Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатики и систем управления

КАФЕДРА Теоретической информатики и компьютерных технологий

ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ

Студент Локшин Вячеслав Андреевич
фамилия, имя, отчество

Группа ИУ9-41Б

Тип практики Технологическая

Название предприятия ЗАО «НОРСИ-ТРАНС»

Студент _____ _____
подпись, дата *фамилия, и.о.*

Рекомендуемая оценка: _____

Руководитель практики
от предприятия: _____ _____
подпись, дата *фамилия, и.о.*

Руководитель практики _____ _____
подпись, дата *фамилия, и.о.*

Оценка _____

2023 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью производственной практики является закрепление знаний по изучаемым дисциплинам и получение студентами практических навыков в период пребывания на предприятии, в связи с чем можно выделить следующие задачи данной практики:

1. Ознакомление с предприятием, его программными продуктами и предметной областью, для которой предназначены эти продукты.
2. Изучение жизненного цикла программного обеспечения, создаваемого на предприятии.
3. Ознакомление с использованием сетевых технологий и распределённых вычислений на предприятии.
4. Изучение профессиональных взаимоотношений в рамках организационной структуры подразделения прохождения практики, организационных мер, направленных на эффективную совместную работу программистов и других специалистов над общим проектом.
5. Изучение процесса разработки программного обеспечения.
6. На примере задания, данного руководителем практики от предприятия, получение навыков работы с промышленными средствами: автоматизированной сборки, отладки, контроля версий и разрешения конфликтов версий, анализа кода, автоматизированного тестирования, профилирования.

ХАРАКТЕРИСТИКА ПРЕДПРИЯТИЯ ЗАО «НОРСИ-ТРАНС»

ЗАО «НОРСИ-ТРАНС» — российский разработчик и производитель серверного оборудования, систем хранения данных на различных платформах, телекоммуникационного оборудования и вычислительных платформ.

ЗАО «НОРСИ-ТРАНС» работает с 1995 года и специализируется на создании и выпуске:

- серверного оборудования;
- высокоплотных систем хранения данных;
- HPC-вычислительных платформ;
- телекоммуникационного оборудования;
- высокосложных корпусов включая полный цикл выпуска КД;
- печатных плат в составе выпускаемой продукции;
- системного и прикладного программного обеспечения для различных аппаратных платформ.

Компания разрабатывает и создает готовые программно-аппаратные комплексы под специализированные требования заказчиков. Клиенты Компании — операторы сотовой подвижной связи, операторы фиксированной связи, интернет — провайдеры. ЗАО "НОРСИ-ТРАНС" ведет сотрудничество с крупнейшими российскими поставщиками услуг связи, такими как «МТС», «Мегафон», «Ростелеком», «Национальные кабельные сети», «Нэт Бай Нэт Холдинг».

Кроме этого, в компании ведутся уникальные разработки с использованием собственных программных продуктов, научно-исследовательские и опытно-конструкторские работы. ЗАО «НОРСИ-ТРАНС» с мая 2009 года входит в состав общественно-государственного объединения «Ассоциация документальной электросвязи» и с 2010 года входит в состав наблюдателей, а с 2015 года - членом Европейского Института Телекоммуникационных Стандартов (ETSI).

В этом году, для студентов провели лекции, на которых представители направлений компании рассказывали о задачах, базовых концепциях и продуктах Компании. Лекции затрагивали следующие темы:

1. Опыт использования методов машинного обучения в проектах ЗАО «НОРСИ-ТРАНС» (биометрическая идентификация, распознавание рукописного текста, распознавание речи, анализ текста).
2. Графовые базы данных.
3. Технология блокчейн. Структура данных и принцип работы сети Bitcoin.
4. Криптовалюта Ethereum. Смарт-контракты.
5. Чем лабораторная работа отличается от промышленного кода.
6. Внутреннее устройство хранилища Clickhouse.
7. Опыт дипломного проектирования в МГТУ им.Н.Э.Баумана.

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

2.1 Постановка задачи

PostGIS — это расширение географической базы данных для PostgreSQL, которое предоставляет возможность эффективно хранить и обрабатывать географические данные. PostGIS использует пространственные индексы и функции для обеспечения быстрого доступа и выполнения сложных запросов.

Elasticsearch — это распределенный поисковый и аналитический движок, который также имеет поддержку гео-пространственных запросов. Elasticsearch использует специализированные индексы и запросы для обработки и анализа данных.

Целью работы является сравнение производительности баз данных Elasticsearch и PostGIS при работе с геометрическими объектами. Суть данного сравнения заключается в оценке производительности обоих инструментов при выполнении гео-пространственных запросов на реальных данных. Будут использоваться стандартные запросы, такие как расчет расстояния между точками, поиск ближайших объектов и проверка принадлежности точки области.

В рамках практики была поставлена задача разработать набор тестов для выполнения гео-пространственных запросов как в PostGIS, так и в Elasticsearch. Тестирование будет проведено на наборе данных, содержащем информацию о географических объектах различных типов.

Целью тестирования будет оценка производительности каждого инструмента и выявление их преимуществ и ограничений при работе с гео-пространственными данными.

Таким образом индивидуальное задание состоит из следующих шагов:

1. Изучить документацию PostGis. [1].
2. Изучить документацию Elasticsearch. [2].
3. Организовать архитектуру и инфраструктуру проекта.

4. Создать генератор географических объектов в пределах Московской области.
5. Написать интерфейс для работы с базой данных.
6. Реализовать интерфейс для PostGis.
7. Реализовать интерфейс для Elasticsearch.
8. Написать бенчмарки для сравнения производительности.
9. Провести анализ и оценить результаты.

2.2 Изучение предметной области

Географические данные представляют собой информацию о местоположении объектов на Земле. Гео-пространственные запросы позволяют работать с этими данными, выполнять операции, такие как расчет расстояния между точками, поиск объектов в определенной области и другие операции. Для этого исследования было решено использовать координаты в пределах Московской области.

PostGIS предоставляет множество функций для выполнения гео-пространственных запросов, таких как определение ближайших объектов, поиск в пределах определенного радиуса и агрегация данных.

Данные в Elasticsearch организовываются в секции и реплики для обеспечения высокой доступности и распределенности. Но в данной работе все замеры будут производиться на одной секции без реплик.

Для сравнения производительности между PostGIS и Elasticsearch в выполнении гео-пространственных запросов, требуется создать тестовые наборы данных с различными типами гео-объектов. Затем нужно выполнить запросы, такие как поиск ближайших объектов, поиск в пределах определенного радиуса и другие типичные операции. После выполнения запросов необходимо измерить время выполнения и использование ресурсов системы. Это поможет сравнить производительность обоих инструментов и определить их преимущества и ограничения при работе с гео-пространственными данными.

Гео-объектами в данной работе будут являться точки, многоугольники и круги, соотвественно будет заведено две таблицы для каждой базы, хранящие точки или многоугольники. Сравнение производительности будет выполнено на запросах получения объектов в радиусе, в границах заданного многоугольника и других, о которых подробнее будет рассказано в пункте о реализации.

2.3 Реализация

2.3.1 Архитектура и инфраструктура

Работа началась с создания обеих баз данных для Elasticsearch и PostGis через docker, используя docker-compose — так приложение становится легко переносимым на другие устройства, им нужно только склонировать репозиторий и запустить. Языком реализации проекта является Golang с использованием чистой архитектуры.

2.3.2 Генерация объектов.

После изучения документации для работы с базами данных, был создан пакет для генерации различных гео-объектов. Пакет это организационная единица, которая содержит связанные между собой функции, типы и другие ресурсы для решения задач. Задача стояла в том, что все фигуры и точки должны лежать в пределах координат Московской области. Было решено пронять за границы области круг, центром которого являются координаты Красной Площади, а радиусом — средний радиус между максимально и минимально удаленной точки на границе Московской области. После этого можно было получать любое множество точек внутри этого круга.

После генерации точек нужно было генерировать замкнутые n-угольники без самопересечений — это условия для типа polygon в PostGis и Elasticsearc. Для генерации таких многоугольников берется N точек, которые сортируются по тангенсу угла относительно предварительно найденной центральной точки, потом многоугольник замыкается с первой точкой многоугольника. Для получения случайных значений использовался пакет rand из стандартной библиотеки языка Go. [3].

2.3.3 Создание интерфейса

Перед созданием запросов было утверждено два интерфейса, которому должны удовлетворять базы данных. Для таблиц, содержащих точки (Листинг 1), и таблиц, содержащих многоугольники (Листинг 2).

```
type Storage interface {
        Init(ctx context.Context) error
        Drop(ctx context.Context) error

        AddPoint(ctx context.Context, p internal.Point) error
        AddPointBatch(ctx context.Context, points []internal.Point) error

        GetInRadius(ctx context.Context, p internal.Point, radius int) ([]internal.Point, error)
        GetInPolygon(ctx context.Context, polygon []internal.Point) ([]internal.Point, error)

        GetInShapes(ctx context.Context, shapes internal.Shapes) ([]internal.Point, error)
}
```

Листинг 1 — Реализация интерфейса для таблиц, содержащих точки

```
type PolygonStorage interface {
        InitPolygon(ctx context.Context) error
        DropPolygon(ctx context.Context) error

        AddPolygon(ctx context.Context, polygon internal.Polygon) error
        AddPolygonBatch(ctx context.Context, polygons []internal.Polygon) error

        GetInRadiusPolygon(ctx context.Context, p internal.Polygon, radius int) ([]internal.Polygon, error)
        GetInPolygonPolygon(ctx context.Context, polygon internal.Polygon) ([]internal.Polygon, error)

        GetIntersectionPolygon(ctx context.Context, polygon internal.Polygon) ([]internal.Polygon, error)
        GetIntersectionPoint(ctx context.Context, point internal.Point) ([]internal.Polygon, error)
}
```

Листинг 2 — Реализация интерфейса для таблиц, содержащих многоугольники

2.3.4 Реализация интерфейса для PostGis

Для работы с PostGis в Golang был выбран драйвер "github.com/jackc/pgx/v5". [4].

Особенность PostGis в том, что это расширение PostgreSQL, соответсвенно весь синтаксис — это SQL-запросы, в которые добавлены новые функции.

Функции PostGis, используемые в проекте, с их кратким описанием:

1. *ST_MakePoint(x, y)*: создает точку с координатами x и y.
2. *ST_GeomFromText(geometry_text)*: создает геометрию из текстового представления.

3. *ST_Intersects(geom1, geom2)*: возвращает true, если геометрии geom1 и geom2 пересекаются.
4. *ST_Within(geom1, geom2)*: возвращает true, если геометрия geom1 полностью находится внутри geom2.
5. *ST_Distance(geom1, geom2)*: возвращает расстояние между двумя геометриями.
6. *ST_Buffer(geom, radius)*: создает буфер (область вокруг геометрии) заданного радиуса.
7. *ST_Union(geom1, geom2)*: возвращает объединение двух геометрий.
8. *ST_Intersection(geom1, geom2)*: возвращает пересечение двух геометрий.

2.3.5 Реализация интерфейса для Elasticsearch

Для работы с Elasticsearch в Golang был выбран драйвер "github.com/elastic/go-elasticsearch/v8". [5].

Особенность Elasticsearch в том, что это noSQL база данных, в которой запросы осуществляются через HTTP RESTful API, а не через SQL-подобный язык, как в случае с PostgreSQL или другими реляционными базами данных. Для отправления запросов в Elasticsearch, используются HTTP-методы (обычно GET или POST) для взаимодействия с индексами и данными.

Синтаксис запросов в Elasticsearch основан на JSON-объектах. Запросы в Elasticsearch могут быть выполнены, как часть URL-адреса с параметрами, так и в теле запроса в формате JSON.

Функции Elasticsearch, используемые в проекте, с их кратким описанием:

1. *Поиск*: выполняется с помощью запросов GET или POST на определенный эндпоинт /index/_search. Вы указываете свой поисковый запрос в теле запроса в формате JSON.
2. *Индексация*: выполняется с помощью запросов PUT, POST или DELETE на определенный эндпоинт /index/_doc/{id}. Вы отправляете

данные в формате JSON в теле запроса, чтобы создать или обновить документ.

3. *Удаление*: выполняется с помощью запросов DELETE на адрес /index/_doc/{id}. Вы указываете идентификатор документа, который хотите удалить.
4. *Match*: это тип запроса, который ищет совпадения в текстовых полях по определенному запросу.
5. *Term*: это тип запроса, который ищет точные совпадения значений полей.
6. *Bool*: это тип запроса, который позволяет объединять несколько условий с использованием операторов must, should и must_not. Это полезно для создания сложных запросов с комбинацией различных условий.
7. *Filter*: это контекст запроса, который позволяет применять фильтры к результатам запроса. Фильтры используются для ограничения набора документов, которые будут возвращены, без влияния на оценку релевантности.
8. *Geo_point*: это тип данных в Elasticsearch, который используется для хранения географических координат (широты и долготы). Он позволяет выполнять географические запросы, такие как поиск ближайших точек и областей.
9. *Geo_distance*: это тип запроса, который позволяет искать документы, находящиеся в заданной географической близости к указанной точке.

2.3.6 Реализация бенчмарков

Для уменьшения количества повторений кода была создана система тестов, которая на вход получает интерфейс хранилища, производит замеры времени для всех методов интерфейса и возвращает документ с результатами замеров. Для тестов использовалось 2 млн точек и 20 тысяч многоугольников.

2.3.7 Результаты замеров

После получения результатов тестов было проведено сравнение характеристик производительности баз данных для точек представлена в Таблице 1.

Таблица 1 — Сравнительная таблица характеристик для точек

Характеристика	PostGis без индекса, ms	Elasticseacrh, ms	PostGis с индексом, ms
Время на удаление	78	235	83
Время на добавление	16641	44854	32699
Время на поиск в радиусе	1.714	1.371	205
Время на поиск в многоугольниках (число углов — время)	3 — 1620 4 — 1505 5 — 1474 6 — 1487 7 — 1494 8 — 1471 9 — 1477 10 — 1477 11 — 1488 12 — 1498 13 — 1490 14 — 1467 15 — 1479 16 — 1477 17 — 1513 18 — 1506 19 — 1477	3 — 116 4 — 51 5 — 35 6 — 30 7 — 35 8 — 37 9 — 43 10 — 32 11 — 34 12 — 37 13 — 42 14 — 44 15 — 31 16 — 41 17 — 50 18 — 51 19 — 39	3 — 331 4 — 235 5 — 224 6 — 163 7 — 237 8 — 398 9 — 348 10 — 283 11 — 212 12 — 196 13 — 231 14 — 226 15 — 254 16 — 219 17 — 289 18 — 1135 19 — 1424
Время на поиск в множестве фигур	1596	85	2756

Сравнительная таблица характеристик производительности баз данных для многоугольников представлена в Таблице 2.

Таблица 2 — Сравнительная таблица характеристик для точек

Характеристика	PostGis без индекса, ms	Elasticseacrh, ms	PostGis с индексом, ms
Время на удаление	14	109	14
Время на добавление	474	195	500
Время на поиск в радиусе	247	3	151
Время на поиск в многоугольнике	39	2	28
Время на поиск пересечений с многоугольном	117	3	98
Время на поиск пересечений с точкой	64	2	14

Повышение скорости работы PostGIS с индексами. Индексы в PostGIS обеспечивают оптимизацию запросов пространственного поиска. По сути, они представляют собой структуры данных, которые позволяют быстро находить объекты внутри определенных геометрических областей (например, объекты внутри радиуса, полигона и т.д.). Когда производится поиск или аналитические операции на пространственных данных, оптимизированный индекс позволяет значительно сократить время выполнения запросов.

Если рассмотреть результаты замеров, можно заметить, что время поиска точек внутри радиуса в PostGIS с использованием индексов значительно меньше, чем время поиска без индексов. Также время поиска точек внутри полигона в PostGIS с индексами также существенно улучшается по сравнению с поиском без индексов.

Это демонстрирует, что использование индексов в PostGIS существенно ускоряет запросы, связанные с пространственными данными, что делает его отличным инструментом для работы с геопространственными структурами.

Преимущества Elasticsearch при работе с простыми структурами. Одной из сильных сторон является специальная индексация, которая позволяет обрабатывать запросы на основе геоинформации в реальном времени.

ES использует инвертированный индекс, который позволяет быстро находить и получать результаты поиска, точки внутри геометрических областей и т.д. Это делает Elasticsearch очень эффективным при работе с простыми структурами данных, такими как точки, линии и полигоны.

В результате использования индексов в Elasticsearch для поиска точек внутри радиуса и точек внутри полигона, мы видим значительное преимущество по скорости по сравнению с PostGIS, даже когда он использует индексы.

Итак, Elasticsearch хорошо подходит для обработки простых структур данных, так как его специализированная индексация обеспечивает быстрый и эффективный поиск и анализ таких данных.

ЗАКЛЮЧЕНИЕ

Исследование и сравнение производительности геометрических запросов между различными базами данных представляет собой важную задачу для определения оптимального инструмента для обработки гео-пространственных данных. В рамках данного проекта были проведены обширные тесты и анализ результатов для оценки возможностей и эффективности различных инструментов.

В процессе сравнения было выявлено, что различные базы данных предоставляют схожие возможности для обработки гео-пространственных данных.

Оптимальный выбор базы данных зависит от требований и характера проекта. Если проект требует сложных запросов и анализа гео-пространственных данных, то специализированные инструменты, такие как PostGIS, могут быть предпочтительны. Однако, если акцент сделан на простых запросах и быстром доступе к данным, то Elasticsearch может предложить достойную альтернативу.

Исследование производительности различных баз данных позволило приобрести ценные знания и опыт в области обработки гео-пространственных данных.

Данное исследование помогло мне в понимании баз данных SQL и NoSQL, а также в Работа с Golang. Сравнение производительности между PostGIS (SQL-ориентированная база данных) и Elasticsearch (NoSQL-ориентированная база данных) дало мне возможность глубже понять различия между двумя типами баз данных, каждый тип баз данных имеет свои преимущества и ограничения в обработке гео-пространственных данных.

Также в ходе проекта было разработано тестовое окружение на языке программирования Golang. Были изучены подходы к написанию эффективного кода. Тестовое окружение можно расширять дальше. Данное сравнение говорит о важности выбора правильного инструмента для выполнения задачи.

Оптимальный выбор базы данных может существенно повлиять на производительность и эффективность проекта в целом.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Документация PostGis [Электронный ресурс] — URL: https://postgis.net/documentation/getting_started/ (дата обращения: 15.07.2023).
2. Документация Elasticsearch [Электронный ресурс] — URL: https://www.elastic.co/guide/index.html (дата обращения: 18.07.2023).
3. Документация rand [Электронный ресурс] — URL: https://pkg.go.dev/math/rand дата обращения: 20.07.2023).
4. Документация драйвера PostGis [Электронный ресурс] — URL: https://pkg.go.dev/github.com/jackc/pgx/v5 (дата обращения: 24.07.2023).
5. Документация драйвера Elasticsearch [Электронный ресурс] — URL: https://pkg.go.dev/github.com/elastic/go-elasticsearch/v8 (дата обращения: 24.07.2023).