Best Practices for Migrating From RDBMS to MongoDB

Sheeri Cabral, Product Manager, Distributed Systems

Safe Harbor Statement

This presentation contains “forward-looking statements” within the meaning of Section 27A of the Securities Act of 1933, as amended, and Section 21E of the Securities Exchange Act of 1934, as amended. Such forward-looking statements are subject to a number of risks, uncertainties, assumptions and other factors that could cause actual results and the timing of certain events to differ materially from future results expressed or implied by the forward-looking statements. Factors that could cause or contribute to such differences include, but are not limited to, those identified our filings with the Securities and Exchange Commission. You should not rely upon forward-looking statements as predictions of future events. Furthermore, such forward-looking statements speak only as of the date of this presentation.

In particular, the development, release, and timing of any features or functionality described for MongoDB products remains at MongoDB’s sole discretion. This information is merely intended to outline our general product direction and it should not be relied on in making a purchasing decision nor is this a commitment, promise or legal obligation to deliver any material, code, or functionality. Except as required by law, we undertake no obligation to update any forward-looking statements to reflect events or circumstances after the date of such statements.

Agenda

60 minutes

Normalization and MongoDB

Schema Design and Performance

Seamless no-downtime Migration

Q&A

Who am I?

Who am I?

Master’s in Computer Science

Who am I?

Master’s in Computer Science

Sysadmin for 4 years

Who am I?

Master’s in Computer Science

Sysadmin for 4 years

MySQL DBA for 14 years

RDBMS = Relational Database Management System

Relation = Table

	User	
ID	**Name**	**Username**
1	Sheeri K. Cabral	sheeri
2	Tony Cabral	magicTony

	User	
ID	**Name**	**Username**
1	Sheeri K. Cabral	sheeri
2	Tony Cabral	magicTony

row ~ document

	User	
ID	**Name**	**Username**
1	Sheeri K. Cabral	sheeri
2	Tony Cabral	magicTony

row ~ document

table ~ collection

	User	
ID	**Name**	**Username**
1	Sheeri K. Cabral	sheeri
2	Tony Cabral	magicTony

```
{
  id: 1,
  name: "Sheeri K. Cabral",
  username: "sheeri"
}
```

	User	
ID	**Name**	**Username**
1	Sheeri K. Cabral	sheeri
2	Tony Cabral	magicTony

```
{
  id: 1,
  name: "Sheeri K. Cabral",
  username: "sheeri"
}
```

```
{
  id: 2,
  name: "Tony Cabral",
  username: "magicTony"
}
```

	User	
ID	**Name**	**Username**
1	Sheeri K. Cabral	sheeri
2	Tony Cabral	magicTony

		Articles		
ID	**Author_ID**	**Title**	**Category**	**Body**
1	1	“First Post”	general	“Welcome to my blog. Hello, world!”
2	1	“Apple Picking”	fun	“Today we went apple picking. What fun!”
3	1	“It’s Been a While”	general	“I’m working on a webinar.”

```
{
  id: 1,
  name: "Sheeri K. Cabral",
  username: "sheeri",
  posts: [
    { title: "First Post",
      category: "general",
      body: "Welcome to my blog. Hello, world!"
    },
    { title: "Apple Picking",
      category: "fun",
      body: "Today we went apple picking. What fun!"
    },
    { title: "It's Been a While",
      body: "I'm working on a webinar.",
      category: "general",
    },
  ]
}
```

```
{
  id: 1,
  name: "Sheeri K. Cabral",
  username: "sheeri",
  posts: [
    { title: "First Post",
      category: "general",
      body: "Welcome to my blog. Hello, world!"
    },
    { title: "Apple Picking",
      category: "fun",
      body: "Today we went apple picking. What fun!"
    },
    { title: "It's Been a While",
      body: "I'm working on a webinar.",
      category: "general",
    },
  ]
}
```

```
{
  id: 2,
  name: "Tony Cabral",
  username: "magicTony",
  posts: [
    { title: "New Trick",
      body: "I'm working on a new trick with dice."
    },
    { title: "Apple Picking",
      body: "Today we went apple picking. It was OK."
    },
  ]
}
```

What problems does normalization solve?

What problems does normalization solve?

Hard to update a multi-value data cell

Sheeri K. Cabral
@sheeri

Yo dawg I heard you like databases so I put 3,915 JSON objects into a MySQL LONGTEXT field.

11:13 AM · Dec 19, 2012 · YoruFukurou

View Tweet activity

77 Retweets **37** Likes

What problems does normalization solve?

Duplicate data leads to **data integrity** problems when doing updates

Hard to update a multi-value data cell

What **problems** does normalization solve?

Duplicate data leads to

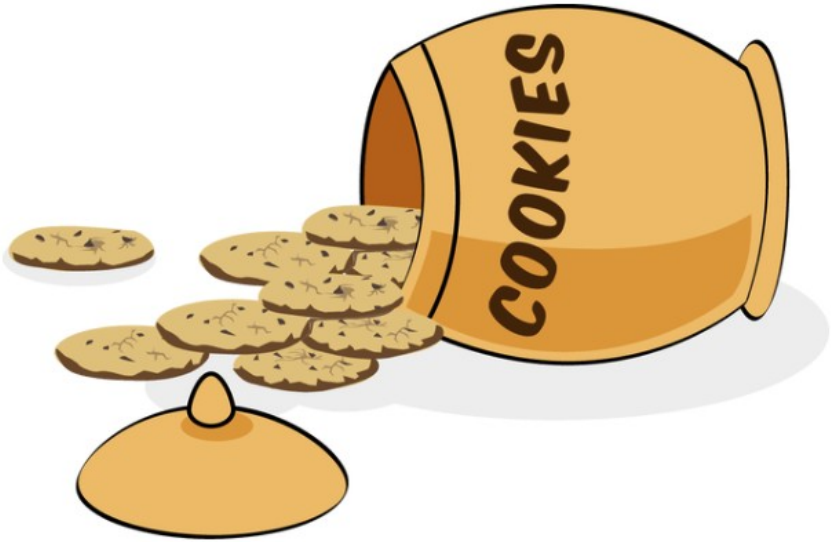
data integrity

problems when doing updates

Hard to update a multi-value data cell

Duplicate data

wastes resources


COOKIES

What **problems** does normalization cause?

What problems does normalization cause?

Transactions

(ACID compliance) more difficult

What **problems** does normalization cause?

Transactions

(ACID compliance) more difficult

Joins are expensive

What problems does normalization cause?

Transactions

(ACID compliance) more difficult

Joins are expensive

Migrations are not convenient

```
{
  id: 1,
  name: "Sheeri K. Cabral",
  username: "sheeri",
  posts: [
    { title: "First Post",
      category: "general",
      body: "Welcome to my blog. Hello, world!"
    },
    { title: "Apple Picking",
      category: "fun",
      body: "Today we went apple picking. What fun!"
    },
    { title: "It's Been a While",
      body: "I'm working on a webinar.",
      category: "general",
    },
  ]
}
```

Data that is **accessed together**

should be **stored together**

```
{
  id: 1,
  name: "Sheeri K. Cabral",
  username: "sheeri"
}
```

```
{
  id: 2,
  name: "Tony Cabral",
  username: "magicTony"
}
```

users

articles

```
{ authorId: 1,
  title: "First Post",
  category: "general",
  body: "Welcome to my blog. Hello, world!"
}
```

```
{ authorId: 2,
  title: "New Trick",
  body: "I'm working on a new trick with dice."
}
```

```
{
  id: 1,
  name: "Sheeri K. Cabral",
  username: "sheeri"
}
```

```
{
  id: 2,
  name: "Tony Cabral",
  username: "magicTony"
}
```

users

articles

```
{ authorId: 2,
  title: "New Trick",
  body: "I'm working on a new trick with dice.",
  comments: [
    { authorId: 1,
      body: "I can't wait to see it!"
    },
    { authorId: 1,
      body: "What are you calling the trick?"
    }
  ]
}
```

```
{
  id: 1,
  name: "Sheeri K. Cabral",
  username: "sheeri"
}
```

```
{
  id: 2,
  name: "Tony Cabral",
  username: "magicTony"
}
```

users

articles

```
{ authorId: 2,
  title: "New Trick",
  body: "I'm working on a new trick with dice.",
  comments: [
    { authorUserName: "sheeri",
      authorName: "Sheeri K. Cabral",
      body: "I can't wait to see it!"
    },
    { authorUserName: "sheeri",
      authorName: "Sheeri K. Cabral",
      body: "What are you calling the trick?"
    }
  ]
}
```

```
{
  id: 1,
  name: "Sheeri K. Cabral",
  username: "sheeri"
}
```

```
{
  id: 2,
  name: "Tony Cabral",
  username: "magicTony"
}
```

users

```
// Get the user object
> user = db.user.findOne({username: “sheeri”});
```

articles

```
{ authorId: 2,
  title: "New Trick",
  body: "I'm working on a new trick with dice.",
  comments: [
    { authorId: 1,
      body: "I can't wait to see it!"
    },
    { authorId: 1,
      body: "What are you calling the trick?"
    }
  ]
}
```

```
{
  id: 1,
  name: "Sheeri K. Cabral",
  username: "sheeri"
}
```

```
{
  id: 2,
  name: "Tony Cabral",
  username: "magicTony"
}
```

users

```
// Get the user object
> user = db.user.findOne({username: “sheeri”});
  // Get all the articles linked to the person
> myArticles = db.articles.find({_id: {
          $in : people.articles.map(authorId => user._id) } } )
```

articles

```
{ authorId: 2,
  title: "New Trick",
  body: "I'm working on a new trick with dice.",
  comments: [
    { authorId: 1,
      body: "I can't wait to see it!"
    },
    { authorId: 1,
      body: "What are you calling the trick?"
    }
  ]
}
```

```
{
  id: 1,
  name: "Sheeri K. Cabral",
  username: "sheeri"
}
```

```
{
  id: 2,
  name: "Tony Cabral",
  username: "magicTony"
}
```

users

```
// Get the user object
> user = db.user.findOne({username: “sheeri”});
```

articles

```
{ authorId: 2,
  title: "New Trick",
  body: "I'm working on a new trick with dice.",
  comments: [
    { authorId: 1,
      body: "I can't wait to see it!"
    },
    { authorId: 1,
      body: "What are you calling the trick?"
    }
  ]
}
```

```
{
  id: 1,
  name: "Sheeri K. Cabral",
  username: "sheeri"
}
```

```
{
  id: 2,
  name: "Tony Cabral",
  username: "magicTony"
}
```

users

```
// Get the user object
> user = db.user.findOne({username: “sheeri”});
  // Get all the articles linked to the person
> myArticles = db.articles.find({_id: {
          $in : people.articles.map(authorId => user._id) } } )
```

articles

```
{ authorId: 2,
  title: "New Trick",
  body: "I'm working on a new trick with dice.",
  comments: [
    { authorId: 1,
      body: "I can't wait to see it!"
    },
    { authorId: 1,
      body: "What are you calling the trick?"
    }
  ]
}
```

Model the objects that your

application uses

```
{
  id: 1,
  name: "Sheeri K. Cabral",
  username: "sheeri"
}
```

```
{
  id: 2,
  name: "Tony Cabral",
  username: "magicTony"
}
```

users

articles

```
{ authorId: 2,
  title: "New Trick",
  body: "I'm working on a new trick with dice.",
  comments: [
    { authorId: 1,
      body: "I can't wait to see it!"
    },
    { authorId: 1,
      body: "What are you calling the trick?"
    }
  ]
}
```

articles

```
{ authorId: 2,
  authorUserName: "magicTony",
  title: "New Trick",
  body: "I'm working on a new trick with dice.",
  comments: [
    { authorId: 1,
      authorUserName: "sheeri",
      body: "I can't wait to see it!"
    },
    { authorId: 1,
      authorUserName: "sheeri",
      body: "What are you calling the trick?"
    }
  ]
}
```

users

```
{
  id: 1,
  name: "Sheeri K. Cabral",
  username: "sheeri",
  permissions: [
    { can_comment: true },
    { can_post: true },
    { can_admin: true },
  ]
}
```

```
{
  id: 2,
  name: "Tony Cabral",
  username: "magicTony",
  permissions: [
    { can_comment: true },
    { can_post: true },
  ]
}
```

articles

```
{ authorId: 2,
  authorUserName: "magicTony",
  title: "New Trick",
  body: "I'm working on a new trick with dice.",
  comments: [
    { authorId: 1,
      authorUserName: "sheeri",
      body: "I can't wait to see it!"
    },
    { authorId: 1,
      authorUserName: "sheeri",
      body: "What are you calling the trick?"
    }
  ]
}
```

users

```
{
  id: 1,
  name: "Sheeri K. Cabral",
  username: "sheeri",
  permissions: [
    { can_comment: true },
    { can_post: true },
    { can_admin: true },
  ]
}
```

```
{
  id: 2,
  name: "Tony Cabral",
  username: "magicTony",
  permissions: [
    { can_comment: true },
    { can_post: true },
  ]
}
```

articles

```
{ authorId: 2,
  authorUserName: "magicTony",
  title: "New Trick",
  body: "I'm working on a new trick with dice.",
  comments: [
    { authorId: 1,
      authorUserName: "sheeri",
      body: "I can't wait to see it!"
    },
    { authorId: 1,
      authorUserName: "sheeri",
      body: "What are you calling the trick?"
    }
  ]
}
```

Extended reference

users

```
{
  _id: ObjectId("ab12"),
  name: "Sheeri K. Cabral",
  username: "sheeri",
  permissions: [
    { can_comment: true },
    { can_post: true },
    { can_admin: true },
  ]
}
```

```
{
  _id: ObjectId("3cd4"),
  name: "Tony Cabral",
  username: "magicTony",
  permissions: [
    { can_comment: true },
    { can_post: true },
  ]
}
```

users

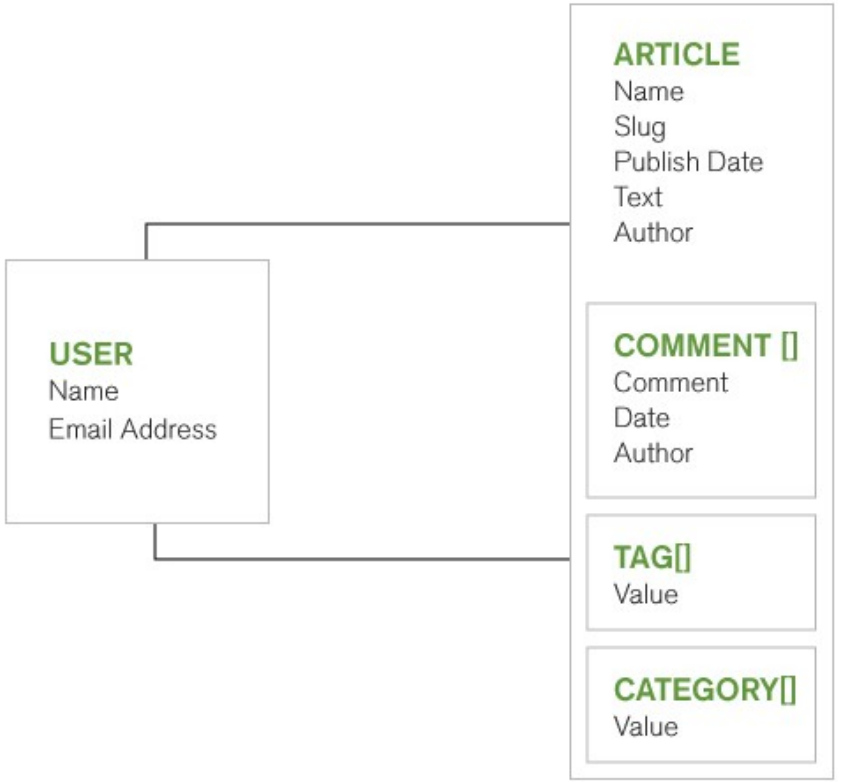
```
{
  _id: ObjectId("ab12"),
  name: "Sheeri K. Cabral",
  username: "sheeri",
  permissions: [
    { can_comment: true },
    { can_post: true },
    { can_admin: true },
  ]
}
```

```
{
  _id: ObjectId("3cd4"),
  name: "Tony Cabral",
  username: "magicTony",
  permissions: [
    { can_comment: true },
    { can_post: true },
  ]
}
```

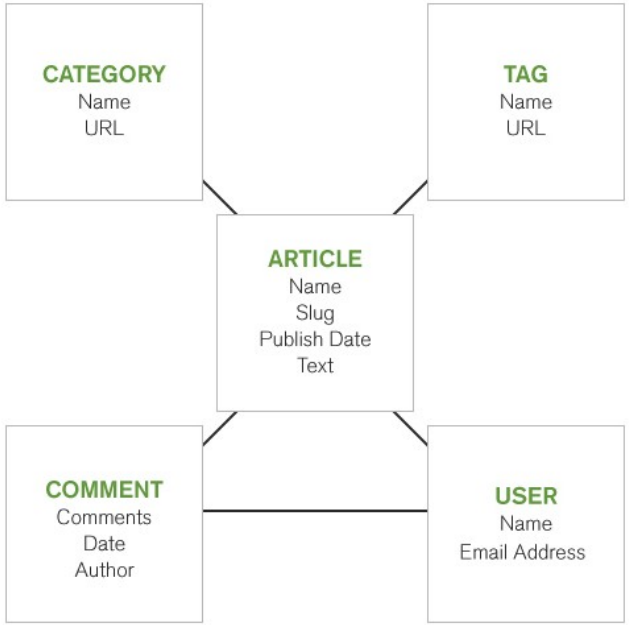
articles

```
{
  _id: ObjectId("aaaa"),
  authorId: ObjectId("3cd4"),
  authorUserName: "magicTony",
  title: "New Trick",
  body: "I'm working on a new trick with dice.",
  comments: [
    { authorId: ObjectId("ab12"),
      authorUserName: "sheeri",
      body: "I can't wait to see it!"
    },
    { authorId: ObjectId("ab12"),
      authorUserName: "sheeri",
      body: "What are you calling the trick?"
    }
  ]
}
```

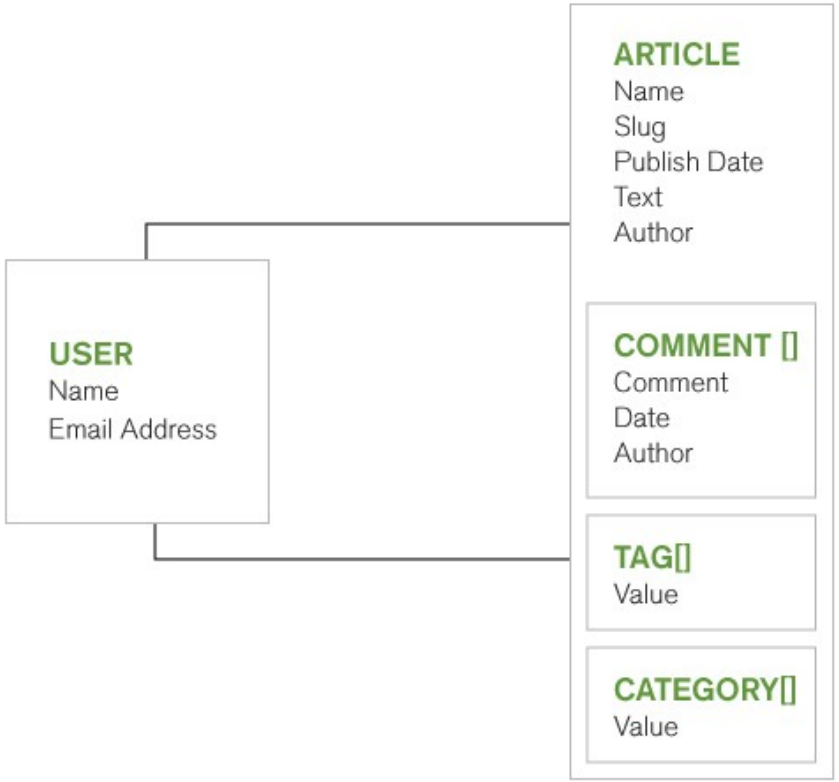
MongoDB


ARTICLE
Name
Slug
Publish Date
Text
Author
USER
Name
Email Address
COMMENT []
Comment
Date
Author
TAG[]
Value
CATEGORY[]
Value

Relational



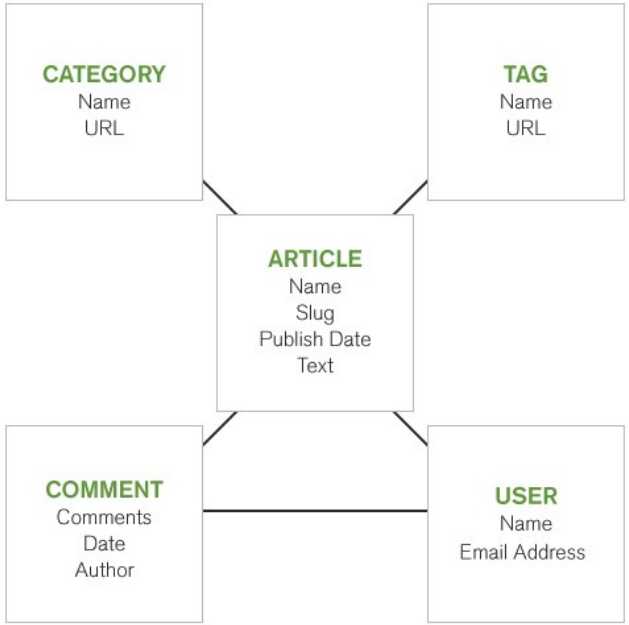
MongoDB



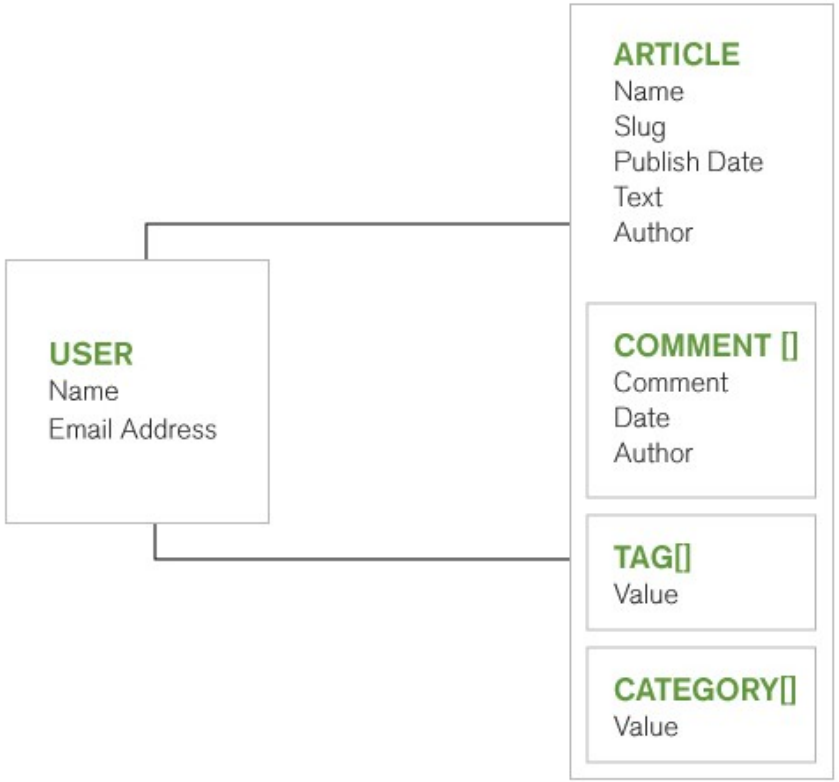
Data that is **accessed together**

should be **stored together**

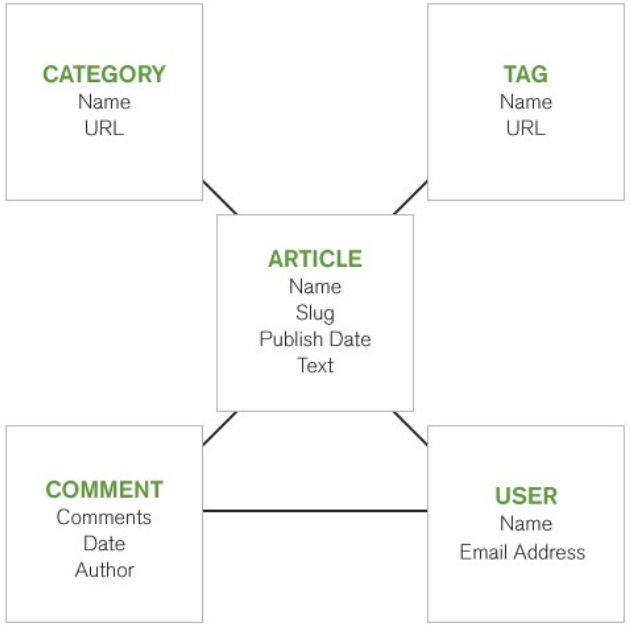
Relational



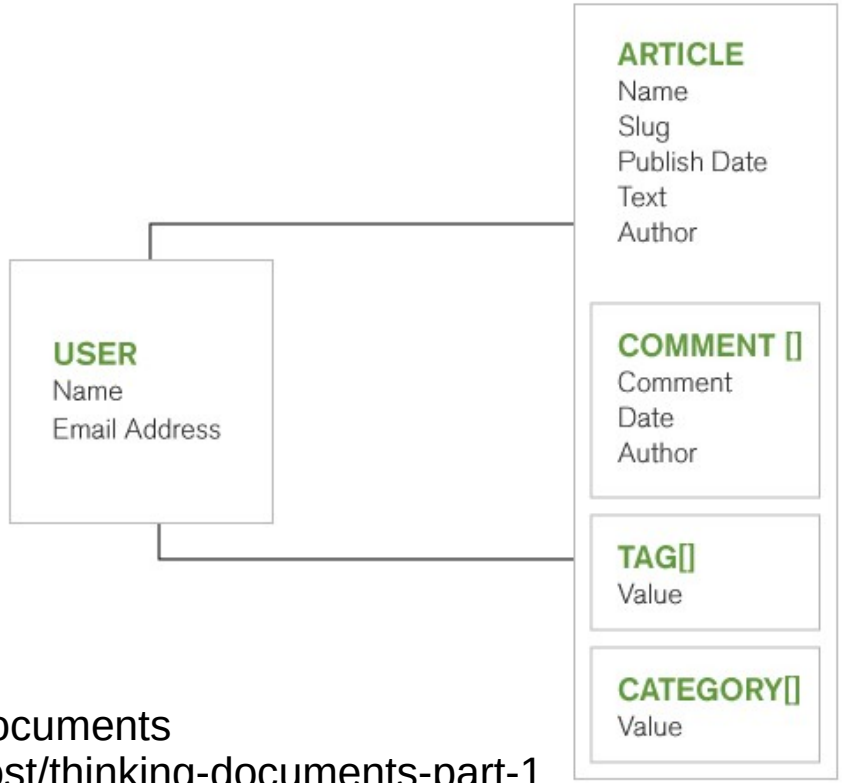
MongoDB



Relational



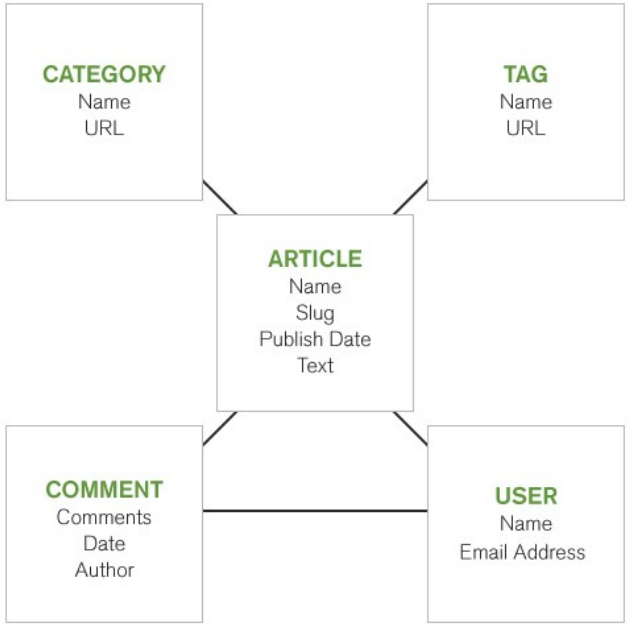
MongoDB



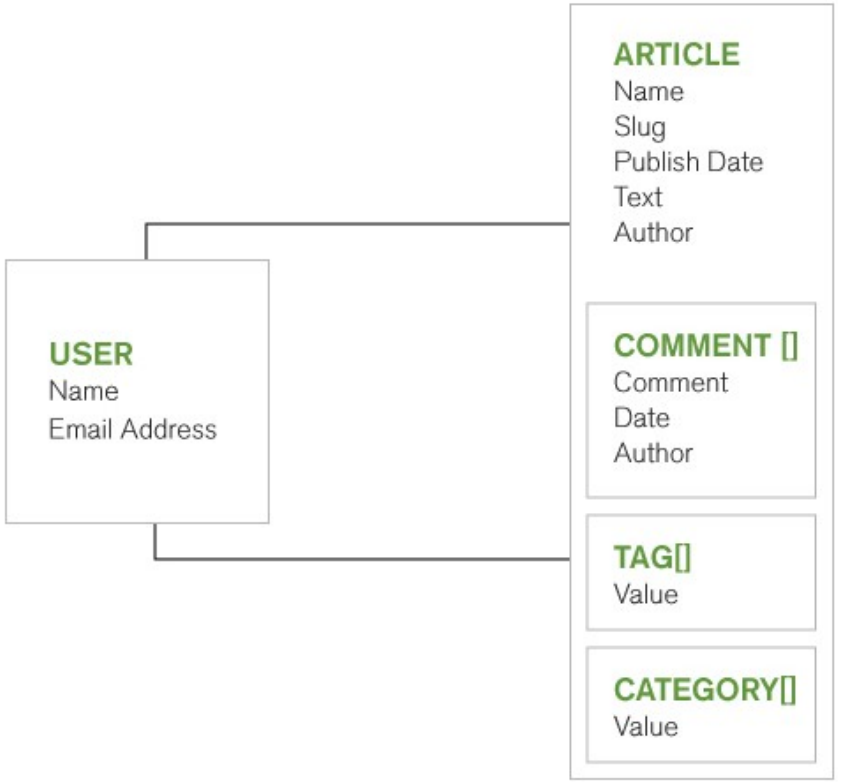
Thinking in Documents
https://www.mongodb.com/blog/post/thinking-documents-part-1

6 Rules of Thumb for MongoDB Schema Design
https://www.mongodb.com/blog/post/6-rules-of-thumb-for-mongodb-schema-design-part-1

Relational

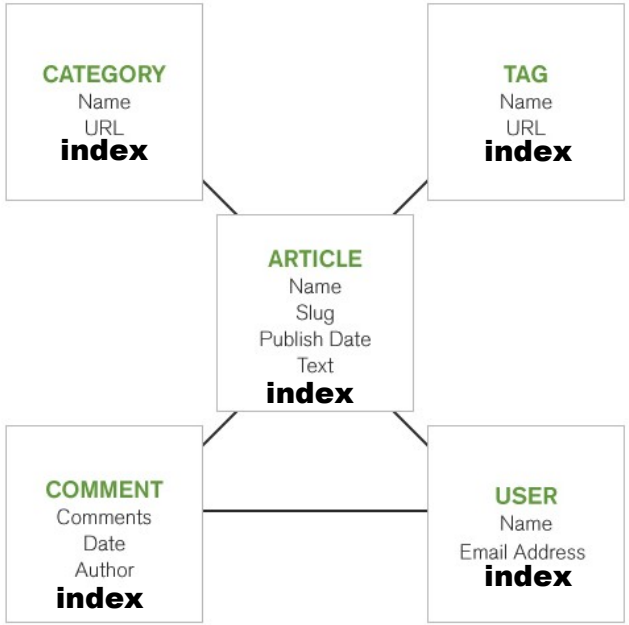


MongoDB

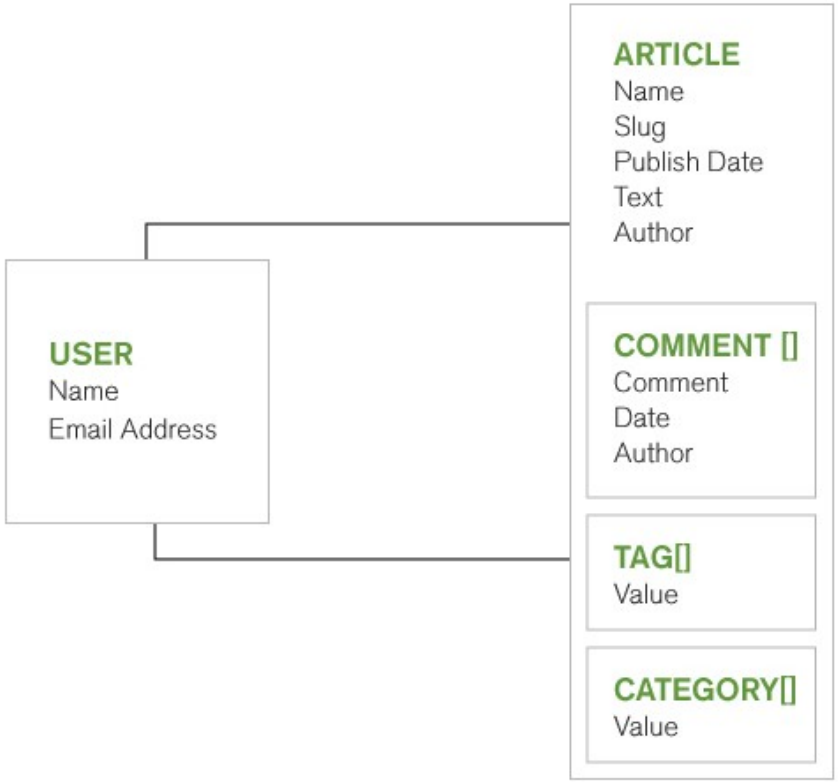


What about indexes?

Relational

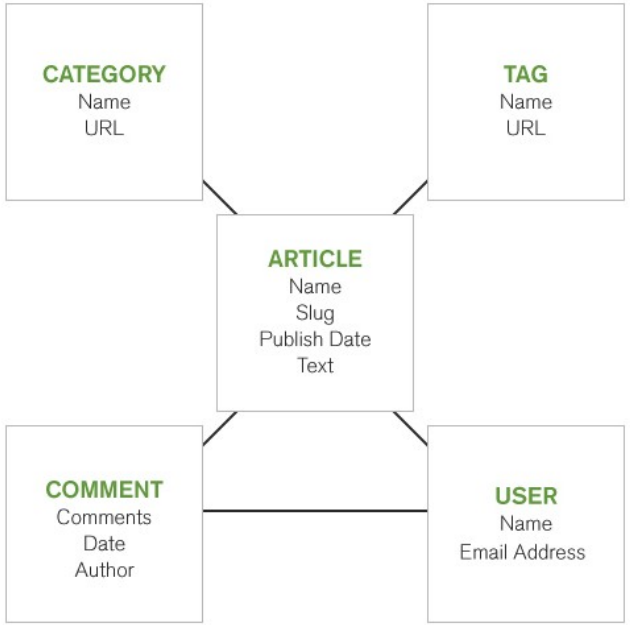


MongoDB

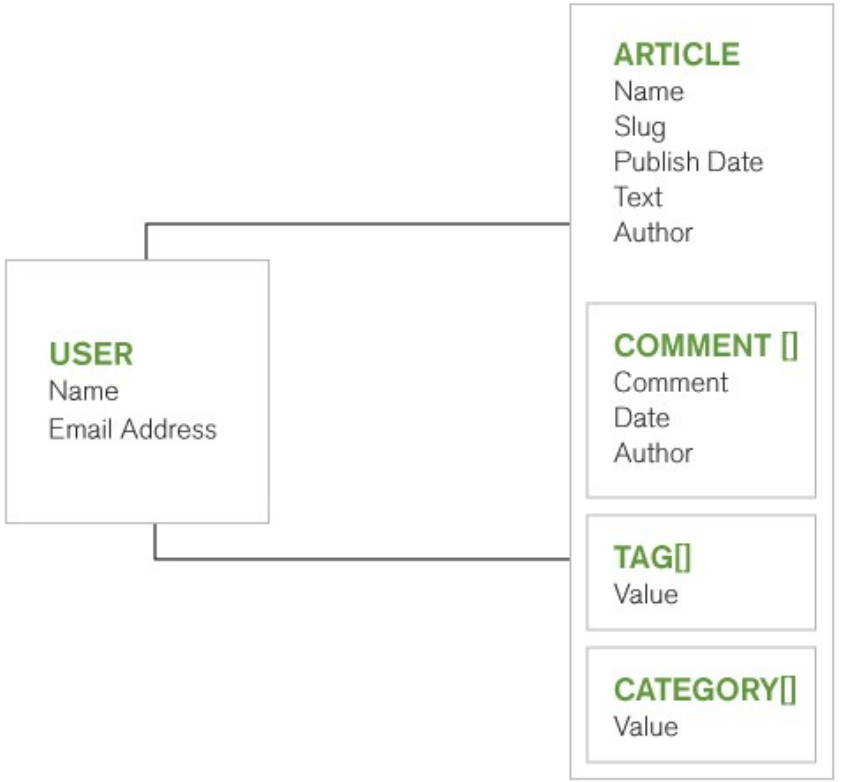


What about indexes?

Relational



MongoDB



What about indexes?

What about indexes?

What about indexes?

simple = single field

What about indexes?

simple = single field

compound = multiple fields

What about indexes?

simple = single field

compound = multiple fields

multi-key = index for arrays and nested arrays

What about indexes?

simple = single field

compound = multiple fields

multi-key = index for arrays and nested arrays

Unique or **non-unique**

What about structure?

What about structure?

schema validation

What about **structure?**

schema validation

require **fields**

What about **structure?**

schema validation

require **fields**

specify **data types**

including **enum**erated **lists**

What about foreign keys?

What about foreign keys?

Do you really need them?

What about **foreign keys?**

Do you **really** need them?

App **validates** from db lookups

What about **foreign keys?**

Do you **really** need them?

App **validates** from db lookups

Why validate **again?**

What about **foreign keys?**

Do you **really** need them?

App **validates** from db lookups

Why validate **again?**

How does your app handle **failures?**

What about foreign keys?

What about foreign keys?

embed for parent/child

What about foreign keys?

embed for parent/child

schema validation and enum for specific values

What about foreign keys?

embed for parent/child

schema validation and enum for specific values

reference

What about transactions?

Atomicity

succeeds or fails completely

What about transactions?

Atomicity

succeeds or fails completely

Consistency

db from one valid state to another

What about **transactions?**

Atomicity

succeeds or fails completely

Consistency

db from one valid state to another

Isolation

how/when changes are seen by ops

What about transactions?

Atomicity

succeeds or fails completely

Consistency

db from one valid state to another

Isolation

how/when changes are seen by ops

Durability

completion is forever

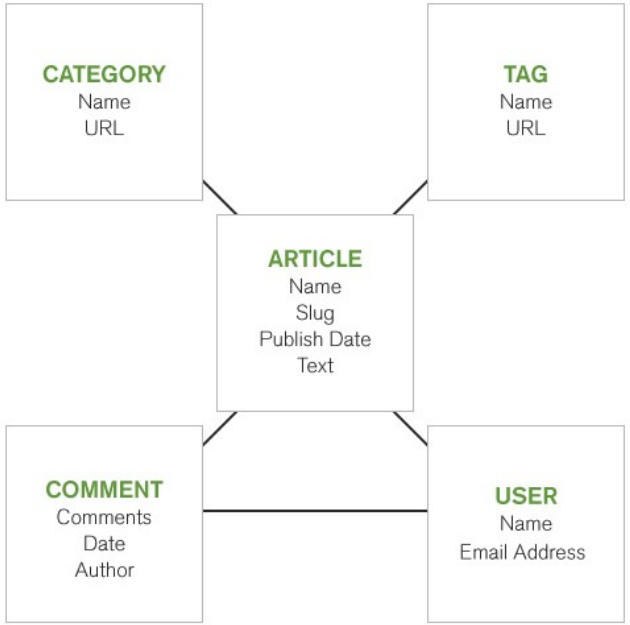
What about **transactions?**

MongoDB has transactions

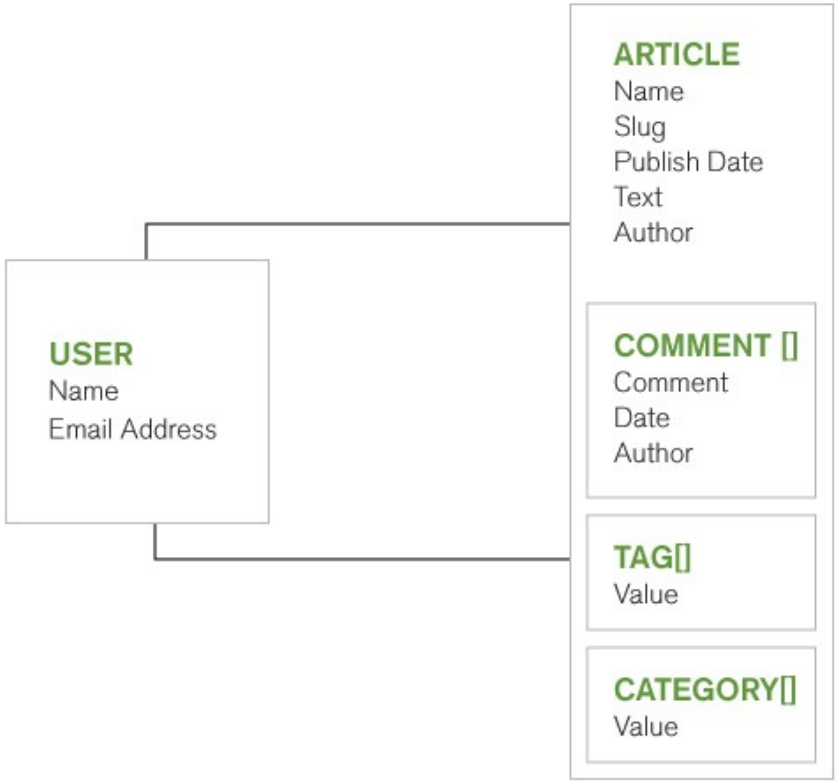
across documents, collections, shards, etc.

What about transactions?

Relational



MongoDB



What about **transactions?**

Lots of transactions?

Rethink your schema

articles

```
{
  _id: ObjectId("aaaa"),
  authorId: ObjectId("3cd4"),
  authorUserName: "magicTony",
  title: "New Trick",
  body: "I'm working on a new trick with dice.",
  comments: [
    { authorId: ObjectId("ab12"),
      authorUserName: "sheeri",
      body: "I can't wait to see it!"
    },
    { authorId: ObjectId("ab12"),
      authorUserName: "sheeri",
      body: "What are you calling the trick?"
    }
  ]
}
```

articles

```
{
  _id: ObjectId("aaaa"),
  authorId: ObjectId("3cd4"),
  authorUserName: "magicTony",
  title: "New Trick",
  body: "I'm working on a new trick with dice.",
  comments: [
    { authorId: ObjectId("ab12"),
      authorUserName: "sheeri",
      body: "I can't wait to see it!"
    },
    { authorId: ObjectId("ab12"),
      authorUserName: "sheeri",
      body: "What are you calling the trick?"
    },
    { authorId: ObjectId("ab12"),
      authorUserName: "sheeri",
      body: "Saw the trick, it was fantastic!",
      ts: 2019-10-02T05:31:19.968+00:00
    },
  ]
}
```

Data that is **accessed together**

should be **stored together**

Data that is **accessed together**

should be **stored together**

No downtime

seamless migrations

Change strings to dates

Change strings to dates

Code application to handle

strings and dates

Change strings to dates

Code application to handle
strings and dates

New data stored as dates

Change strings to dates

Code application to handle
strings and dates

New data stored as dates

update documents
one at a time

articles

```
{
  _id: ObjectId("aaaa"),
  authorId: ObjectId("3cd4"),
  authorUserName: "magicTony",
  title: "New Trick",
  body: "I'm working on a new trick with dice.",
  comments: [
    { authorId: ObjectId("ab12"),
      authorUserName: "sheeri",
      body: "I can't wait to see it!"
    },
    { authorId: ObjectId("ab12"),
      authorUserName: "sheeri",
      body: "What are you calling the trick?"
    },
    { authorId: ObjectId("ab12"),
      authorUserName: "sheeri",
      body: "Saw the trick, it was fantastic!",
      ts: 2019-10-02T05:31:19.968+00:00
    },
  ]
}
```

16Mb document size limit

16Mb document size limit

Hot documents

Activity hot spots

16Mb document size limit

Hot documents

Activity hot spots

Embed = **fast access**

16Mb document size limit

Hot documents

Activity hot spots

Embed = **fast access**

Large docs use

more memory

articles

```
{
  _id: ObjectId("aaaa"),
  authorId: ObjectId("3cd4"),
  authorUserName: "magicTony",
  title: "New Trick",
  body: "I'm working on a new trick with dice.",
  comments: [
    { authorId: ObjectId("ab12"),
      authorUserName: "sheeri",
      body: "I can't wait to see it!"
    },
    { authorId: ObjectId("ab12"),
      authorUserName: "sheeri",
      body: "What are you calling the trick?"
    },
    { authorId: ObjectId("ab12"),
      authorUserName: "sheeri",
      body: "Saw the trick, it was fantastic!",
      ts: 2019-10-02T05:31:19.968+00:00
    },
  ]
}
```

articles

```
{
  _id: ObjectId("aaaa"),
  authorId: ObjectId("3cd4"),
  authorUserName: "magicTony",
  title: "New Trick",
  body: "I'm working on a new trick with dice.",
  comments_ref: [ ObjectId("ef89"),
                  ObjectId("98ef"),
                  ObjectId("e98f")  ]
}
```

articles

```
{
  _id: ObjectId("aaaa"),
  authorId: ObjectId("3cd4"),
  authorUserName: "magicTony",
  title: "New Trick",
  body: "I'm working on a new trick with dice.",
  comments_ref: [ ObjectId("ef89"),
                  ObjectId("98ef"),
                  ObjectId("e98f")  ]
}
```

---

comments

```
{ _id: ObjectId("ef89"),
  authorId: ObjectId("ab12"),
  authorUserName: "sheeri",
  body: "I can't wait to see it!"
},
```

```
{ _id: ObjectId("98ef"),
  authorId: ObjectId("ab12"),
  authorUserName: "sheeri",
  body: "What are you calling the trick?"
},
```

```
{ _id: ObjectId("e98f"),
  authorId: ObjectId("ab12"),
  authorUserName: "sheeri",
  body: "Saw the trick, it was fantastic!"
  ts: 2019-10-02T05:31:19.968+00:00
},
```

articles

```
{
  _id: ObjectId("aaaa"),
  authorId: ObjectId("3cd4"),
  authorUserName: "magicTony",
  title: "New Trick",
  body: "I'm working on a new trick with dice.",
  comments_ref: [ ObjectId("ef89"),
                  ObjectId("98ef"),
                  ObjectId("e98f")  ]
  comments: [
    { authorId: ObjectId("ab12"),
      authorUserName: "sheeri",
      body: "I can't wait to see it!"
    },
    { authorId: ObjectId("ab12"),
      authorUserName: "sheeri",
      body: "What are you calling the trick?"
    },
    { authorId: ObjectId("ab12"),
      authorUserName: "sheeri",
      body: "Saw the trick, it was fantastic!",
      ts: 2019-10-02T05:31:19.968+00:00
    },
  ]
}
```

articles

```
{
  _id: ObjectId("aaaa"),
  authorId: ObjectId("3cd4"),
  authorUserName: "magicTony",
  title: "New Trick",
  body: "I'm working on a new trick with dice.",
  comments: [
    { authorId: ObjectId("ab12"),
      authorUserName: "sheeri",
      body: "What are you calling the trick?"
    },
    { authorId: ObjectId("ab12"),
      authorUserName: "sheeri",
      body: "Saw the trick, it was fantastic!",
      ts: 2019-10-02T05:31:19.968+00:00
    },
  ]
}
```

articles

```
{
  _id: ObjectId("aaaa"),
  authorId: ObjectId("3cd4"),
  authorUserName: "magicTony",
  title: "New Trick",
  body: "I'm working on a new trick with dice.",
  comments: [
    { authorId: ObjectId("ab12"),
      authorUserName: "sheeri",
      body: "What are you calling the trick?"
    },
    { authorId: ObjectId("ab12"),
      authorUserName: "sheeri",
      body: "Saw the trick, it was fantastic!",
      ts: 2019-10-02T05:31:19.968+00:00
    },
  ]
}
```

comments

```
{ _id: ObjectId("ef89"),
  articleId: ObjectId("aaaa"),
  authorId: ObjectId("ab12"),
  authorUserName: "sheeri",
  body: "I can't wait to see it!"
},
```

```
{ _id: ObjectId("e98f"),
  articleId: ObjectId("aaaa"),
  authorId: ObjectId("ab12"),
  authorUserName: "sheeri",
  body: "Saw the trick, it was fantastic!"
  ts: 2019-10-02T05:31:19.968+00:00
},
```

```
{ _id: ObjectId("98ef"),
  articleId: ObjectId("aaaa"),
  authorId: ObjectId("ab12"),
  authorUserName: "sheeri",
  body: "What are you calling the trick?"
},
```

articles

```
{
  _id: ObjectId("aaaa"),
  authorId: ObjectId("3cd4"),
  authorUserName: "magicTony",
  title: "New Trick",
  body: "I'm working on a new trick with dice.",
  comments: [
    { authorId: ObjectId("ab12"),
      authorUserName: "sheeri",
      body: "What are you calling the trick?"
    },
    { authorId: ObjectId("ab12"),
      authorUserName: "sheeri",
      body: "Saw the trick, it was fantastic!",
      ts: 2019-10-02T05:31:19.968+00:00
    },
  ]
}
```

subset

comments

```
{ _id: ObjectId("ef89"),
  articleId: ObjectId("aaaa"),
  authorId: ObjectId("ab12"),
  authorUserName: "sheeri",
  body: "I can't wait to see it!"
},
```

```
{ _id: ObjectId("e98f"),
  articleId: ObjectId("aaaa"),
  authorId: ObjectId("ab12"),
  authorUserName: "sheeri",
  body: "Saw the trick, it was fantastic!"
  ts: 2019-10-02T05:31:19.968+00:00
},
```

```
{ _id: ObjectId("98ef"),
  articleId: ObjectId("aaaa"),
  authorId: ObjectId("ab12"),
  authorUserName: "sheeri",
  body: "What are you calling the trick?"
},
```

articles

```
{ authorId: ObjectId("3cd4"),
  authorUserName: "magicTony",
  title: "New Trick",
  body: "I'm working on a new trick with dice.",
  comments: [
    { authorId: ObjectId("ab12"),
      authorUserName: "sheeri",
      body: "I can't wait to see it!"
    },
    { authorId: ObjectId("ab12"),
      authorUserName: "sheeri",
      body: "What are you calling the trick?"
    },
  has_extras: true,
  ]
}
```

articles

```
{ authorId: ObjectId("3cd4"),
  authorUserName: "magicTony",
  title: "New Trick",
  body: "I'm working on a new trick with dice.",
  comments: [
    { authorId: ObjectId("ab12"),
      authorUserName: "sheeri",
      body: "I can't wait to see it!"
    },
    { authorId: ObjectId("ab12"),
      authorUserName: "sheeri",
      body: "What are you calling the trick?"
    },
  has_extras: true,
  ]
}
```

---

overflow_comments

```
{ _id: ObjectId("e98f"),
  articleId: ObjectId("aaaa"),
  authorId: ObjectId("ab12"),
  authorUserName: "sheeri",
  body: "Saw the trick, it was fantastic!"
  ts: 2019-10-02T05:31:19.968+00:00
},
```

articles

```
{ authorId: ObjectId("3cd4"),
  authorUserName: "magicTony",
  title: "New Trick",
  body: "I'm working on a new trick with dice.",
  comments: [
    { authorId: ObjectId("ab12"),
      authorUserName: "sheeri",
      body: "I can't wait to see it!"
    },
    { authorId: ObjectId("ab12"),
      authorUserName: "sheeri",
      body: "What are you calling the trick?"
    },
  has_extras: true,
  ]
}
```

outlier

---

overflow_comments

```
{ _id: ObjectId("e98f"),
  articleId: ObjectId("aaaa"),
  authorId: ObjectId("ab12"),
  authorUserName: "sheeri",
  body: "Saw the trick, it was fantastic!"
  ts: 2019-10-02T05:31:19.968+00:00
},
```

Building a MongoDB schema

Building a MongoDB schema

Embed if you can

1:few

Building a MongoDB

schema

Embed if you can

1:few

Array of references

for separate data

1:many

Building a MongoDB

schema

Embed if you can

1:few

Array of references

for separate data

1:many

Reference for unbounded arrays

1:zillion

Schema Patterns

Polymorphic

flexible schema

Schema Patterns

Polymorphic

flexible schema

extended reference

not just _id

Schema Patterns

Polymorphic
flexible schema

extended reference
not just _id

subset
part of data is duplicated by embedding

Schema Patterns

Polymorphic
flexible schema

extended reference
not just _id

subset
part of data is duplicated by embedding

outlier
a few documents will overflow

Schema Patterns

Polymorphic
flexible schema

extended reference
not just _id

subset
part of data is duplicated by embedding

outlier
a few documents will overflow

Building with Patterns blog series:
https://www.mongodb.com/blog/post/building-with-patterns-a-summary

mongoDB

From RDBMS to MongoDB

Documents do not need to have

identical fields

From RDBMS to MongoDB

Documents do not need to have
identical fields

Data that is **accessed together**
should be **stored together**

From RDBMS to MongoDB

Documents do not need to have
identical fields

Data that is **accessed together**
should be **stored together**

Rethink if you have lots of
references or **transactions**

Credit, Thanks and Links

Asya Kamsky
Evin Roesle
Nick Larew
Aly Cabral
Wikipedia

Thinking in Documents
https://www.mongodb.com/blog/post/thinking-documents-part-1

6 Rules of Thumb for MongoDB Schema Design
https://www.mongodb.com/blog/post/6-rules-of-thumb-for-mongodb-schema-design-part-1

Building with Patterns blog series:
https://www.mongodb.com/blog/post/building-with-patterns-a-summary

Q&A