-

MongoDB

---

mongo design for blog

posts

```
{ _id
  author
  body
  comments: { }
  date
  permalink
  tags: [ ]
  title
}
```

comments: body, email, author

tags: strings

Supported data access patterns: →

> collecting most recent entries for homepage
> collecting all information to display one post
> collecting all comments by a single author

bad access pattern for:
providing a table of contents by tag.

Alt. Schema for Blog

```
posts {
  _id
  title
  body
  author
  date
}

comments {
  _id
  post_id
  author
  author_email
  order: 0
}

tags {
  _id
  tag
  post_id
}
```

doesn't work particularly well

If it feels relational, it is probably wrong schema for MDB

Living w/o constraints

REL → foreign key constraints

MDB → data consistency is programers responsibility

> comments embedded in post → "pre-join"

→ embed data in ways that make sense

Life w/o transactions

MDB HAS ATOMIC OPERATIONS

— when working on a single doc. no one else can access it until you finished

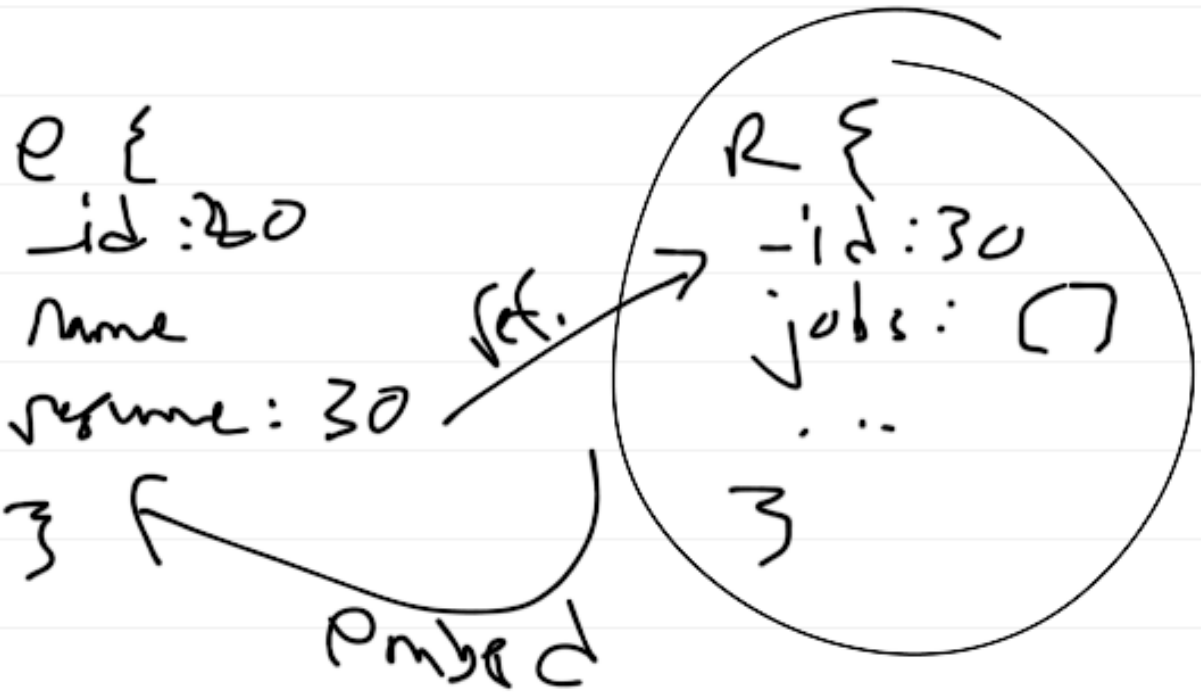
① restructure	② implement in SW → software	③ tolerate

approaches to handle things.

one-to-one relations

employee : resume

building : floorplan



- frequency of access
  - if less frequent use ref (store seperately)
- size : > 16mb must be REF.
  - anonymicity of data

one-to-many

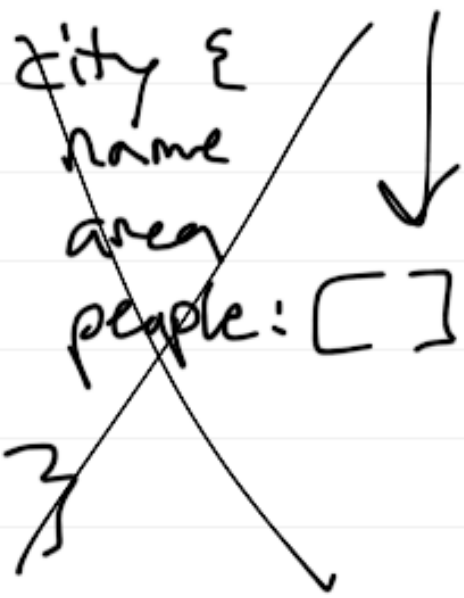
city : person

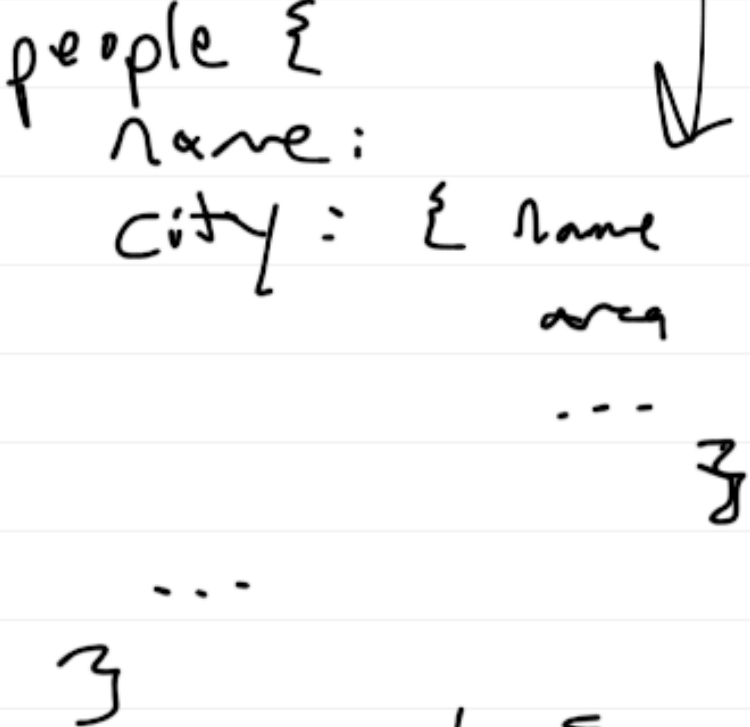
NYC

on people

too large

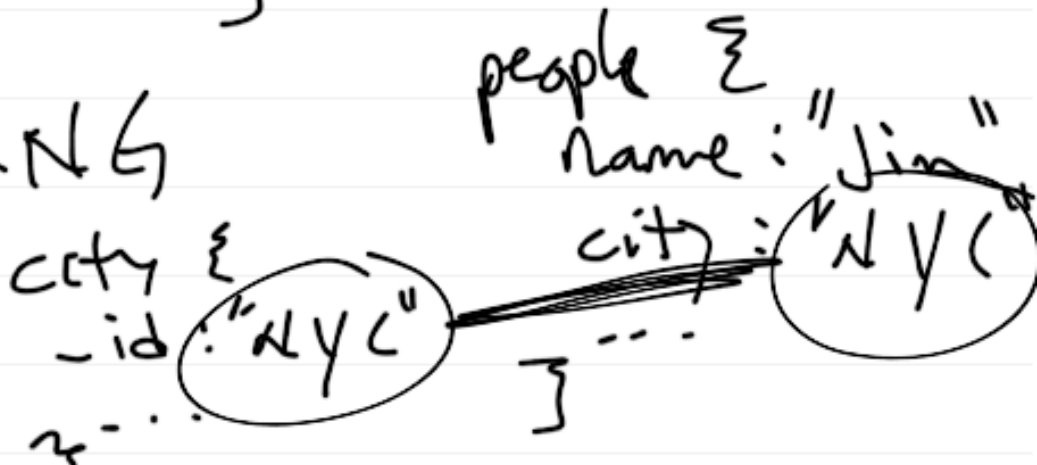
duplicate





TRUE LINKING

Two collections



one to few

blog post : comments

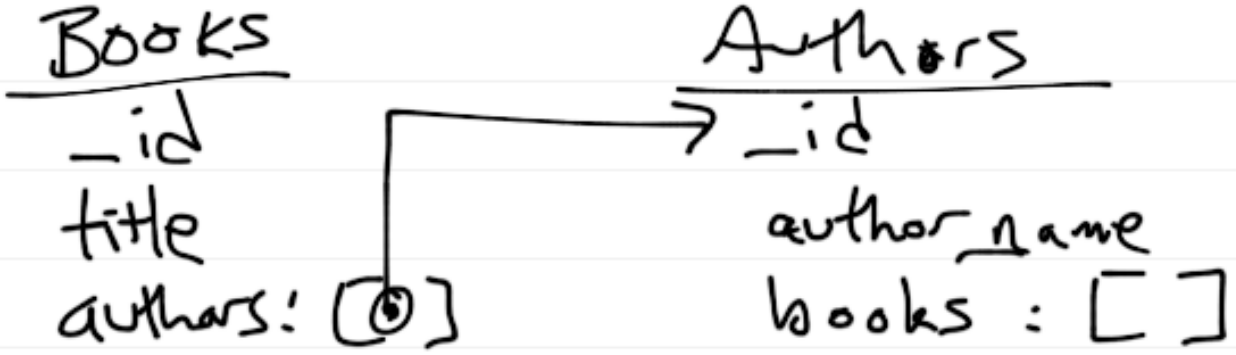
```
posts {
  name
  comments : [ ]
  ...
}
```

Many-to-many

BOOKS ⟶ AUTHORS

STUDENTS ⟶ TEACHERS

FEW : FEW (USUALLY)



* could embed books in authors doc

students : teachers

teachers: [] students: [ ]

MULTIKEYS (multikey index)

students

```
{ _id : 0
  name: "Jim"
  teachers : [0,1,2] }
```

teachers

```
{ _id : 2
  name : "TS" }
```

```
db.students.ensureIndex({'teachers':1})
db.students.find({'teachers':
                  {$all : [0,1]}})
```

append .explain() to see what it did

Benefits of Embedding

- improved read performance

High latency (ms)
High Bandwidth

- one roundtrip to the DB

eComm categories

TREES

HOME : OUTDOORS : WINTER : SNOW

products:
category : 7
product_name :

category:
_id : 7
cat_name:
(1) parent : 6
(2) children : [ ]
(3) ancestors :
[3, 5, 9, 8]
(in order)

options

db.categories.find ({ ancestors: 34 }) — an element

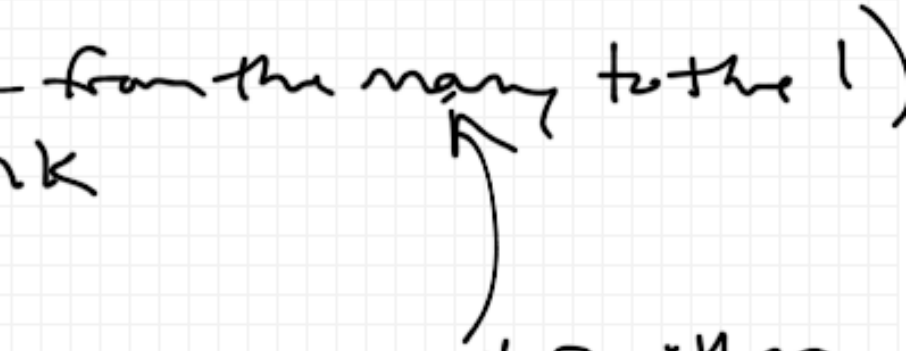
— finds all documents when '34' is in the ancestors array

WHEN TO DENORMALIZE

1:1 — embed

1:Many — (embed from the many to the 1)

many:many — link



to support application access pattern, embedding is perfectly acceptable