_audit_file. Обычно процесс создания и использования аудита происходит следующим образом.

1. Создайте аудит и определите цель.

2. Создается либо спецификация аудита сервера, либо спецификация аудита базы данных, которая сопоставляет аудит. Включается спецификация аудита.

3. Включите аудит.

4. Считывание событий аудита можно осуществить с помощью оснастки Просмотр событий Windows, Средства просмотра журнала или функции fn_get_audit_file.

Создание аудита на Transact-SQL заключается в реализации всех аспектов аудита среды SQL Server, для этого можно использовать инструкции DDL, представления каталогов и динамические административные представления и функции; чтобы создать, изменить или удалить спецификацию аудита, можно использовать инструкции DDL.

Далее представим несколько варианта решения задачи аудита для SQL Server 2008.

1) CT (Change Tracking)

Зачастую путают с CDC (Change Data Capture). Но эти инструменты различны как в назначении, так и в реализации. CT предназначен для отслеживания фактов изменений (в каких строках, какие данные были изменены (CRUD)), в то время как CDC хранит историю изменений (все версии строк, в том числе те, которые были удалены). Что касается реализации, CDC основан на чтении журнала транзакций (асинхронен), в то время как CT работает синхронно. Для каждой таблицы, для которой включено отслеживание изменений, создается системная таблица, в которой хранился ID измененной строки, битовая маска для идентификации измененных колонок, тип операции. Для включения CT нужно активировать его на уровне БД и для конкретной таблицы:

ALTER DATABASE ChangeTracking SET change_tracking = ON (change_retention = 10

minutes, auto_cleanup = ON)

ALTER TABLE Orders enable change_tracking WITH (track_columns_updated = ON)

2) CDC (Change Data Capture) Средство для отслеживания измененных данных. Основными отличиями от CT являются асинхронная реализация (как писалось выше) и хранение всех версий измененных (CRUD) данных. Для хранения измененных данных CDC использует системные таблицы в схеме cdc. Для каждой таблицы, для которой активирован CDC, создается таблица. Для активации CDC Вам нужно активировать его на уровне БД для конкретной таблицы. С чисто практической точки зрения, значительный минус CDC это то, что невозможно зафиксировать автора изменений.

3. SQL Server Audit

Мощное средство, предназначенное для отслеживания всех событий и запросов и серверу (в том числе select). Область применения этого средства достаточно широка — от профилирования до вопросов, связанных с безопасностью и выявление активности пользователей в не предназначенной им части БД. SQL Server Audit позволяет гибко настраивать фильтры отслеживаемых событий. Для использования аудита необходимо активировать его на уровне сервера:

CREATE server audit ServerAudit TO FILE (filepath = 'D:', maxsize = 1GB) WITH (on_failture = CONTINUE)

ALTER server audit ServerAudit WITH (STATE=ON)

Пример создания спецификации аудита (трейса) на уровне сервера:

CREATE server audit specification ServerAudit_Permissions FOR server audit ServerAudit ADD (server_principal_change_group),

ADD (server_permission_change_group),

ADD (server_role_member_change_group);

ALTER server audit specification ServerAudit_Permissions WITH (STATE=ON);

Пример создания спецификации аудита на уровне БД:

USE MyDb CREATE DATABASE audit specification SA_MyDb_Orders FOR server audit ServerAudit ADD (SELECT, UPDATE, INSERT, DELETE ON dbo.Orders BY PUBLIC), ADD (SELECT, UPDATE, INSERT, DELETE ON dbo.OrderDetails BY PUBLIC)

Выбор СУБД

При планировании проекта была выбрана СУБД PostgreSQL. В Таблице 1 приведен сравнительный анализ трех распространенных систем управления базами данных, конкурирующих на рынке программного обеспечения по основным показателям.

Таблица 1: Сравнительная таблица СУБД

Показатели	Microsoft SQL Server 2008	MySQL 5.1	PostgreSQL 8.4
Поддерживаемые операционные системы	Windows Desktop/Server	Windows Desktop/Server , Linux, Unix, Mac	Windows1 Desktop/S22erver, Linux, Unix, 2Mac

Условии лицензирования	Коммерческий продукт с закрытым исходным кодом. Есть бесплатная версия с ограничением оперативной памяти до 4 Гб.	Коммерческая лицензия и GNU GPL.	Лицензия BSD Open Source.
Наличие предустановленных драйверов в ОС семейства Windows	Да	Нет	Нет
Наличие драйверов ODBC, JDBC, ADO.NET	Да	Да	Да
Наличие программных продуктов с открытым исходным кодом, основанных на этой СУБД	Несколько	Несколько	Несколько, но их число растет
Использование в коммерческих проектах	Среднее	Среднее	Среднее (чуть реже, чем MySQL)
Наличие графического ПО для конструирования и оптимизации запросов	Да (SQL Management Studio и Studio Express)	Нет	Да (PgAdminIII)
Поддержка частичных индексов	Да (называются “фильтрованные индексы”)	Нет	Да
Поддержка фунциональных индексов	Нет	Нет	Нет
Поддержка ACID-требований к транзакциям	Да	Да	Да
Каскадное обновление/удаление внешних ключей	Да	Да	Да
Внесение данных в несколько строк	Да	Да	Да
Возможность писать хранимые функции на разных языках программирования	Да, теоретически на любом языке, поддерживающим CLR	Нет (кроме C и Pl/SQL)	Да, наиболее полная поддержка из всех рассматриваемых.
Возможность создавать пользовательские аггрегированные функции	Да — любой .NET язык, кроме TRANSACT SQL	Да, только на C	Да — на PL language и встроенных C, SQL, PLPgSQL.
Поддержка триггеров	Да	Да	Да
Партицирование таблиц	Да (в Enterprise версии)	Да	Да

Возможность создавать функции, возвращающие таблицу или набор таблиц, которые можно использовать в секции FROM запросов.	Да	Нет	Да
Поддержка создания функций	Да	Да	Да
Поддержка хранимых процедур	Да	Да	Да (с помощью CREATE FUNCTION)
Поддержка динамического SQL в функциях	Нет	Нет	Да
Бесплатное ПО для графического управления БД	Да (SQL Management Studio/Express)	Нет	Да (PgAdmin III)
Наличие встроенного планировщика (не CronTab)	Да (SQL Agent не для Express версии)	Да (только для SQL-запросов)	Да (PgAgent)
Возможность доступа к таблице из другой базы данных, находящейся на том же хосте	Да	Да	Да
Чувствительность к регистру	По умолчанию — не чувствительна	Нет	Да
Поддержка даты и времени	Да	Да (но без временной зоны)	Да
Аутентификация	Средставими БД и ActiveDirectory	Средствами БД	Много разных методов, включающих предыдущие
Разграничение доступа к столбцам	Да	Да	Да
Поддержка рекурсивных запросов	Да	Нет	Да
Поддержка COUNT(DISTINCT), AGGREGATE(DISTINCT)	Да	Да	Да
Поддержка схем	Да	Нет	Да
Поддержка DISTINCT ON	Нет	Нет	Да
Поддержка функций OVER..PARTITION BY	Да	Нет	Да, причем лучше, чем в MS SQL
Поддержка LIMIT .. OFFSET	Нет	Да	Да
Наличие Advanced Database Tuning Wizard	Да (технолгия Microsoft)	Нет	Нет

Наличие Maintenance Plan Wizard	Да (технолгия Microsoft)	Нет	Нет
Наличие Pluggable Storage Engine	Нет	Да	Нет
Поддержка связанных подзапросов	Да	Да	Да
Производительность планировщика запросов для сложных запросов	Средняя (умеет параллельные запросы «из коробки»)	Плохая	Очень хорошая (GridSQL)
Наличие текстового процессора	Да	Да	Да
Поддержка последовательностей и автоматической нумерации	Да	Да	Да
Возможность откатить CREATE, ALTER	Да	Нет	Да

По результатам сравнительного анализа СУБД PostgreSQL опережает своих конкурентов.

Так же приведем список проектов и компаний, которые используют в качестве СУБД PostgreSQL:

1. Yahoo!
2. MySpace
3. OpenStreetMap
4. Sony Online Entertainment
5. BASF
6. hi5.com
7. Skype
8. Sun xVM
9. Evergreen

Глава 2

Диаграммы развертывания

Физическое представление программной системы не может быть полным, если отсутствует информация о том, на каких вычислительных средствах она реализована. Для представления общей конфигурации и топологии распределенной программной системы предназначены диаграммы развертывания. Диаграмма развертывания предназначена для визуализации элементов и компонентов программы. При этом представляются только компоненты-экземпляры

программы, являющиеся исполняемыми файлами или динамическими библиотеками. Диаграмма развертывания содержит графические изображения процессоров, устройств, процессов и связей между ними. При разработке диаграммы развертывания преследуются следующие цели: 1) определить распределение компонентов системы по ее физическим узлам; 2) показать физические связи между всеми узлами реализации системы на этапе ее исполнения; В качестве элементов диаграммы используются: аппаратные компоненты («узлы») существуют (например, веб-сервер, сервер базы данных, сервер приложения), программные компоненты («Artifact») работают на каждом узле (например, веб-приложение, база данных), различные части этого комплекса соединяются, друг с другом изображаются отрезками линий без стрелок. Узлы представляются как прямоугольные параллелепипеды с артефактами, расположенными в них, изображенными в виде прямоугольников. Узлы могут иметь подузлы, которые представляются как вложенные прямоугольные параллелепипеды. Существует два типа узлов: узел устройства («Device»), узел среды выполнения (execution environment).

Опишем несколько представлений программной системы разрабатываемой в рамках дипломного проекта. На первой диаграмме развертывания покажем связь и работу, которая реализована на данный момент: На диаграмме показано, что программная система состоит из сервера базы данных (DataBaseServer) и рабочей станции, которая имеет подключение к данному серверу. Справа представлена рабочая станция, на которой инсталлировано и выполняется приложение podsystem.exe, которое является приложением, разработанным в дипломном проекте. Слева представлен узел устройства – сервер базы дынных, на котором развернуто такая среда выполнения как СУБД PostgreSQL, на которой в свою очередь может быть развернута любая база данных со своей группой таблиц (Group tables of DBMS) и двумя таблицами, таких как: таблица настроек (Table Settings), таблица записей аудита (Table Logs); которые создаются приложением podsystem.exe и инсталлируются в базу данных, к которой приложение имеет подключение, а также с триггером (Trigger Log), генерируемым в приложении podsystem.exe и также инсталлируемым в базу данных, как и две выше описанных таблицы. Общий смысл диаграммы в том, что на рабочей станции имеется приложение podsystem, которое подключается к базе данных, находящейся под управлением системы PostgreSQL и располагающейся на каком-то сервере базе данных, и добавляет в эту базу данных необходимые таблицы, а так же триггер со своими параметрами для аудита.

Данная диаграмма представлена на Рисунке 1.

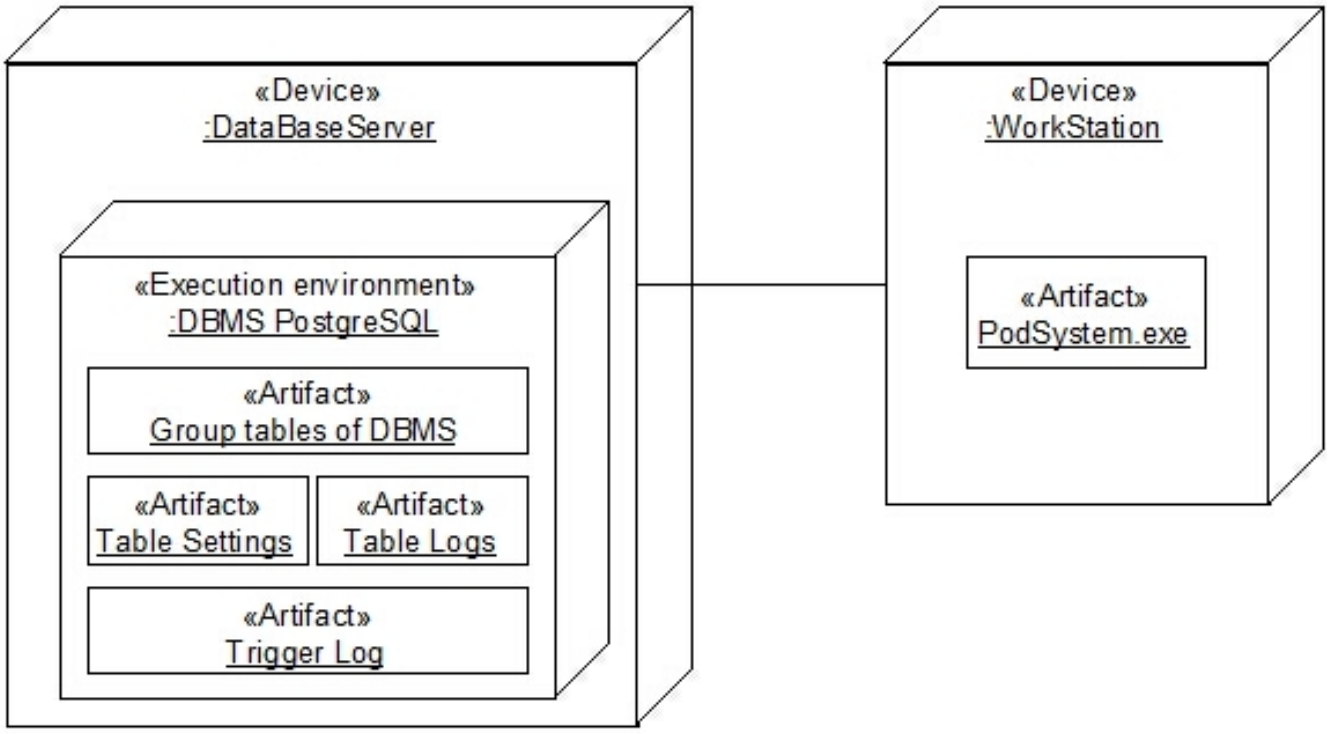

Рис. 1: Диаграмма

На второй диаграмме развертывания покажем связь и работу при более широком возможном использовании данного проекта: На диаграмме представлена расширенная версия первой диаграммы. Дополнение: есть узел устройства веб-сервер (WebServer), на котором развернута некая среда выполнения приложений (Application) и есть сервер приложений (ApplicationServer), между этими двумя серверами установлено соединение. Сам сервер приложений может использовать базу данных на сервере базы данных, по также установленному между ними соединению. Общий смысл диаграммы в том, что также имеется рабочая станция, подключение к БД, сервер БД, как и в предыдущей диаграмме. Добавляется только возможность использования сервера баз данных неким другим сервером приложений. То есть можно предположить существование некоего веб-сервера, на котором установлены свои приложения, обслуживающие его работу, и работающие с сервером приложений, приложения которого могут использовать базу данных на сервере БД. Данная диаграмма представлена на Рисунке 2.

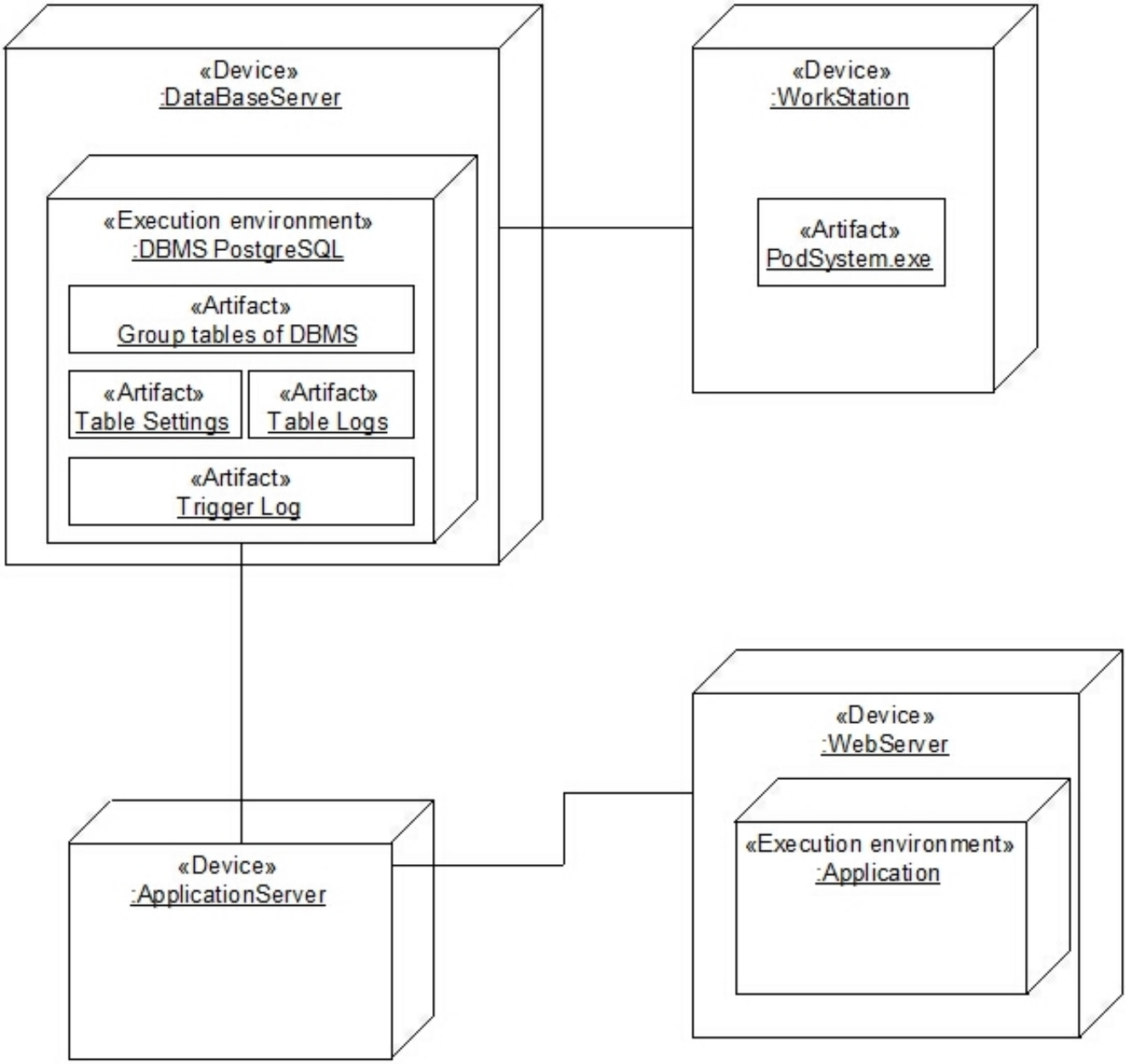

Рис. 2: Диаграмма

На третьей диаграмме развертывания покажем связь и работу при более широком возможном использовании проекта из второй диаграммы: На диаграмме представлена расширенная версия второй диаграммы. Дополнение: есть ещё один узел устройства - сервер баз данных (DataBaseServer2), на котором, как и на первом сервере БД, развернута среда выполнения - СУБД PostgreSQL, на которой в свою очередь может быть развернута любая база данных со своей группой таблиц (Group tables of DBMS2) и двумя таблицами: таблица настроек (Table Settings2), таблица записей аудита (Table Logs2); которые создаются приложением podsystem и инсталлируются в базу данных, к которой приложение имеет подключение, а также с триггером (Trigger Log2), генерируемым в приложении podsystem и также инсталлируемым в базу данных, как и две выше описанных таблицы. Общий смысл диаграммы в том, что также имеется рабочая станция, подключение к БД, сервер БД, сервер приложений, веб-сервер, как и в предыдущей диаграмме. Добавляется только возможность подключения приложения podsystem ко второй базе данных, также под управлением системы PostgreSQL и распола-

гающейся на другом сервере базе данных. Приложение podsystem также добавляет в эту базу данных необходимые таблицы и триггер со своими параметрами для аудита. В данном проекте возможность работы с несколькими базами данных не реализована, но в дальнейшем может быть добавлена. Данное добавление значительно расширяет функциональность проекта. Данная диаграмма представлена на Рисунке 3.

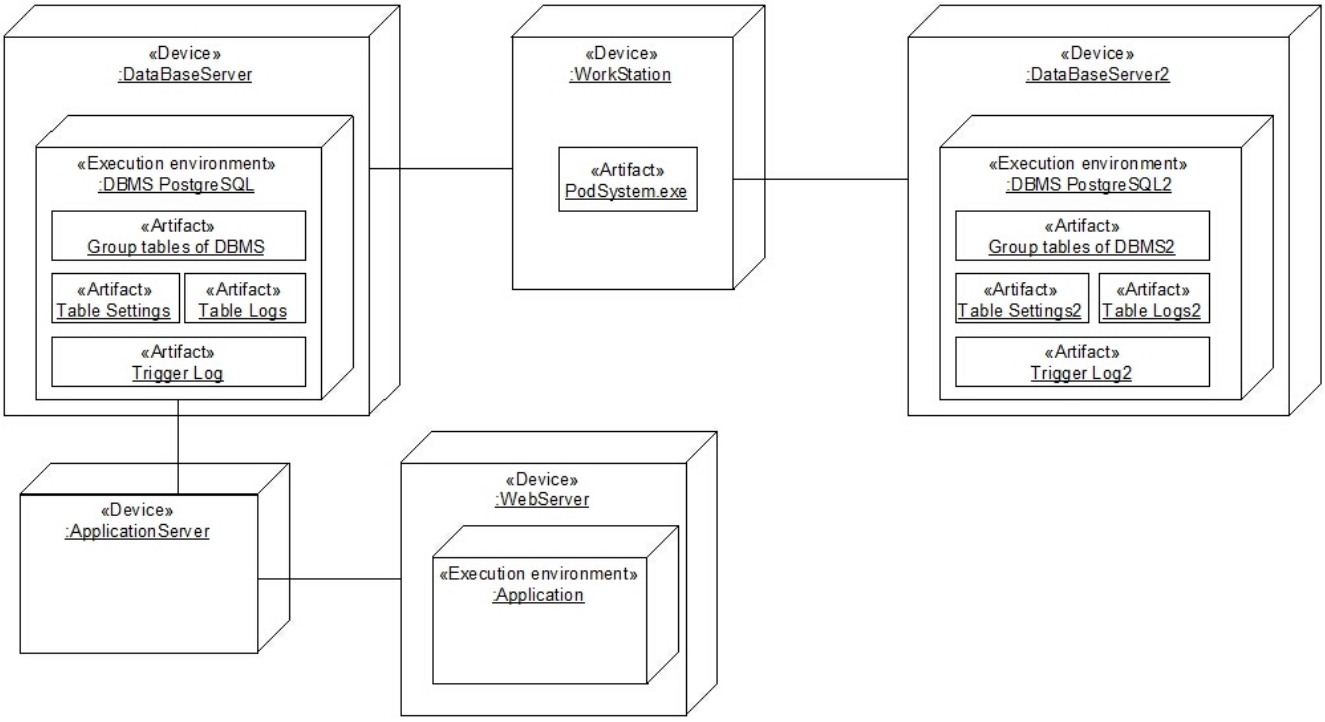

Рис. 3: Диаграмма

На четвертой диаграмме развертывания покажем связь и работу при самом широком возможном использовании проекта: На диаграмме представлена расширенная версия третьей диаграммы. Дополнение: есть соединение между узлом - сервер приложений (ApplicationServer) и узлом - сервер баз данных (DataBaseServer2). Сервер приложений может использовать, по установленным между ними соединению, обе базы данных на сервере базы данных и сервере базы данных 2. Общий смысл диаграммы в том, что также имеется рабочая станция, подключение к БД, сервер БД, сервер приложений, веб-сервер, возможность подключения приложения podsystem ко второй базе данных, также под управлением системы PostgreSQL и располагающейся на другом сервере базе данных. Добавляется возможность использования приложениями с сервера приложений второй базы данных, таким ж образом как это происходит в случае работы приложения podsystem. Данная диаграмма представлена на Рисунке 4.

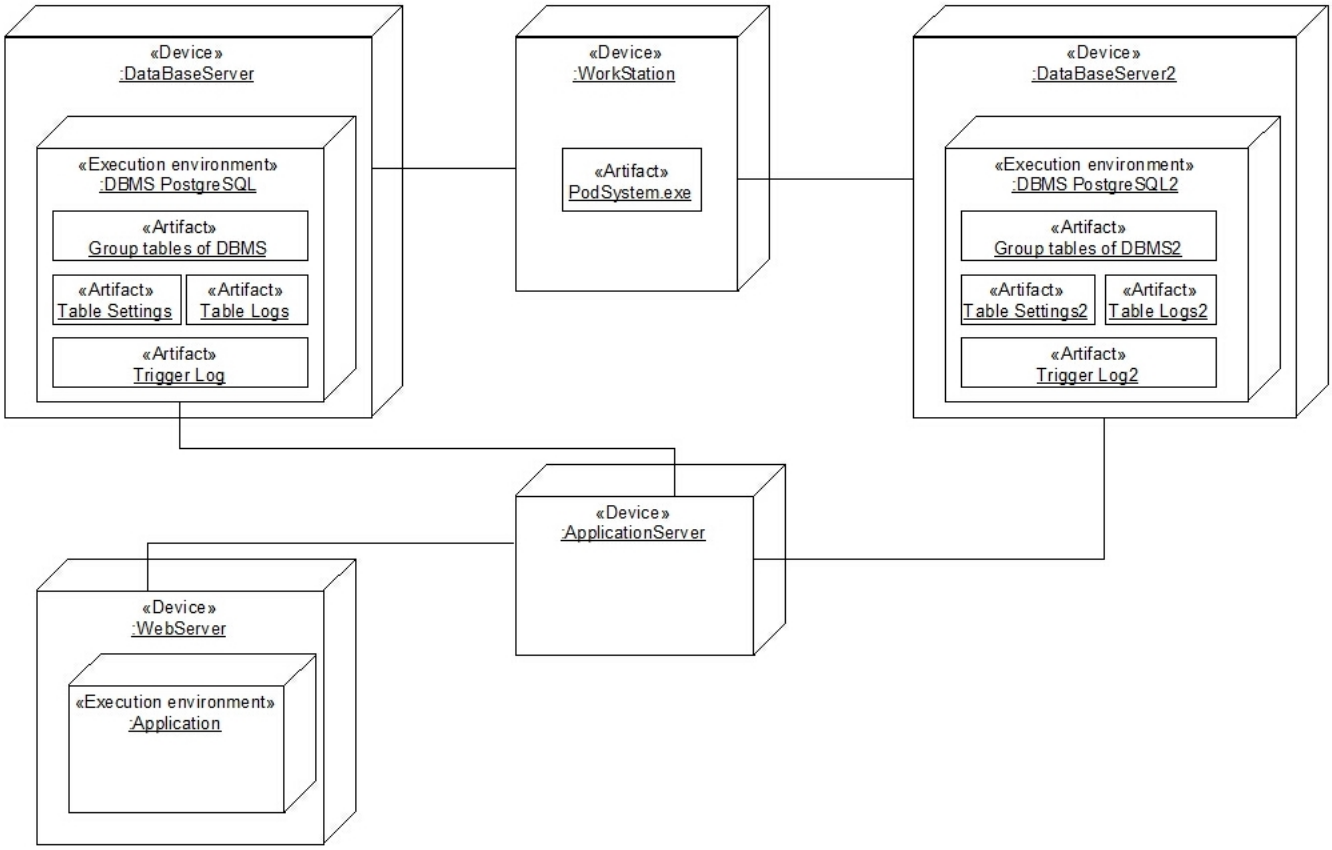

Рис. 4: Диаграмма

Диаграммы деятельности

Диаграммы деятельности позволяют моделировать сложный жизненный цикл объекта, с переходами из одного состояния (деятельности) в другое. Но этот вид диаграмм может быть использован и для описания динамики совокупности объектов. Они применимы и для детализации некоторой конкретной операции. Диаграммы деятельности описывают переход от одной деятельности к другой. Действия показаны скругленными прямоугольниками; ромб - символ принятия решения с обозначениями условий возле переходов; жирная линия означает распараллеливание, а затем опять слияние воедино (синхронизацию) потоков управления. Далее рассмотрим диаграмму деятельности для разрабатываемого проекта, которая представлена на рисунке 5.

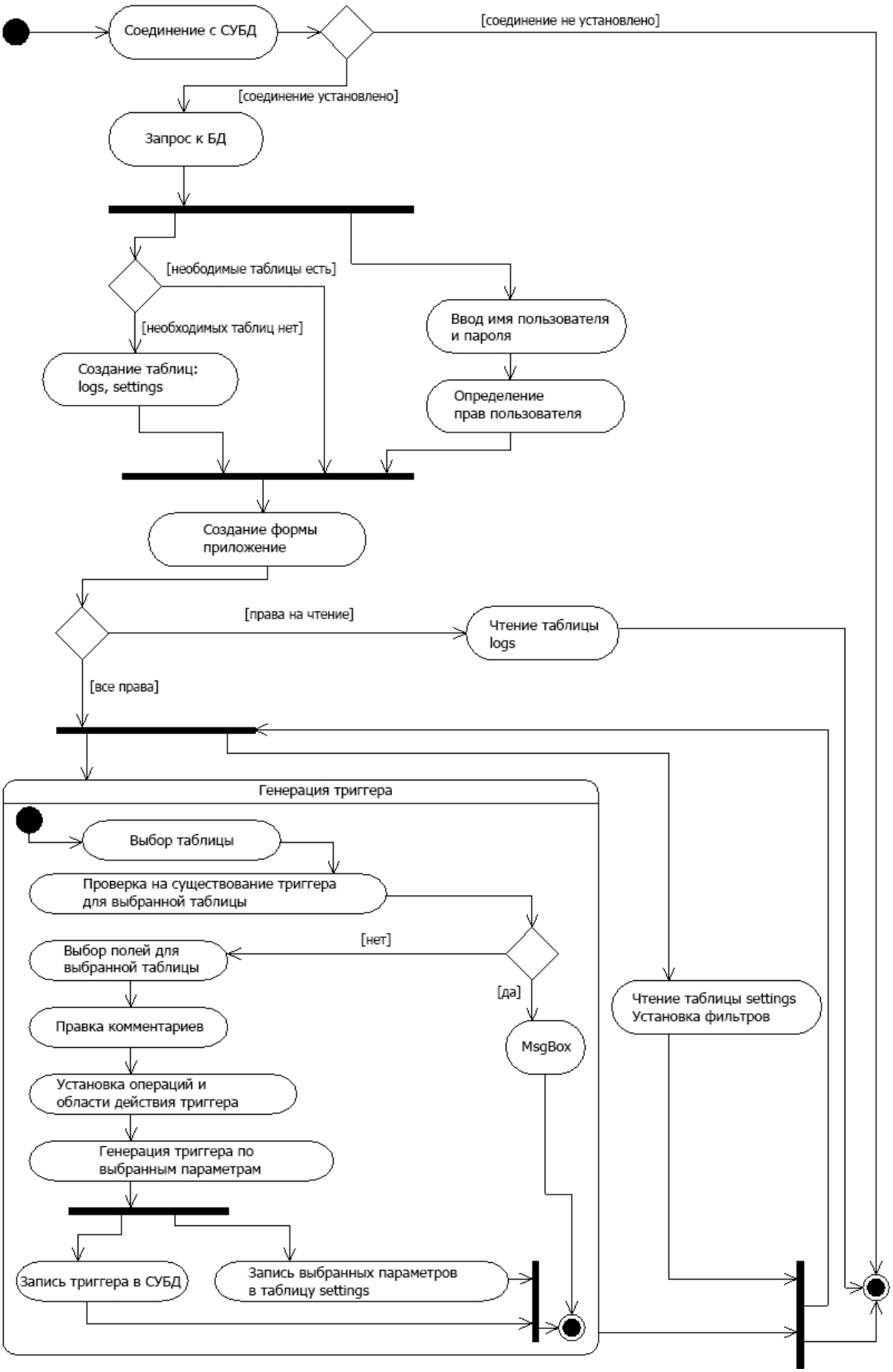

Рис. 5: Диаграмма

По диаграмме видно, что после запуска программы происходит операция соединение с СУБД, данное соединение происходит благодаря заранее созданному драйверу PostgreSQL. Далее следует символ принятия решения – если соединение не установлено, то дальнейшая работа приложения невозможна, и оно приходит к завершению, если же соединение установлено, то переходим к следующему этапу – запросу к БД. Операция запроса БД определяет, есть ли уже в БД необходимые для работы приложения таблицы. Далее идет распараллеливание, а именно одновременно выполняется две операции: 1) в зависимости от результата предыдущей операции принимается решение, если таблицы существуют, то происходит переход к следующей операции, если нет, то необходимые таблицы, а это таблицы настроек и таблица записей аудита, создаются в БД. 2) осуществляется операция ввода пароля и пользователя и дальнейшее определения прав этого пользователя. Далее происходит слияние потоков управления и создается форма приложения в которую загружается таблица записей аудита, если она была создана на предыдущем этапе, то пустая, если с ней была произведена какая-то работа, то с набором записей. В зависимости от того какие права были определены на предыдущем этапе для данного пользователя далее поток опять разветвляется. Если у данного пользователя имеются права только на чтение, то для него следует операция чтения без редактирования таблицы записей аудита и без изменения и чтения настроек, после чего возможен только выход из приложения. Если ж для пользователя определены все права, то переходим к следующей операции-генерации триггера, которая будет рассмотрена более подробно отдельно. Параллельно операции генерации, либо после нее возможна операция чтение и редактирование таблицы настроек, а также установка фильтров для просмотра таблицы записей аудита, которые находятся на форме приложения. После этих операций возможен либо выход либо их неоднократный повтор. Операция генерации триггера: генерация триггера начинается с выбора таблицы, далее выполняется операция проверки на существование в БД триггера на выбранную таблицу, поскольку создание нескольких триггерных функций на одну таблицу невозможна; После этого происходит ветвлении на две операции: 1) если есть такая триггерной функция, то появляется сообщение об этом 2) если нет, то переходим к следующему этапу. Следующий этап заключается в операции выбора полей для выбранной таблицы. Потом можно добавить комментарии для выбранных полей или же оставить их пустыми. Также возможно дальнейшая операция установки области действия триггера и установки операций на которые будет срабатывать данный триггер. После всех этих операция происходит распараллеливание. Параллельно выполняется операция записи триггера в БД и операция записи в таблицу настроек выбранных параметров (выбранные поля, выбранная таблица, имя пользователя, который совершил все операции при генерации). На этом генерация заканчивается.

Генерация триггера

Рассмотрим более подробно, как происходит генерация триггера в разрабатываемом проекте