

Odisee
DE CO-HOGESCHOOL

NoSqlDatabases



Jens Baetens



NoSQL

- ▣ Staat voor non-SQL of not-only SQL
- ▣ Database systeem voor verwerken van niet-gestructureerde data.
- ▣ Waarom?
 - Eenvoudiger ontwerp
 - Eenvoudig horizontaal te schalen
 - Availability beter te controleren
 - Object-relational mismatch beperkt



Voordelen

- ▣ Availability
- ▣ Snelheid
- ▣ Fout-tolerantie
- ▣ Flexibiliteit
- ▣ Schaalbaarheid





Nadelen

- ▣ Niet meer gegarandeerd consistent
- ▣ Lower-level query languages than sql
- ▣ Joins zijn moeilijker
- ▣ (Nog) geen standaardisatie





Gebruikte datastructuren

- ▣ Geen vaste tabellen maar
 - Key-value pairs
 - Documents
 - Graph
 - Wide column (structuur afhankelijk van rij tot rij)
- ▣ Deze structuren zijn flexibeler en soms sneller



Consistentie

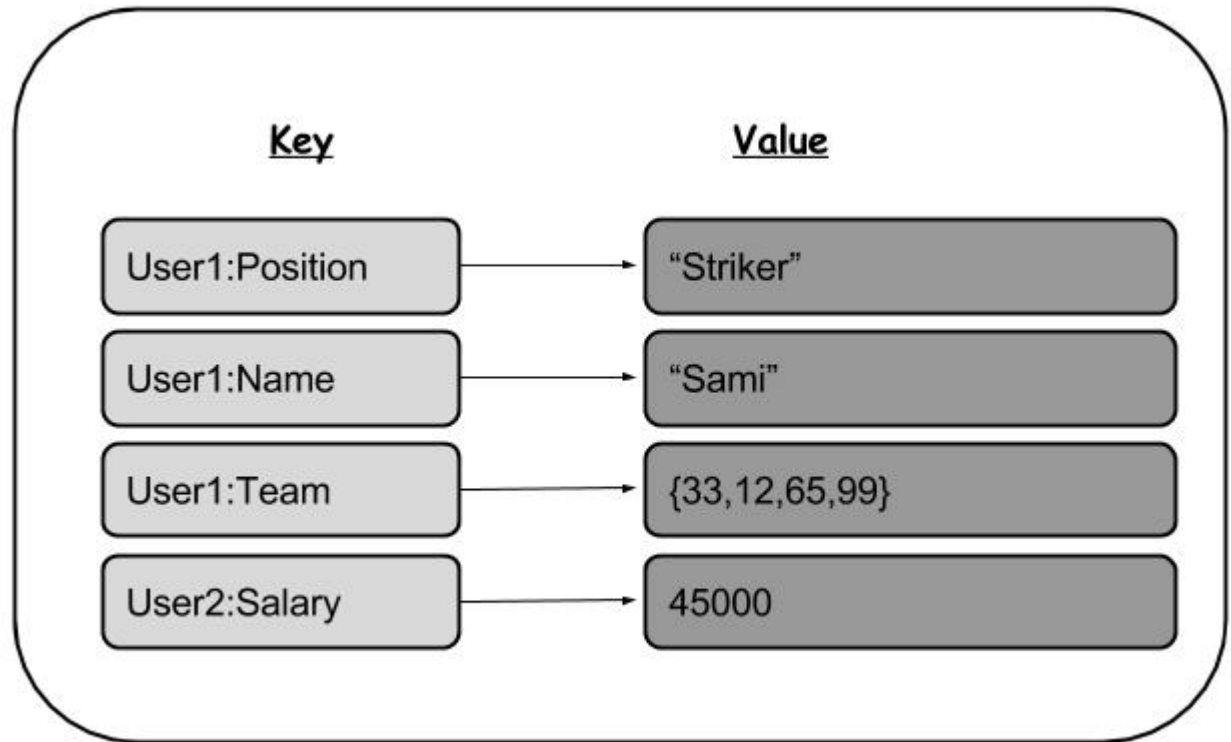
- ▣ NoSQL databases gebruiken “Lazy Consistency” model
 - Replicaties worden aangepast na een write maar dit kan even duren
 - Na het schrijven van een waarde kan een andere replica nog de oude waarde bevatten

Types

Type	Notable examples of this type
Key-value cache	Apache Ignite, Couchbase, Coherence, eXtreme Scale, Hazelcast, Infinispan, Memcached, Redis, Velocity
Key-value store	Azure Cosmos DB, ArangoDB, Aerospike, Couchbase, Redis
Key-value store (eventually consistent)	Azure Cosmos DB, Oracle NoSQL Database, Dynamo, Riak, Voldemort
Key-value store (ordered)	FoundationDB, InfinityDB, LMDB, MemcacheDB
Tuple store	Apache River, GigaSpaces
Object database	Objectivity/DB, Perst, ZopeDB
Document store	Azure Cosmos DB, ArangoDB, BaseX, Clusterpoint, Couchbase, CouchDB, DocumentDB, eXist-db, IBM Domino, MarkLogic, MongoDB, Qizx, RethinkDB, Elasticsearch
Wide Column Store	Azure Cosmos DB, Amazon DynamoDB, Bigtable, Cassandra, Google Cloud Datastore, HBase, Hypertable,
Native multi-model database	ArangoDB, Azure Cosmos DB, OrientDB, MarkLogic

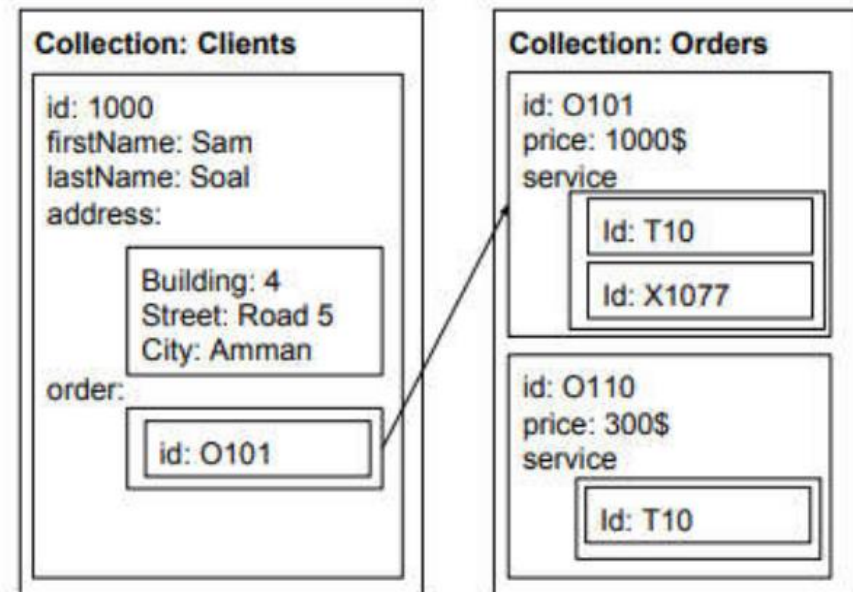
Key-value pairs

- ▣ Analooq aan map/dictionary
- ▣ Elke key is uniek
- ▣ Values
 - Geen vast schema/type hiervoor
- ▣ Basis voor complexere modellen
- ▣ Kan in Ram, HDD of SDD
- ▣ Focus vooral op schaalbaarheid, minder op consistentie
- ▣ Voorbeelden: Amazon Dynamo, Redis, LinkedIn Voldemort , ...



Document store

- ▣ Centrale concept: Document
 - objecten die informatie encapsuleert
- ▣ Encoded als XML / Json / YAML / ...
- ▣ Semi-structured data in de vorm van key-value pairs
- ▣ Document bestaat uit
 - Key/identifier voor indexing / searching
 - Set of key-value pairs
- ▣ Flexible schema
 - Documents as subdocuments
 - Documents as values



```
{ "client": {  
  "id": 1000,  
  "firstName": "Sam",  
  "lastName": "Soal",  
  "address": {  
    "building": "4B",  
    "street": "Road 5",  
    "city": "Amman"  
  },  
  "order": [  
    "O101"  
  ],  
  "order1": {  
    "id": "O101",  
    "price": 1000,  
    "service": [  
      "T10",  
      "X1077"  
    ]  
  },  
  "order2": {  
    "id": "O110",  
    "price": 300,  
    "service": [  
      "T10"  
    ]  
  }  
}
```

Document store

▣ Vooral bruikbaar als data er zich toe leent

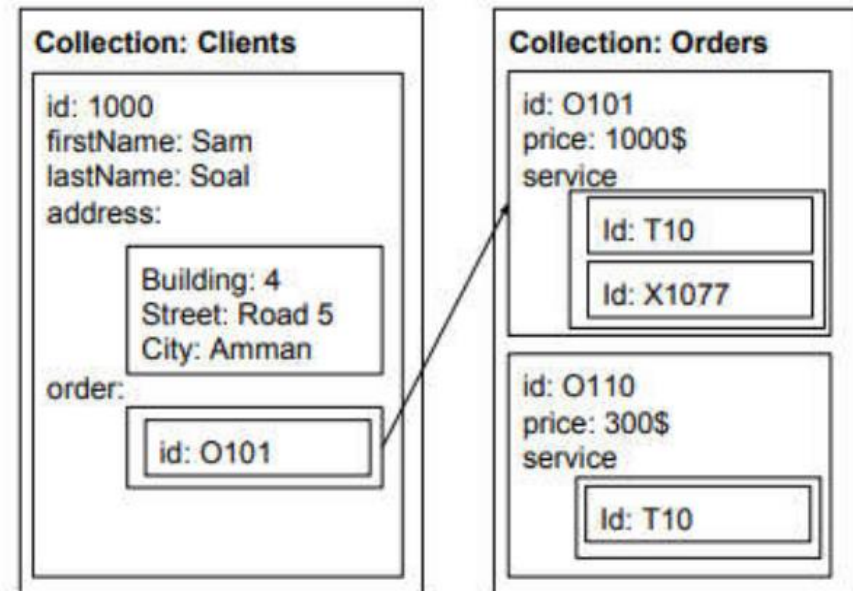
- ▬ CMS
- ▬ Blogging sites
- ▬ Webshop

▣ Nadeel

- ▬ Geen verband tussen documenten
- ▬ Wel met identifier maar bestaan niet gegarandeerd

▣ Voorbeelden:

- ▬ MongoDB
- ▬ Apache CouchDB

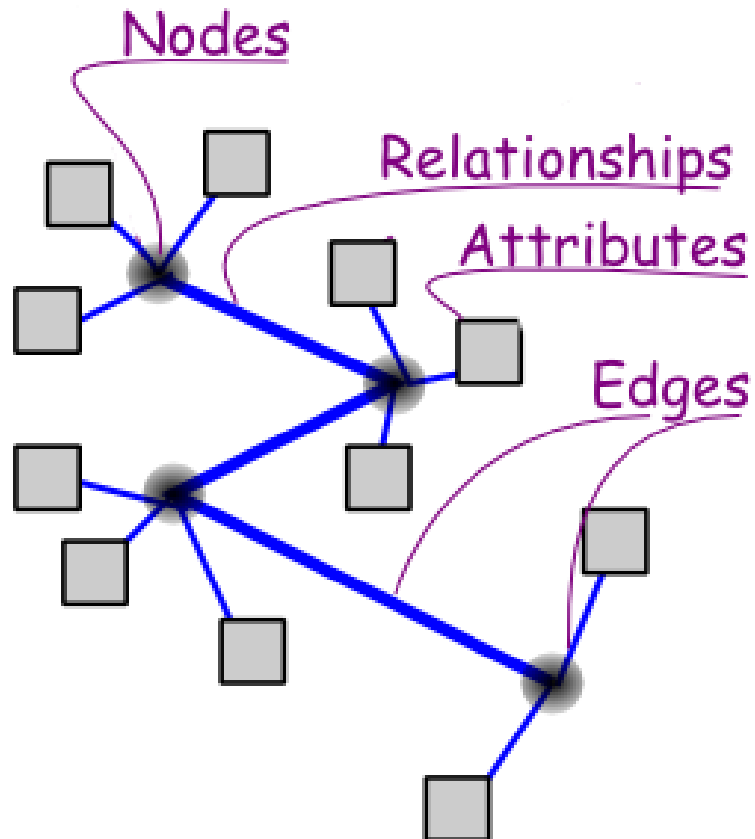


```
{ "client": {  
  "id": 1000,  
  "firstName": "Sam",  
  "lastName": "Soal",  
  "address": {  
    "building": "4B",  
    "street": "Road 5",  
    "city": "Amman"  
  },  
  "order": [  
    "O101"  
  ],  
  "order1": {  
    "id": "O101",  
    "price": 1000,  
    "service": [  
      "T10",  
      "X1077"  
    ]  
  },  
  "order2": {  
    "id": "O110",  
    "price": 300,  
    "service": [  
      "T10"  
    ]  
  }  
}
```

Graph

■ Voor data bestaande uit geconnecteerde elementen

- Social media
- Straten, mobiliteit, ...
- Netwerken, ...



Wide Column Store

- ▣ Gebruikt tables, rows, columns
 - ▬ Lijkt sterk op SQL-database
- ▣ Naam en format van de kolommen kan elke rij anders zijn
- ▣ 2 dimensionale key-value store
- ▣ Google BigData, Amazon DynamoDB, Cassandra, HBase

Row A	Column 1	Column 2	Column 3	...
	Value	Value	Value	
Row B	Column 2	Column 3	Column 4	...
	Value	Value	Value	

Overzicht

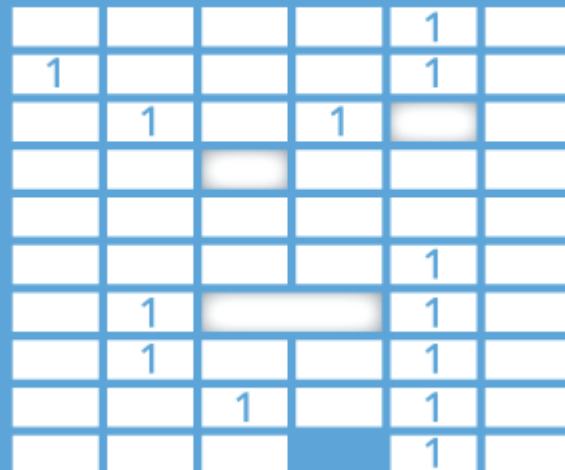
Key-Value



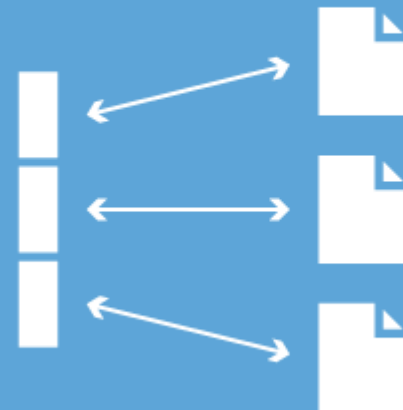
Graph DB



Column Family



Document



Overzicht

Data model ↕	Performance ↕	Scalability ↕	Flexibility ↕	Complexity ↕	Functionality ↕
Key-value store	high	high	high	none	variable (none)
Column-oriented store	high	high	moderate	low	minimal
Document-oriented store	high	variable (high)	high	low	variable (low)
Graph database	variable	variable	high	high	graph theory
Relational database	variable	variable	low	moderate	relational algebra



NoSQL Databases

Simply Explained



Self – hosted options - MongoDB

- ▣ Document oriented
- ▣ BSON documentformaat
 - ▬ Binary Json (Javascript Object Notation)
- ▣ Geen ondersteuning voor
 - ▬ Joins
 - ▬ ACID-regels (atomicity, consistency, isolation, and durability)
 - ▣ Belangrijk voor transacties van operaties

Self – hosted options - MongoDB

- ▣ Ondersteuning voor
 - B(inary) L(arge) OB(ject)
 - MapReduce via javascript voor analyse en aggregatie

Self – hosted options: Cassandra

- ▣ Wide-column store released by Facebook
- ▣ Amazon
 - ▬ Storage en replication
- ▣ Google
 - ▬ Data en storage engine model
- ▣ Ondersteuning voor
 - ▬ Map Reduce
 - ▬ Pig
 - ▬ Hive
- ▣ Apache Query Language

Cloud – Azure – CosmosDB

- ▣ Multi-model NoSQL database
- ▣ Items in containers in databases
 - Databases = Namespaces
 - Containers = collection of documents
 - Items = documents
- ▣ JSON-friendly SQL-API of 5 complementary API's
 - MongoDB
 - Cassandra
 - ...

Cloud – Azure – CosmosDB

■ Different consistency levels

- ▬ Eventual
- ▬ Consistent Prefix
- ▬ Session: read-your-own writes (Default)
- ▬ Bounded Staleness: at most x updates or ms behind
- ▬ Strong consistency: always up to date

Cloud – Amazon - DynamoDB

- ▣ Wide-column store
 - Key-value
 - Document data
- ▣ Integration with Hadoop through Elastic Map Reduce



Cloud – Amazon - DocumentDB

- ▣ Document data
- ▣ Json Data
- ▣ Limited support for Mongo DB

Cloud – Google - BigTable

- ▣ 3 dimensional keys :
 - row/column key
 - Timestamp
- ▣ Value is een byte-array
- ▣ Tables gesplitst in tablets
 - Stukjes van de table op rij-niveau
 - 100 MB tot een aantal GB per tablet

Big Table – Voorbeeld voor kopie van het Web voor Google Search

▣ Keys

- Rij = url
- Kolom = Karakteristieken / Html page
- Timestamp voor verschillende versies

Cloud – Google – Datastore of Firestore (nieuwe versie)

- ▣ Data opgeslagen in documenten die in collecties gesorteerd worden
- ▣ Elk document is een set key-value pairs
- ▣ Document is klein (max 1MB) json-record
 - Bijvoorbeeld voor een user

```
alovelace
name :
  first : "Ada"
  last  : "Lovelace"
born  : 1815
```