NoSQL databases

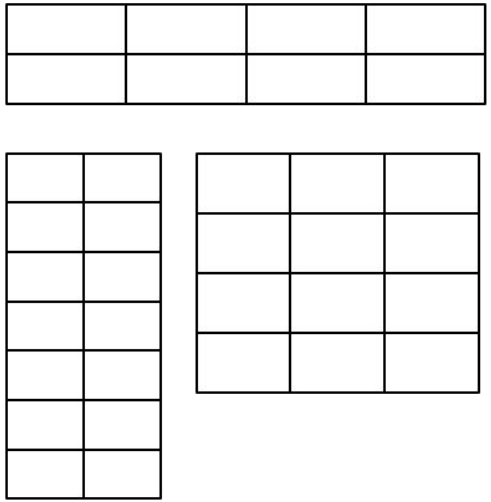
Eelco Dijkstra
4 februari 2016

Overzicht

- Relationele (SQL) databases
- Waarom NoSQL databases?
- Soorten NoSQL databases
- Voorbeeld NoSQL db: MongoDB
- Vervolg

SQL: relationele databases

- schema: beschrijft structuur
  - namen van tabellen, kolommen,
  - type van kolom-data (uniform voor kolom)
  - statisch (veranderen off-line)
- tabellen
  - vaste kolommen
  - “rechthoekige data”
  - “key” voor identificatie van rij
- normalisatie
  - vermijden van redundante representaties
  - eenvoudiger consistent te houden
  - gevolg: “joins” bij opvragen van data



Transacties: ACID

Database moet *consistent* blijven onder transacties

- *Atomicity* - transactie slaagt, of heeft geen effect;
- *Consistency* - db-invarianten gelden na elke transactie;
- *Isolation* - “gelijktijdige” transacties hebben effect van opeenvolging van transacties
- *Durability* - effect van transactie is permanent (ook bij fysieke problemen)

ACID versus BASE: Basic Availability, Soft state, Eventual consistency

Waarom NoSQL?

- *variatie* in structuur van data
  - data niet altijd “rechthoekig”; bijv. Internet of Things, tijdreeksen
- *flexibiliteit (evolutie)* in structuur van data
  - structuur “evolueert” tijdens agile ontwikkeling
- *schaalbaarheid*: zeer grote aantallen gebruikers
  - verdelen van gebruikers, *replicatie* van data
- *schaalbaarheid*: zeer grote hoeveelheden data
  - verdelen van data (sharding)
- *beschikbaarheid (snelheid)* vóór consistentie
  - “eventual consistency” voldoende (BASE)
- relationele database *onvoldoende efficiënt* voor toepassing
  - bijv. analyseren van relaties(!)

Voorbeeld: sensor data

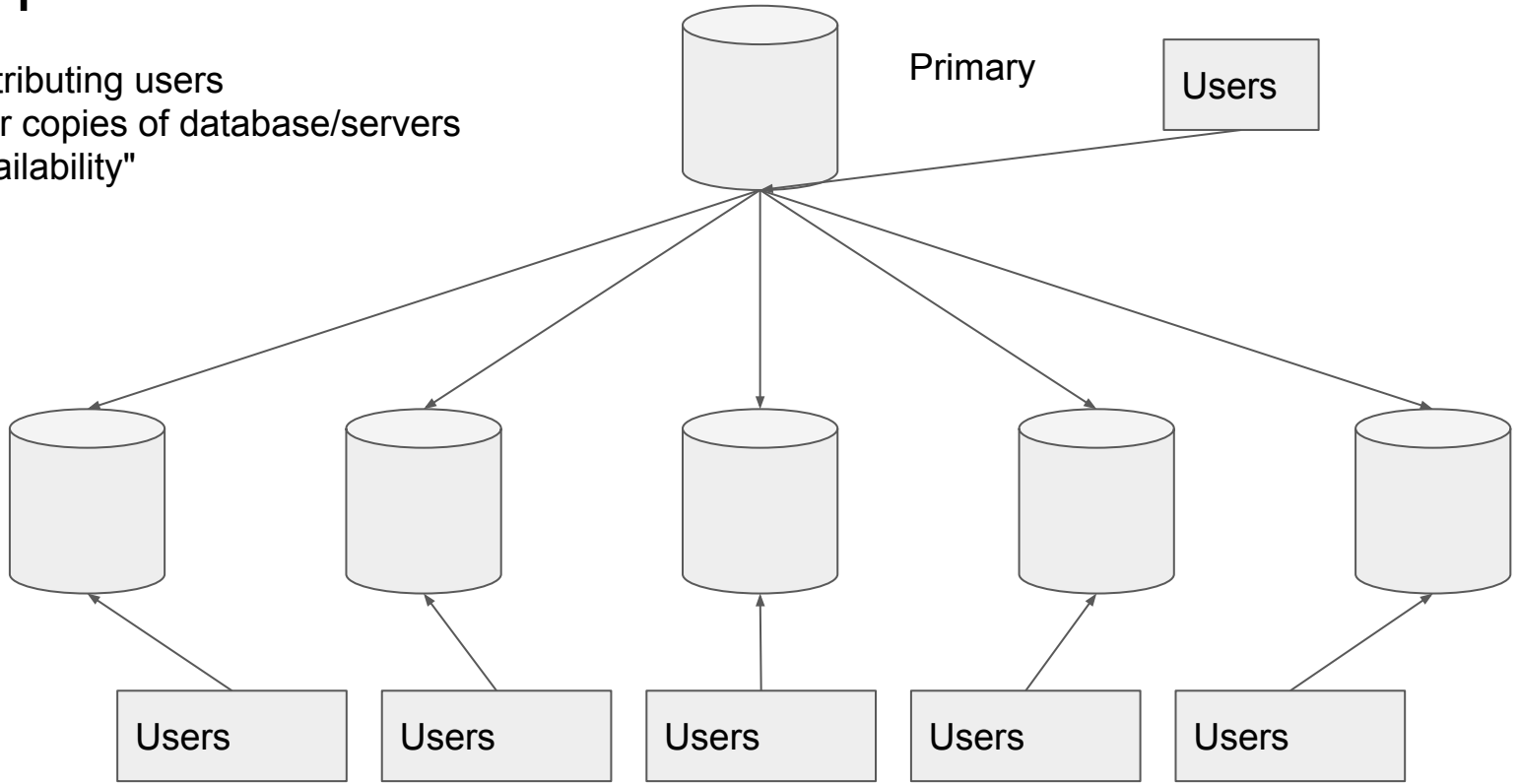
Verschillende sensoren leveren verschillende *soorten* waarden.

```
{sensor: "temperature", time: "12:35", value: 12.7, location: "garden", }
{sensor: "pressure", time: "12:17", value: 1024.3}
{sensor: "light", time: "12:19", value: [200, 150, 150, 30]}
```

(Vgl. ook log-files en andere tijdreeks-data: verschillende events met verschillende attributen.)

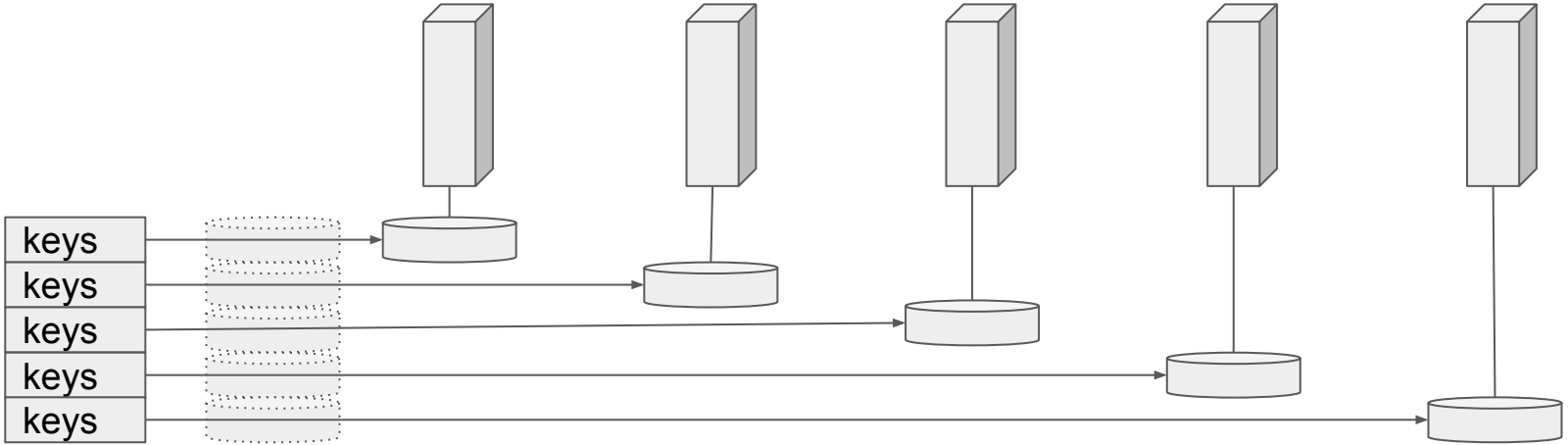
Replication

Distributing users
over copies of database/servers
"availability"



Sharding

Distribute data over servers
Key determines shard



Principe van *lokaliteit*

"Houd gegevens die samen/tegelijk worden gebruikt, bij elkaar in de buurt."

nabijheid in tijd -> nabijheid in plaats

- caching
- virtueel geheugen
- harde schijf
- distributie over servers
- distributie over wereld

Soorten NoSQL databases

- key->value store
  - LocalStorage (HTML5/Browser)
- key->value indexed DB
  - IndexedDB (HTML5/browser)
- column database
  - Cassandra
- document DB
  - MongoDB; CoachDB
  - Amazon SimpleDB; Google Cloud DataStore
- graph DB
  - e.g. Neo4J
  - tuple store (semantische DB)

MongoDB: document database

SQL database	**mongoDB**
Database	Database
Table	Collection
Row	Document
Index	Index
Join	Document embedding or reference

Database

MongoDB instantie: meerdere databases

> show databases

Aangeven welke database je wilt gebruiken:

> use test

Een database is een verzameling *collections*.

Collection

Collection: verzameling (min of meer) gelijksoortige documenten,

Bij opdrachten (queries) aangeven welke collection je gebruikt:

> db.docenten.find({naam: “Hans Hoekstra”})

Document

Document: verzameling velden (eigenschappen): naam -> waarde

document geïdentificeerd (in collection) door *key* (document-id): “_id”-veld

In JavaScript-interface: object; in Python: dictionary

BSON representatie: uitbreiding van JSON (met extra types)

Voor uitwisseling: Extended JSON

Document: schema en document-validatie

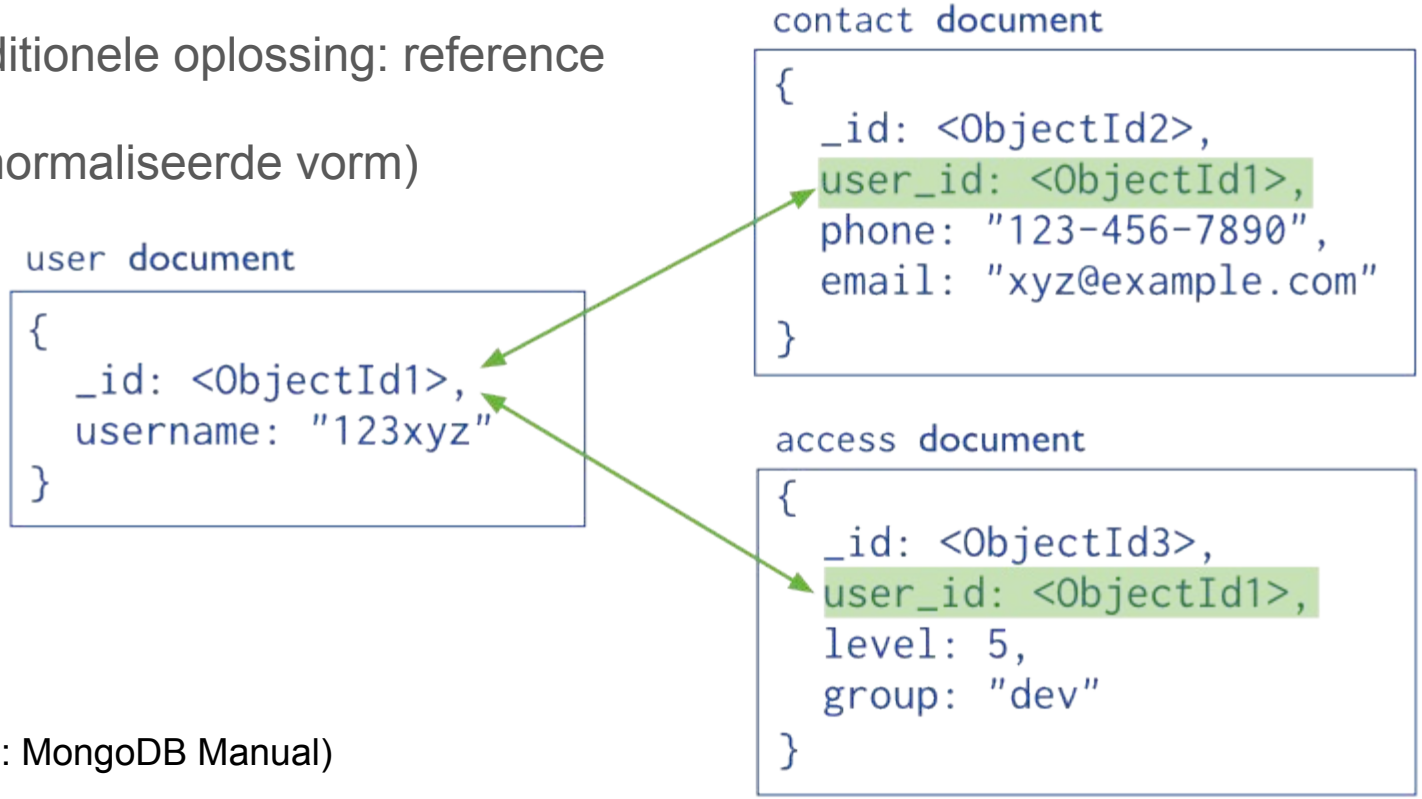
Documenten in een collectie kunnen onderling variëren. Document-validatie (optioneel): controleren van minimaal aanwezige velden.

```
db.createCollection( "contacts",
   { validator: { $or:
      [  { phone: { $type: "string" } },
         { email: { $regex: /@mongodb\.com$/ } },
         { status: { $in: [ "Unknown", "Incomplete" ] } }
      ]
   }
} )
```

Document-reference of embedding

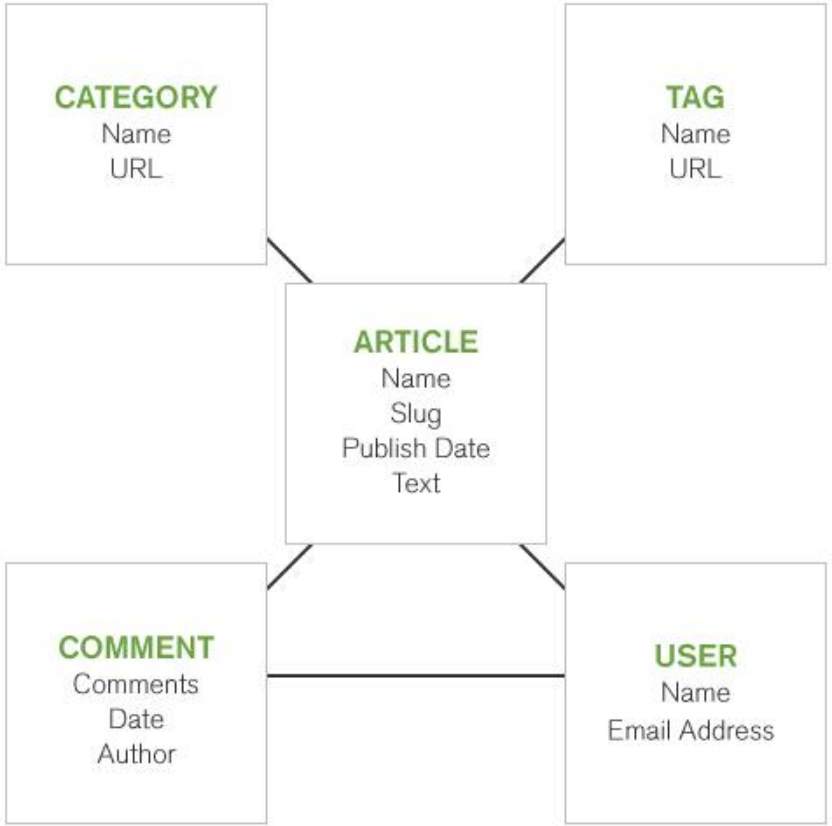
Traditionele oplossing: reference

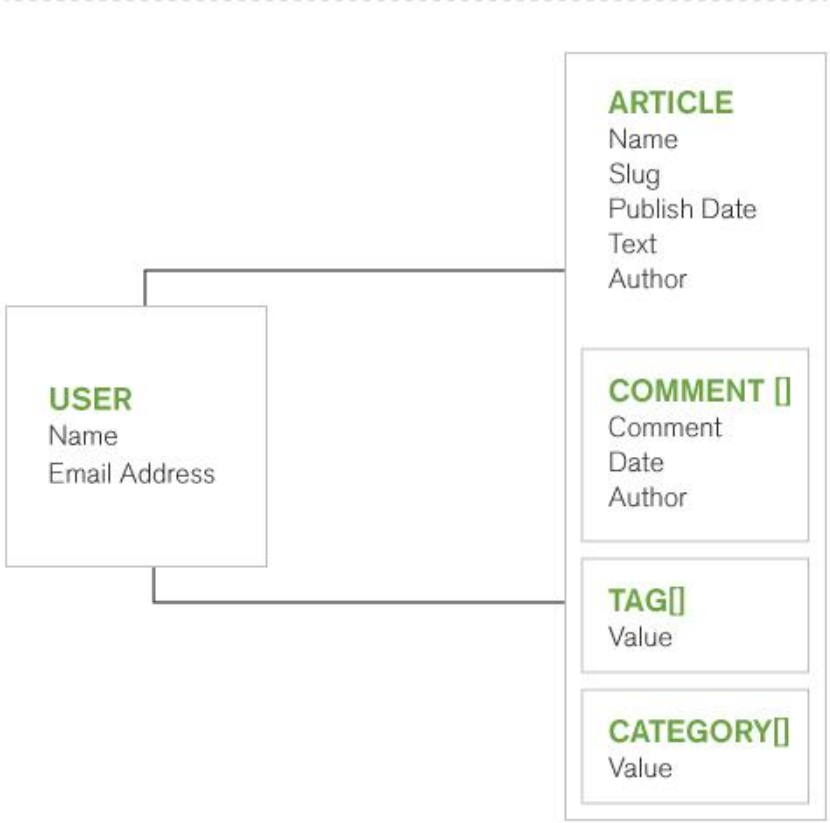
(genormaliseerde vorm)



(Bron: MongoDB Manual)

Embedding (de-normalisatie)





Voorbeeld: embedding van 3 deel-documenten (bron: MongoDB - Thinking in documents)

Joins, transactions: programmeren in de toepassing

Join

- extra database-lookups

Transactions over multiple documents:

- (only) document-action is atomic
- complex transaction: 2-phase commit, log-document

Oefening

- Installeren/activeren MongoDB (Cloud9)
- Werken met MongoDB shell (interactief)
- (eventueel: oefenen met queries - top2000)
- MongoDB vanuit Python
  - met “embedding” versus “referencing”
- MongoDB/Python voor websites

Docent

Naam	Vak

School

Naam	Plaats	Betapartner

Docent-school

Docent-naam	School-naam

Voor de N-M relatie tussen Docent en School heb je een aparte tabel nodig.

In MongoDB kun je deze relatie "embedden" in beide documenten.

Vervolg

- ontwikkelen lesmateriaal - inleiding databases
- ontwikkelen lesmateriaal MongoDB
  - eenvoudige website (Python)
  - Node-Red: afhandelen van sensor-data (JavaScript)
- uitgebreidere cursus NoSQL
  - Graph databases
  - localStorage; IndexedDB (en varianten)
  - ...