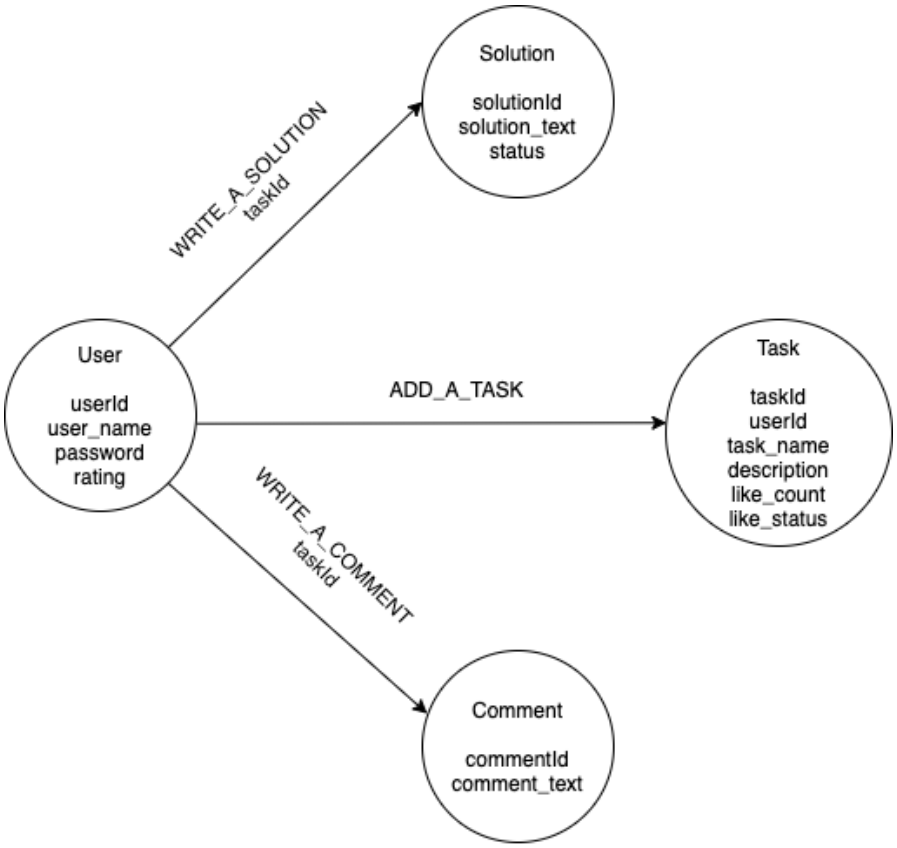
NoSQL Модель данных

1. Графическое представление

Графическое представление модели данных Neo4j:



2. Описание сущностей, коллекций, типов данных

Модель состоит из 4 сущностей:

1. User – хранит информацию о пользователе. Содержит поля:
    - “userId”, int – идентификатор. 4B
    - “user_name”, string – имя пользователя. 50*2B
    - “password”, string – пароль пользователя. 50*2B
    - “rating”, int – рейтинг. 4B

Итого: 208B

2. Task
    - “taskId”, int – идентификатор. 4B
    - “userId”, int – id пользователя. 4B

- “task_name”, string – название задачи. 50*2B
- “description”, string – описание задачи. 100*2B
- “like_count”, int – количество лайков у задачи. 4B
- “task_status”, int – статус задачи. 4B

Итого: 316B

3. Comment
    - “commentId”, int – идентификатор. 4B
    - “comment_text” – string. 100 * 2B

Итого: 204B

4. Solution
    - “solutionId”, int – идентификатор. 4B
    - “solution_text”, string – текст решения. 100*2B
    - “status”, int – флаг успешности решения. 4B

Итого: 208B

Существует 3 связи между сущностями:

1) WRITE_A_SOLUTION – User->Solution.

Данный пользователь пишет данные решения.

2) ADD_A_TASK – User->Task.

Пользователь создает данную задачу.

3) WRITE_A_COMMENT – User->Comment.

Пользователь пишет данный комментарий.

3. Расчет объема

Предположим, что имеется A пользователей, каждый из которых создал по 2 задачи, написал по 3 решения и оставил 5 комментариев.

Чистый объем:

- A * 204B – пользователи
- A * 2 * 308B – задачи

- A * 5 * 200B – комментарии
- A * 3 * 204B – решения

Чистый объем БД: A * 3052

Фактический объем:

- A * 464B – пользователи
- A * 2 * 576B – задачи
- A * 5 * 336B – комментарии
- A * 3 * 336B – решения
- WRITE_A_SOLUTION – A * 3 * 34B
- ADD_A_TASK – A * 2 * 34B
- WRITE_A_COMMENT – A * 5 * 34B

Фактический объем БД: A * 4644

Избыточность модели: (A * 4644) / (A * 3052)

4. Примеры запросов:

1) Добавить пользователя

CREATE (e:User {...})

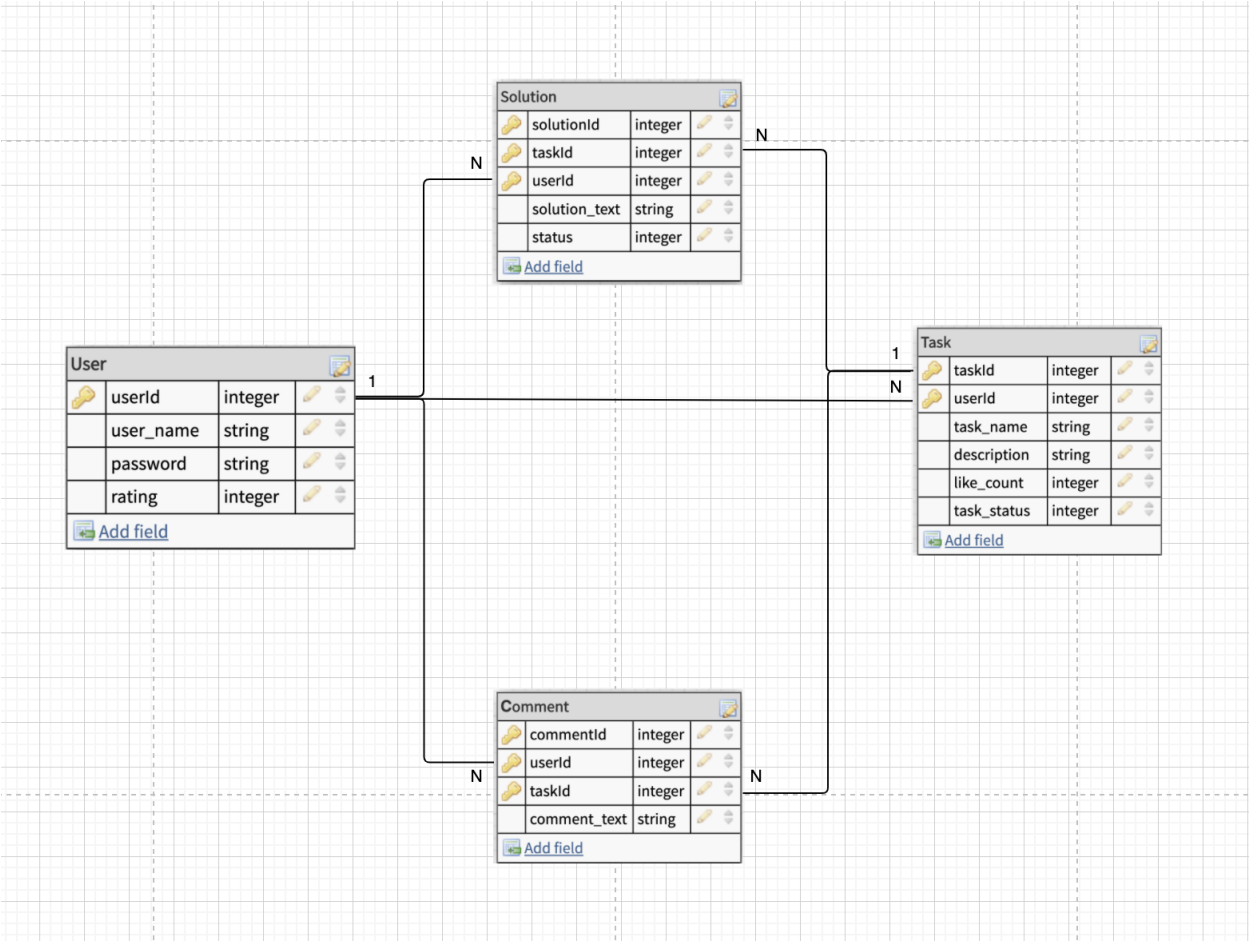
2) Найти задачу, добавленную пользователем, с id 12

MATCH (k {_id: 12})-[:ADD_A_TASK]-(e)
RETURN e.task_name, …..

SQL Модель данных

1. Графическое представление

Графическое представление модели данных MySQL:



2. Сущности модели данных

В качестве реляционной СУБД использована MySQL, в которой созданы 4 таблицы:

5. User – хранит информацию о пользователе. Содержит поля:
    - “userId”, int – идентификатор. 4B
    - “user_name”, string – имя пользователя. 50*2B
    - “password”, string – пароль пользователя. 50*2B
    - “rating”, int – рейтинг. 4B

Итого: 208B

6. Task
    - “taskId”, int – идентификатор. 4B
    - “userId”, int – id пользователя. 4B
    - “task_name”, string – название задачи. 50*2B
    - “description”, string – описание задачи. 100*2B
    - “like_count”, int – количество лайков у задачи. 4B
    - “task_status”, int – статус задачи. 4B

Итого: 316B

7. Comment
    - “commentId”, int – идентификатор. 4B
    - “userId”, int – id пользователя автора комментария. 4B
    - “taskId”, int – id задачи. 4B
    - “comment_text” – string. 100 * 2B

Итого: 212B

8. Solution
    - “solutionId”, int – идентификатор. 4B
    - “taskId”, int – id задачи. 4B
    - “userId”, int – id пользователя автора решения. 4B
    - “solution_text”, string – текст решения. 100*2B
    - “status”, int – флаг успешности решения. 4B

Итого: 216B

3. Оценка объема информации

Предположим, что имеется A пользователей, каждый из которых создал по 2 задачи, написал по 3 решения и оставил 5 комментариев. Чистый объем:

- A * 204B – пользователи

- A * 2 * 308B – задачи
- A * 5 * 200B – комментарии
- A * 3 * 204B – решения

Чистый объем БД: A * 2432

Фактический объем:

- A * 208B – пользователи
- A * 2 * 316B – задачи
- A * 5 * 212B – комментарии
- A * 3 * 216B – решения

Фактический объем БД: A * 2548

Избыточность модели: (A * 2548) / (A * 2432)

4. Примеры запросов:

1) Добавить пользователя

INSERT INTO User VALUES(…)

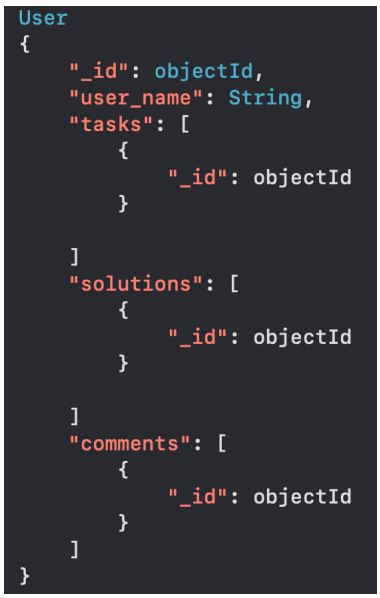
INSERT INTO Task VALUES(…)

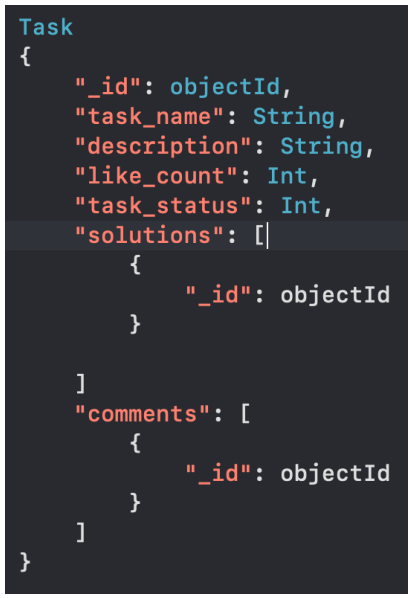
2) Найти задачу, добавленную пользователем, с id 12

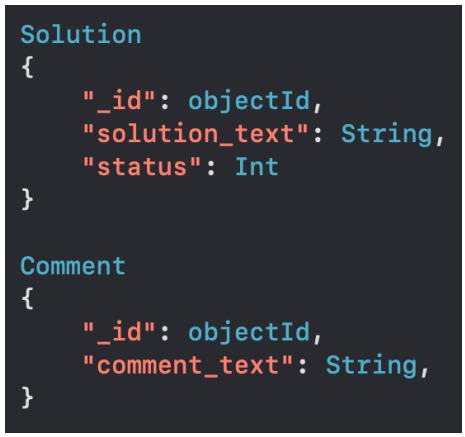
SELECT *FROM Task INNER JOIN User ON User.userId = Task.userId AND User.userId = 12.

1. Графическое представление

Графическое представление модели данных mongodb:







2. Сущности модели данных

Модель состоит из 4 сущностей

1) Пользователь (A – количество пользователей) 62B + 12B * B + 12B * C + 12B * D

- “_id”: objectId – идентификатор, 12B
- “user_name”: String – имя пользователя, 50 * 1B
- “tasks”: objectId[] – список задач, количество задач * 12B
- “solutions”: objectId[] – список решений, количество решений * 12B
- “comments”: objectId[] – список комментариев, количество комментариев * 12B

2) Задача (B – количество задач), 124B + 12B * C + 12B * D

- “_id”: objectId – идентификатор, 12B
- “task_name”: String – название задачи, 50 * 1B
- “description”: String – описание задачи, 50 * 1B
- “like_count”: Int – количество лайков, 8 * 1B
- “task_status”: Int – статус задачи, 4 * 1B

- “solutions”: objectId[] – список решений, количество решений * 12B
- “comments”: objectId[] – список комментариев, количество комментариев * 12B

3) Решение (C – количество решений), 66B

- “_id”: objectId – идентификатор, 12B
- “solution_text ”: String – текст решения задачи, 50 * 1B
- “status”: Int – статус решения, 4 * 1B

4) Комментарий (D – количество комментариев), 62B

- “_id”: objectId – идентификатор, 12B
- “comment_text ”: String – текст решения задачи, 50 * 1B

3. Оценка объема информации

Предположим, что имеется A пользователей, каждый из которых создал по 2 задачи, написал по 3 решения и оставил 5 комментариев.

Чистый объем: 62B * A + 2 * 124B + 3 * 66B + 5 * 62B = 818B * A

Фактический объем: 62B * A + 12B * 2 * (124B + 12B * 66B+ 12B * 62B) + 12B * 66B * 3 + 12B * 12B * 5 = A*42998

Избыточность модели: (A*42998) / (818*A)

4. Примеры запросов:

Добавление задачи:

```
db.tasks.insertOne({
    _id:1,
    "task_name": "TaskOne",
    "description": "This task about One",
    "like_count": 34,
    "task_status": 1,
    "solutions": [
        {
            "_id": 1
        },
        {
            "_id": 2
        }
    ]
    "comments": [
         {
             "_id": 3
         },
         {
             "_id": 6
         }
    ]
})
```

Поиск задачи:

```
db.tasks.find({_id:1})
```

Сравнение Neo4j, mongodb и SQL модели данных

SQL требует меньше памяти: A * 2548 против A * 4644 (Neo4j) и A * 42998 (mongodb), однако написание SQL запросов занимает больше времени. Запросы на SQL – языке являются более громоздкими, чем на языке Neo4j Cypher. Реализация модели данных в MySQL потребовало бы создать большее количество связей. Исходя из вышеперечисленного, можно сказать, что для рассматриваемой задачи Neo4j подходит в большей степени. Он побеждаен mongodb, так как требует меньше памяти.