Databasteknik

MongoDB

Utbildare: Mikael Olsson

mikael.olsson@emmio.se
076-174 90 43

NACKADEMIN

Socrative

https://www.socrative.com/

- Frågehanterare
  - Logga in som student
  - Ange rum “Emmio”
  - Få upp en vänta-skärm

Snabbfråga

- MySQL är en relationsdatabas
- MongoDB är en dokumentdatabas

Modelling

Users

ID	first_name	surname	cell	city	location_x	location_y
1	Paul	Miller	447557505611	London	45.123	47.232

Professions

ID	user_id	profession
10	1	banking
11	1	finance
12	1	trader

Cars

ID	user_id	model	year
20	1	Bentley	1973
21	1	Rolls Royce	1965

Modelling

- Det viktiga när man designar modeller i MongoDB är hur man ska använda datat, inte dess inbördes relationer.

Modelling

Relational

Users

ID	first_name	surname	cell	city	location_x	location_y
1	Paul	Miller	447557505611	London	45.123	47.232

Professions

ID	user_id	profession
10	1	banking
11	1	finance
12	1	trader

Cars

ID	user_id	model	year
20	1	Bentley	1973
21	1	Rolls Royce	1965

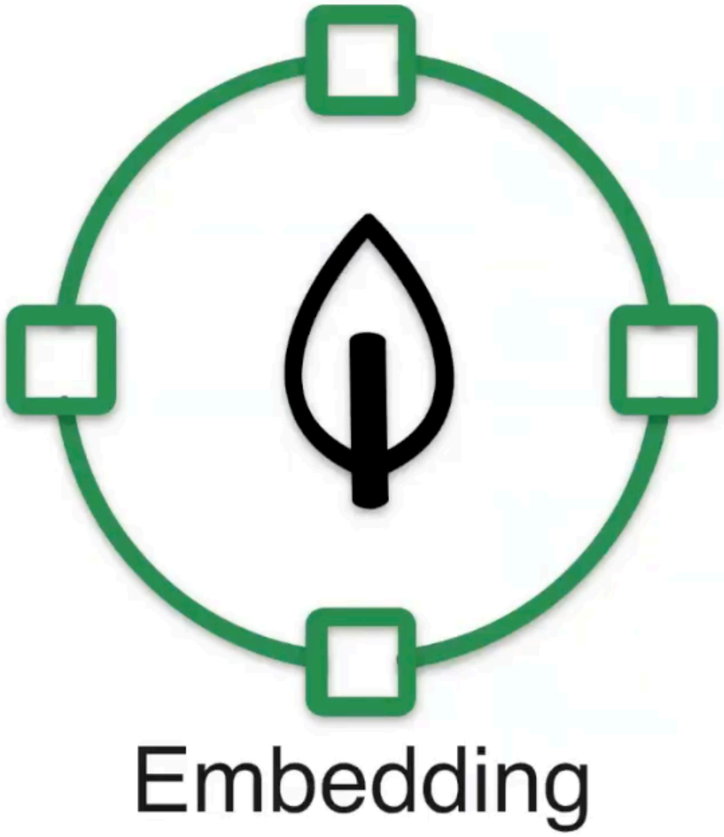
MongoDB

```
{
    first_name: "Paul",
    surname: "Miller",
    cell: "447557505611",
    city: "London",
    location: [45.123,47.232],
    profession: ["banking", "finance", "trader"],
    cars: [
            {
                model: "Bentley",
                year: 1973
            },
            {
                model: "Rolls Royce",
                year: 1965
            }
    ]
}
```

@JoeKarlsson1

Embedding vs referencing

Embedding - det vi gjort hittills i våra modeller




```
{
    _id : ObjectId('AAA'),
    name: 'Kate Monster',
    ssn: '123-456-7890',
    addresses: [
        {
            street: '123 Sesame St',
            city: 'Anytown', cc: 'USA'
        },
        {
            street: '123 Avenue Q',
            city: 'New York',
            cc: 'USA'
        }
    ]
}
```

Fristående objekt, ingen egen entitet.

$lookup

”Left outer join”

- https://docs.mongodb.com/manual/reference/operator/aggregation/lookup/

```
{
   $lookup:
     {
       from: <collection to join>,
       localField: <field from the input documents>,
       foreignField: <field from the documents of the "from" collection>,
       as: <output array field>
     }
}
```

Embedding

Embedding

Pro

Retrieve all data with a single query

Avoids expense JOINs or $lookup

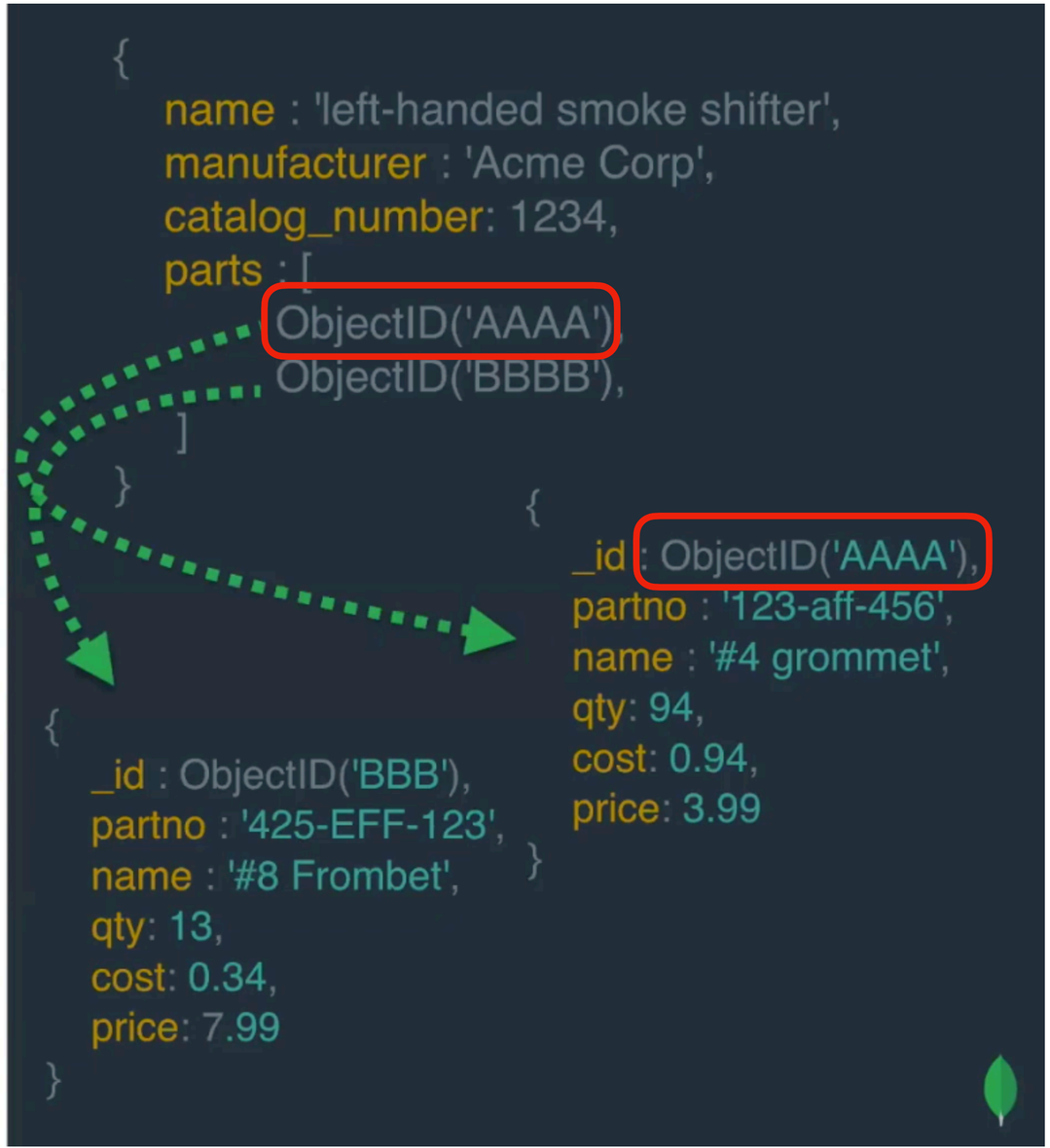
Update all data with a single atomic operation

Con

Large docs === more overhead

16 MB Document size limit

Referencing



Inte helt olikt främmande nycklar

Referencing

Referencing

Pro

- Smaller documents
- Less likely to reach 16 MB limit
- No duplication of data
- Infrequently accessed data not accessed on every query

Con

- Two queries or $lookup required to retrieve all data

Embedding vs referencing

- Här har vi lagt in författaren som ett textvärde, vilket kan vara okej, beroende på vad vi behöver för data i vår applikation.


```
{
    "_id": {
        "$oid": "601d1f6104eae3ec898d97b5"
    },
    "title": "The Hobbit: An Unexpected Journey",
    "writer": "J.R.R. Tolkein",
    "year": 2012,
    "franchise": "The Hobbit",
    "synopsis": "A reluctant hobbit, Bilbo Baggins, sets out to
}
```

Embedding vs referencing

Embedding

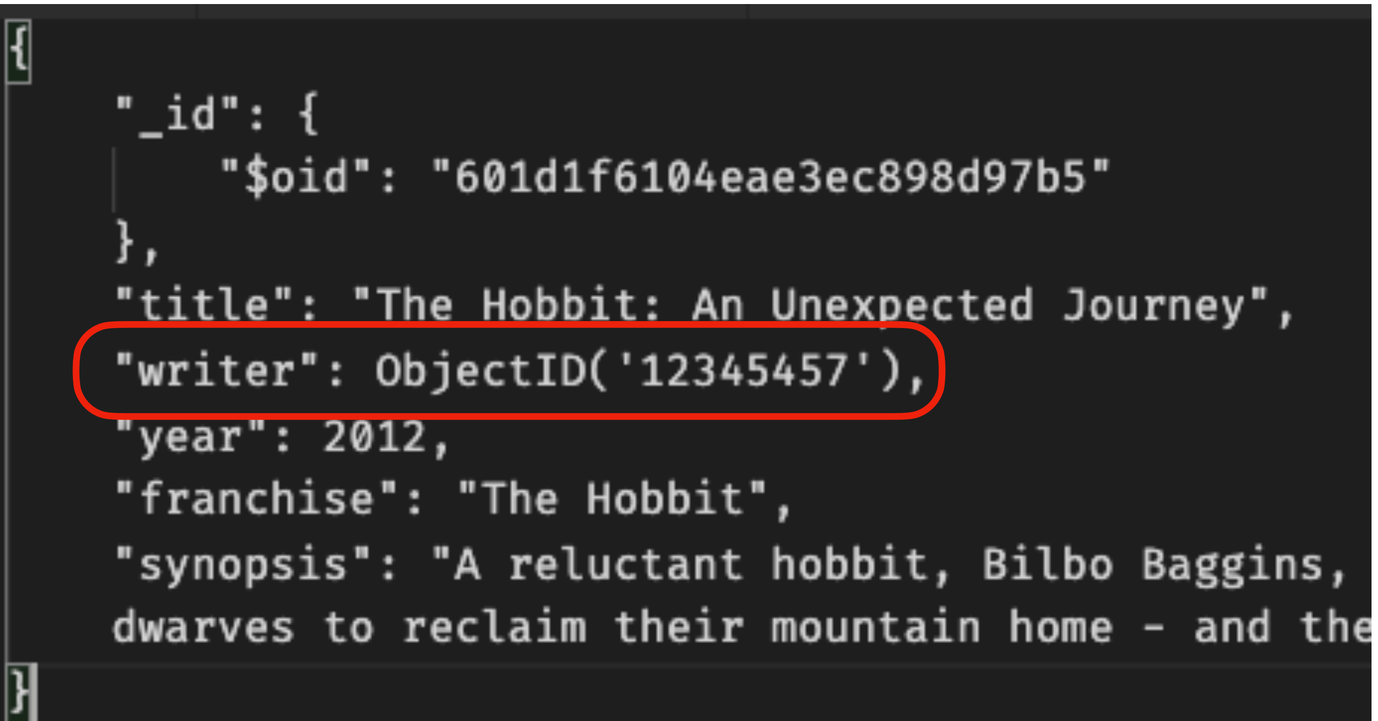
- Vi skulle också kunna tänka oss att embedda ett fristående objekt.

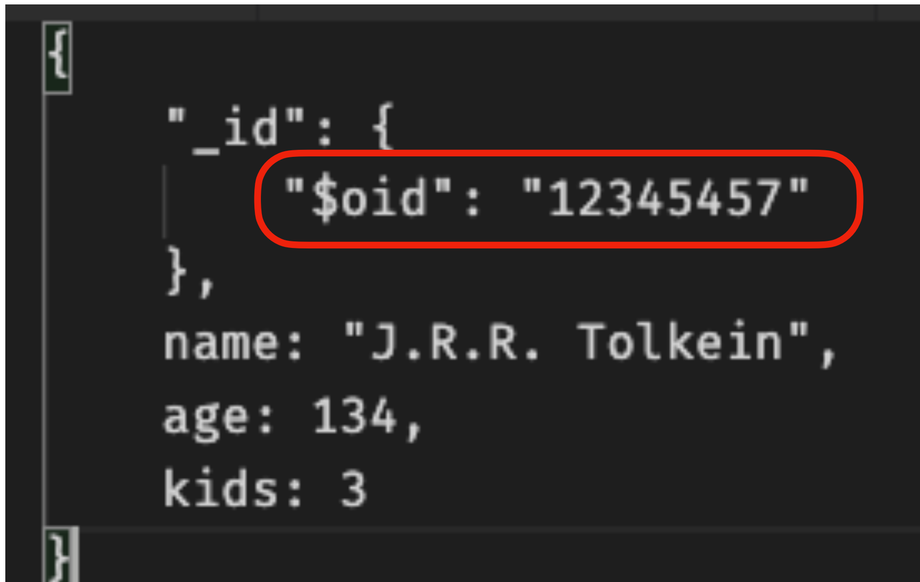
```
{
    "_id": {
        "$oid": "601d1f6104eae3ec898d97b5"
    },
    "title": "The Hobbit: An Unexpected Journey",
    "writer": {
        name: "J.R.R. Tolkein",
        age: 134,
        kids: 3
    },
    "year": 2012,
    "franchise": "The Hobbit",
    "synopsis": "A reluctant hobbit, Bilbo Baggins, sets out to the Lonely Mountain with a spirited group of
    dwarves to reclaim their mountain home - and the gold within it - from the dragon Smaug."
}
```

Embedding vs referencing

Referencing

- Eller så skapar vi en Writer-collection, skapar ett dokument där och refererar in det i vårt Movie-dokument.

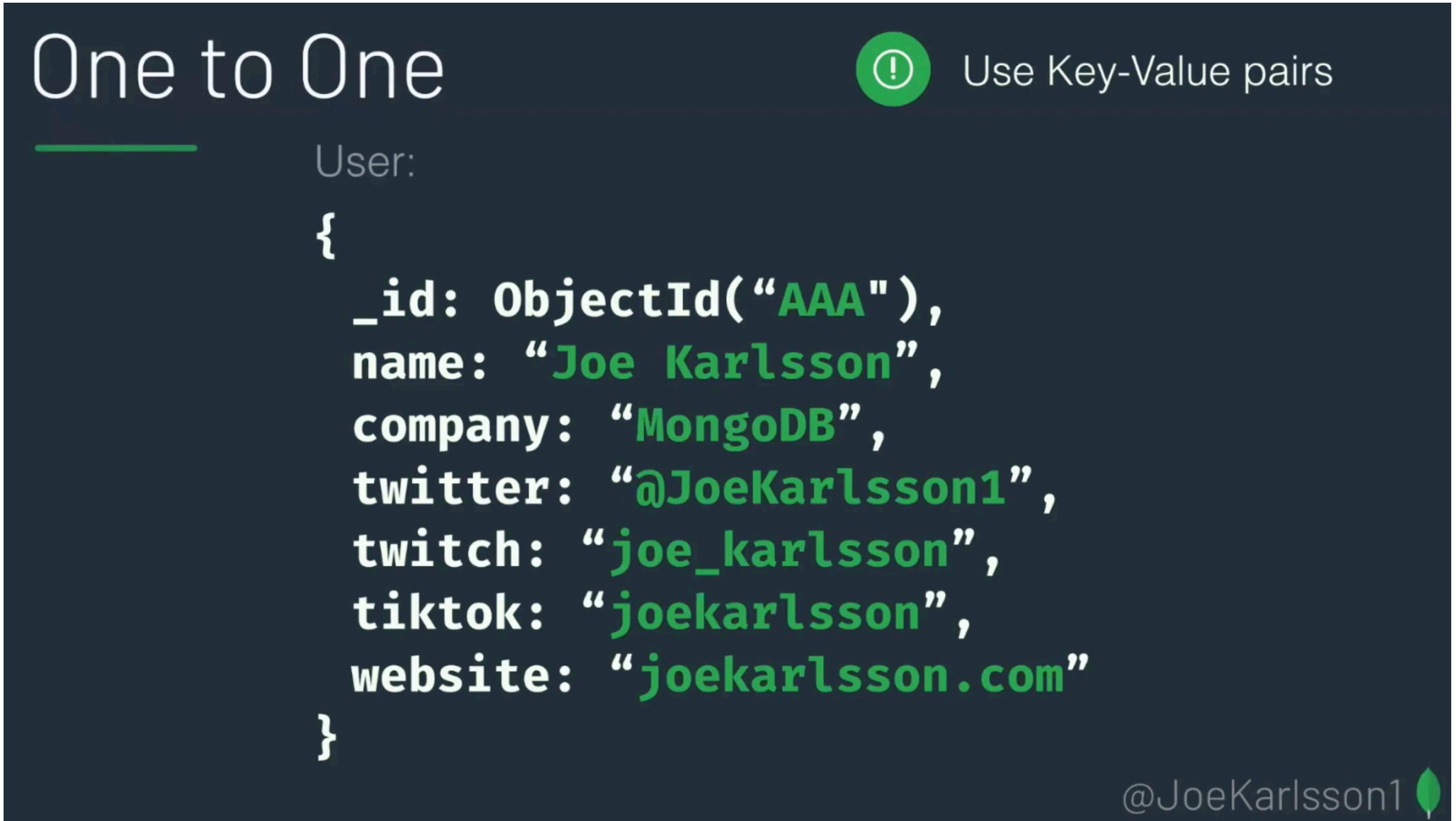




Referencing

- Hur vet vi vilket id vi vill spara?
  - Både i relationsdatabaser och i dokumentdatabaser behöver vi hålla reda på vilket id som ska in som ”främmande nyckel”.
  - I våra experiment blir det ett manuellt arbete, i våra webbapplikationer kommer vi att kunna hantera detta mer automatiskt på backend-sidan.
  - *Psuedo-kod* (kod som är lite lik riktig kod, men som inte fungerar utan är mer gjord för att visualisera vad som ska hända för våra mänskliga ögon):
    - `let writerId = createWriter(”Tolkien”).getID(); // Påhittade metoder, men vi förstår tanken`
    - `addMovie(”LOTR”, writerId);`

Kardinalitet



Kardinalitet

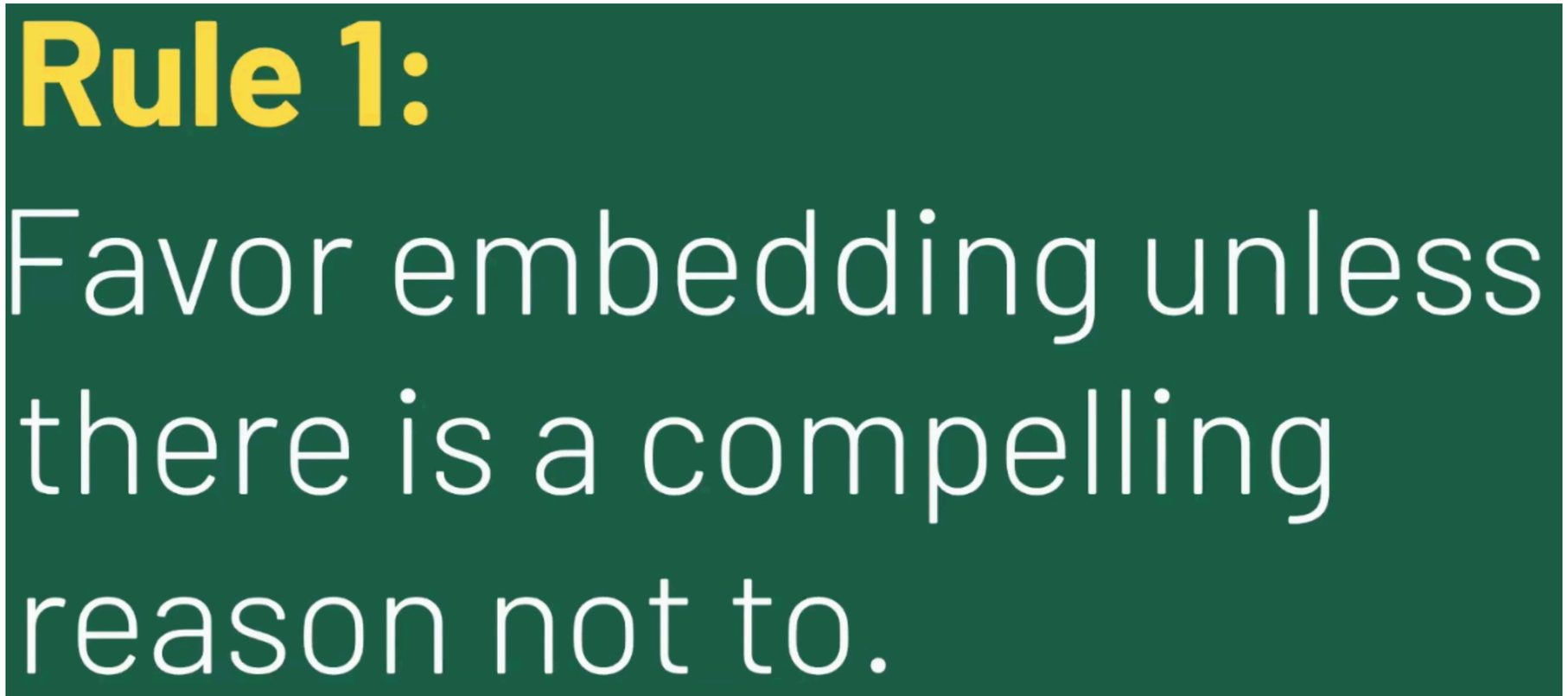
One to Few

```
{
  _id: ObjectId(“AAA"),
  name: “Joe Karlsson”,
  company: “MongoDB”,
  twitter: “@JoeKarlsson1”,
  twitch: “joe_karlsson”,
  tiktok: “joekarlsson”,
  website: “joekarlsson.com”,
  addresses: [
    { street: “123 Sesame St", city: “Anytown”, cc: “USA" },
    { street: "123 Avenue Q",  city: “New York”, cc: ”USA” }
  ]
}
```

@JoeKarlsson1

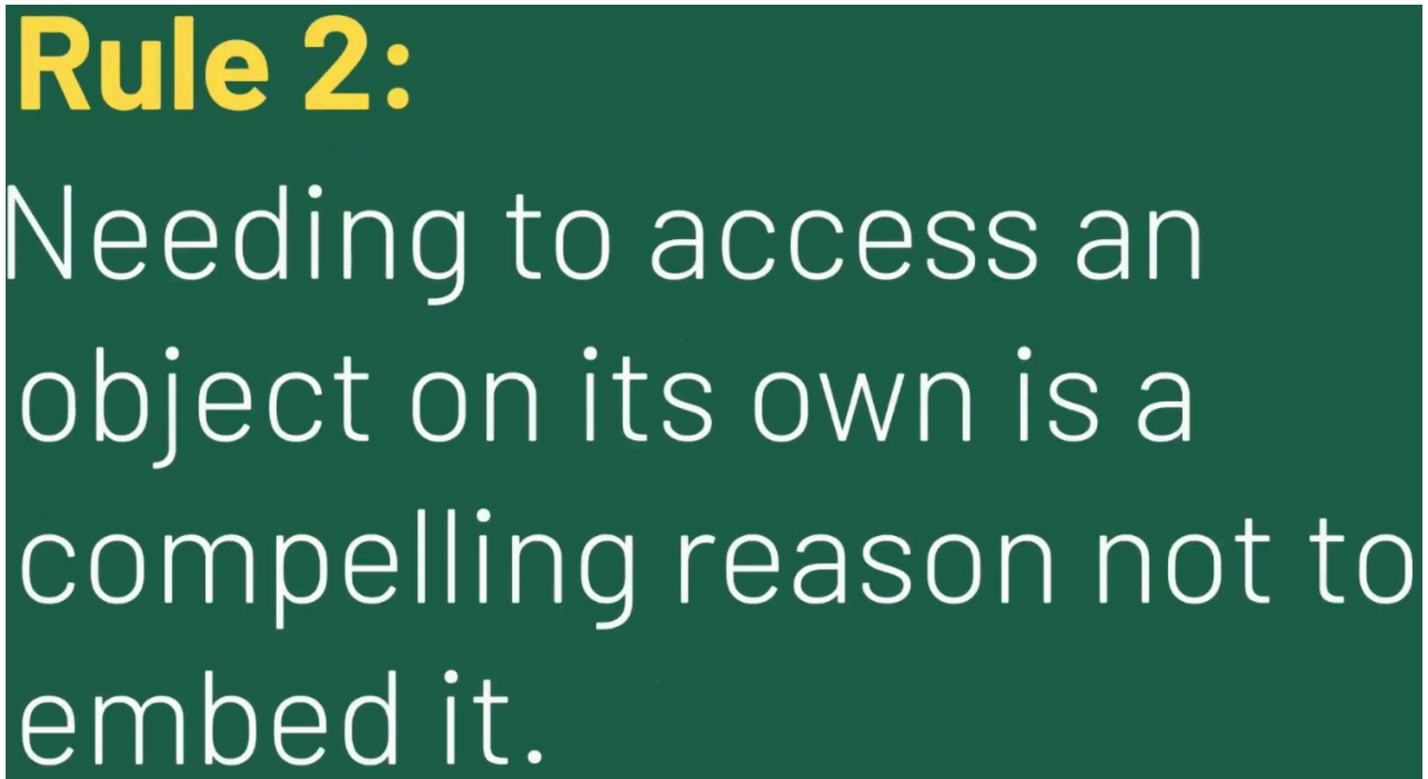
Embedding vs referencing

- Joe pratar om några regler som han har kommit fram till och även om det inte är ”branschregler” på samma sätt som t ex normalisering, så känns de vettiga.



Embedding vs referencing

- T ex om vi behöver komma åt alla Writers ”för sig”, inte som en del av Movie-dokumentet.



Kardinalitet

One to Many

Products:

```
{
  _id: ObjectId("123"),
  name: "left-handed smoke shifter",
  manufacturer: "Acme Corp",
  catalog_number: 1234,
  parts: [
    ObjectId("AAA"),
    ObjectId("BBB"),
    ObjectId("CCC"),
  ]
}
```

Parts:

```
{
  _id: ObjectId("AAA"),
  partno: "123-ABC-456",
  name: "#4 grommet",
  qty: 94,
  cost: 0.54,
  price: 2.99,
}
```

@JoeKarlsson1

$lookup

Exempel

- https://docs.mongodb.com/manual/reference/operator/aggregation/lookup/#examples

```
db.orders.insert([
   { "_id" : 1, "item" : "almonds", "price" : 12, "quantity" : 2 },
   { "_id" : 2, "item" : "pecans", "price" : 20, "quantity" : 1 },
   { "_id" : 3  }
])
```

```
db.inventory.insert([
   { "_id" : 1, "sku" : "almonds", "description": "product 1", "instock" : 120 },
   { "_id" : 2, "sku" : "bread", "description": "product 2", "instock" : 80 },
   { "_id" : 3, "sku" : "cashews", "description": "product 3", "instock" : 60 },
   { "_id" : 4, "sku" : "pecans", "description": "product 4", "instock" : 70 },
   { "_id" : 5, "sku": null, "description": "Incomplete" },
   { "_id" : 6 }
])
```

$lookup

Exempel

- https://docs.mongodb.com/manual/reference/operator/aggregation/lookup/#examples

```
db.orders.aggregate([
    {
      $lookup:
        {
          from: "inventory",
          localField: "item",
          foreignField: "sku",
          as: "inventory_docs"
        }
   }
])
```

```
{
   "_id" : 1,
   "item" : "almonds",
   "price" : 12,
   "quantity" : 2,
   "inventory_docs" : [
      { "_id" : 1, "sku" : "almonds", "description" : "product 1", "instock" : 120 }
   ]
}
{
   "_id" : 2,
   "item" : "pecans",
   "price" : 20,
   "quantity" : 1,
   "inventory_docs" : [
      { "_id" : 4, "sku" : "pecans", "description" : "product 4", "instock" : 70 }
```

Embedding vs referencing

Rule 3:

Avoid JOINs and $lookups if they can be, but don't be afraid if they can provide a better schema design.

Kardinalitet

Lite som skillnaden mellan 1:n mot n:1

Referencing

- Tänk att ha egenskapen tv-shows som innehåller referenser till alla program som någonsin visats på kanalen.
- Vänd på det istället och sätt referensen på tv-programmet.

Kardinalitet

Embedding vs referencing

- Skillnad mot RDBMS.
  - RDBMS: Utgår från hur data ser ut.
  - Dokument-db: Utgår ifrån hur du vill använda data.