МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра математической кибернетики и компьютерных наук

ПРИМЕНЕНИЕ OLAP-ТЕХНОЛОГИИ ПРИ РАЗРАБОТКЕ АНАЛИТИЧЕСКОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 5 курса 551 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Багрова Ильи Юрьевича

Научный руководитель
старший преподаватель _____ М. И. Сафрончик

Заведующий кафедрой
к. ф.-м. н. доцент _____ С. В. Миронов

Саратов 2022

ВВЕДЕНИЕ

В настоящее время бизнесу постоянно требуется производить анализ данных, получаемых в ходе его деятельности. Поскольку приходится обрабатывать значительные объемы данных, возникает необходимость вынесения аналитической информации из оперативной базы данных.

Целью настоящей выпускной квалификационной работы является разработка веб-сервиса для анализа данных с применением технологий оперативной аналитической обработки данных (OLAP). Для достижения данной цели поставлены следующие задачи:

- рассмотреть концепцию OLAP;
- изучить основы языка MDX;
- выбрать датасет для анализа содержащейся в нем информации;
- спроектировать хранилище данных для этой информации;
- построить ETL-процесс для переноса данных из датасета в хранилище;
- построить OLAP-куб;
- разработать аналитические запросы к полученному кубу;
- разработать веб-интерфейс для отображения результатов выполнения этих запросов.

Для разработки приложения были использованы следующие инструменты:

- Microsoft SQL Server Developer Edition — сервер баз данных;
- SQL Server Integration Services — службы, позволяющие организовать процесс переноса данных;
- Pentaho Mondrian — OLAP-клиент для Java;
- Spring Boot — Java фреймворк для разработки веб-приложений;
- React — библиотека для построения пользовательских интерфейсов.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

Первый раздел «Обзор концепции хранения и анализа больших объемов данных» посвящен описанию основных концепций хранения и анализа больших объемов данных.

В подразделе 1.1 приведено описание многомерной модели данных [1].

Подраздел 1.2 посвящен архитектуре OLAP-систем.

В подразделе 1.2.1 описывается реляционный OLAP.

В подразделе 1.2.1 описывается многомерный OLAP, дается 12 основных принципов определяющих многомерную OLAP.

В подразделе 1.3 посвящен обзору языка MDX для доступа к многомерным структурам данных.

Второй раздел «Реализация хранилища данных» посвящен описанию реализации хранилища данных.

В подразделе 2.1 описывается проектирование хранилища данных на основе полей датасета содержащем информацию о поездках такси города Нью-Йорка.

Датасет представляет собой набор файлов формата `.parquet` и содержит следующую информацию о поездках желтых и зеленых такси:

- `Tpep_pickup_datetime` — время и дата, когда включился таксометр;
- `Tpep_dropoff_datetime` — время и дата, когда выключился таксометр;
- `Passenger_count` — количество пассажиров (вводится водителем);
- `Trip_distance` — длина поездки в милях согласно показаниям таксометра;
- `PULocationID` — ID зоны, в которой включился таксометр;
- `DOLocationID` — ID зоны, в которой выключился таксометр;
- `RateCodeID` — код вида поездки;
  - `1` — стандартный тариф;
  - `2` — поездка в аэропорт JFK;
  - `3` — поездка в Ньюарк;
  - `4` — Нассо или Уэстчестер;
  - `5` — договорная цена;
  - `6` — групповая поездка;
- `Payment_type` — код, обозначающий способ оплаты поездки:
  - `1` — кредитная карта;

  - 2 — оплата наличными;
  - 3 — бесплатно;
  - 4 — спорная ситуация;
  - 5 — неизвестно;
  - 6 — аннулированная поездка;
- Fare_amount — стоимость поездки, рассчитанная таксометром;
- Extra — различные надбавки к стоимости поездки. Включает в себя надбавки в 0.5$ и 1$ в час пик и ночные надбавки;
- MTA_tax — 0.5$ налога для MTA, которая автоматически срабатывает на основе показаний счетчика;
- Improvement_surcharge — минимальная стоимость поездки в 0.3$, взымаемая с 2015 года;
- Tip_amount — чаевые, заполняются только для поездок с оплатой по кредитной карте. Чаевые, выданные наличными, не учитываются;
- Tolls_amount — сумма всех пошлин в поездке;
- Total_amount — итоговая сумма поездки, взятая с пассажиров. Не учитывает наличные чаевые;
- Trip_type — тип поездки: было ли такси поймано на улице или вызвано через оператора.

Подраздел 2.2 посвящен разработке ETL процесса для загрузки данных в хранилище [2].

Процесс загрузки данных в хранилище представлен тремя основными шагами:

1. Извлечение данных из внешних источников.
2. Трансформация данных.
3. Непосредственная загрузка преобразованных данных в хранилище.

Третий раздел «Разработка аналитического Web-приложения» посвящен описанию разработки Web-приложения, позволяющего визуализировать полученную из хранилища информацию.

В подразделе 3.1 описывается разработка серверной части. Было разработано Web-приложение, позволяющее визуализировать полученную из хранилища информацию. Данное приложение является клиент-серверным: в роли клиента выступает браузер; сервер реализован при помощи фреймворка Spring [3].

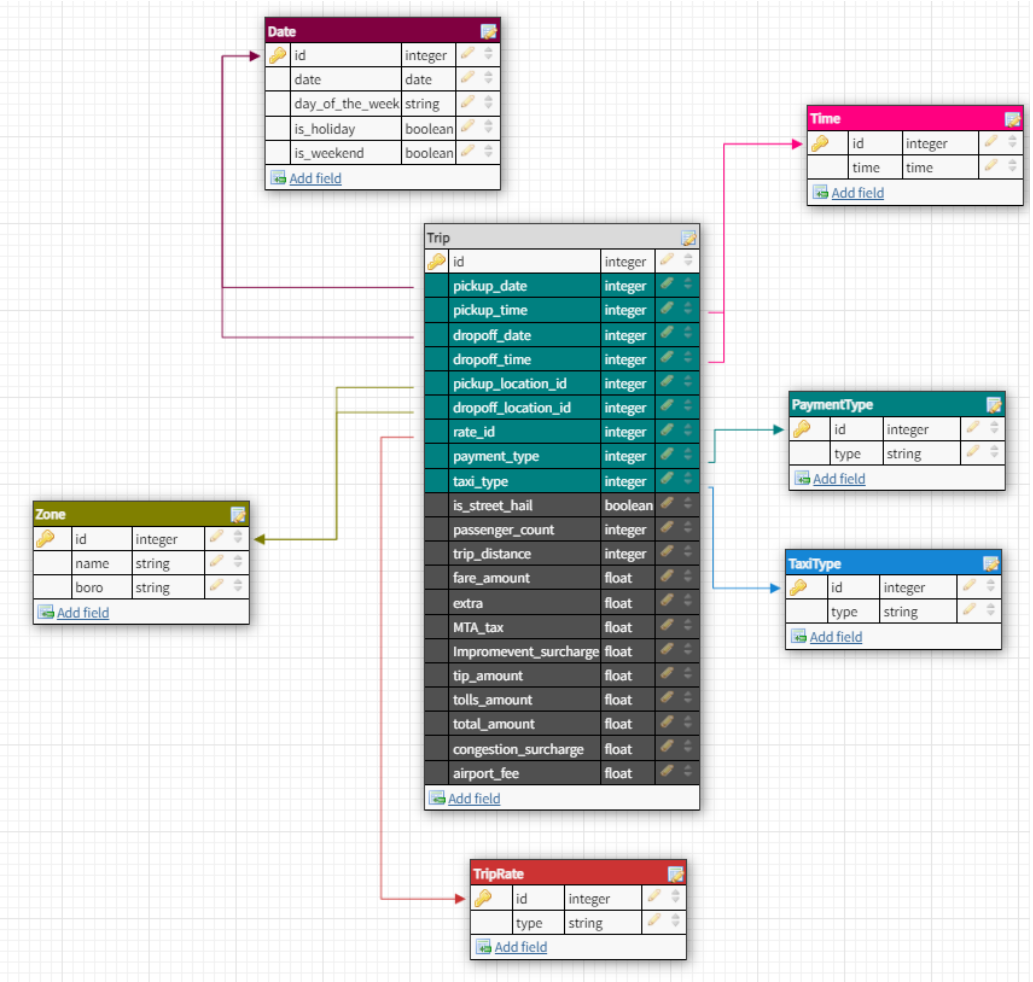

Рисунок 1 – Схема хранилища данных

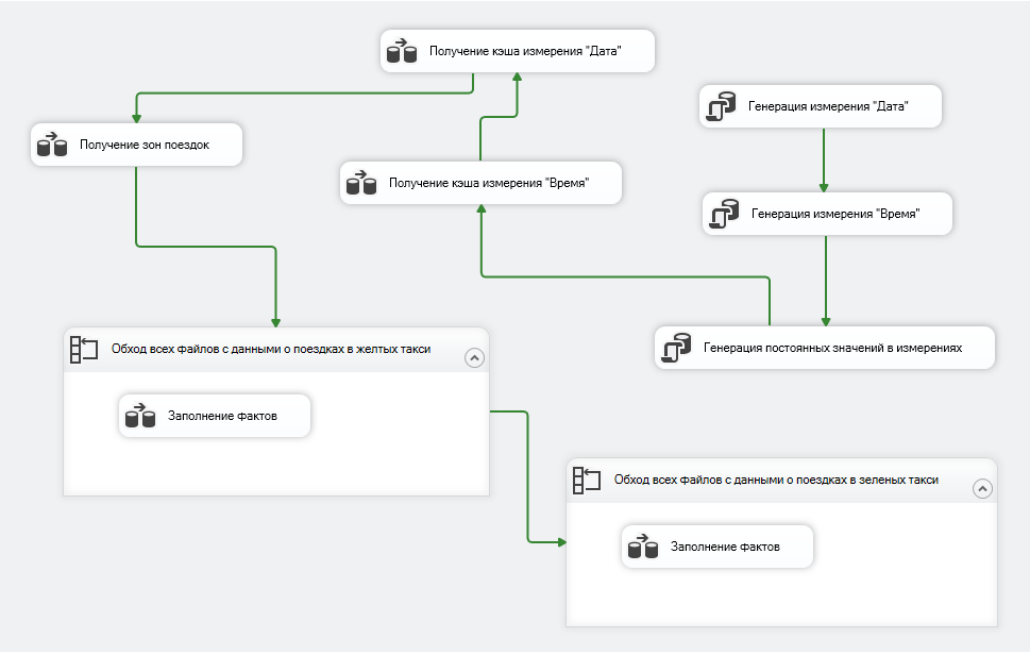

Рисунок 2 – Поток управления ETL процесса для загрузки данных датасета в хранилище

В подразделе 3.1.1 описание модели OLAP-куба. Для построения OLAP-куба и выполнения запросов к нему была использована библиотека Mondrian. Данная библиотека позволяет описать куб при помощи XML-файла, который называется схемой куба. Основными элементами схемы являются элементы `<Cube/>`, `<Dimension/>`, `<Hierarchy/>`, `<Level/>` и `<Measure/>`, описывающие соответственно куб, измерение, иерархию, уровень и меру. Так, для разработанного хранилища данных была написана схема приведенная в приложении.

Построенный OLAP-куб имеет 4 измерения `<Dimension/>`:

— `<Pickup Location/>` — район из которого началась поездка;
— `<Pickup Date/>` — дата начала поездки;
— `<Pickup Time/>` — время начала поездки;
— `<Dropoff Location/>` — район окончания поездки;
— `<Payment Type/>` — тип оплаты.

В подразделе 3.2.1 представлена разработка API для получения данных из куба.

Для выполнения запросов к кубу из браузера было разработано API, конечные точки которого приведены в таблице 1.

Таблица 1 – Конечные точки API для доступа к кубу

URL	**Описание**
`/date/year`	Возвращает список годов
`/date/holidays`	Возвращает для заданного месяца в заданном году список праздничных и выходных дней.
`/stat/average-cost`	Возвращает среднюю стоимость одной поездки
`/stat/total-cost`	Возвращает общую стоимость всех поездок за все время
`/stat/payment-type/distribution`	Возвращает доли количества поездок по способам оплаты
`/stat/location/start-distribution`	Возвращает доли районов, в которых начинались поездки

`/trip/count`	Возвращает для заданной даты количество поездок в эту дату. Если дата не задана, возвращается количество поездок для каждого года. Если задан только год, возвращается количество поездок для каждого месяца этого года. Если заданы год и месяц, возвращается количество поездок для каждого дня этого месяца. Если заданы год, месяц и день, то возвращается количество поездок по часам в этом дне.
`/trip/count/total`	Возвращает количество всех поездок за все время
`/trip/count/zone/pickup`	Возвращает распределение количества поездок по районам. Каждой зоне города ставится в соответствие число от 0 до 1, показывающее, насколько количество поездок в этой части города близко к максимуму.

Для получения данных из куба, при помощи Mondrian к нему отправляются запросы на языке MDX — языке доступа к многомерным базам данных.

Так, для получения количества поездок за все время был написан следующий запрос:

```
1 SELECT [Measures].[Trip Count] ON COLUMNS FROM [TaxiTrips];
```

Данный запрос возвращает единственную ячейку, в которой содержится значение меры `[Measures].[Trip Count]`, взятое по всему кубу.

Для получения долей поездок, приходящихся на способы оплаты, был написан следующий запрос:

```
WITH MEMBER Measures.[Total Trip Count] AS
 100 * SUM([Payment Type].[Type].CurrentMember,
 Measures.[Trip Count]) / SUM([Payment Type].[Type].members,
 Measures.[Trip Count])
SELECT Measures.[Total Trip Count] ON COLUMNS,
[Payment Type].[Type].members ON ROWS
FROM [TaxiTrips];
```

Данный запрос получает искомые проценты путем деления количества поездок с каждым способом оплаты на общее количество поездок.

Метод, обрабатывающий данный запрос, выглядит следующим образом:

```
@GetMapping("stat/payment-type/distribution")
public TableResult getPaymentTypeDistribution() throws SQLException {
    final String query = "WITH MEMBER Measures.[Total Trip Count] AS 100 *
    ↪  SUM([Payment Type].[Type]." +
            "CurrentMember, Measures.[Trip Count]) / SUM([Payment
            ↪  Type].[Type].members, Measures.[Trip Count]) " +
            "SELECT Measures.[Total Trip Count] ON COLUMNS," +
            "[Payment Type].[Type].members ON ROWS FROM [TaxiTrips]";

    // Объект запроса
    OlapStatement statement = DBConfig.connection.createStatement();
    // Исполнение запроса
    CellSet cs = statement.executeOlapQuery(query);
    // Возврат обертки над результатом запроса
    return new TableResult(cs);
}
```

Подраздел 3.2 посвящен разработке клиентской части приложения.

Для разработки Web-интерфейса была использована библиотека React [4]. Для визуализации полученных с сервера данных была использована библиотека Chart.js. Данная библиотека предоставляет набор компонентов для отрисовки различных графиков. Для получения данных с сервера используется модуль Axios.

Вид разработанного приложения представлен на рисунке 3.

На данном рисунке представлены агрегированные данные о всех поездках — общее количество поездок, средняя стоимость поездки, а также суммарная стоимость всех поездок. Под этими данными отрисован график количества

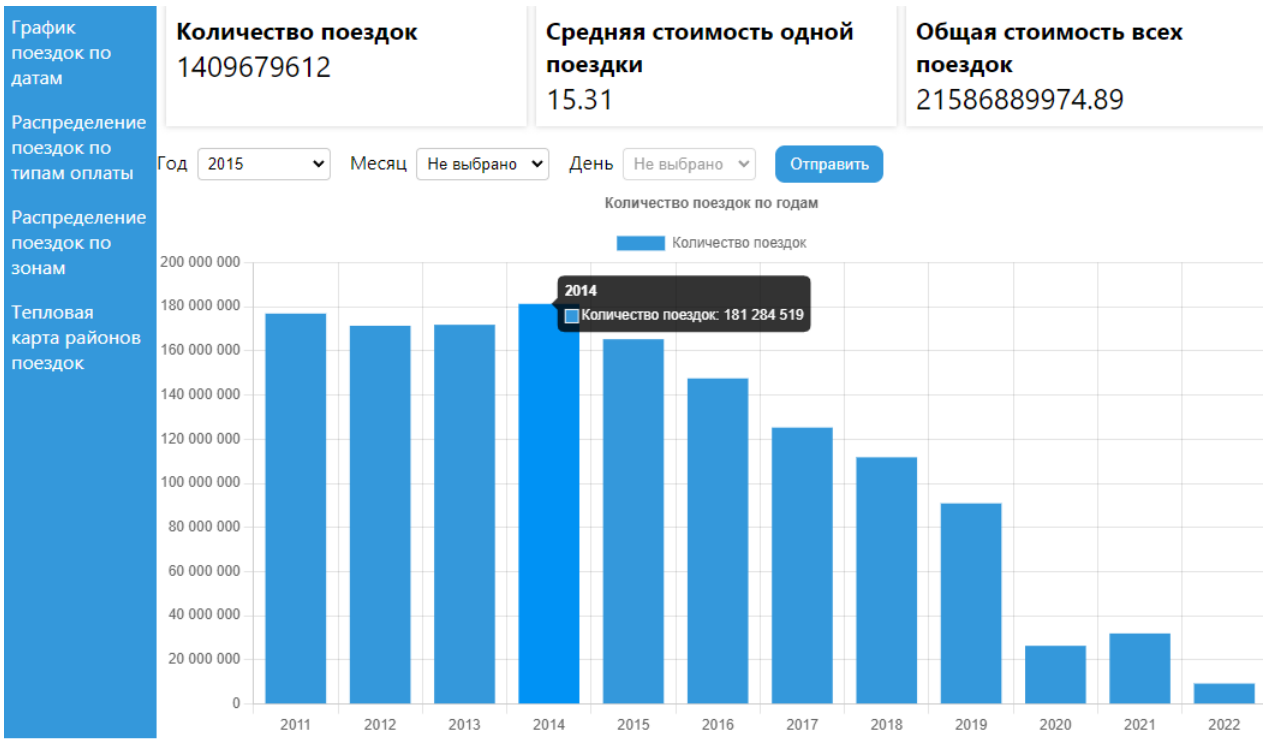

Рисунок 3 – Интерфейс разработанного приложения

поездок по годам.

На карте, приведенной на рисунке, показаны районы, которые выделены разными цветами в зависимости от количества начинающихся в них поездок. Районы, в которых начинается меньше всего поездок, обозначаются зеленым цветом. Чем больше поездок, тем более красным цветом выделен район.

Для отрисовки данной карты была использована JavaScript-библиотека `@pbe/react-yandex-maps`, представляющая собой React-обертку над API сервиса «Яндекс Карты» [5]. Яндекс Карты позволяют отрисовывать объекты, заданные в формате GeoJSON. Однако, исходные данные для отрисовки частей города Нью-Йорк, представленные на сайте с датасетом, были представлены в формате Shapefile. Для преобразования данных из Shapefile в GeoJSON была использована программа QGIS версии 3.22.7. Данное приложение позволяет сконвертировать географические данные в различные форматы, изменив систему координат. Система координат, поддерживаемая Яндекс Картами — WGS 84.

Далее, для преобразования полученных из куба чисел от 0 до 1 в цвет от зеленого до красного были написаны две следующие функции:

```
// Преобразует цвет из формата HSL в RGB
const HSLToRGB = (h, s, l) => {
    s /= 100;
    l /= 100;
    const k = n => (n + h / 30) % 12;
    const a = s * Math.min(l, 1 - l);
    const f = n =>
      l - a * Math.max(-1, Math.min(k(n) - 3, Math.min(9 - k(n), 1)));
    return [255 * f(0), 255 * f(8), 255 * f(4)];
};

function numberToColorHsl(i) {
    var hue = (-i *  120) + 120;
    // преобразование HSL в RGB с параметрами насыщенности 100 и яркости в 50
    ↪  процентов
    var rgb = HSLToRGB(hue, 100, 50);
    // форматирование результата
    return 'rgb(' + rgb[0] + ',' + rgb[1] + ',' + rgb[2] + ')';
}
```

По карте видно, что большинство поездок начинается в районах Манхэттена. Следующими по популярности районами начала поездок являются аэропорт Джона Кеннеди и аэропорт Ла-Гуардия.

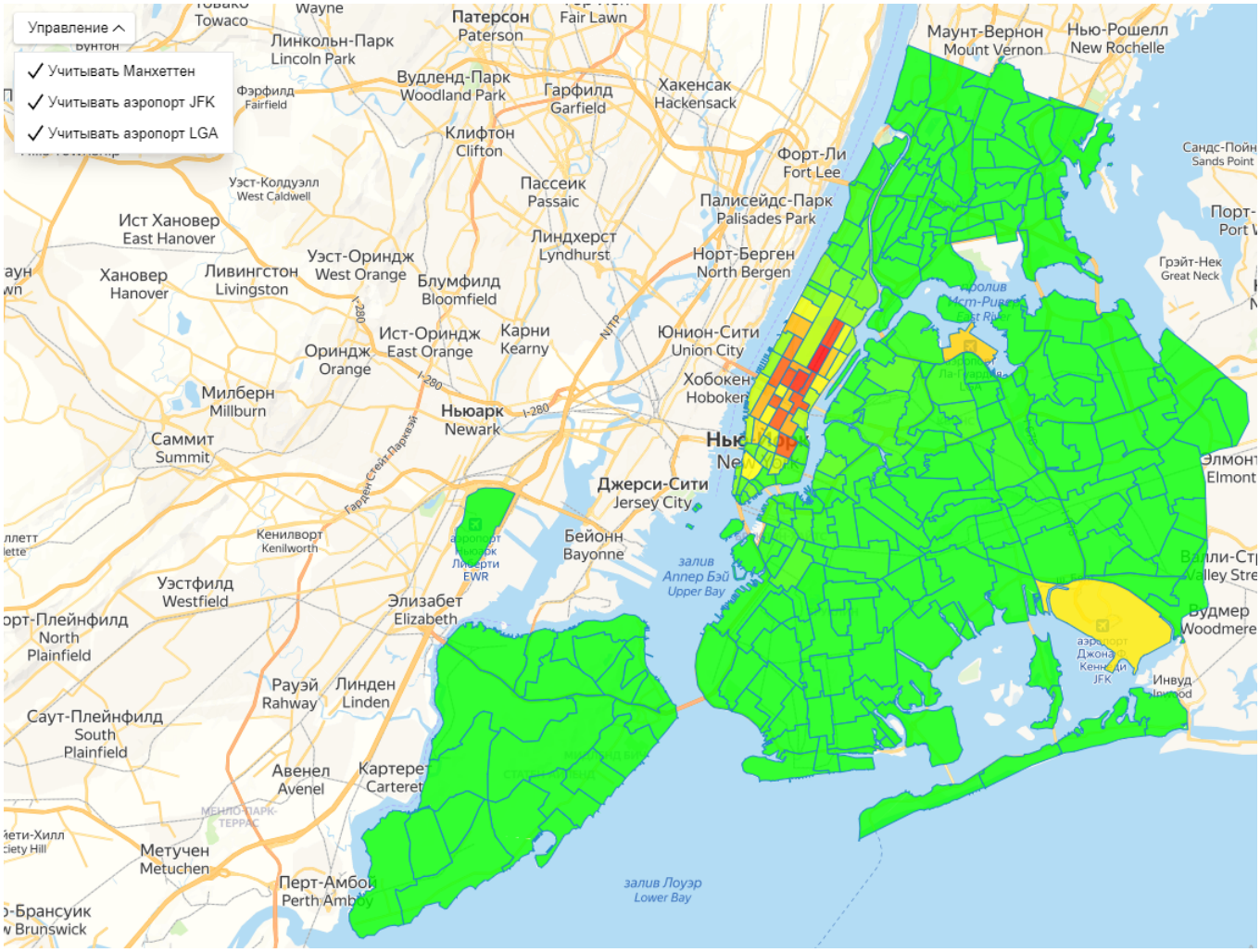

Рисунок 4 – Распределение поездок по районам

ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы было разработано Web-приложение, позволяющее получать аналитическую информацию из многомерной базы данных. В процессе разработки было создано и заполнено хранилище данных, создан OLAP-куб и написаны различные аналитические запросы. Для тестирования приложения был использован датасет поездок в такси Нью-Йорка, объем которого составляет 156 ГБ и 1.4 миллиарда строк. При таких объемах данных время обработки одного запроса составило порядка 30 секунд.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 *Барсегян, А. А.* Анализ данных и процессов / А. А. Барсегян, М. С. Куприянов, И. И. Холод, М. Д. Тесс, С. И. Елизаров. — Санкт-Петербург: БХВ-Петербург, 2009.

2 *Сарка, Д.* Microsoft SQL Server 2012. Реализация хранилищ данных / Д. Сарка, М. Лах, Г. Йеркич. — Москва: Русская редакция, 2014.

3 *Гутьеррес, Фелипе*. Spring Boot 2 лучшие практики для профессионалов / Фелипе Гутьеррес. — Санкт-Петербург: Питер, 2020.

4 *Томас, Марк Тиленс*. React в действии / Марк Тиленс Томас. — Санкт-Петербург: Питер, 2019.

5 ApiYandexMap [Электронный ресурс]. — URL: https://yandex.ru/dev/maps/?p=realty (Дата обращения 15.05.2022). Загл. с экр. Яз. англ.