Rui Ma V00800795

1. Read (At least) part 1 of the following article (All parts are worth reading):
What are the 3 basic ways to model a 'one-to-many' relationship in MongoDB and when would you use each one?

Basics: Modeling One-to-Few

Embed the N side if the cardinality is one-to-few and there is no need to access the embedded object outside the context of the parent object.

Basics: One-to-Many

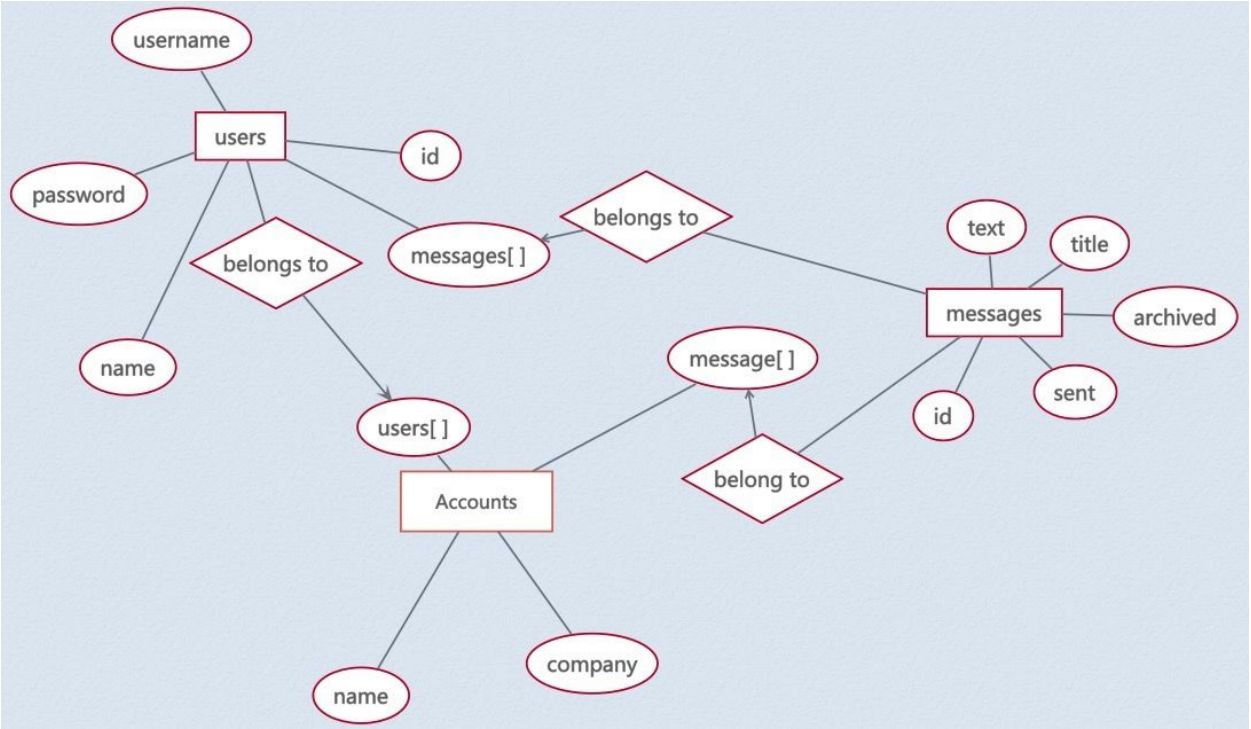
Use an array of references to the N-side objects if the cardinality is one-to-many or if the N-side objects should stand alone for any reasons.

Basics: One-to-Squillions

Use a reference to the One-side in the N-side objects if the cardinality is one-to-squillions.

2. Using the class slides on NoSQL, Google, and the requirements for a message system in Assignment 1, design an equivalent NoSQL schema in MongoDB.

a. **Using the E/R Diagram notation as best you can, create the E/R Diagram for your new schema (Won't be perfect – don't worry. We are using relational terminology to model something that isn't relational). Use a collection as an entity set – in this conversion you can have lists as attributes.**



b. List the collections you used and for each listed collection, explain why you chose to use it in your schema?

Accounts
name Company Users[ ] Messages[ ]

Users
Id username name Password Messages[ ]

Messages
id text title sent Archived

Accounts have two members of Users and messages, and for Users and messages, they have their own attributes and quantity. And each user and message has their own attributes too. So I need to use this one to represent relationships between Accounts, Users and Messages. Because the relationship between Users and Accounts is many-to-one, and for each User, the attributes are id, username, name, password, Messages[ ]. The messages list is made of messages with attributes.
And the relationship between Messages and Accounts is many-to-one, and for each Message, the attributes are id, text, title, sent, archived.

3. Compare the E/R Diagram from Assignment 1 with this newly created diagram:

a. Are there any differences? Why?

The Messages is an entity, whereas, it becomes an attribute list under Accounts, and Users. The Users is an entity, however, it becomes an attribute list under Accounts. Because Users and Messages become attribute right now, and they have their own attributes, they become a ‘list’.

b. What are the implications of these differences?

These differences make relational terminology to model something that isn’t relational.

4. Given the MySQL(relational) schema in part 3, design SQL queries to answer the following questions:

a. What are the names of all of the students?

```
SELECT name
FROM Students;
```

b. What is the average total mark?

```
SELECT AVG(mark)
FROM Takes;
```

c. What is the id and average mark for each class?

```
SELECT class_id, AVG(mark)
FROM Takes
GROUP BY class_id;
```

d. **What are the names and specialties of the instructors who teach ‘CSC370’?**

```
SELECT name, specialty
FROM Teaches, Instructors
WHERE Teaches.class_id=‘CSC370’ AND Teachrs.inst_id=Instructors.id
GROUP BY name;
```

e. **What are the names of the instructors for every class that Bob(student id = 123) is taking?**

```
SELECT name
FROM Instructors, Takes, Teaches;
WHERE Takes.students_id=’123’ AND Takes.class_id=Teaches.class_id AND
Teaches.inst_id=Instructors.id;
```