MongoDB CRUD

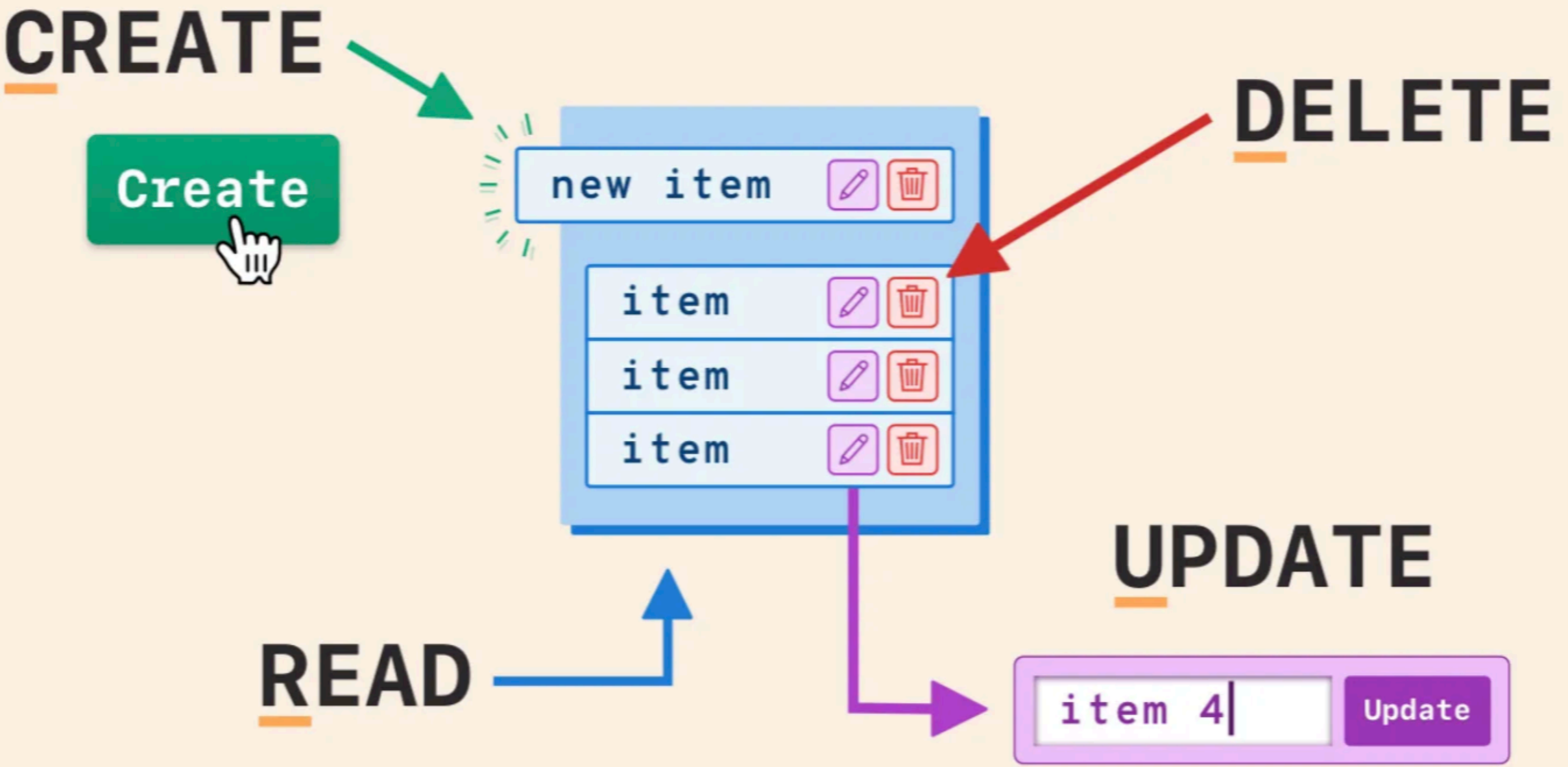
Data literacy

NIFR 2024

MongoDB CRUD & Data modelling

Data literacy

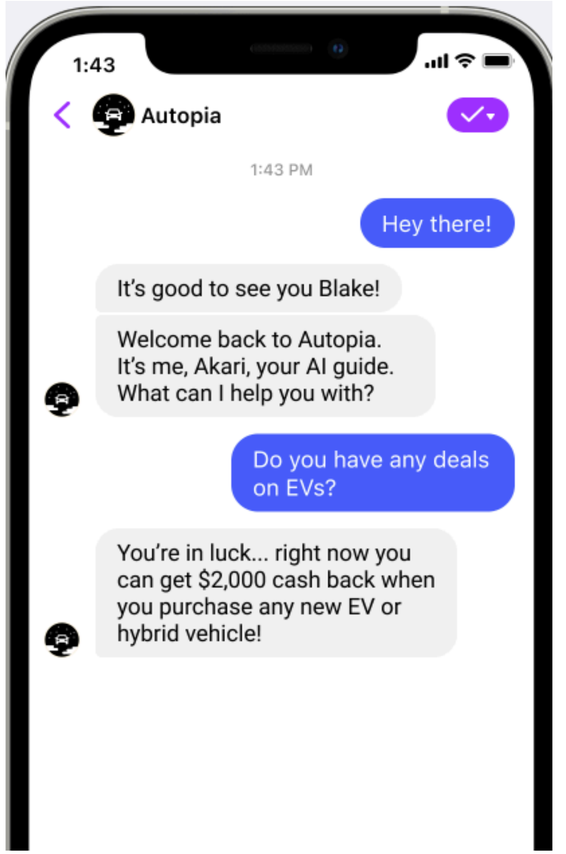
- CRUD applications
- Data modelling with MongoDB
- CRUD with mongoDB
- The mongoDB shell

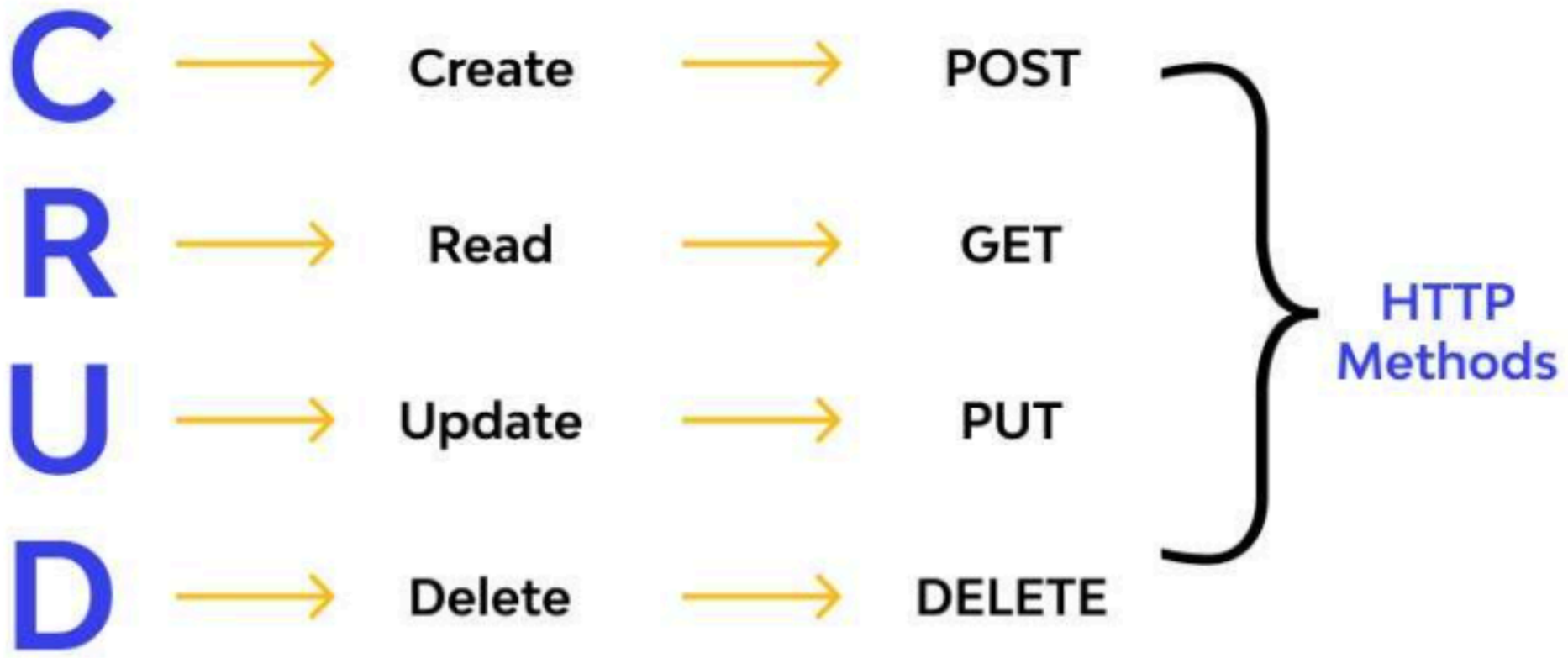

CREATE
Create
new item
item
item
item
DELETE
UPDATE
item 4
Update
READ

CRUD applications

Everyday example: Facebook messenger

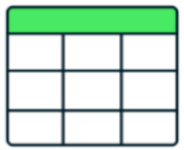
- Writing a new message corresponds to creating a data entity
- Fetching messages corresponds to reading a collection of data entities
- Editing an existing message corresponds to updating a data entity
- Deleting a message corresponds to deleting a data entity




C
Create
POST
R
Read
GET
U
Update
PUT
D
Delete
DELETE
HTTP
Methods

Data modelling with MongoDB

RDBMS vs NoSQL (Document)



Relational Database

MongoDB

User table

ID	first_name	last_name	cell	city
1	Leslie	Yepp	8125552344	Pawnee

Hobbies table

ID	user_id	hobby
10	1	scrapbooking
11	1	eating waffles
12	1	working

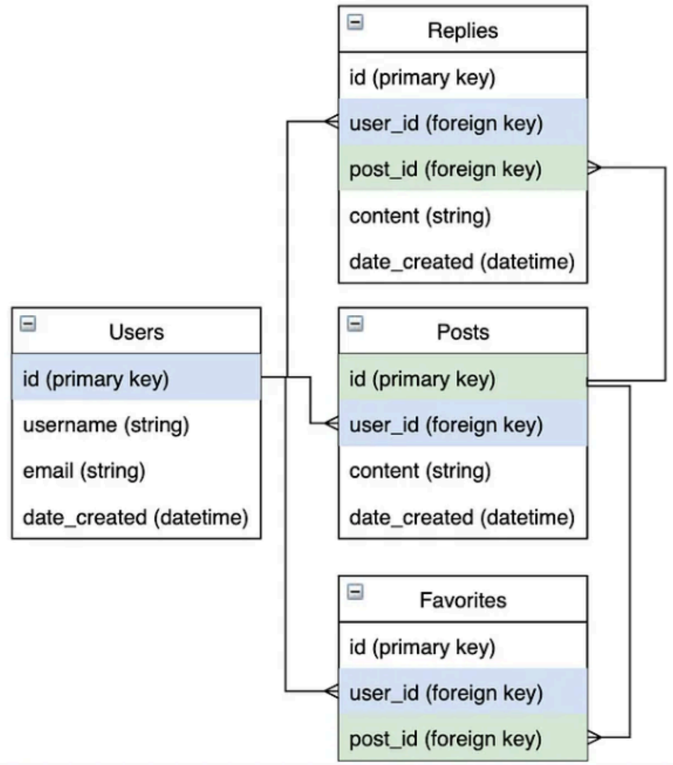
```
{
   "_id": 1,
   "first_name": "Leslie",
   "last_name": "Yepp",
   "cell": "8125552344",
   "city": "Pawnee",
   "hobbies": ["scrapbooking", "eating
waffles", "working"]
}
```

- No need for joins
- No need for data normalization

Designing a database schema

Relational database model

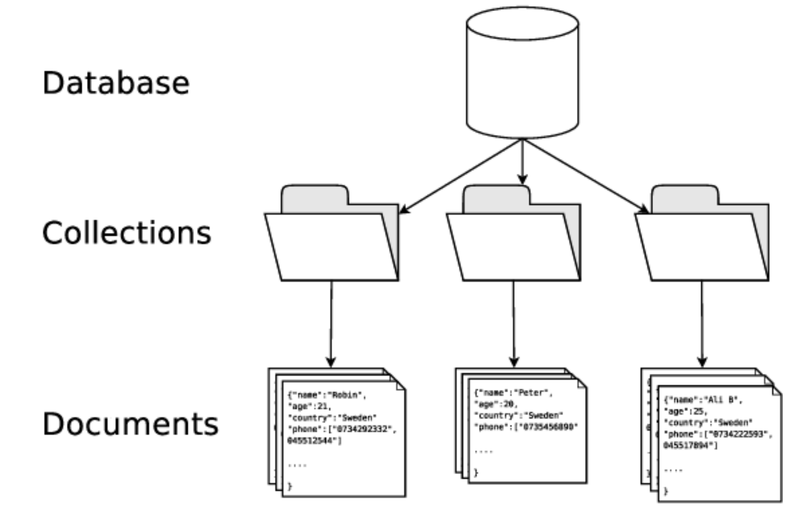
- Define the purpose of the Database (Gather requirements)
- Organise the data in tables and specify a primary key
- Create relationships among the tables
- Normalise the design



Designing a database schema

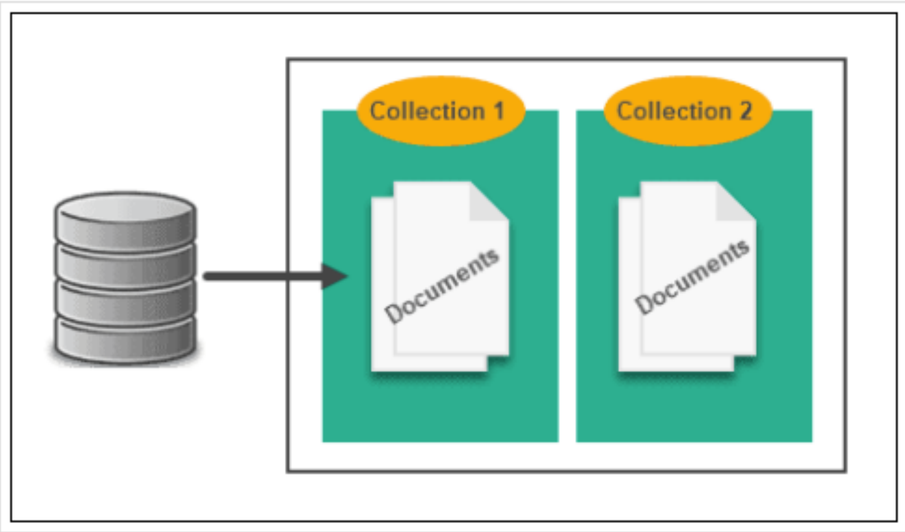
Document oriented database

- Define the purpose of the Database (Gather requirements)
- Identifying the workload pattern
- Identifying relationships
- Applying design patterns



NoSQL

Document database abstraction



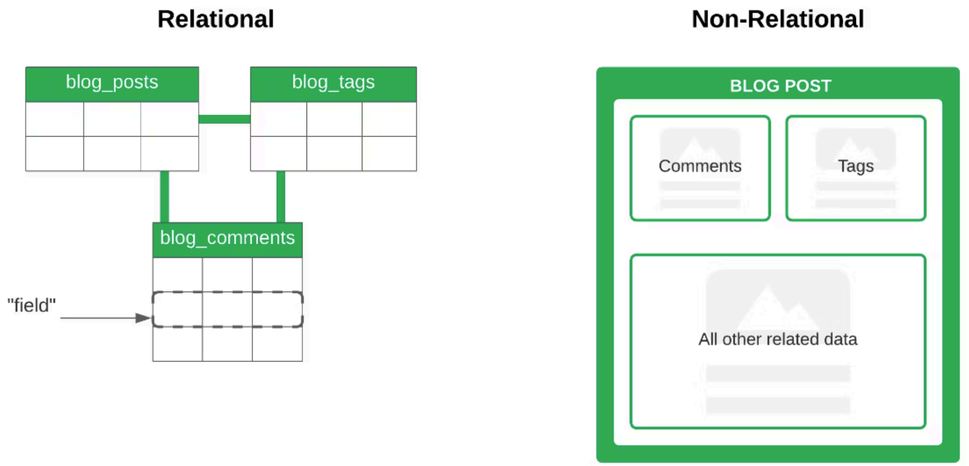

```
{
  "_id": "12345",
  "name": "foo bar",
  "email": "foo@bar.com",
  "address": {
    "street": "123 foo street",
    "city": "some city",
    "state": "some state",
    "zip": "123456"
  },
  "hobbies": ["music", "guitar", "reading"]
}
```

MongoDB: Data that is accessed together should be stored together

Document Database: MongoDB

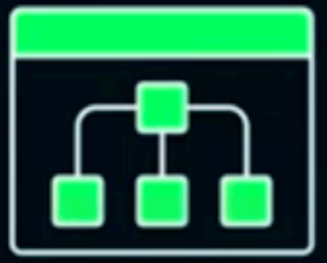
Identifying the workload pattern

- MongoDB has been written and designed with software applications in mind.
- Relational databases was developed conceptually before the modern application scope.
- Entities: Rows / Documents



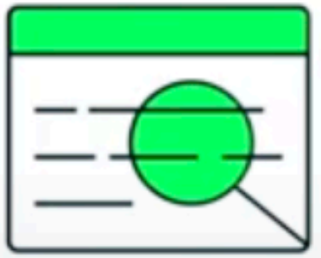
Tabular/
Relational DB

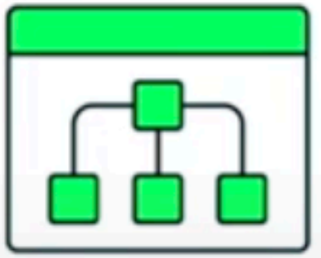
1. Data Model
2. Workload identification



MongoDB

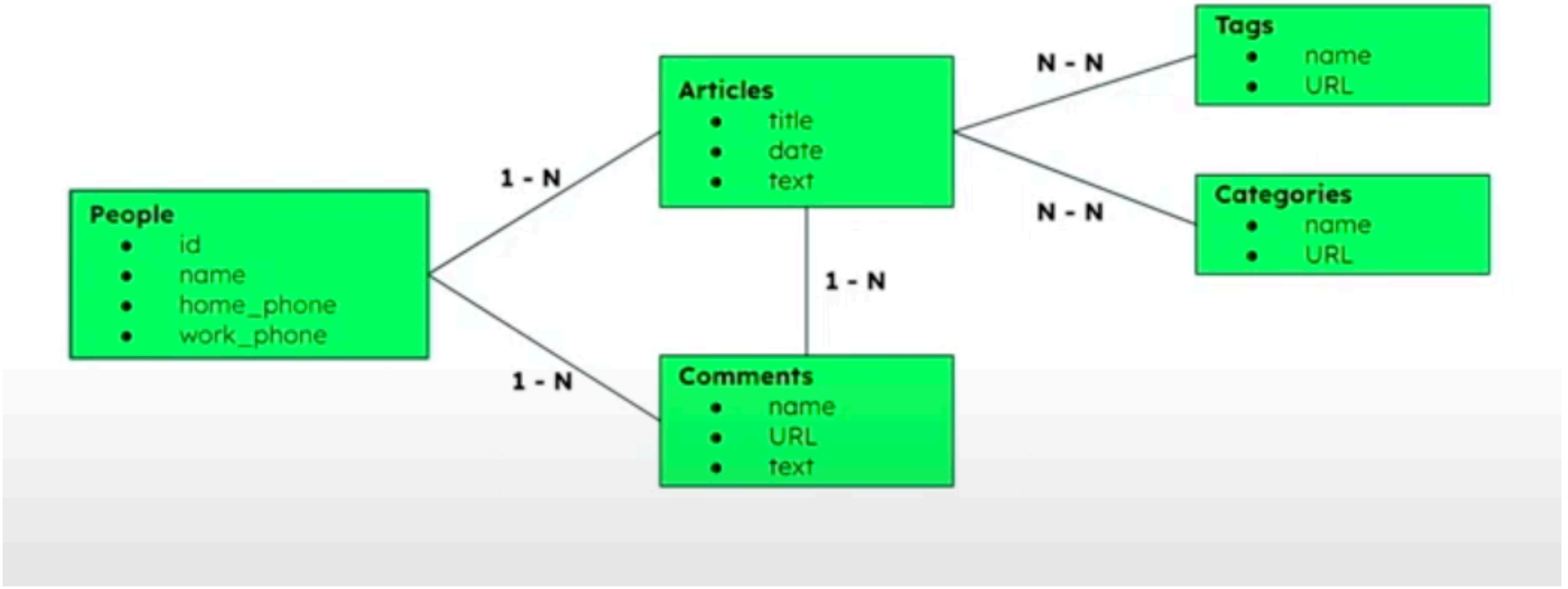
1. Workload identification
2. Data Model





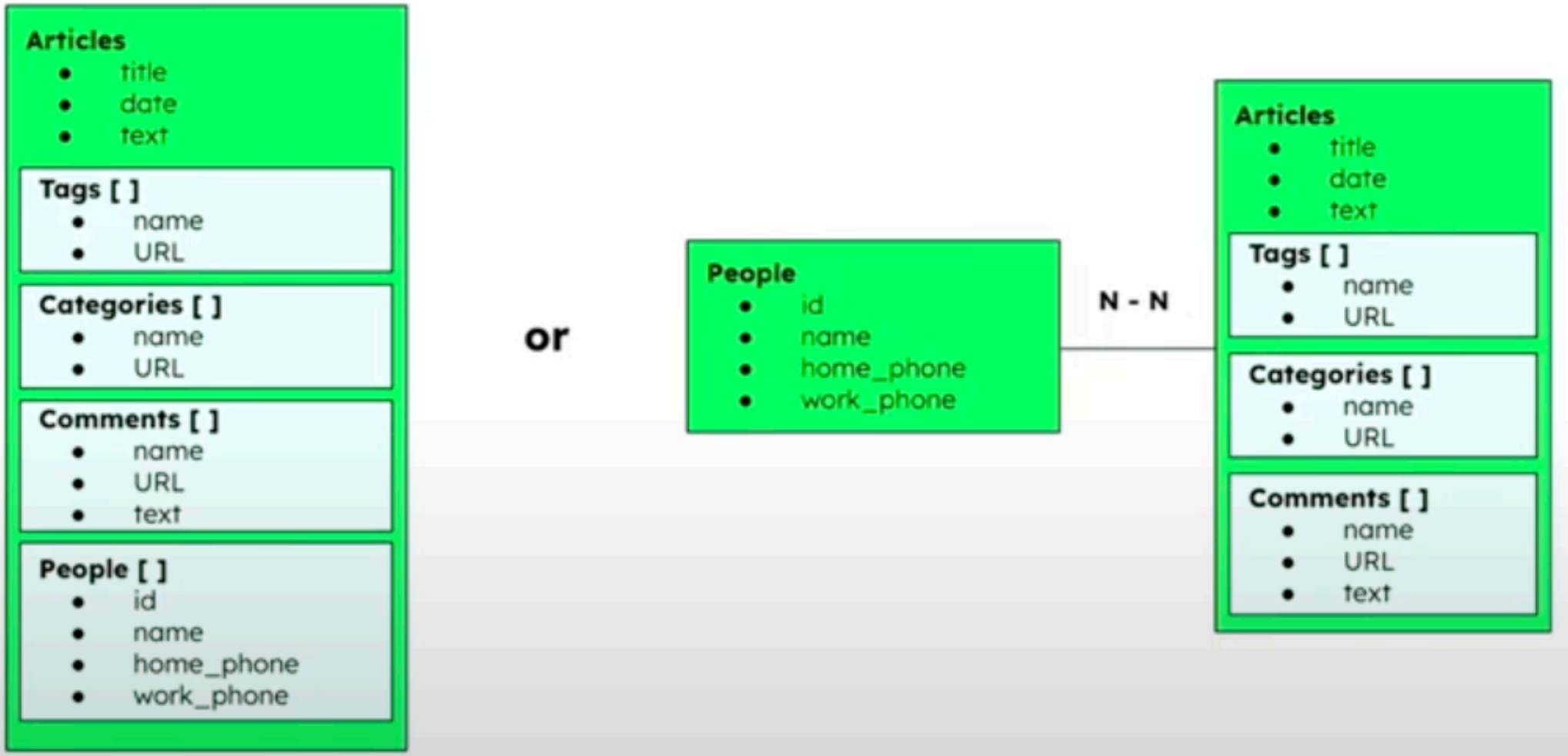
https://youtu.be/BC-NahZ5jsY?si=pK3H9Vpssj42vd-Y

Tabular/Relational Database: 5 Tables



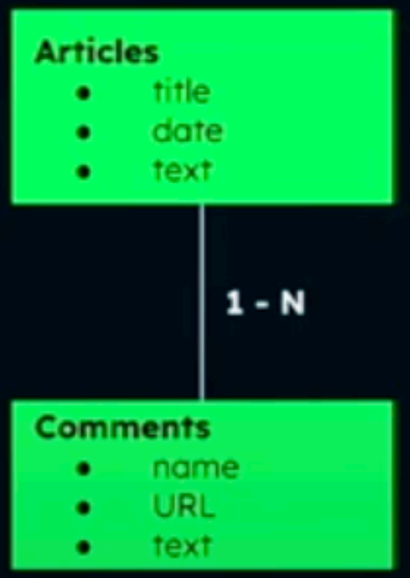
https://youtu.be/BC-NahZ5jsY?si=pK3H9Vpssj42vd-Y

MongoDB: 1 or 2 Collections?

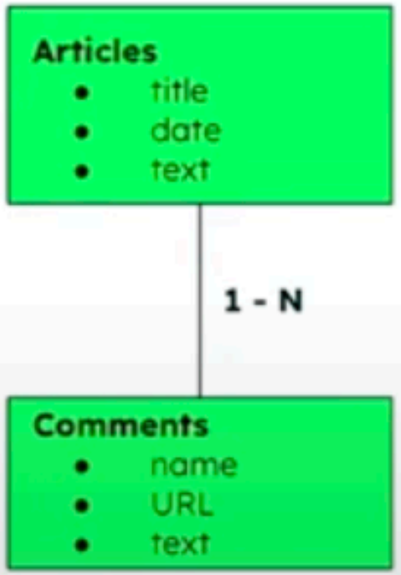


https://youtu.be/BC-NahZ5jsY?si=pK3H9Vpssj42vd-Y

Tabular/Relational

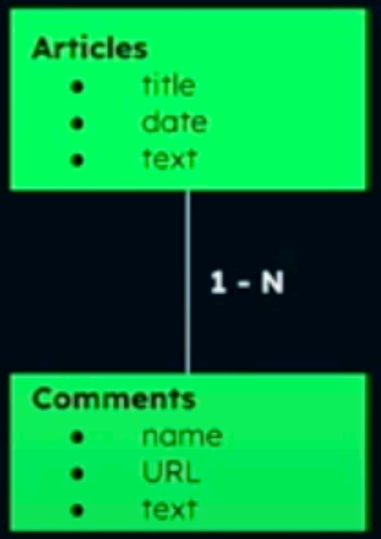


Reference/Link is two collections in MongoDB



https://youtu.be/BC-NahZ5jsY?si=pK3H9Vpssj42vd-Y

Tabular/Relational



Embedding is using one collection in MongoDB

https://youtu.be/BC-NahZ5jsY?si=pK3H9Vpssj42vd-Y

Embedded data

```
{
    "_id": ObjectId("5f6d7f3b5c1234567890abcd"),
    "title": "The Art of MongoDB",
    "genre": "Non-fiction",
    "published_year": 2021,
    "price": 29.99,
    "author": {
        "first_name": "Jane",
        "last_name": "Doe",
        "birth_year": 1980,
        "nationality": "American"
    },
    "reviews": [
        {
            "review_id": ObjectId("5f6d7f3b5c1234567890abce"),
            "user": "reader123",
            "rating": 5,
            "comment": "A comprehensive guide to MongoDB!",
            "date": ISODate("2023-08-15T10:45:00Z")
        },
        {
            "review_id": ObjectId("5f6d7f3b5c1234567890abcf"),
            "user": "tech_guru",
            "rating": 4,
            "comment": "Informative but slightly dense.",
            "date": ISODate("2023-09-01T14:30:00Z")
        }
    ],
    "in_stock": true,
    "tags": ["database", "NoSQL", "MongoDB", "technology"]
}
```

References data

```
{
    "_id": ObjectId("5f6d7f3b5c1234567890abcd"),
    "title": "The Art of MongoDB",
    "genre": "Non-fiction",
    "published_year": 2021,
    "price": 29.99,
    "author_id": ObjectId("5f6d8a9c6c1234567890bcde"),  // Reference to the author
    "review_ids": [
        ObjectId("5f6d8a9c6c1234567890bcdf"),
        ObjectId("5f6d8a9c6c1234567890bce0")
    ],  // References to review documents
    "in_stock": true,
    "tags": ["database", "NoSQL", "MongoDB", "technology"]
}
```

json

```
{
    "_id": ObjectId("5f6d8a9c6c1234567890bcde"),
    "first_name": "Jane",
    "last_name": "Doe",
    "birth_year": 1980,
    "nationality": "American"
}
```

```
{
    "_id": ObjectId("5f6d8a9c6c1234567890bcdf"),
    "user": "reader123",
    "rating": 5,
    "comment": "A comprehensive guide to MongoDB!",
    "date": ISODate("2023-08-15T10:45:00Z"),
    "book_id": ObjectId("5f6d7f3b5c1234567890abcd")
}
```

```
{
    "_id": ObjectId("5f6d8a9c6c1234567890bce0"),
    "user": "tech_guru",
    "rating": 4,
    "comment": "Informative but slightly dense.",
    "date": ISODate("2023-09-01T14:30:00Z"),
    "book_id": ObjectId("5f6d7f3b5c1234567890abcd")
}
```

Referencing or embedding

MongoDB Database design

Reference

- When the “many” side is a huge number
- For integrity on write operations on many-to-many
- When an entity is frequently but not the other

Embedded

- For integrity of read operations
- For integrity of write operations on one-to-one and one-to-many
- For data that is deleted together
- By default

What is huge?

Around 16.777.216 characters

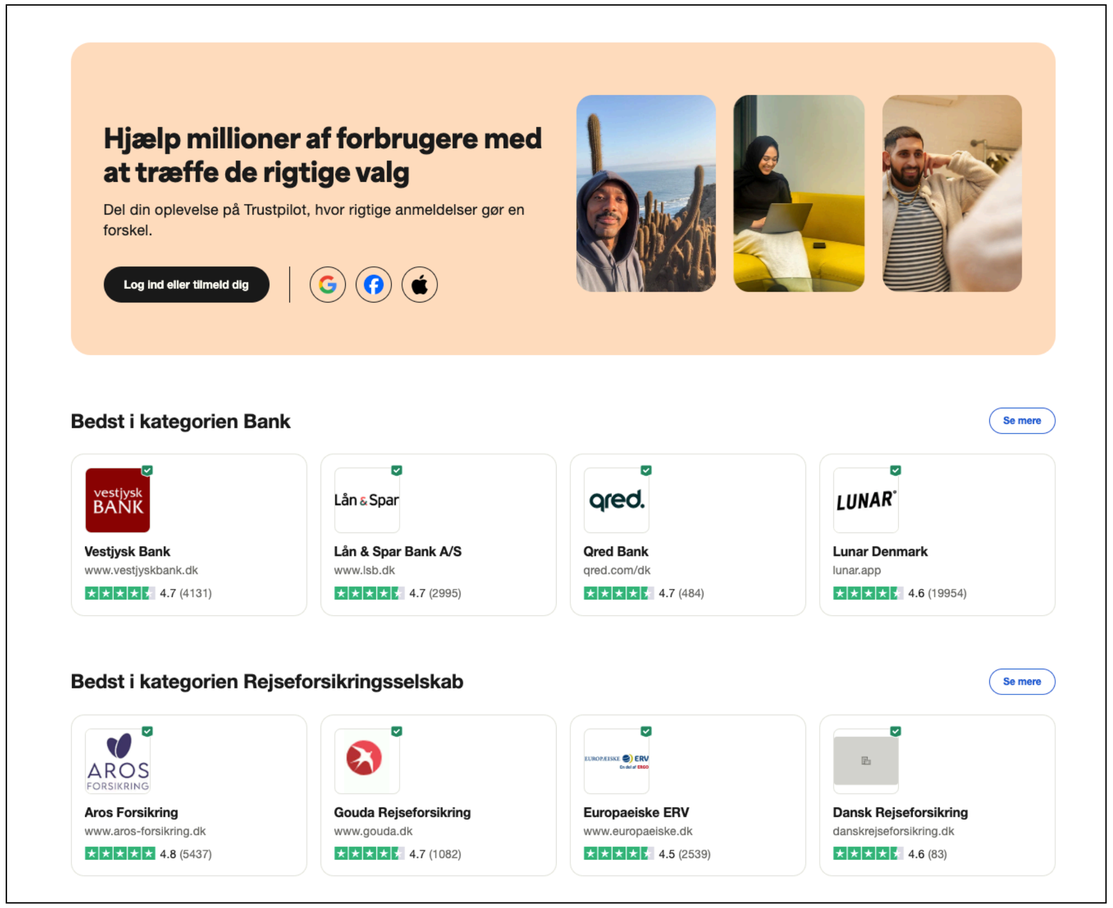
Car in a Tabular/Relational Database (RDBMS)

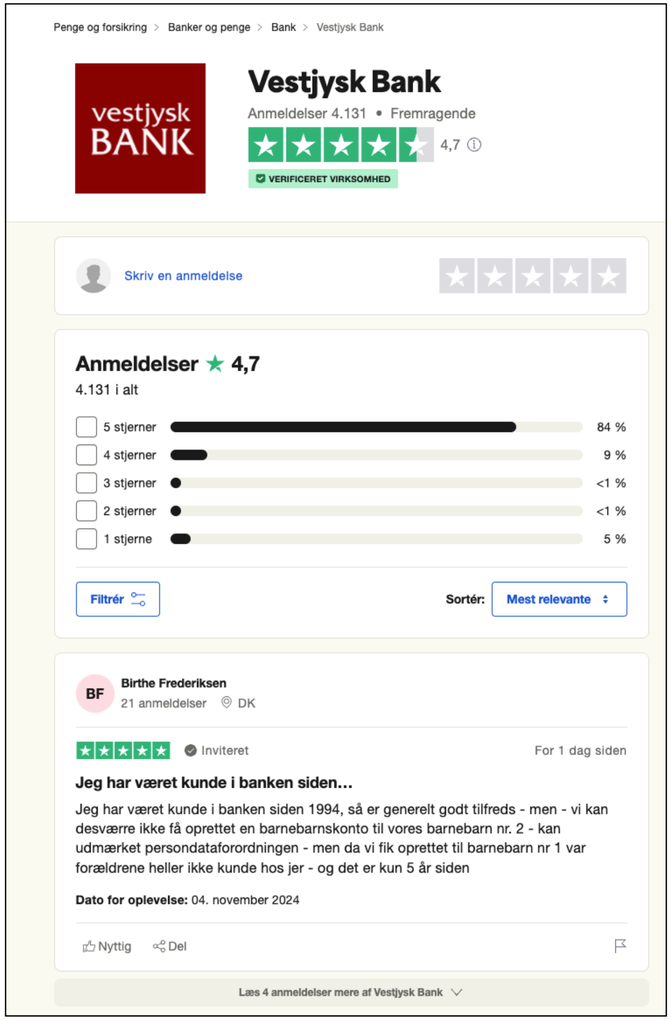
Car in MongoDB

https://youtu.be/BC-NahZ5jsY?si=pK3H9Vpssj42vd-Y

Document Database: MongoDB

Identifying the workload pattern

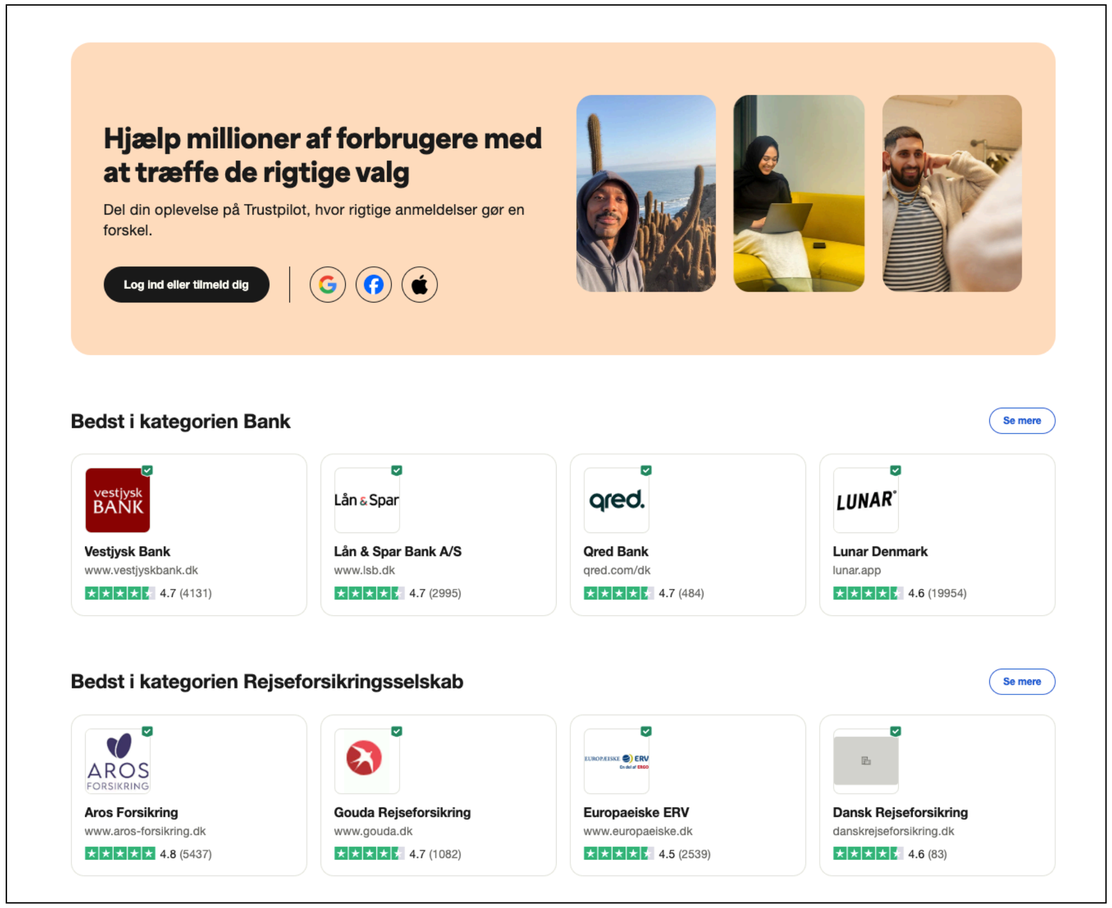


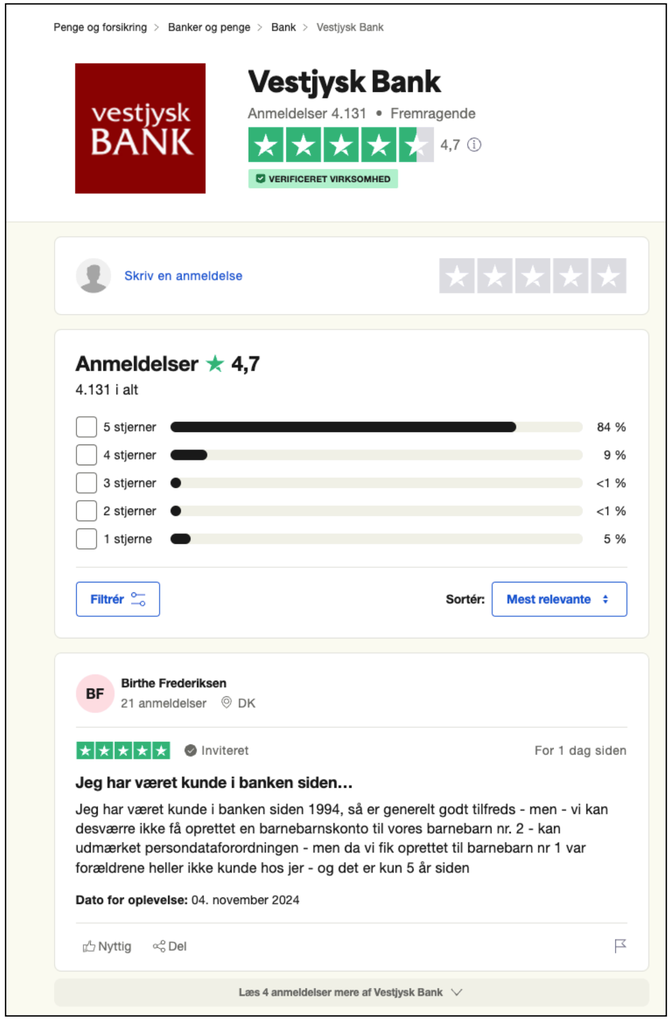


Document Database: MongoDB

Identifying the workload pattern

4131 reviews





4131 reviews

Relational Database: MySQL

Relational data modelling

Bank

- Name of Bank
- No. Of Reviews
- Homepage
- Status
- Image
- Etc.

User

- Name of user
- No. Of reviews made
- Country of origin
- Etc.

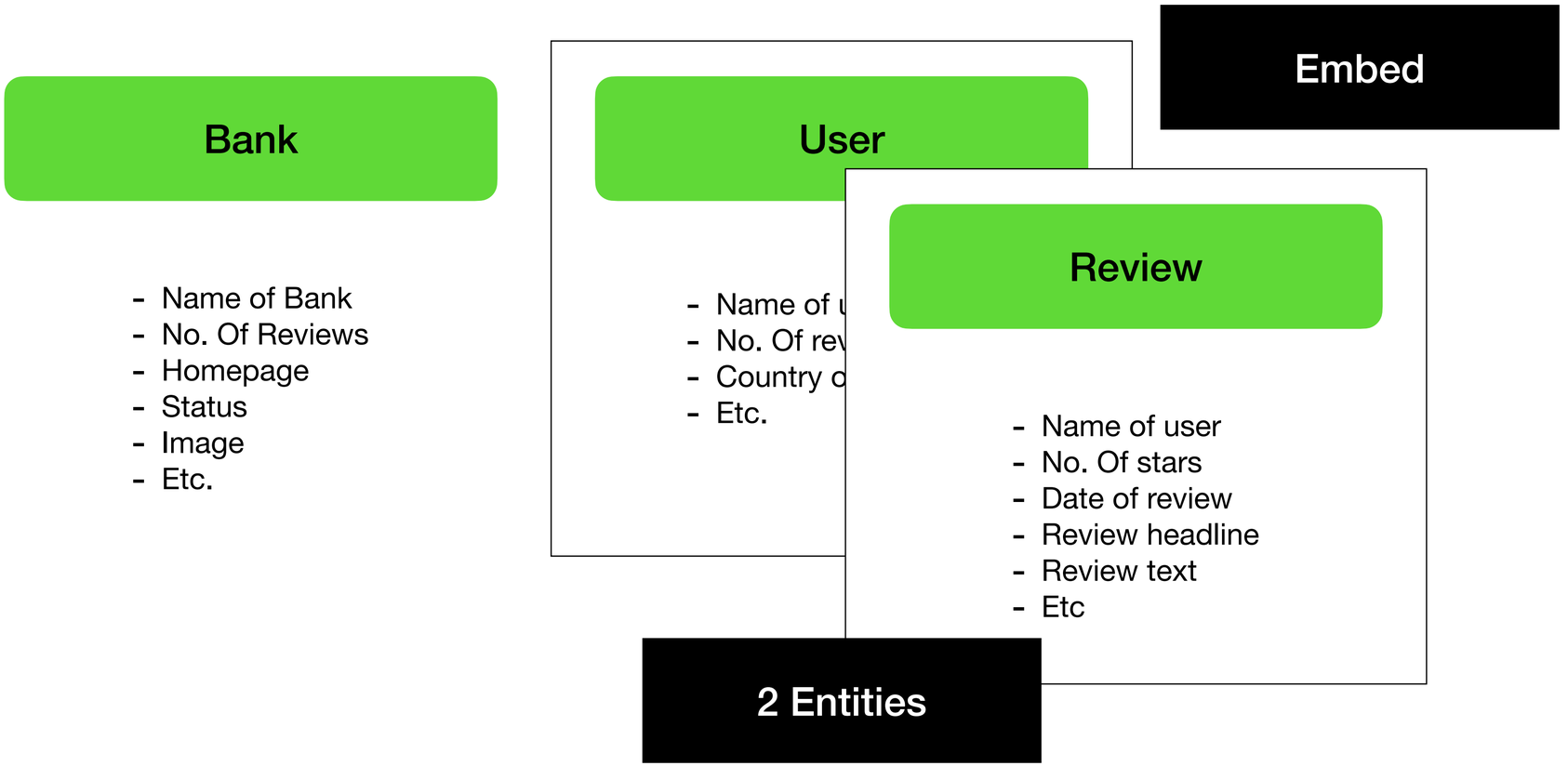
Review

- Name of user
- No. Of stars
- Date of review
- Review headline
- Review text
- Etc

3 Entities?

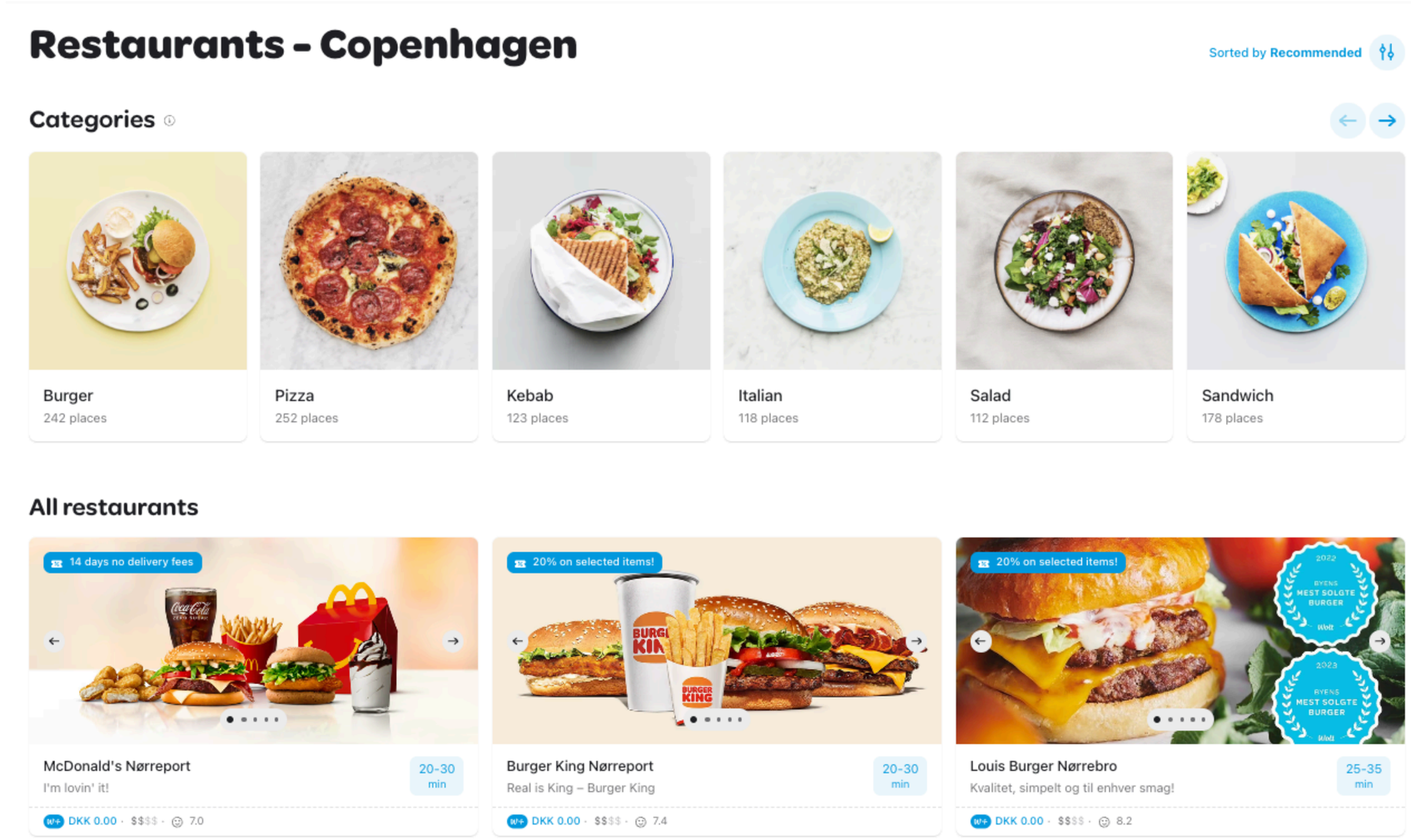
Relational Database: MySQL

Relational data modelling



Case study: Wolt web application

Identifying the workload pattern

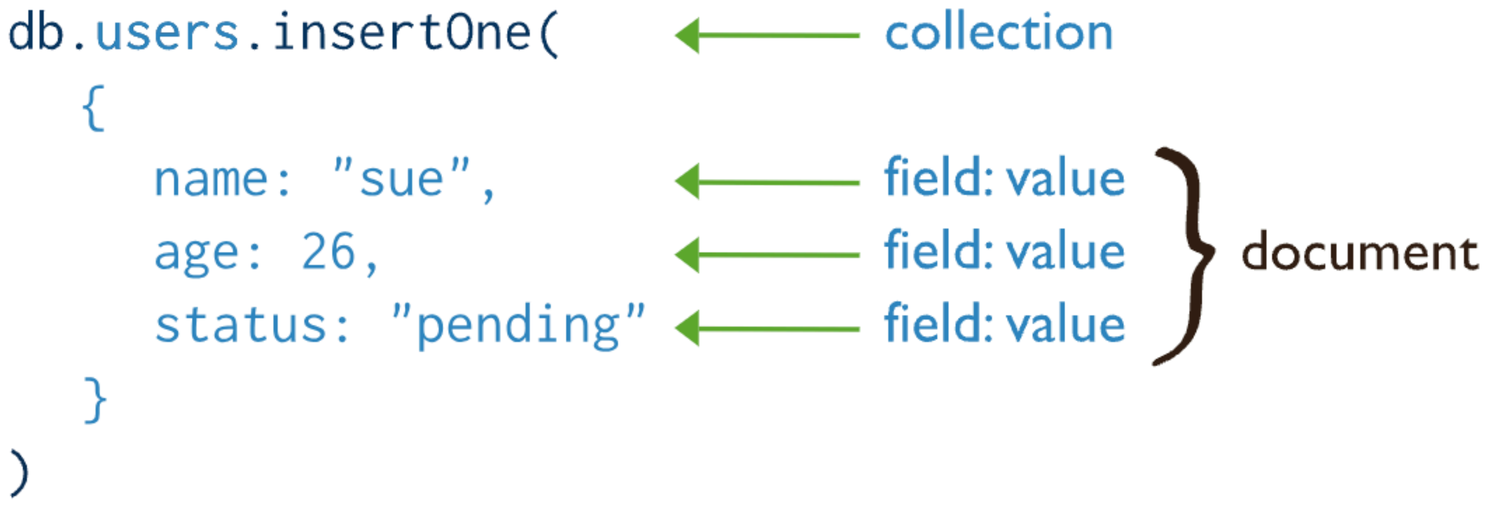


CRUD in mongoDB

https://www.mongodb.com/docs/manual/crud/

MongoDB CRUD

Creating data in MongoDB




```
db.products.insertMany( [
    { _id: 10, item: "large box", qty: 20 },
    { _id: 11, item: "small box", qty: 55 },
    { _id: 12, item: "medium box", qty: 30 }
] );
```

MongoDB CRUD

Finding data in MongoDB

```
db.users.find(
    { age: { $gt: 18 } },
    { name: 1, address: 1 }
).limit(5)
```

← collection
← query criteria
← projection
← cursor modifier

MongoDB CRUD

Finding nested data in MongoDB

```
await db.collection('inventory').insertMany([
  {
    item: 'journal',
    qty: 25,
    size: { h: 14, w: 21, uom: 'cm' },
    status: 'A'
  },
  {
    item: 'notebook',
    qty: 50,
    size: { h: 8.5, w: 11, uom: 'in' },
    status: 'A'
  },
  {
    item: 'paper',
    qty: 100,
    size: { h: 8.5, w: 11, uom: 'in' },
    status: 'D'
  },
  {
    item: 'planner',
    qty: 75,
    size: { h: 22.85, w: 30, uom: 'cm' },
    status: 'D'
  },
  {
    item: 'postcard',
    qty: 45,
    size: { h: 10, w: 15.25, uom: 'cm' },
    status: 'A'
  }
]);
```

```
const cursor = db.collection('inventory').find({
  'size.h': { $lt: 15 }
});
```

MongoDB CRUD

Updating data in MongoDB

```
await db.collection('inventory').updateMany(
  { qty: { $lt: 50 } },
  {
    $set: { 'size.uom': 'in', status: 'P' },
    $currentDate: { lastModified: true }
  }
);
```

Using the mongo shell

Exercise 2

Aggregation: Filter & Group

Exercise 3

MongoDB CRUD

Aggregate functions

```
db.orders.insertMany( [
   { _id: 0, name: "Pepperoni", size: "small", price: 19,
     quantity: 10, date: ISODate( "2021-03-13T08:14:30Z" ) },
   { _id: 1, name: "Pepperoni", size: "medium", price: 20,
     quantity: 20, date : ISODate( "2021-03-13T09:13:24Z" ) },
   { _id: 2, name: "Pepperoni", size: "large", price: 21,
     quantity: 30, date : ISODate( "2021-03-17T09:22:12Z" ) },
   { _id: 3, name: "Cheese", size: "small", price: 12,
     quantity: 15, date : ISODate( "2021-03-13T11:21:39.736Z" ) },
   { _id: 4, name: "Cheese", size: "medium", price: 13,
     quantity:50, date : ISODate( "2022-01-12T21:23:13.331Z" ) },
   { _id: 5, name: "Cheese", size: "large", price: 14,
     quantity: 10, date : ISODate( "2022-01-12T05:08:13Z" ) },
   { _id: 6, name: "Vegan", size: "small", price: 17,
     quantity: 10, date : ISODate( "2021-01-13T05:08:13Z" ) },
   { _id: 7, name: "Vegan", size: "medium", price: 18,
     quantity: 10, date : ISODate( "2021-01-13T05:10:13Z" ) }
] )
```

```
db.orders.aggregate( [

   // Stage 1: Filter pizza order documents by pizza size
   {
      $match: { size: "medium" }
   },

   // Stage 2: Group remaining documents by pizza name and calculate total
   {
      $group: { _id: "$name", totalQuantity: { $sum: "$quantity" } }
   }

] )
```

https://www.mongodb.com/docs/manual/core/aggregation-pipeline/

MongoDB CRUD

To aggregate on full collection_id: null

```
db.inventory.aggregate([
    {
        $group: {
            _id: null,                  // No grouping field, compute
            averageQty: { $avg: "$qty" } // Calculate average 'qty'
        }
    }
]);
```

https://www.mongodb.com/docs/manual/core/aggregation-pipeline/