Document-based Database System Modelling Approaches

This document provides an explanation of the example of database modelling for a document-based database system like MongoDB.

We will see in detail about modelling a document-based database model based on a relational model. There is no standard way to model a NoSQL database e.g. a document-based database system unlike relational database management systems [1]. However, there are multiple ways to do it based on business requirements. Please refer to the example that you have learned during your lecture in the Week 4 folder on the BB. Below are the relational and document-based data models that you learned about in that class. Figure 1 depicts a relational model and Figure 2 depicts a document-based database model.

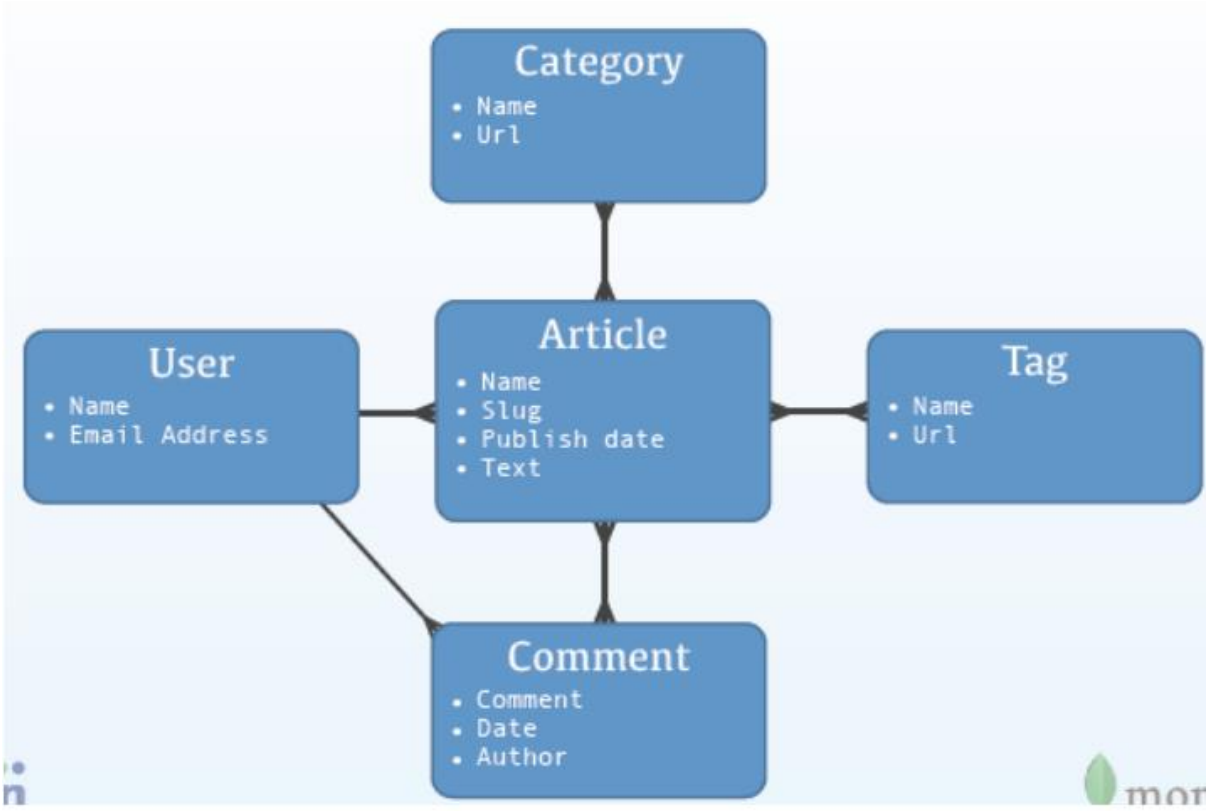

Figure 1Relational Database model.

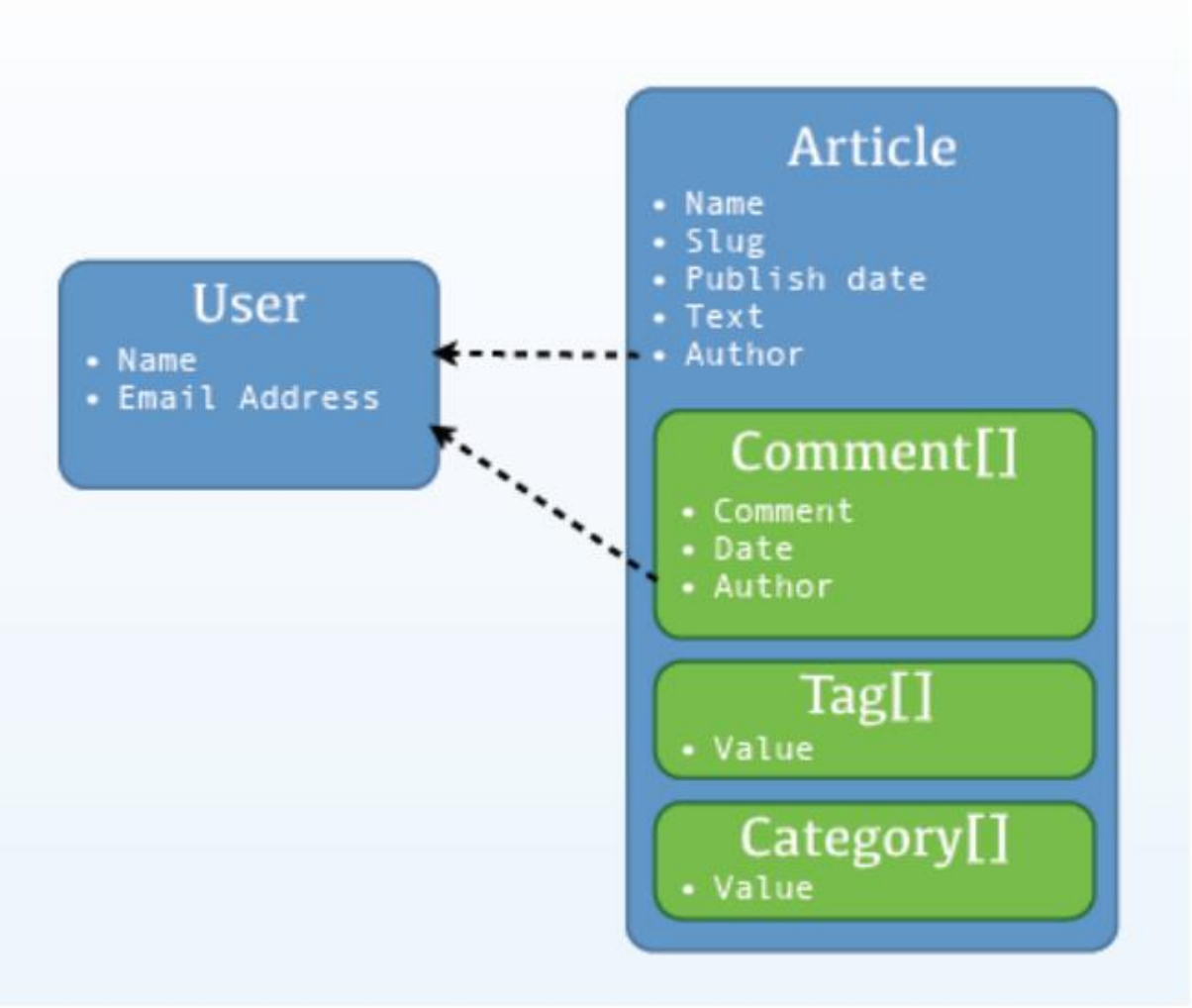

Figure 2Document-based database model.

A document-based database modelling generally uses 1) embedding and/or 2) referencing approach.

1) Embedding:
   In a document-based database, embedding refers to nesting or rather de-normalizing entities into another as sub-document(s). As can be seen in Figure 2 where the collection Article has sub-documents like comment, tag and category.

2) Referencing:
   In a document-based database, referencing refers to relationships like a primary/foreign key relationship in a relational database. Referencing in a document-based system results in normalized data – i.e. one collection referring to the other via a relationship. As can be seen between collections User and Article in Figure 2.

Embedding and/or Referencing approaches can be used based on business requirements. Let's see different cases and their pros and cons. We will look at all the relationships in comparison to the relational database modelling approach.

1) **Many-many relationships**
   Generally, this kind of relationship can be modelled using an embedding approach or a referencing approach based on business requirements. Both have their pros and cons. The de-normalizing approach i.e. embedding approach can be beneficial as you can achieve scalable and efficient solutions while still managing relationships between entities. It simplifies data modelling and querying. In case the referencing approach is used, you will need to use join operations and to some extent, it is okay to use the referencing approach. However, it may increase the complexity if the number of collections grows. Please refer to Figure 1 and Figure 2. Entities like comments, tag and category share a many-to-many relationship with Article in Figure 1 and are modelled using the embedded approach in Figure 2. They are embedded as sub-documents in the Article collection allowing simplicity in data modelling and efficient data querying.

2) **One-many relationships and vice vera**
   Generally, this kind of relationship can be modelled either using an embedding approach or a referencing approach. Both have their pros and cons and let us look at them in detail. The embedding approach can reduce the number of read operations and hence increase the efficiency of querying. It also decreases complexity as it may not require a join operation because all the documents can be retrieved from the same collection. However, if there is an exact pattern associated where one needs to repeat the same data again and again, and there are less number of collections involved, it is advisable to use the referencing approach. For eg. Figure 2 uses a referencing approach to model the relationship between User and Article.

3) **One-one relationships**
   Generally, this kind of relationship is modelled using the embedding approach. When the data is connected within a single document, it can minimize read operations that are required to retrieve the data. Moreover, it reduces the complexity as well because it may not require joins to retrieve the data as you can retrieve the data from the same collection.

References:

[1] How To Design Schema For NoSQL Data Models | MongoDB, Accessed 15th March 2024.