МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ

по дисциплине «Введение в нереляционные базы данных»

Тема: Архив погодных наблюдений MongoDB

Студенты гр. 8303 _____ Дирксен А.А.

_____ Лисок М.А.

_____ Сенюшкин Е.В.

Преподаватель _____ Заславский М.М.

Санкт-Петербург

2021

ЗАДАНИЕ

НА ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ

Студенты

Дирксен А.А.

Лисок М.А.

Сенюшкин Е.В.

Группа 8303

Тема работы: Архив погодных наблюдений MongoDB

Исходные данные:

Необходимо реализовать приложение для просмотра, агрегации, фильтрации и построения графиков для погодных наблюдений.

БД – MongoDB

Наборы данных - https://www.metoffice.gov.uk/research/climate/maps-and-data/data/haduk-grid/datasets

Содержание пояснительной записки:

«Введение»

«Качественные требования»

«Сценарии использования»

«Модель данных»

«Разработанное приложение»

«Выводы»

«Приложение»

«Литература»

Предполагаемый объем пояснительной записки:

Не менее 10 страниц.

Дата выдачи задания:

Дата сдачи реферата:

Дата защиты реферата:

Студенты	_____	Дирксен А.А.
	_____	Лисок М.А.
	_____	Сенюшкин Е.В.
Преподаватель	_____	Заславский М.М.

АННОТАЦИЯ

Была поставлена задача реализовать веб приложения для работы с архивом погодных наблюдений с использованием MongoDB. Веб приложение должно предоставлять следующий функционал: просмотр, агрегация, фильтрация и построение графиков для архива погодных наблюдений.

SUMMARY

The task was to implement a web application to work with the archive of weather observations using MongoDB. The web application should provide the following functionality: viewing, aggregation, filtering and graphing for the archive of weather observations.

СОДЕРЖАНИЕ

1. ВВЕДЕНИЕ

Цель работы — создать приложение, которое позволяет пользователю просматривать архив погодных наблюдений.

Было решено разработать web-приложение, позволяющее просматривать графики и таблицы с погодными наблюдениями за выбранный период времени.

2. КАЧЕСТВЕННЫЕ ТРЕБОВАНИЯ К РЕШЕНИЮ

Требуется разработать приложение с использованием MongoDB, позволяющее получить информацию о погодных наблюдениях за данный период времени.

3. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

3.1. Макет UI

1. Макет приложения (рис. 1)

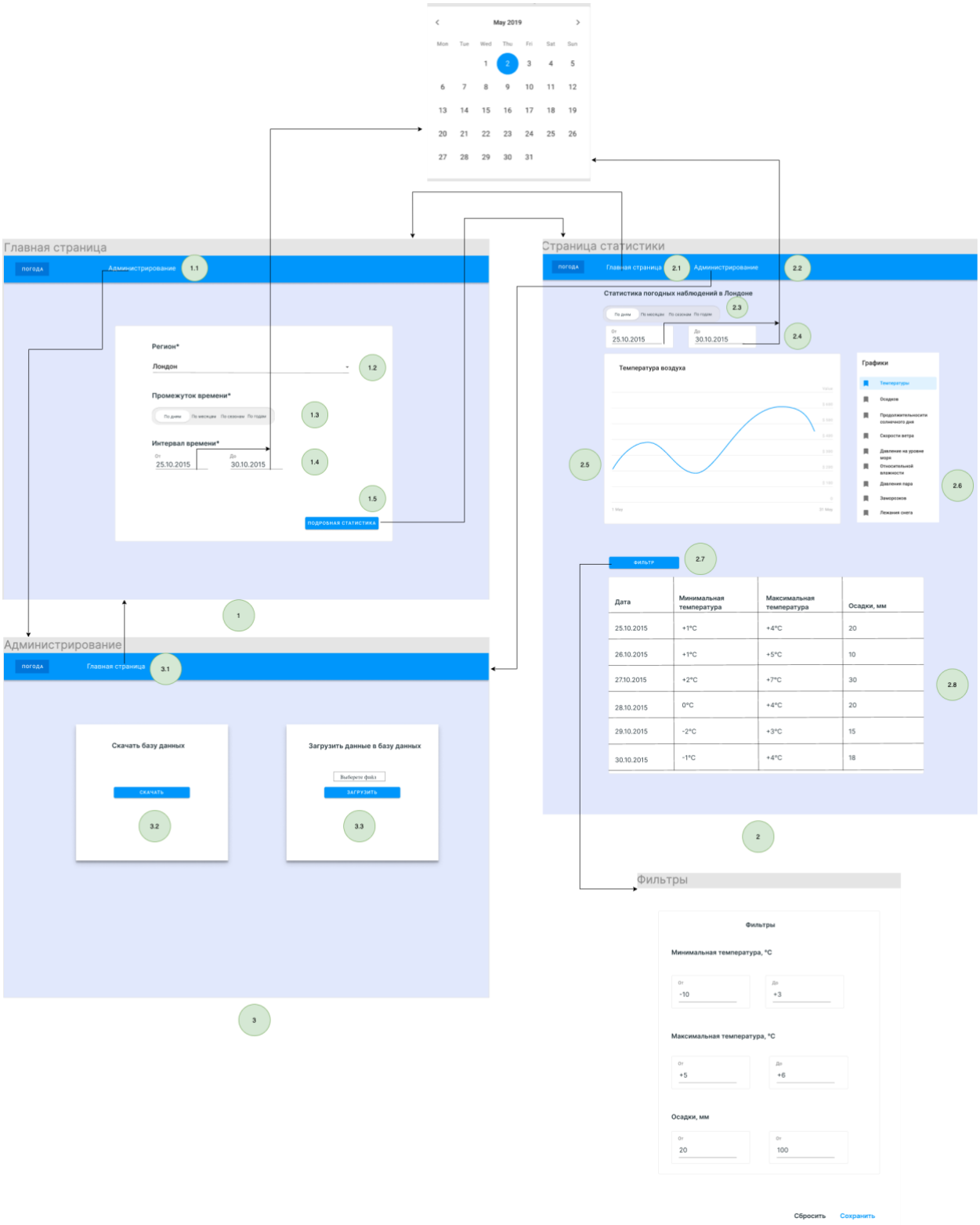

Рисунок 1. Макет основного приложения.

3.2. Сценарии использования.

Сценарий использования - «Просмотр архива погодных наблюдений»:

Действующее лицо: Пользователь

Основной сценарий:

1. Пользователь заходит на главную страницу, отображаются:
   1.1 Кнопка перехода на страницу администрирования;
   1.2 Список всех доступных регионов;
   1.3 Промежуток времени (по дням, по месяцам, по сезонам, по годам), для которого формируются данные погодных наблюдений;
   1.4 Интервал времени (начальный и конечный периоды), для которого формируются данные погодных наблюдений;
   Пользователь выбирает регион (вводит название региона или выбирает из списка), промежуток времени, интервал времени(в случае дней пользователь использует календарь, в случае месяцев список месяцев и годов, в случае сезонов список сезонов и годов, в случае годов список годов), кликает по кнопке "Подробная статистика", переходит на страницу со статистикой.

2. Пользователю отображаются на странице статистики:
   2.1 Кнопка перехода на главную страницу;
   2.2 Кнопка перехода на страницу администрирования;
   2.3 Выбранный промежуток времени (по дням, по месяцам, по сезонам, по годам) для которого формируются данные погодных наблюдений;
   2.4 Выбранный интервал времени (начальный и конечный периоды), для которого формируются данные погодных наблюдений;
   2.5 График погодного наблюдения(по умолчанию – Температура воздуха);
   2.6 Панель переключения графиков погодных наблюдений(в случае выбора промежутка времени "по дням" графики температуры и осадков; в

случае выбора остальных промежутков времени графики температуры, осадков, продолжительности солнечного дня, скорости ветра, давление на уровне моря, относительной влажности, давление пара, заморозков, лежания снега).

2.7 Кнопка фильтрации таблицы;

2.8 Таблица погодных наблюдений за заданный промежуток и интервал времени;

Пользователь нажимает на кнопку "Фильтр", задает нужные интервалы у погодного наблюдения, нажимает кнопку сохранить, видит отфильтрованную таблицу погодных наблюдений.

Пользователь кликает по панели переключения графиков, видит выбранный график.

Пользователь изменяет промежуток времени, интервал времени, видит график и таблицу за указанный промежуток и интервал времени.

3. Пользователь заходит на страницу администрирования, отображаются:

   3.1 Кнопка перехода на главную страницу;

   3.2 Кнопка скачивания базы данных;

   3.3 Кнопка загрузки данных в базу данных;

   Пользователь кликает по кнопке скачивания, видит файл в формате json, со всеми данными, содержащимися в базе данных.

   Пользователь выбирает файл с данными в формате json, нажимает кнопку загрузки, данные добавляются в базу данных.

Альтернативный сценарий:

- Пользователь не выбирает регион, интервал или промежуток времени, видит окно предупреждения "Заполните все поля помеченные *" (страница 1).

- Пользователь выбирает начальный интервал времени больше, чем конечный, видит окно предупреждения "Неправильно заданы интервалы времени" (страница 1 и 2).
- Пользователь выбирает интервал времени, за который нет погодных наблюдений, вместо графика и таблицы видит "За данный интервал времени, данных погодных наблюдений нет"(страница 2).
- Пользователь нажимает на кнопку "Фильтр", задает начальный интервал у погодного наблюдения больше, чем конечный, нажимает кнопку сохранить, видит "Неправильно заданы интервалы погодного наблюдения"(страница 2).

4. МОДЕЛЬ ДАННЫХ

Схема бд

SQL:

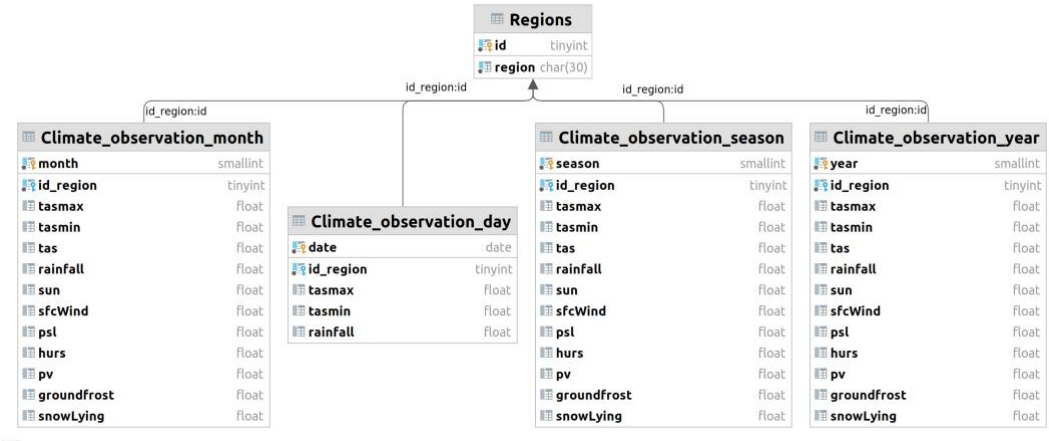

Рисунок 2. SQL дата модель.

MongoDB:

{

_id: <ObjectId>

region: Integer

year?: Integer

month?: Integer

day?: Integer

season?: Integer

tasmax?: Double

tasmin?: Double

rainfall?: Double

tas?: Double

sun?: Double

sfcWind?: Double

psl?: Double

hurs?: Double

pv?: Double

groundfrost?: Double

snowLying?: Double

}

Список сущностей:

SQL:

1. Regions – сущность региона включает в себя следующие атрибуты:
    - id: Tinyint – уникальный идентификатор региона(1 - 16)
    - region: Char(30) – название региона
2. Climate_obseravation_days
    - date: Date – дата погодного наблюдения. Используется как уникальный идентификатор.
    - id_region: Tinyint - поле идентификатора региона
    - tasmax: Float – максимальная температура воздуха, измеренная между 09:00 UTC в день D и 09:00 UTC в день D + 1.
    - tasmin: Float – минимальная температура воздуха, измеренная между 09:00 UTC в день D-1 и 09:00 UTC в день D.
    - rainfall: Float – общее количество осадков, измеренное между 09:00 UTC в день D и 09:00 в день D + 1.

3. Climate_obseravation_month
    - month: Smallint – количество месяцев, начиная с 1800 года. Используется как уникальный идентификатор.
    - id_region: Tinyint - поле идентификатора региона
    - tasmax: Float – средняя суточная максимальная температура воздуха за календарный месяц.
    - tasmin: Float – средняя дневная минимальная температура воздуха за календарный месяц.
    - tas: Float – среднесуточная средняя температура воздуха за календарный месяц.
    - groundfrost: Float – подсчет дней, когда минимальная температура травы ниже 0 за календарный месяц.
    - snowLying: Float – подсчет дней, когда более 50% территории покрыто снегом в 09:00 UTC за календарный месяц.
    - psl: Float – среднее часовое (или трехчасовое) среднее давление на уровне моря за календарный месяц.
    - hurs: Float – средняя почасовая (или трехчасовая) относительная влажность за календарный месяц.
    - pv: Float – среднее почасовое (или трехчасовое) давление пара за календарный месяц.
    - sun: Float – продолжительность яркого солнечного сияния в течение месяца.
    - rainfall: Float – общее количество осадков за календарный месяц.
    - sfcWind: Float – средняя часовая скорость ветра на высоте 10 м над уровнем земли за месяц.
4. Climate_obseravation_season
    - season: Smallint – количество сезонов, начиная с 1800 года. Используется как уникальный идентификатор.
    - id_region: Tinyint - поле идентификатора региона

- tasmax: Float – средняя суточная максимальная температура воздуха за сезон.
- tasmin: Float – средняя дневная минимальная температура воздуха за сезон.
- tas: Float – среднесуточная средняя температура воздуха за сезон.
- groundfrost: Float – подсчет дней, когда минимальная температура травы ниже 0 за сезон.
- snowLying: Float – подсчет дней, когда более 50% территории покрыто снегом в 09:00 UTC за сезон.
- psl: Float – среднее часовое (или трехчасовое) среднее давление на уровне моря за сезон.
- hurs: Float – средняя почасовая (или трехчасовая) относительная влажность за сезон.
- pv: Float – среднее почасовое (или трехчасовое) давление пара за сезон.
- sun: Float – продолжительность яркого солнечного сияния в течение сезона.
- rainfall: Float – общее количество осадков за сезон.
- sfcWind: Float – средняя часовая скорость ветра на высоте 10 м над уровнем земли за сезон.

5. Climate_obseravation_year
    - year: Smallint – номер года. Используется как уникальный идентификатор.
    - id_region: Tinyint - поле идентификатора региона
    - tasmax: Float – средняя суточная максимальная температура воздуха за год.
    - tasmin: Float – средняя дневная минимальная температура воздуха за год.
    - tas: Float – среднесуточная средняя температура воздуха за год.

- groundfrost: Float – подсчет дней, когда минимальная температура травы ниже 0 за год.
- snowLying: Float – подсчет дней, когда более 50% территории покрыто снегом в 09:00 UTC за год.
- psl: Float – среднее часовое (или трехчасовое) среднее давление на уровне моря за год.
- hurs: Float – средняя почасовая (или трехчасовая) относительная влажность за год.
- pv: Float – среднее почасовое (или трехчасовое) давление пара за год.
- sun: Float – продолжительность яркого солнечного сияния в течение года.
- rainfall: Float – общее количество осадков за календарный год.
- sfcWind: Float – средняя часовая скорость ветра на высоте 10 м над уровнем земли за год.

MongoDB:

Каждое погодное наблюдение представляет собой отдельный документ, в котором одно из полей {year | season | month | day} отвечает за период погодных наблюдений. При этом сезон, месяц и день хранятся в виде количества сезонов, месяцев или дней, прошедших с 1800 года. У документа может быть любое подмножество полей из множества {tasmax, tasmin, tas, rainfall, sun, sfcWind,psl, hurs, pv, groundfrost, snowLying}, которое отвечает за конкретные погодные наблюдения.

Оценка объема информации:

"Чистый" объем:

1. Данные о регионах:
   region – 30B

2. Данные о погоде за года
   tasmax – 4B
   tasmin – 4B
   tas – 4B
   rainfall – 4B
   sun – 4B
   sfcWind – 4B
   psl – 4B
   hurs – 4B
   pv – 4B
   groundfrost – 4B
   snowLying – 4B
3. Данные о погоде за месяца
   tasmax – 4B
   tasmin – 4B
   tas – 4B
   rainfall – 4B
   sun – 4B
   sfcWind – 4B
   psl – 4B
   hurs – 4B
   pv – 4B
   groundfrost – 4B
   snowLying – 4B
4. Данные о погоде за сезоны
   tasmax – 4B
   tasmin – 4B
   tas – 4B
   rainfall – 4B
   sun – 4B

sfcWind – 4B

psl – 4B

hurs – 4B

pv – 4B

groundfrost – 4B

snowLying – 4B

5. Данные о погоде за дни

   tasmax – 4B

   tasmin – 4B

   rainfall – 4B

"Чистый" объем:

Тогда "чистый" объем информации, посчитанный на реальных данных, будет равен

(

(2020 - 1884) * (tasmax + tasmin + tas) + (2020 - 1862) * rainfall + (2020 - 1929) * sun +

(2020 - 1969) * sfcWind + (2020 - 1961) * (psl + hurs + pv + groundfrost) + (2020 - 1971) * snowLying +

12 * ((2020 - 1884) * (tasmax + tasmin + tas) + (2020 - 1862) * rainfall + (2020 - 1929) * sun + (2020 - 1969) *

sfcWind + (2020 - 1961) * (psl + hurs + pv + groundfrost) + (2020 - 1971) * snowLying) +

4 * ((2020 - 1884) * (tasmax + tasmin + tas) + (2020 - 1862) * rainfall + (2020 - 1929) * sun + (2020 - 1969) *

sfcWind + (2020 - 1961) * (psl + hurs + pv + groundfrost) + (2020 - 1971) * snowLying) +

(12 * 365 * (2020 - 1960) + 16) * (tasmax + tasmin) + (12 * 365 * (2020 - 1891) + 32) * rainfall

) * 16 = 4 426 248 * 16 = 70 819 968 B

Тогда "чистый" объем информации, посчитанный в общем виде

X * (tasmax + tasmin + tas + psl + hurs + pv + groundfrost + snowLying + sun + sfcWind + rainfall) +

21.5 * X * (tasmax + tasmin + rainfall)

= X * (22.5 * tasmax + 22.5 * tasmin + tas + psl + hurs + pv + groundfrost + snowLying + sun + sfcWind + 22.5 * rainfall)

= 302 * X

Где

- X – количество записей.

Mongo

MongoDB (фактический объем):

Фактический объем информации, посчитанный на реальный данных, будет равен

(

(2020 - 1862) * (__id + region + year) +

(2020 - 1884) * (tasmax + tasmin + tas) + (2020 - 1862) * rainfall + (2020 - 1929) * sun +

(2020 - 1969) * sfcWind + (2020 - 1961) * (psl + hurs + pv + groundfrost) + (2020 - 1971) * snowLying +

12 *

(

(2020 - 1862) * (__id + region + month) +

(2020 - 1884) * (tasmax + tasmin + tas) + (2020 - 1862) * rainfall + (2020 - 1929) * sun + (2020 - 1969) *

sfcWind + (2020 - 1961) * (psl + hurs + pv + groundfrost) + (2020 - 1971) * snowLying
) +

4 *
(
(2020 - 1862) * (__id + region + season) +
(2020 - 1884) * (tasmax + tasmin + tas) + (2020 - 1862) * rainfall + (2020 - 1929) * sun + (2020 - 1969) *
sfcWind + (2020 - 1961) * (psl + hurs + pv + groundfrost) + (2020 - 1971) * snowLying
) +

(__id + region + day) * ((2020 - 1862) * 12 * 365 + 32) +
(12 * 365 * (2020 - 1960) + 16) * (tasmax + tasmin) + (12 * 365 * (2020 - 1891) + 32) * rainfall
)
* 16 = (11104 + 133248 + 44416 + 18204176) * 16 = 18 392 944 * 16 = 294 287 104 B
Фактический объем в mongoDB 294 287 104 B
Фактический объем в mongoDB в общем виде
X * (__id + region + {year | season | month | day} + tasmax + tasmin + tas + psl + hurs + pv + groundfrost + snowLying + sun + sfcWind + rainfall) +
21.5 * X * (__id + region + {year | season | month | day} + tasmax + tasmin + rainfall) =
X * (22.5 * _id + 22.5 * region + 22.5 * {year | season | month | day} + 22.5 * tasmax + 22.5 * tasmin + 22.5 * rainfall + tas + psl + hurs + pv + groundfrost + snowLying + sun + sfcWind)
= 1054 * X
Где

- X – количество записей.

Избыточность модели данных: 294 287 104 B / 70 819 968 B = 4.1554255432592119

Избыточность модели данных в общем виде: 1054 * X / 302 * X = 3,490066225

Направление роста модели

- По годам

  Climate_Observation_year = Y * (_id + tasmax + tasmin + tas + rainfall + sun + sfcWind + psl + hurs + pv + groundfrost + snowLying), где Y – количество годов
- По месяцам

  Climate_Observation_month = M * (_id + tasmax + tasmin + tas + rainfall + sun + sfcWind + psl + hurs + pv + groundfrost + snowLying), где M – количество месяцев
- По сезонам

  Climate_Observation_season = S * (_id + tasmax + tasmin + tas + rainfall + sun + sfcWind + psl + hurs + pv + groundfrost + snowLying), где S – количество сезонов
- По дням

  Climate_Observation_day = D * (tasmax + tasmin + rainfall), где D – количество дней
- По регионам

  N * (Climate_Observation_year + Climate_Observation_month + Climate_Observation_season + Climate_Observation_day)

  Где
- N – количество регионов,
- Climate_Observation_year – климатические наблюдения за года,

- Climate_Observation_month – климатические наблюдения за месяца,
- Climate_Observation_season – климатические наблюдения за сезоны,
- Climate_Observation_day – климатические наблюдения за дни

SQL

SQL (фактический объем):

Фактический объем информации будет равен

(

((2020 - 1862) * (year + id_region) +

(2020 - 1884) * (tasmax + tasmin + tas) + (2020 - 1862) * rainfall + (2020 - 1929) * sun +

(2020 - 1969) * sfcWind + (2020 - 1961) * (psl + hurs + pv + groundfrost) +

(2020 - 1971) * snowLying) +

12 * (

(2020 - 1862) * (month + id_region) +

((2020 - 1884) * (tasmax + tasmin + tas) + (2020 - 1862) * rainfall + (2020 - 1929) * sun +

(2020 - 1969) * sfcWind + (2020 - 1961) * (psl + hurs + pv + groundfrost) +

(2020 - 1971) * snowLying)

) +

4 * (

(2020 - 1862) * (season + id_region) +

((2020 - 1884) * (tasmax + tasmin + tas) + (2020 - 1862) * rainfall + (2020 - 1929) * sun +

(2020 - 1969) * sfcWind + (2020 - 1961) * (psl + hurs + pv + groundfrost) +

(2020 - 1971) * snowLying)

) +

(((2020 - 1862) * 12 * 365 + 32) * (date + id_region) + ((2020 - 1960) * 12 * 365 + 16) * (tasmax + tasmin) +
((2020 - 1891) * 12 * 365 + 32) * rainfall)

) * 16 = (4446 + 53352 + 17784 + 6438952) * 16 = 104 232 544 B

Фактический объем информации: 104 232 544 B

Фактический объем информации в общем виде:
X * (id_region + {year | season | month | day} + tasmax + tasmin + tas + psl + hurs + pv + groundfrost + snowLying + sun + sfcWind + rainfall) +
21.5 * (id_region + {year | season | month | day} + tasmax + tasmin + rainfall) =
X * (22.5 * id_region + 22.5 * {year | season | month | day} + 22.5 * tasmax + 22.5 * tasmin + tas + psl + hurs + pv + groundfrost + snowLying + sun + sfcWind + 22.5 * rainfall)
= 414,5 * X
Где

- X – количество записей.
-

Избыточность модели данных: 104 232 544 B / 70 819 968 B = 1.4717959770893994

Избыточность модели данных в общем виде: 414,5 * X / 302 * X = 1,372516556

Направление роста модели

- По годам

Climate_observation_year = Y * (year + id_region + tasmax + tasmin + tas + rainfall + sun + sfcWind + psl + hurs + pv + groundfrost + snowLying), где Y – количество годов

- По месяцам

  Climate_observation_month = M * (month + id_region + tasmax + tasmin + tas + rainfall + sun + sfcWind + psl + hurs + pv + groundfrost + snowLying), где M – количество месяцев
- По сезонам

  Climate_observation_season = S * (season + id_region + tasmax + tasmin + tas + rainfall + sun + sfcWind + psl + hurs + pv + groundfrost + snowLying), где S – количество сезонов
- По дням

  Climate_observation_day = D * (date + id_region + tasmax + tasmin + rainfall), где D – количество дней
- По регионам

  N * (Climate_observation_year + Climate_observation_month + Climate_observation_season + Climate_observation_day)

  Где
- N – количество регионов,
- Climate_observation_year – климатические наблюдения за года,
- Climate_observation_month – климатические наблюдения за месяца,
- Climate_observation_season – климатические наблюдения за сезоны,
- Climate_observation_day – климатические наблюдения за дни

Запросы:

SQL

Получение списка регионов

```
select region
    from Regions;
```

Пример получения данных погодных наблюдений в период с 2017-03-14 по 2017-03-18 для Лондона.

```
select date, tasmax, tasmin, rainfall
    from Climate_observation_day as observ
    where date >= '2017-03-14'
            && date <= '2017-03-18'
            && id_region = (select DISTINCT Regions.id
                    from Regions
                    where Regions.region = 'London');
```

Пример фильтрации данных погодных наблюдений по максимальной температуре(tasmax < 12.3), по минимальной температуре (tasmin > 0.4) и продолжительности сияния солнца(sun >= 300) в период с 24200 до 24220 месяца, начиная с 0 года(с августа 2016 года по апрель 2018). Для Восточной Шотландии.

```
select region, month, tasmax, tasmin, tas, rainfall, sun, sfcWind, psl, hurs, pv,
groundfrost, snowLying
    from Regions, Climate_observation_month as observ
    where Regions.id = 3 && id_region = 3
            && month >= 24200
            && month <= 24220
            && tasmax < 12.3
            && tasmin > 0.4
            && sun >= 300;
```

Из-за того, что в sql можно проиндексировать уникальные поля, запрос на извлечения данных за временной промежуток упирается только в извлечение данных с жесткого диска. Если нужно отфильтровать данные по определенным значениям, то запрос будет работать O(n), где n длинна

временного промежутка, потому что каждую запись нужно будет проверить на соответствие предикату.

MongoDB

Пример получения обобщеных данных для каждого региона за 2017 год.

db.weathers.find({year: 2017})

Пример получения обобщеных данных для каждого региона за все доступные года.
db.weathers.find({month: null, season: null})

Пример получения данных погодных наблюдений в период с 2017-03-14 по 2017-03-18 для Лондона.

db.weathers.find({day: {$gte: 79330, $lt: 79334}, region: 4})

Пример фильтрации данных погодных наблюдений по максимальной температуре(tasmax < 12.3), по минимальной температуре (tasmin > 0.4) и продолжительности сияния солнца(sun >= 30) в период с 2600 до 2620 месяца, начиная с 1800 года(с августа 2016 года по апрель 2018). Для Восточной Шотландии.

db.weathers.find({region: 3, tasmax: {$lt: 12.3}, tasmin: {$gt: 0.4}, sun: {$gte: 30}, month: {$gte: 2600, $lt: 2620}}).count()

Благодаря тому, что mongodb позволяет индексировать только те записи, в которых поле действительно есть, мы можем сделать 4 индекса(year, season, month или day) и значительно ускорить выполнение запросов.

Таким образом, скорость выполнения запросов будет примерно такая же, как и в SQL.

Пример хранения данных

MongoDB

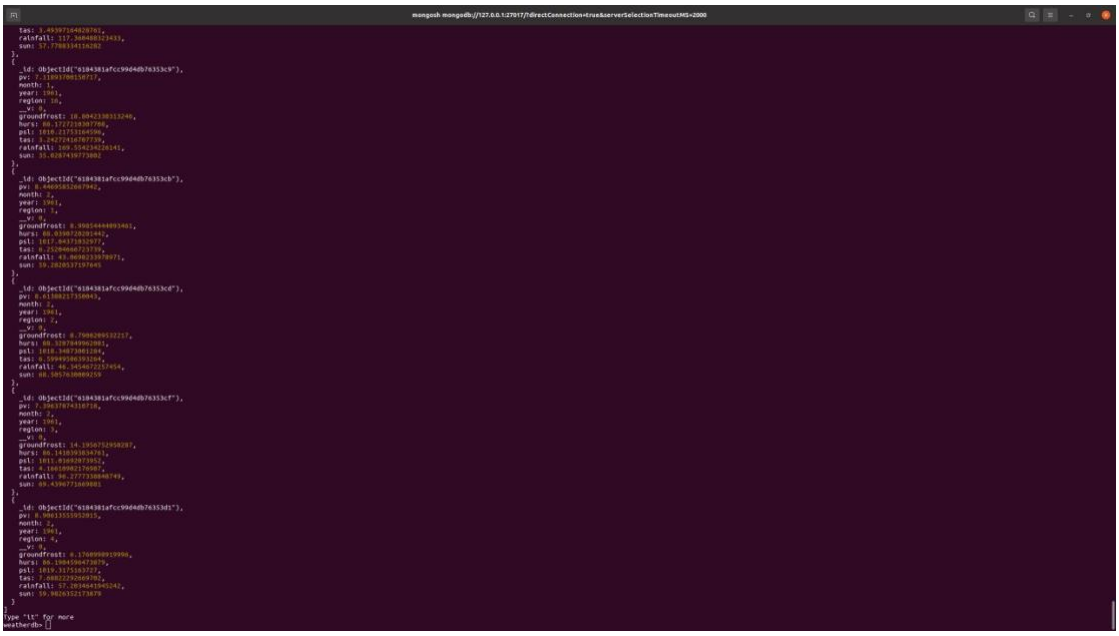

Рисунок 3 – Пример хранения данных в MongoDB

SQL

	month	id_region	tasmax	tasmin	tas	rainfall	sun	sfcWind	psl	hurs	pv	groundfrost	snowLying
1	24207	2	17.5888	11.3674	9.67849	194.395	317.132	6.4807	600.99	5.27337	9.11783	246.915	70.3387
2	24208	3	5.74193	16.2221	19.1035	394.835	948.157	4.68964	187.658	66.2759	1.41401	691.432	29.9931
3	24209	4	1.75836	13.9355	8.6174	14.8201	6.10522	0.880872	1313.1	44.6858	11.1413	430.795	25.4304
4	24210	3	4.42412	11.8985	13.4961	34.7703	344.454	1.73755	614.28	30.9224	0.696749	168.991	70.1159
5	24211	3	12.502	7.87723	2.99376	13.6489	528.802	4.77583	1242.96	59.7455	0.602106	1702.27	77.568
6	24212	4	8.20006	1.74595	13.0402	60.708	255.139	2.01955	822.885	38.0893	5.83305	1846.32	3.34554
7	24213	3	1.56937	6.35747	13.5096	48.3914	587.851	7.77789	387.855	57.4606	11.8754	1199.03	13.2264
8	24214	2	11.1745	0.0229351	9.43499	90.8872	287.478	1.30499	1456.79	48.2307	7.51302	1340.82	21.7137
9	24215	4	2.93758	10.1171	16.0826	23.5815	524.013	1.78821	597.393	56.9447	4.35177	885.877	85.9924

Рисунок 4 – Пример хранения данных в SQL

Вывод

Если сравнивать mongoDB и SQL, то скорость поиска и извлечения данных будет примерно одинаковая (O(n)), такая в обеих базах данных данные проиндексированы. Для нашего датасета было бы удобнее пользоваться mongoDB, потому что многие параметры погодных наблюдений в датасете присутствуют не для всех периодов, соответственно SQL заполняет такие поля null, тратит на это память и такие поля впоследствии придется дополнительно обрабатывать на бэкенде. Но в mongoDB невозможно точно настроить типы данных для

полей коллекции, таким образом mongoDB занимает больше памяти(294 287 104 B против 104 232 544 B в SQL).

5. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

5.1 Краткое описание.

Было реализовано веб приложения для работы с архивом погодных наблюдений с использованием MongoDB. Веб приложение предоставляет следующий функционал: просмотр, агрегация, фильтрация и построение графиков для архива погодных наблюдений.

Бэкенд реализован на NestJs, фронтентд на React.

Для импорта новых погодных наблюдений в БД нужно передать json, который будет иметь определенный набор полей.

Во время экспорта данных пользователь получает json файл.

5.2 Использованные технологии.

БД: MongoDB

Бэк: NestJs

Фронт: React

5.3 Ссылки на приложение.

https://github.com/moevm/nosql2h21-weather-mongo

5.4 Схема экранов приложения

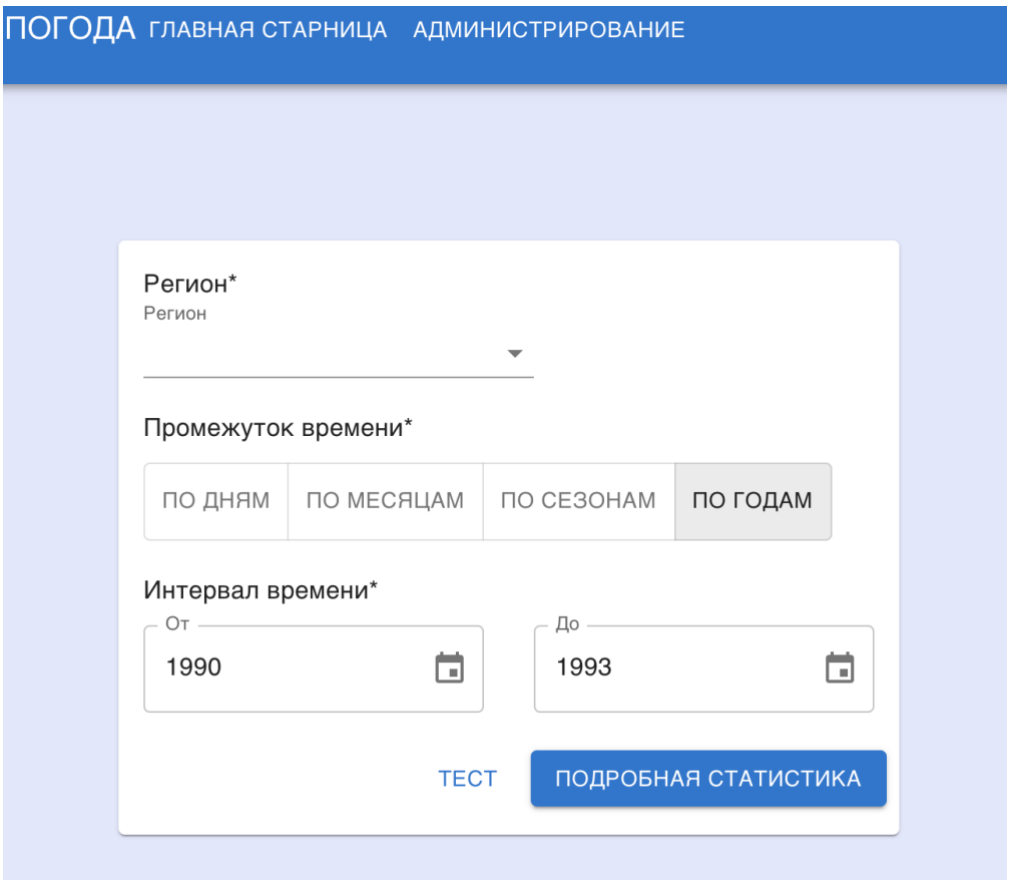

Рисунок 5 – Главная страница.

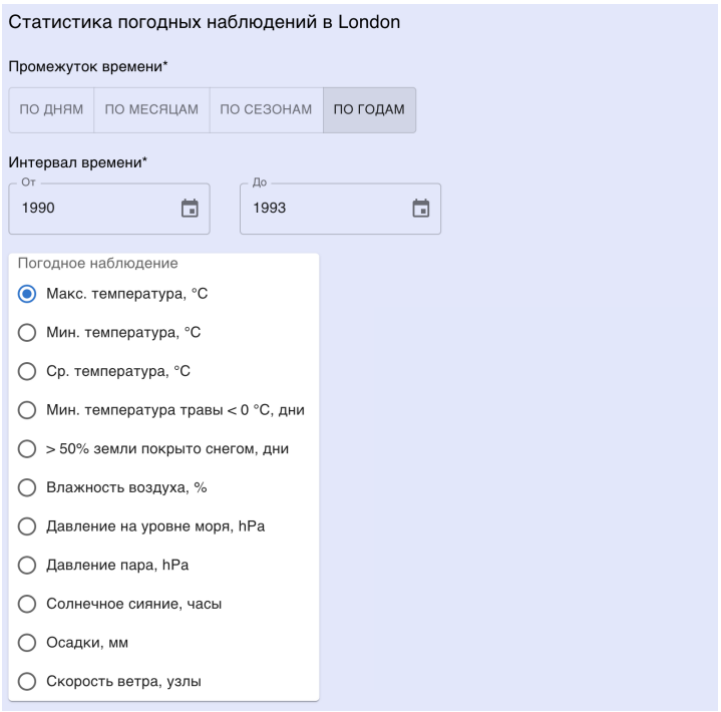

Рисунок 6 – Страница статистики.

Рисунок 7 – Страница статистики.

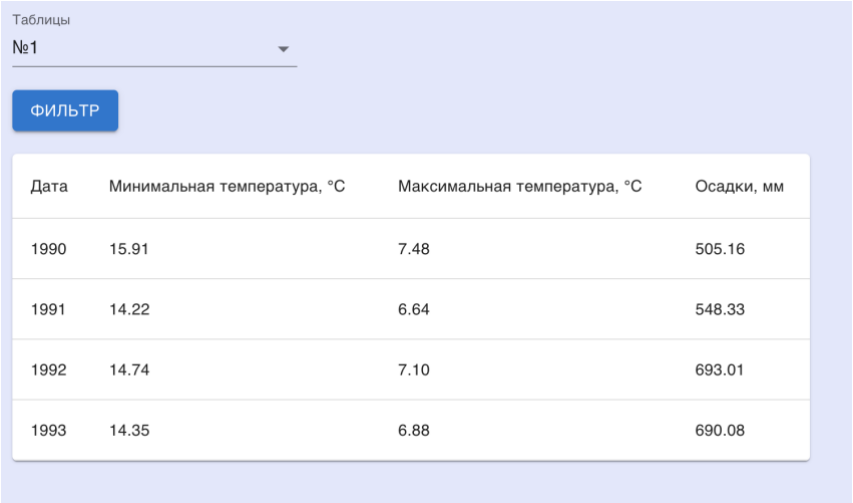

Рисунок 8 – Страница статистики.

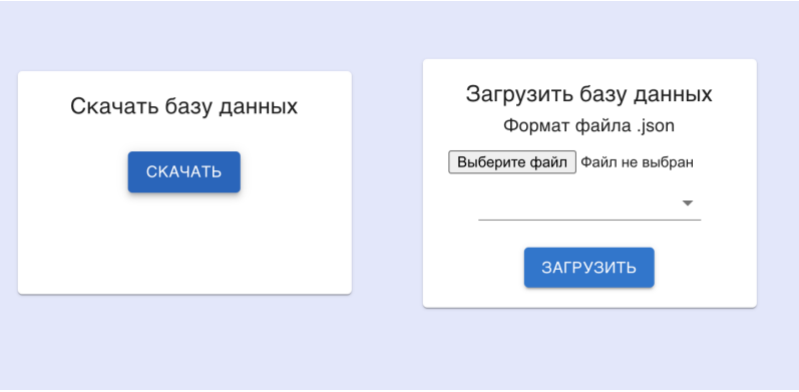

Рисунок 9 – Страница администрирования.

6. ВЫВОДЫ

6.1. Достигнутые результаты

В ходе выполнения работы было разработано приложение для просмотра архива погодных наблюдений в виде графиков и таблиц за заданный период времени. В качестве НСУБД используется MongoDB, для данной задачи было проведено сравнение нереляционной и реляционной модели.

6.2. Недостатки и пути для улучшения полученного решения

Недостатком разработанного приложения является то, что БД хранит данные, не позволяющие работать с картой.

Для решение данной проблемы можно использовать более мощный сервер.

6.3. Будущее развитие решения

В дальнейшем может быть реализована версия для мобильных платформ, также может быть добавлена возможность просматривать карту с погодными условиями, импортировать и экспортировать данные в разных форматах(на данный момент только json), работа с данными разных стран (на данный момент только Великобритания).

7. ПРИЛОЖЕНИЕ

a. Документация по сборке и развертыванию.

   Склонировать репозиторий,

   Перейдите в папку проекта

   Напишите в командой строке

      docker-compose build --no-cache

      docker-compose up

   Если на вашей машине не установлен docker или docker-compose установите их по ссылке.

b. Инструкция для пользователя

   Перейдите по ссылке http://localhost:8080

8. ЛИТЕРАТУРА

1. Документация MongodDB: https://docs.mongodb.com/
2. Документация MongooseJS: https://mongoosejs.com/docs/