

## 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:  $N * 104 + M * 320$  байта

Фактически объем:

HAS(k) - 34 байта

Города -  $M * 232$  байта

Дорожные работа -  $N * 720$  байт

Итого:  $N * 232 + M * 720 + k * 34$  байта

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

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 -  $N \cdot 104$  байта.

Объем дорожных работ -  $M \cdot 320$  байта.

Итого:  $N \cdot 104 + M \cdot 320$  байта

Фактический объем:

Объем городов -  $N \cdot 104$  байта.

Объем дорожных работ -  $M \cdot 324$  байта.

Итого:  $N \cdot 104 + M \cdot 324$  байта

Избыточность модели:  $I = (N \cdot 104 + M \cdot 324 \text{ байта}) / (N \cdot 104 + M \cdot 320) \text{ байта}$

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:

Почитав разные статейки пришли к выводу, что для данной задачи больше подойдет Neo4j, так как время

выполнения запросов быстрее, чем допустим в MySQL. Но при этом всем Neo4j «ест» больше памяти по сравнению с MySQL. Мы делаем выбор в сторону скорости выполнения запросов.