**Федеральное государственное автономное образовательное учреждение**

**высшего образования   
«Национальный исследовательский университет   
«Высшая школа экономики»**

**Московский институт электроники и математики им. А.Н.Тихонова**

Направление подготовки

**«10.04.01 Информационная безопасность»**

Образовательная программа **«Кибербезопасность»**

**О Т Ч Е Т**

**о прохождении**

**производственной практики**

**Студент** Шараев Е.В.МКБ231

(Фамилия И.О.) номер группы

**Руководитель практики студента:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Руководитель группы систем мониторинга |  | Зубков А.А. |  |  |
| должность и место работы |  | Фамилия И.О. |  | подпись |

**Руководитель практики от НИУ ВШЭ:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Зав. каф. информационной безопасности киберфизических систем департамента электронной инженерии МИЭМ НИУ ВШЭ |  | Евсютин О.О. |  |  |
| должность и место работы |  | Фамилия И.О. |  | подпись |

**Практика пройдена с оценкой 10**

**Дата 17.06.2024**

**Москва, 2024**

**СОДЕРЖАНИЕ**

[1 Введение 3](#_heading=h.gjdgxs)

[2 Краткая характеристика организации 4](#_heading=h.30j0zll)

[3 Описание профессиональных задач 5](#_heading=h.1fob9te)

[4 Исполненное индивидуальное задание 6](#_heading=h.3znysh7)

4.1 Сборка и установка из исходного кода СУБД PostgreSQL 6

4.2 Архитектура СУБД PostgreSQL 6

4.3 Методы обеспечения безопасности данных в СУБД PostgreSQL 6

4.4 Методы обеспечения надежности хранения данных в СУБД PostgreSQL 6

[5 Заключение 7](#_heading=h.2et92p0)

[6 Список использованных источников 8](#_heading=h.tyjcwt)

[ПРИЛОЖЕНИЕ А. Основные требования к оформлению отчета 9](#_heading=h.1t3h5sf)

[ПРИЛОЖЕНИЕ Б. Пример списка использованных источников 14](#_heading=h.4d34og8)

# 1 Введение

Целью прохождения целью прохождения производственной практики является получение профессиональных умений и опыта профессиональной деятельности, в первую очередь научно-исследовательской работы.

Задачи практики:

– закрепление и углубление теоретических знаний по прослушанным за время обучения в университете дисциплинам;

– адаптация студента к реальным условиям работы в различных учреждениях и организациях, приобретение опыта работы в трудовых коллективах, планирование работы в организации, коммуникация и общение в сфере будущей профессиональной деятельности;

– создание условий для практического применения знаний в области кибербезопасности;

– формирование и совершенствование базовых профессиональных навыков и умений в области применения современных информационных технологий к задачам кибербезопасности;

– выполнений обязанностей на первичных должностях в области обеспечения кибербезопасности;

– диагностика профессиональной пригодности студента к профессиональной деятельности;

– формирование информационной компетентности с целью успешной работы в профессиональной деятельности;

– обеспечение успеха дальнейшей профессиональной карьеры.

# 2 Краткая характеристика организации

Приводится краткая характеристика организации (места прохождения практики) с описанием сферы деятельности, организационной структуры, экономическими показателями.

Общество с ограниченной ответственностью «Постгрес Профессиональный» (ООО «ППГ»), располагается по адресу: 117292, г. Москва, ул. Дмитрия Ульянова, д. 7А.

ООО «Постгрес Профессиональный» является российским разработчиком системы управления базами данных Postgres Pro на основе PostgreSQL. СУБД Postgres Pro входит в Единый реестр российского программного обеспечения и выпускается в четырех вариантах: Postgres Pro Standard, Postgres Pro Enterprise, а также их сертифицированные ФСТЭК версии.

В команде ООО «Постгрес Профессиональный» — все ключевые российские разработчики Postgres, благодаря чему компания эффективно решает любые сложные задачи: дополняет Postgres новой функциональностью (включая разработку ядра), обеспечивает безопасность и оказывает поддержку полного цикла: аудит, помощь в миграции, техническую поддержку 24х7 и SLA, обучение, заказные разработки, сертификацию специалистов.

ООО «Постгрес Профессиональный» — самая популярная для импортозамещения зарубежных решений СУБД в России. Среди заказчиков ООО «Постгрес Профессиональный» – крупнейшие федеральные и частные компании, в том числе объекты критической инфраструктуры, требующие наиболее высоких стандартов обеспечения безопасности информации.

ООО «Постгрес Профессиональный» занимает второе место в мировом рейтинге контрибьюторов открытой СУБД PostgreSQL, являясь одной из крупнейших команд проекта.

# 3 Описание профессиональных задач

В рамках производственной практики были изучены методы установки и настройки СУБД PostgreSQL и Postgres Pro, их внутреннее устройство и архитектура, а также рассмотрены вопросы обеспечения безопасности данных и надежности их хранения.

# 4 Исполненное индивидуальное задание

**4.1 Сборка и установка из исходного кода СУБД PostgreSQL**

PostgreSQL является программой, входящей в категорию систем управления базами данных. Когда эта программа запущена, мы называем ее сервером PostgreSQL или экземпляром сервера или просто сервером, для краткости.

Данные, управляемые сервером PostgreSQL, хранятся в базах данных. Один экземпляр сервера может работать с несколькими базами данных одновременно. Этот набор баз данных называется кластером баз данных. Таким образом, кластер баз данных представляет собой данные в файлах, а сервер или экземпляр сервера - это программа, управляющая этим кластером баз данных.

Практическая реализация сборки и установки сервера СУБД PostgreSQL в Приложении 1.

Демонстрация конфигурационного файла сервера и примеры установки параметров конфигурации в Приложении 2.

**4.2 Архитектура СУБД PostgreSQL**

В PostgreSQL применяются принципы изоляции и многоверсионности, чтобы обеспечить целостность данных и согласованность транзакций. Давайте рассмотрим эти принципы подробнее.

**4.2.1 Изоляция транзакций**

Существуют четыре уровня изоляции транзакций, определенные стандартом SQL:

1. Read Uncommitted (чтение незафиксированных данных): Транзакция может читать изменения, сделанные другими транзакциями, даже если эти транзакции еще не зафиксированы. Этот уровень изоляции в PostgreSQL не поддерживается, так как даже на уровне Read Committed транзакция может читать только зафиксированные данные.

2. Read Committed(чтение зафиксированных данных): Транзакция видит только те изменения, которые были зафиксированы другими транзакциями на момент начала каждого запроса внутри транзакции. Это уровень изоляции по умолчанию в PostgreSQL.

3. Repeatable Read (повторяемое чтение): Транзакция видит только те изменения, которые были зафиксированы на момент начала самой транзакции, и все изменения, сделанные самой транзакцией. В течение всей транзакции видимость данных не изменяется, несмотря на изменения, сделанные параллельными транзакциями.

4. Serializable(сериализуемость): Транзакции выполняются таким образом, как если бы они были выполнены последовательно, одна за другой. Это самый строгий уровень изоляции, предотвращающий любые возможные проблемы параллельного выполнения, такие как фантомные чтения, неповторяемые чтения и грязные чтения.

**4.2.2 Многоверсионность** (MVCC - Multi-Version Concurrency Control)

Многоверсионность в PostgreSQL реализована с помощью механизма MVCC, который позволяет каждой транзакции видеть "снимок" базы данных на определенный момент времени. Это достигается за счет создания новых версий строк в таблицах при внесении изменений. Старые версии строк сохраняются, пока они могут быть видимы каким-либо действующим транзакциям. Основные принципы MVCC включают:

1. Снимки транзакций: Каждая транзакция видит только те изменения, которые были зафиксированы до ее начала, и свои собственные изменения. Это обеспечивает согласованность данных без необходимости блокировок на чтение.

2. Создание новых версий строк: При обновлении строки создается новая версия этой строки с новым идентификатором транзакции (XID). Старая версия сохраняется до тех пор, пока не будет больше транзакций, которые могли бы ее видеть.

3. Удаление старых версий (VACUUM): Процесс VACUUM удаляет старые версии строк, которые больше не нужны, освобождая место в базе данных. Это необходимо для предотвращения раздувания базы данных и поддержания производительности.

4. Таймстампы и XID: Каждая строка имеет метки времени или идентификаторы транзакций, которые определяют, в какой момент времени эта версия строки была создана и когда она была удалена (если была). Это позволяет PostgreSQL определять видимость строки для каждой конкретной транзакции.

Эти принципы обеспечивают высокую производительность и согласованность данных в PostgreSQL, делая его мощным инструментом для работы с высоконагруженными и многопользовательскими системами.

Подробный практический разбор принципов изоляции и многоверсионности в Приложении 3.

**4.2.3 Буферный кеш и журнал предзаписи**

Буферный кэш служит для сглаживания разницы в скорости между оперативной памятью и дисковыми накопителями. Он представляет собой массив буферов, содержащих страницы данных и дополнительную информацию, такую как имя файла и положение страницы внутри него. Размер страницы обычно составляет 8 Кбайт и может быть изменен только при сборке PostgreSQL.

Любая работа с данными осуществляется через буферный кэш. Если процессу требуется страница, он сначала ищет ее в кэше. При отсутствии страницы в кэше процесс запрашивает операционную систему для ее чтения и затем помещает страницу в буферный кэш. Операционная система может либо прочитать страницу с диска, либо обнаружить ее в собственном кэше. После размещения страницы в буферном кэше к ней можно обращаться многократно без дополнительных затрат на вызовы операционной системы. Если процесс изменил данные на странице, буфер помечается как "грязный". Измененная страница должна быть записана на диск, но для повышения производительности эта запись происходит асинхронно и может быть отложена.

Буферный кэш, как и другие структуры общей памяти, защищен блокировками для управления одновременным доступом. Несмотря на эффективную реализацию блокировок, доступ к буферному кэшу не так быстр, как прямое обращение к оперативной памяти. Поэтому запросы, которые читают и изменяют меньше данных, выполняются быстрее.

Размер буферного кэша обычно недостаточен, чтобы вместить всю базу данных. Его объем ограничивается доступной оперативной памятью и увеличивающимися накладными расходами при его росте. Поэтому при чтении новой страницы рано или поздно место в буферном кэше заканчивается, и используется алгоритм вытеснения страниц. Этот алгоритм выбирает страницу, которая использовалась реже других. Если выбранный буфер помечен как "грязный", страница записывается на диск, чтобы сохранить изменения. Затем в освобожденный буфер загружается новая страница.

Такой алгоритм вытеснения называется LRU (Least Recently Used). Он сохраняет в кэше данные, с которыми активно работают. Эти "горячие" данные обычно составляют небольшую часть всей базы данных, поэтому при достаточном объеме буферного кэша удается существенно сократить количество обращений к операционной системе и дисковых операций.

**4.3 Методы обеспечения безопасности данных в СУБД PostgreSQL**

**4.3.1 Управление доступом к данным: Роли и их привилегии**

В основном, роль в PostgreSQL представляет собой пользователя базы данных. Роль также может выполнять функцию группы пользователей. Роли не связаны с именами пользователей операционной системы, хотя стандартные программы часто используют имя пользователя ОС в качестве имени роли по умолчанию. Например, при запуске psql без указания имени роли, будет использоваться имя текущего пользователя ОС. Роли являются общими для всего кластера: одна роль может подключаться к различным базам данных и владеть объектами в этих базах.

Роль обладает определенными атрибутами, которые определяют ее основные характеристики и права, не касающиеся доступа к объектам. Эти атрибуты обычно имеют два состояния, например, CREATEDB (разрешает создание баз данных) и NOCREATEDB (запрещает создание баз данных). Как правило, по умолчанию выбирается ограничивающий вариант. В таблице указаны только некоторые из атрибутов. Атрибуты INHERIT и BYPASSRLS будут рассмотрены позже в этом модуле.

Практическая реализация создания ролей, их атрибутов и выдачи / отъем привилегий в Приложении 4.

**4.3.1 Настройка подключения и аутентификация и СУБД PostgreSQL**

При подключении клиента сервер выполняет несколько задач:

1. Идентифицируется пользователь, то есть определяет его имя. Пользователь представляется, но указанное имя может отличаться от имени пользователя базы данных (например, если он использует свое имя в ОС).

2. Сервер аутентифицирует пользователя, проверяя, что он действительно тот, за кого себя выдает, например, с помощью ввода пароля.

3. Сервер авторизует пользователя, проверяя, имеет ли он право на подключение (эта задача частично пересекается с управлением привилегиями).

Иногда все три задачи объединяются под общим термином «аутентификация». PostgreSQL предоставляет гибкие настройки для этого процесса.

**4.4 Методы обеспечения надежности хранения данных в СУБД PostgreSQL**

В PostgreSQL существует несколько методов резервного копирования (бэкапа) базы данных, каждый из которых имеет свои преимущества и применяется в различных сценариях. Рассмотрим основные методы резервного копирования в PostgreSQL.

**4.4.1. Логическое резервное копирование - pg\_dump**

`pg\_dump` — это утилита для создания логических резервных копий отдельных баз данных. Она может создавать дампы в различных форматах: текстовом, пользовательском (custom), tar и directory.

*$ pg\_dump -U username -F c -b -v -f mydatabase.backup mydatabase*

-U username - Имя пользователя для подключения к базе данных.

-Fc - Формат пользовательской резервной копии.

-b - Включить большой объект в дамп.

-v - Подробный вывод (verbose).

-f mydatabase.backup - Имя файла резервной копии.

*$ pg\_restore -U username -d mydatabase -v mydatabase.backup*

-d mydatabase - Имя базы данных, в которую будет выполнено восстановление.

**4.4.2. Физическое резервное копирование**

`pg\_basebackup` — утилита для создания физической резервной копии всего кластера баз данных PostgreSQL. Этот метод полезен для создания бэкапов в случае необходимости полной копии базы данных с минимальными простоями.

*$ pg\_basebackup -U username -D /path/to/backup -Fp -Xs -P -v*

-U username - Имя пользователя для подключения.

-D /path/to/backup - Директория для сохранения резервной копии.

-Fp - Формат простого файла (plain file).

-Xs - Включить архив WAL в резервную копию.

-P - Показать прогресс выполнения.

-v - Подробный вывод (verbose).

**4.4.3. Снимок файловой системы (Filesystem Snapshot)**

Этот метод включает создание моментального снимка (snapshot) файловой системы, на которой находится база данных PostgreSQL. Он может быть использован вместе с механизмами файловой системы, такими как LVM (Logical Volume Manager) или ZFS.

**4.4.4. Архивирование WAL (Write-Ahead Logging)**

Этот метод позволяет выполнять непрерывное резервное копирование базы данных, архивируя файлы журналов изменений (WAL). Это обеспечивает возможность восстановления базы данных на любой момент времени.

В файле настройки сервера PostgreSQL нужно добавить следующие параметры:

*archive\_mode = on (Включить режим архивации)*

*archive\_command = 'cp %p /path/to/archive/%f' (Команда для копирования файлов)*

*archive\_timeout = 60 (Периодичность создания архива)*

Выбор метода резервного копирования зависит от преследуемых целей и методов использования дампа. Логическое резервное копирование (pg\_dump) подходит для резервного копирования отдельных баз данных, обеспечивает гибкость в восстановлении отдельных объектов и структур. Физическое резервное копирование (pg\_basebackup) идеально для создания полной копии всего кластера баз данных, включая настройки и файлы конфигурации. Снимок файловой системы - быстрый метод создания моментальных снимков текущего состояния базы данных, полезен для крупных баз данных с минимальными простоями. Архивирование WAL - обеспечивает возможность восстановления базы данных до любого момента времени, полезен для систем с высокими требованиями к отказоустойчивости и минимизации потерь данных.

Каждый из методов может быть использован отдельно или в комбинации в зависимости от требований к резервному копированию и восстановлению данных.

**4.5 Репликация**

Один единственный сервер, управляющий всеми базами данных, может не соответствовать требованиям из-за двух основных причин. Во-первых, это отказоустойчивость: такая архитектура представляет собой потенциальную точку отказа. Если сервер выходит из строя, вся система становится недоступной. Во-вторых, производительность: один сервер может не справиться с большой нагрузкой. Часто более предпочтительным является масштабирование, то есть распределение нагрузки на несколько серверов, вместо увеличения мощности одного сервера.

Таким образом, речь идет о наличии нескольких серверов, работающих с теми же базами данных. Под репликацией понимается процесс синхронизации данных между этими серверами.

Существуют различные способы организации взаимодействия между несколькими серверами, и в PostgreSQL доступны два основных подхода: физическая и логическая репликация.

**4.5.1 Физическая репликация**

В физической репликации серверы выполняют определенные роли: мастер и реплика. Мастер передает журнальные записи на реплику в форме файлов или потока записей. Реплика затем применяет эти записи к своим данным.

Процесс применения происходит механически, без осознания смысла изменений, поэтому важна совместимость в двоичном формате между серверами (то есть одинаковые платформы и версии PostgreSQL). Поскольку журнал является общим для всего кластера, репликация возможна только для всего кластера в целом.

**4.5.2 Логическая репликация**

Логическая репликация включает в журнал информацию более высокого уровня, которая позволяет реплике понимать изменения на уровне строк таблиц (устанавливается параметр wal\_level = logical). Для такого типа репликации не требуется двоичная совместимость; важно лишь, чтобы реплика могла декодировать логическую информацию, содержащуюся в журнале. Логическая репликация позволяет выбирать, какие изменения применять, например, применять изменения только для определенных таблиц. Этот метод доступен с версии 10 PostgreSQL; более ранние версии должны использовать расширение pg\_logical или организовывать репликацию с помощью триггеров.

Примеры настройки резервного копирования и репликации в Приложении 5.

# 5 Заключение

Приводятся выводы о результатах прохождения практики.

# 6 Список использованных источников

Список библиографических ссылок, оформленных по ГОСТ приведены в Приложении 5.