NoSQL:

Описание сущностей:

1.City - информация о городе.

Содержит:

- «cityId» - int, идентификатор, 4 байта.
- «cityName» - string, название города, 50*2 байта.

Итого: 104 байта.

2.Roadwork - информация о дорожной работе. Содержит:

- «workId» - int, идентификатор, 4 байта.
- «workName» - string, название города, 50*2 байта.
- «workAddress» - string, название города, 50*2 байта.
- «workDate» - string, название города, формат: xx\xx\xx = 8*2 байта.
- «type» - string, название города, 50*2 байта.

Итого: 320 байта.

Существует связь между сущностями:

HAS - City->Roadwork.

Расчет объема:
Имеется N городов и M дорожных работ.
Объем городов - N*104 байта.
Объем дорожных работ - M*320 байта.
Итого: R1 = (N*104 + M*320) байта

Фактически объем:
HAS(k) - 34 байта
Города - M*232 байта
Дорожные работа - N*720 байт
Итого: R2 = (N*232 + M*720 + k*34) байта

Избыточность модели: I = (N*232 + M*720 + k*34)/(N*104 + M*320)

Пусть имеется M = 1000000 и k=3000000. Получаем:
Чистый объем от количества городов: N*104+1000000*320 (байт)= N*104+320000000 (байт) =N*104 (байт) + 0,32 (гигабайт)
Фактический объем от количества

городов:
N*104+1000000*324+3000000*34 (байт) = N*104+324000000+102000000 (байт) = N*104 (байт) + 0,426 (гигабайт)

Примеры запросов:

- Create (work{…}) - добавить работу
- Match (a {_id:34})-[:HAS]-(e) Return e.workName - найти название работ по id города

SQL

Описание сущностей:
1.City - информация о городе.
Содержит:

- «cityId» - int, идентификатор, 4 байта.
- «cityName» - string, название города, 50*2 байта.

Итого: 104 байта.
2.Roadwork - информация о дорожной работе. Содержит:

- «workId» - int, идентификатор, 4 байта.

- «workName» - string, название города, 50*2 байта.
- «workAddress» - string, название города, 50*2 байта.
- «workDate» - string, название города, формат: xx\xx\xx = 8*2 байта.
- «cityId» - int, идентификатор города, 4 байта.
- «type» - string, название города, 50*2 байта.

Итого: 324 байта.

Расчет объема:
Имеется N городов и M дорожных работ.
Объем городов - N*104 байта.
Объем дорожных работ - M*320 байта.
Итого: R3 = N*104 + M*320 байта

Фактический объем:
Объем городов - N*104 байта.
Объем дорожных работ - M*324 байта.
Итого: R4 = N*104 + M*324 байта

Избыточность модели: I = (N*104 +

M*324 байта)/(N*104 + M*320) байта

Пусть имеется M = 1000000. Получаем:
Чистый объем от количества городов: N*104+1000000*320 (байт)= N*104+320000000 (байт) =N*104 (байт) + 0,32 (гигабайт)
Фактический объем от количества городов: N*104+1000000*324 (байт) = N*104+324000000 (байт) = N*104 (байт) + 0,324 (гигабайт)

Примеры запросов:

- INSERT INTO Work VALUES(…) - добавить работу
- SELECT * FROM City INNER JOIN Work ON Work.workId = City.cityId - найти название работ по id города

Вывод:
Время выполнения запросов в Neo4j быстрее, чем допустим в MySQL. Но при этом Neo4j занимает больше памяти по сравнению с MySQL. Мы делаем выбор в сторону скорости

выполнения запросов.