DATABASE DESIGN FOR DOCUMENT DATABASES

AMR ELHELW

TECH VAULT

Blog Application

Action	Type	Information	Frequency	Priority
Submit a new article	Write	author, text	10 per day	High
Submit a comment on an article	Write	user, text	1,000 per day (100 per article)	Medium
View an article	Read	article id, text, comments	1,000,000 per day	High

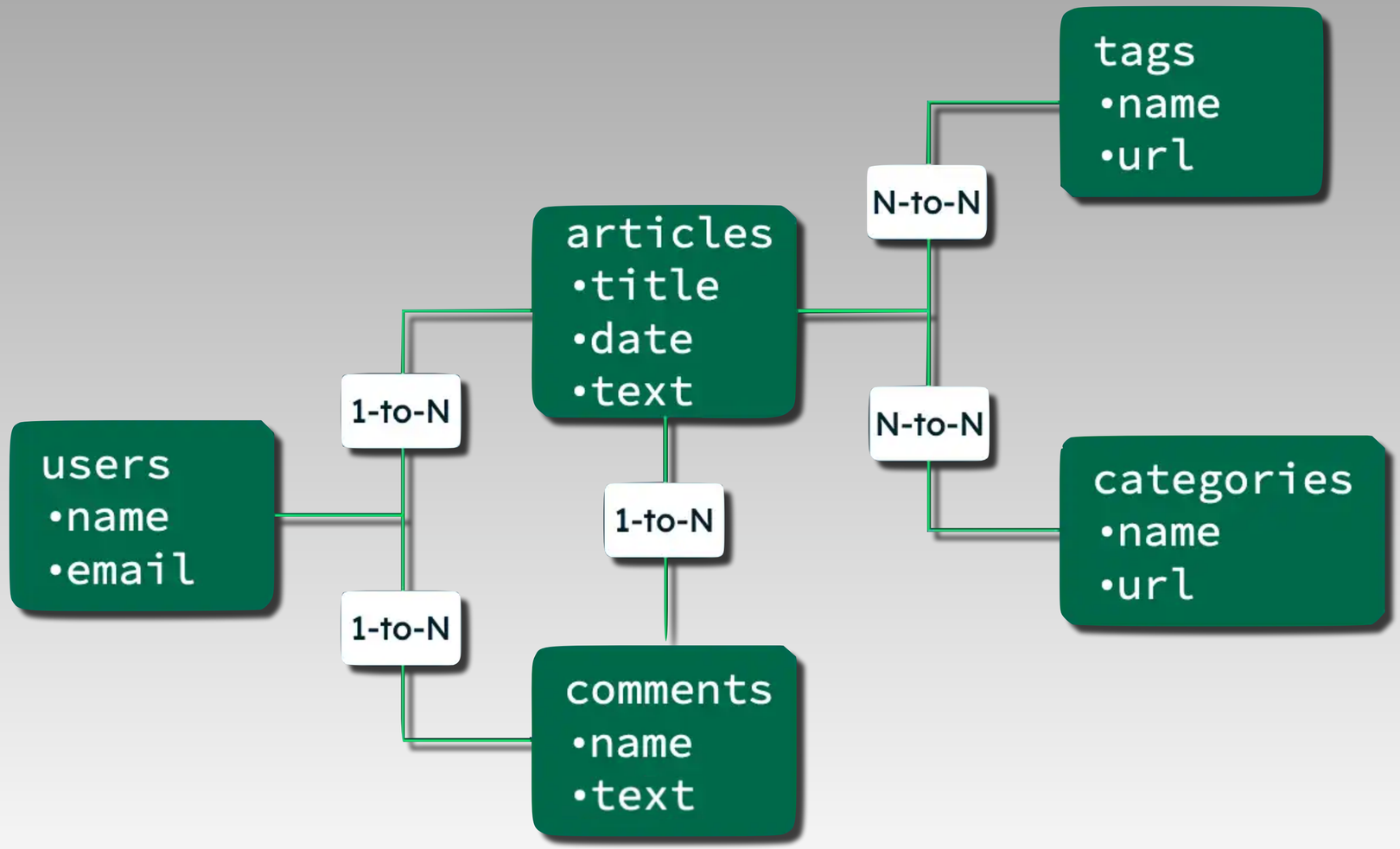
... and more!

Blog Application

- **User/author** - Name, email, ...

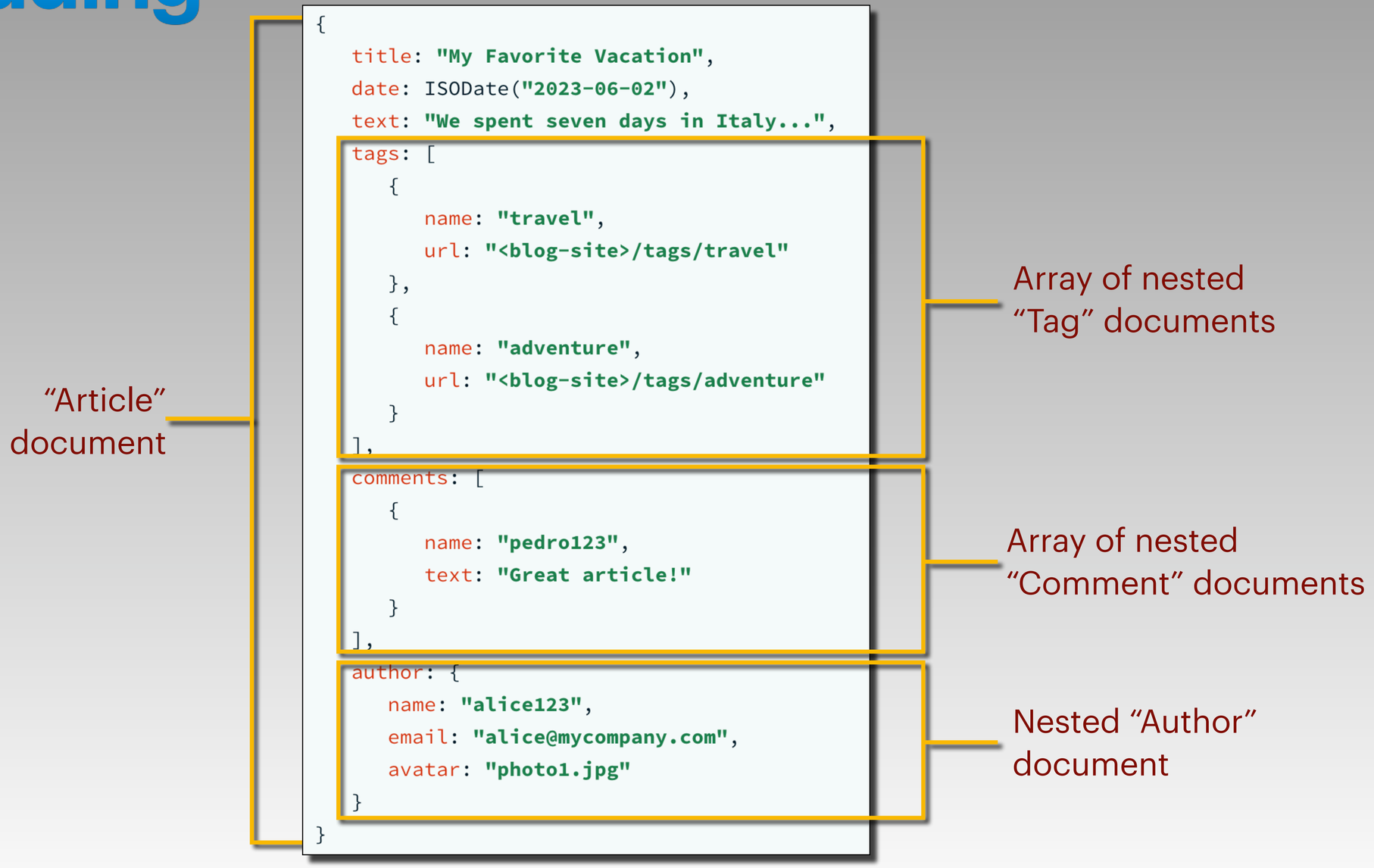
- **Article** - Title, text, date, ...
  - **Tags** - Name, url, ...
  - **Categories** - Name, url, ...
  - **Comments** - Text, name, ...

Blog Application



	Normalization	Denormalization
Document Database Terminology	*"Referencing"*	*"Embedding"*
Representation of related data	Multiple collections with "references"	Nested documents
Updating	Easy to update, since there are no duplicates	Slow updates for duplicated info
Reading	Slower since we need to join multiple collections	Faster reads, since everything is stored together

Embedding



Embedding

- **Pros**
  - Retrieve all article information in a single query (avoid joins)
  - Update all information for a given article in a single atomic operation

- **Cons**
  - Document size limit (16-MB limit for a single document in MongoDB)
  - Duplication - e.g. same author information can be repeated for different articles
    - Updating author information becomes more expensive
    - Potential for inconsistency

Referencing

"Authors" collection

```
{
    _id: 987,
    name: "alice123",
    email: "alice@mycompany.com",
    avatar: "photo1.jpg"
}
```

"Articles" collection

```
{
    title: "My Favorite Vacation",
    date: ISODate("2023-06-02"),
    text: "We spent seven days in Italy...",
    authorId: 987,
    tags: [
        {
            name: "travel",
            url: "<blog-site>/tags/travel"
        },
        {
            name: "adventure",
            url: "<blog-site>/tags/adventure"
        }
    ],
    comments: [
        {
            name: "pedro345",
            text: "Great article!"
        }
    ]
}
```

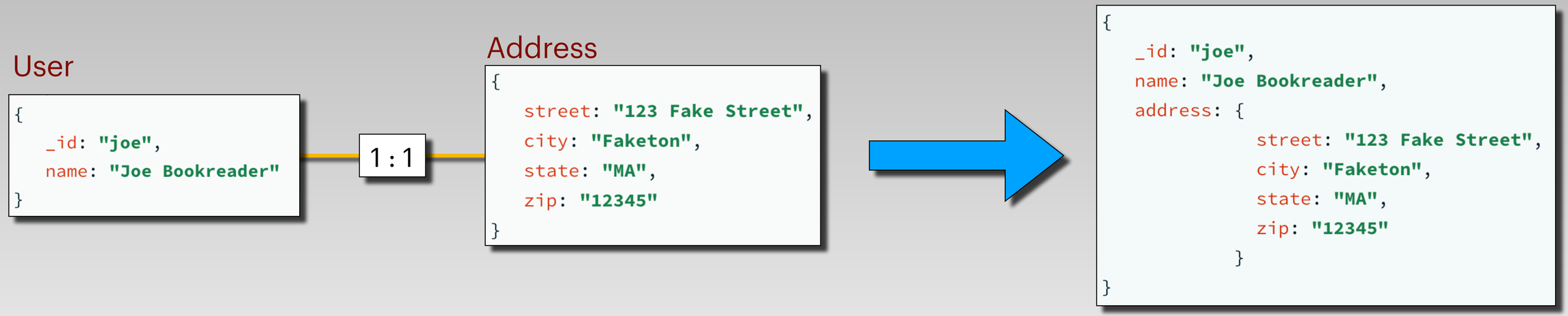
Referencing

- **Pros**
  - No duplication of author info
    - Easy to update, no risk of inconsistency
  - Smaller article document
    - Easier/faster to read if we don’t need author info

- **Cons**
  - Need to join collections if we want to read article + author —> slow
  - Adding new article with new author requires several writes

When to use embedding vs. referencing?

One-to-one relationships



When to use embedding vs. referencing?

One-to-many relationships

Articles

```
{
   title: “Article 1",
   date: ISODate("2023-06-02"),
   text: “Some text...",
   author: {
      name: “John Smith",
      email: “js@mycompany.com",
      avatar: "photo1.jpg"
   }
},
{
   title: “Article 2",
   date: ISODate("2024-04-16"),
   text: “More text...",
   author: {
      name: “John Smith",
      email: “js@mycompany.com",
      avatar: "photo1.jpg"
   }
}
```

Authors

```
{
   _id: 987
   name: “John Smith",
   email: “js@mycompany.com",
   avatar: "photo1.jpg"
}
```

Articles

```
{
   title: “Article 1",
   date: ISODate("2023-06-02"),
   text: “Some text...",
   author_id: 987
},
{
   title: “Article 2",
   date: ISODate("2024-04-16"),
   text: “More text...",
   author_id: 987
}
```

Authors

```
{
   name: “John Smith",
   email: “js@mycompany.com",
   avatar: “photo1.jpg”,
   articles: [
     {
        title: “Article 1",
        date: ISODate("2023-06-02"),
        text: “Some text..."
     },
     {
        title: “Article 2",
        date: ISODate("2024-04-16"),
        text: “More text..."
     }
   ]
}
```

Amr Elhelw's TECH VAULT

When to use embedding vs. referencing?

Many-to-many relationships

Courses

```
{
   _id: “CS101”
   name: “Data Structures",
   instructor: "Tom"
},
{
   _id: “ECON101”
   name: “Economics",
   instructor: “Alice"
},
...
```

Students

```
{
   name: “John Smith",
   email: “js@mycompany.com",
   avatar: “photo1.jpg”,
   courses: [
     {
        course_id: “CS101”,
        grade: “B+”
     },
     {
        course_id: “ECON101”,
        grade: “A”
     }
   ]
},
...
```