Neo4j

1. Графическое представление на рис. 1.

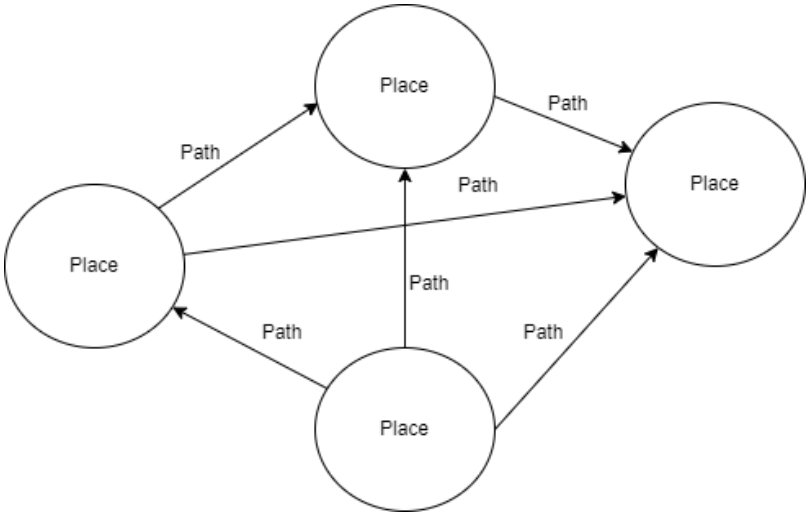

Рисунок 1 – Графическое представление данных Neo4j

2. Описание назначений коллекций, типов данных и сущностей

Вершина Place – достопримечательность.

Обладает следующими свойствами:

guid id – идентификатор, 16 b

string name – название, 200*2b

string description – описание, 400*2b

string imageUrl – ссылка на изображение, 200*2b

double latitude – широта, 8b

double longitude – долгота, 8b

string address – адрес, 400*2b

Итоговый размер – 2432b

Отношение Path – путь из одной достопримечательности к другой (считаем, что ребро графа направленно в обе стороны).

Обладает следующими свойствами:

guid id – идентификатор, 16b

string name – название, 200*2b

double distance – расстояние (в м), 8b

Итоговый размер – 424b

3. Оценка объема

Фактический размер коллекций:

Place – 2905b

Path – 446b

Пусть будет X достопримечательностей с количеством путей Y между ними, тогда для хранения чистых данных потребуется (2432b * X + 424b * Y) b, а фактический объем равен (2905b * X + 446b * Y) b

Избыточность модели $\frac{(2905\text{b} * \text{X} + 446\text{b} * \text{Y})\ \text{b}}{(2432\text{b} * \text{X} + 424\text{b} * \text{Y})\ \text{b}}$

4. Запросы

- Запрос на создание достопримечательностей и пути между ними

```
CREATE
    (p01:Place {
        id: apoc.create.uuid(),
        name: 'Казанский собор',
        description: 'lorem',
        imageUrl: 'lorem',
        latitude: 60.9344301,
        longitude: 60.9344301,
        address: "lorem"
    }),
    (p02:Place {
        id: apoc.create.uuid(),
        name: 'Исаакиевский собор',
        description: 'lorem',
```

```
        imageUrl: 'lorem',
        latitude: 60.9344301,
        longitude: 60.9344301,
        address: "lorem"
    } ),
    (p01)-[:path {distance: 1200, name: "lorem", id:
apoc.create.uuid()}  ]->(p02)
```

- Запрос на получение пути между двумя достопримечательностями

```
MATCH (x:Place)-[p:path*]->(z:Place)
WHERE x.name = 'Казанский собор' AND z.name = ‘Исаакиевский
собор'
RETURN p
```

MongoDB

1. Графическое представление на рис. 2.

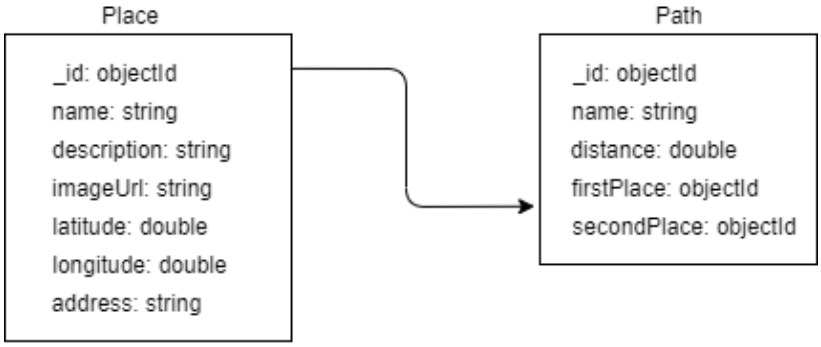

Рисунок 2 – Графическое представление данных MongoDB

2. Описание назначений коллекций, типов данных и сущностей

Коллекция Place – достопримечательности.

Обладает следующими полями:

objectId _id – идентификатор, 12b

string name – название, 200*2 b

string description – описание, 400*2b

string imageUrl – ссылка на изображение, 200*2b

double latitude – широта, 8b

double longitude – долгота, 8b

string address – адрес, 400*2b

Итоговый размер – 2428b

Коллекция Path – путь из одной достопримечательности к другой

Обладает следующими полями:

objectId _id – идентификатор, 12b

string name – название, 200*2b

double distance – расстояние (в м), 8b

objectId firstPlace – идентификатор первой достопримечательности, 12b

objectId secondPlace – идентификатор второй достопримечательности, 12b

Итоговый размер – 444b

3. Оценка объема

Фактический размер коллекций:

Place – 2389b

Path – 494b

Пусть будет X достопримечательностей с количеством путей Y между ними, тогда для хранения чистых данных потребуется (2428b * X + 444b * Y) b, а фактический объем равен (2389b * X + 494b * Y) b

Избыточность модели $\frac{(2389\text{b} * X + 494\text{b} * Y)\ \text{b}}{(2428\text{b} * X + 444\text{b} * Y)\ \text{b}}$

4. Запросы

- Запрос на создание достопримечательностей

```
db.Place.insert([{
   name: 'Казанский собор',
   description: 'lorem',
```

```
    imageUrl: 'lorem',
    latitude: 60.9344301,
    longitude: 60.9344301,
    address: "lorem"
},
{
    name: 'Исаакиевский  собор',
    description: 'lorem',
    imageUrl: 'lorem',
    latitude: 60.9344301,
    longitude: 60.9344301,
    address: "lorem"
}])
```

- Запрос на получение достопримечательности

db.Place.find({ name: "Казанский собор" })

MSSQL

1. Графическое представление на рис. 3.

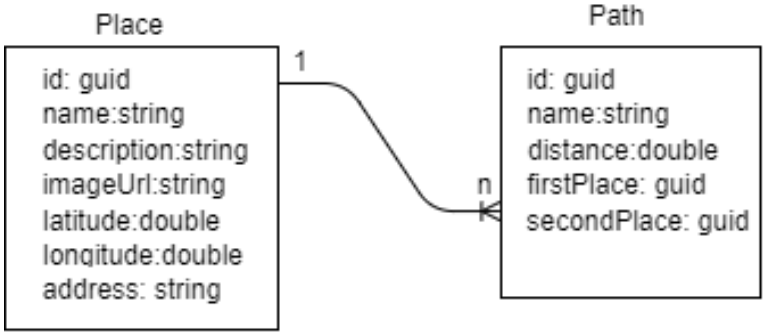

Рисунок 3 – Графическое представление данных MSSQL

2. Описание назначений коллекций, типов данных и сущностей

Таблица Place – достопримечательности.

Обладает следующими полями:

guid id – идентификатор, 16 b

string name – название, 200*2 b

string description – описание, 400*2 b

string imageUrl – ссылка на изображение, 200*2b

double latitude – широта, 8b

double longitude – долгота, 8b

string address – адрес, 400*2b

Итоговый размер – 2434b

Таблица Path – путь из одной достопримечательности к другой

Обладает следующими полями:

guid id – идентификатор, 16b

string name – название, 200*2b

double distance – расстояние (в м), 8b

guid firstPlace – идентификатор первой достопримечательности, 16b

guid secondPlace – идентификатор второй достопримечательности, 16b

Итоговый размер – 456b

3. Оценка объема

Фактический размер коллекций:

Place – 4096b

Path – 1792b

Пусть будет X достопримечательностей с количеством путей Y между ними, тогда для хранения чистых данных потребуется (2428b * X + 444b * Y) b, а фактический объем равен (4096b * X + 1792b * Y) b

Избыточность модели $\frac{(4096\text{b} * X + 1792\text{b} * Y)\ \text{b}}{(2428\text{b} * X + 444\text{b} * Y)\ \text{b}}$

4. Запросы

- Запрос на создание достопримечательностей

```
insert into Place (Name, Description, ImageUrl, Latitude, Longitude, Address)
VALUES
(
    N'Казанский собор',
    N'lorem',
    N'lorem',
    60.9344301,
    60.9344301,
    N'lorem'
),
(
    N'Исаакиевский собор',
    N'lorem',
    N'lorem',
    60.9344301,
    60.9344301,
    N'lorem'
)
```

- Запрос на получение достопримечательности

select * from Place where Name = N'Исаакиевский собор'

Сравнение моделей для Neo4j, MongoDB и MSSQL

1. Удельный объём информации

Для MSSQL требуется больше всего памяти для хранения сущностей (4096b и 1792b), на втором месте Neo4j (2905b и 446b), меньше всего памяти требуется для MongoDB (2389b и 494b).

2. Запросы по отдельным юзкейсам

В MSSQL и MongoDB для получения пути между достопримечательностями надо знать их идентификаторы, значит придётся делать (2 + 1) запрос. В Neo4j этого можно добиться одним запросом.

Вывод

В рамках задачи взаимодействия хранения геомаршрутов Neo4j использовать предпочтительнее, чем MSSQL или MongoDB, так как требуется меньшее количество запросов.

В плане объёмов занимаемой памяти из лидирует MongoDB.