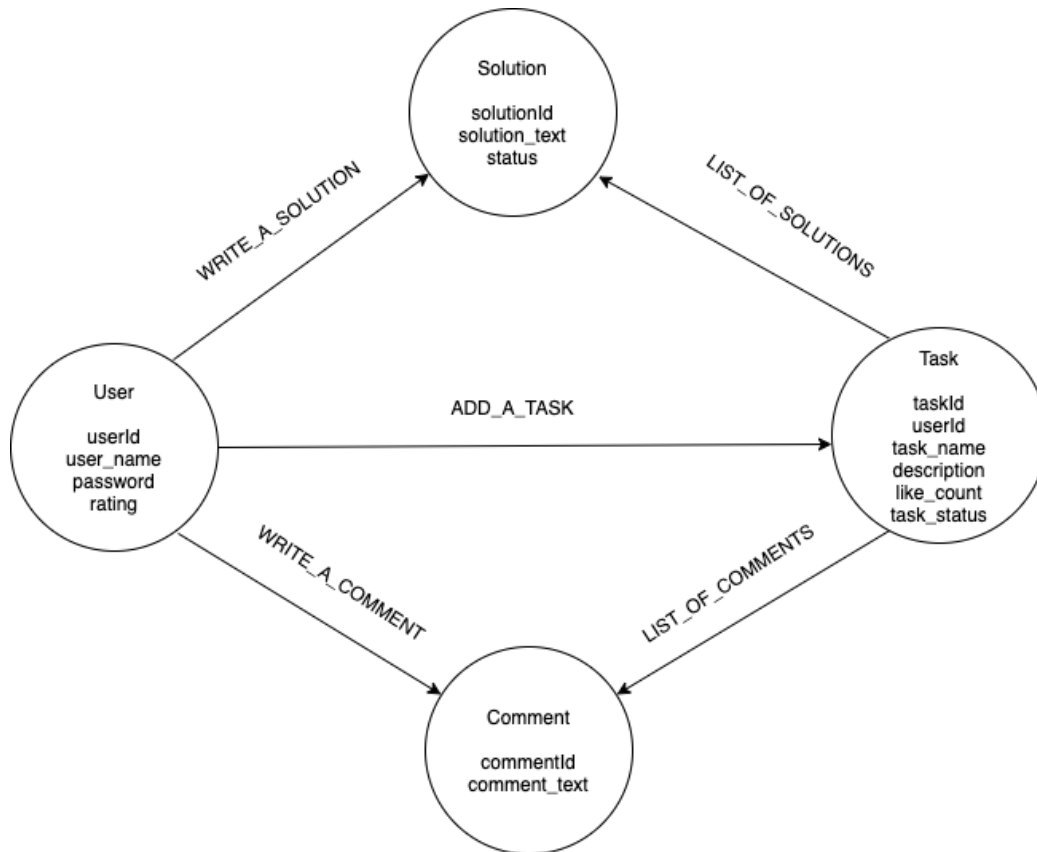


NoSQL Модель данных

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 5 связей 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.

4) LIST_OF_SOLUTIONS – Task->Solution

Данная задача содержит данные решения.

5) LIST_OF_COMMENTS – Task->Comments

Данная задача содержит данные комментарии.

3. Расчет объема

Предположим, что имеется А пользователей и В задач, в каждой из которых имеется С решений и D комментариев.

Чистый объем:

- $A * 204B$ – пользователи
- $B * 308B$ – задачи
- $B * C * 200B$ – комментарии
- $B * D * 204B$ – решения

Чистый объем БД: $A*204B + B*308B + B(C*200B + D*204B)$

Фактический объем:

- $A * 470B$ – пользователи
- $B * 750B$ – задачи
- $B * C * 444B$ – комментарии
- $B * D * 467B$ – решения
- WRITE_A_SOLUTION – 34B
- ADD_A_TASK – 34B
- WRITE_A_COMMENT – 34B
- LIST_OF_SOLUTIONS – 34B
- LIST_OF_COMMENTS – 34B

Фактический объем БД: $A*470B + B*750B + B(C*444B + D*467B)$

Избыточность модели: $A*470B + B*750B + B(C*444B + D*467B) / A*204B + B*308B + B(C*200B + D*204B)$

4. Примеры запросов:

1) Добавить пользователя

CREATE (e:User {...})

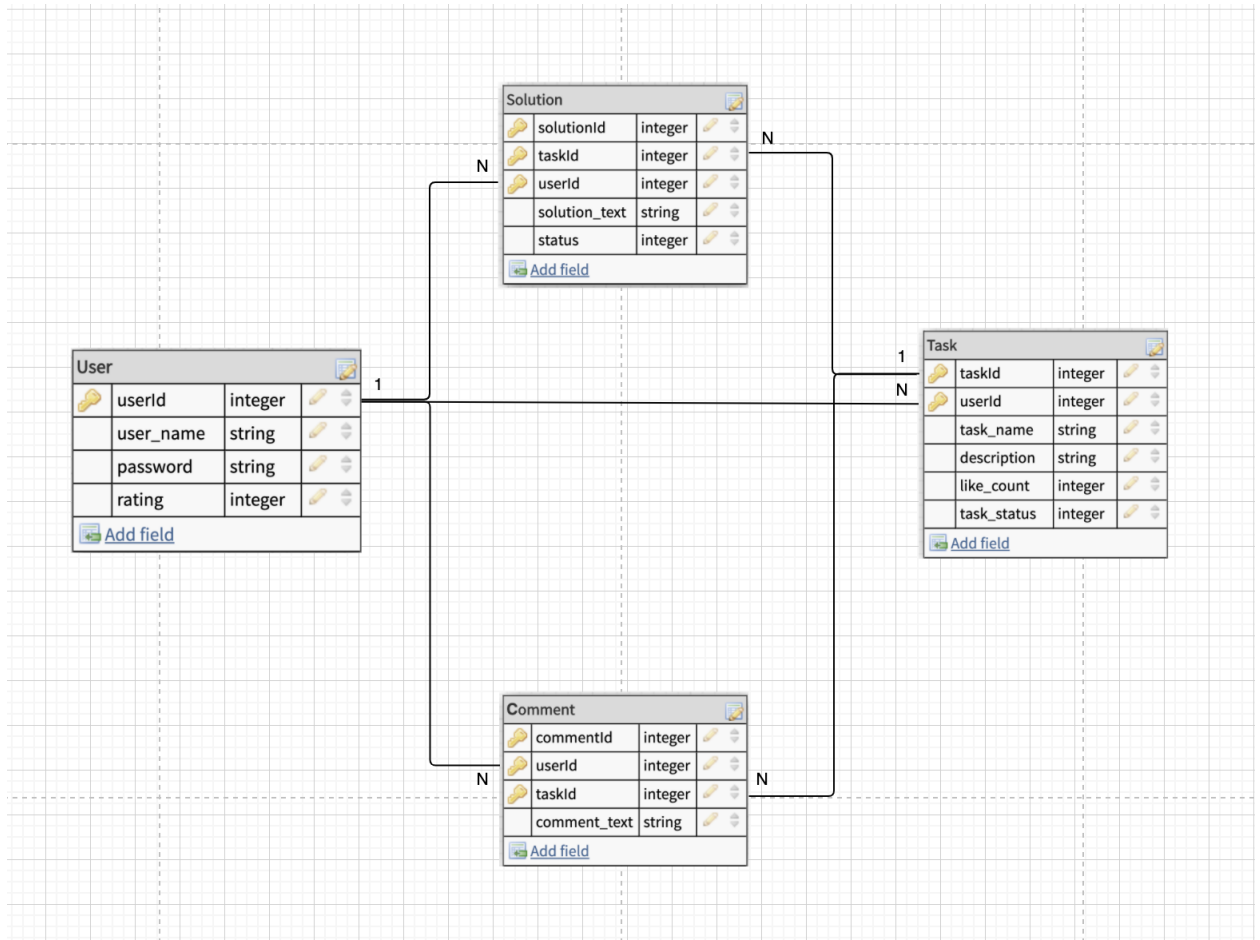
2) Найти задачу, добавленную пользователем, с id 12

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

Итого: 316В

7. Comment

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

Итого: 212В

8. Solution

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

Итого: 216В

3. Оценка объема информации

Предположим, что имеется А пользователей и В задач, в каждой из которых имеется С решений и D комментариев.

Чистый объем:

- $A * 204В$ – пользователи

- $B * 308B$ – задачи
- $B * C * 200B$ – комментарии
- $B * D * 204B$ – решения

Чистый объем БД: $A * 204B + B * 308B + B(C * 200B + D * 204B)$

Фактический объем:

- $A * 208B$ – пользователи
- $B * 316B$ – задачи
- $B * C * 212B$ – комментарии
- $B * D * 216B$ – решения

Фактический объем БД: $A * 208B + B * 316B + B(C * 212B + D * 216B)$

Избыточность модели: $(A * 208B + B * 316B + B(C * 212B + D * 216B)) / (A * 204B + B * 308B + B(C * 200B + D * 204B))$

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.
```

Сравнение Neo4j и SQL модели данных

Запросы на SQL – языке являются более громоздкими, чем на языке Neo4j Cypher. Neo4j требует больше памяти по сравнению с MySQL для хранения данных, но время выполнения запросов в Neo4j в среднем составляет меньше, чем в MySQL (Neo4j – 67 мс, MySQL – 123 мс) Исходя из вышеперечисленного, можно сказать, что для рассматриваемой задачи Neo4j подходит в большей мере.