Napredni modeli i baze podataka

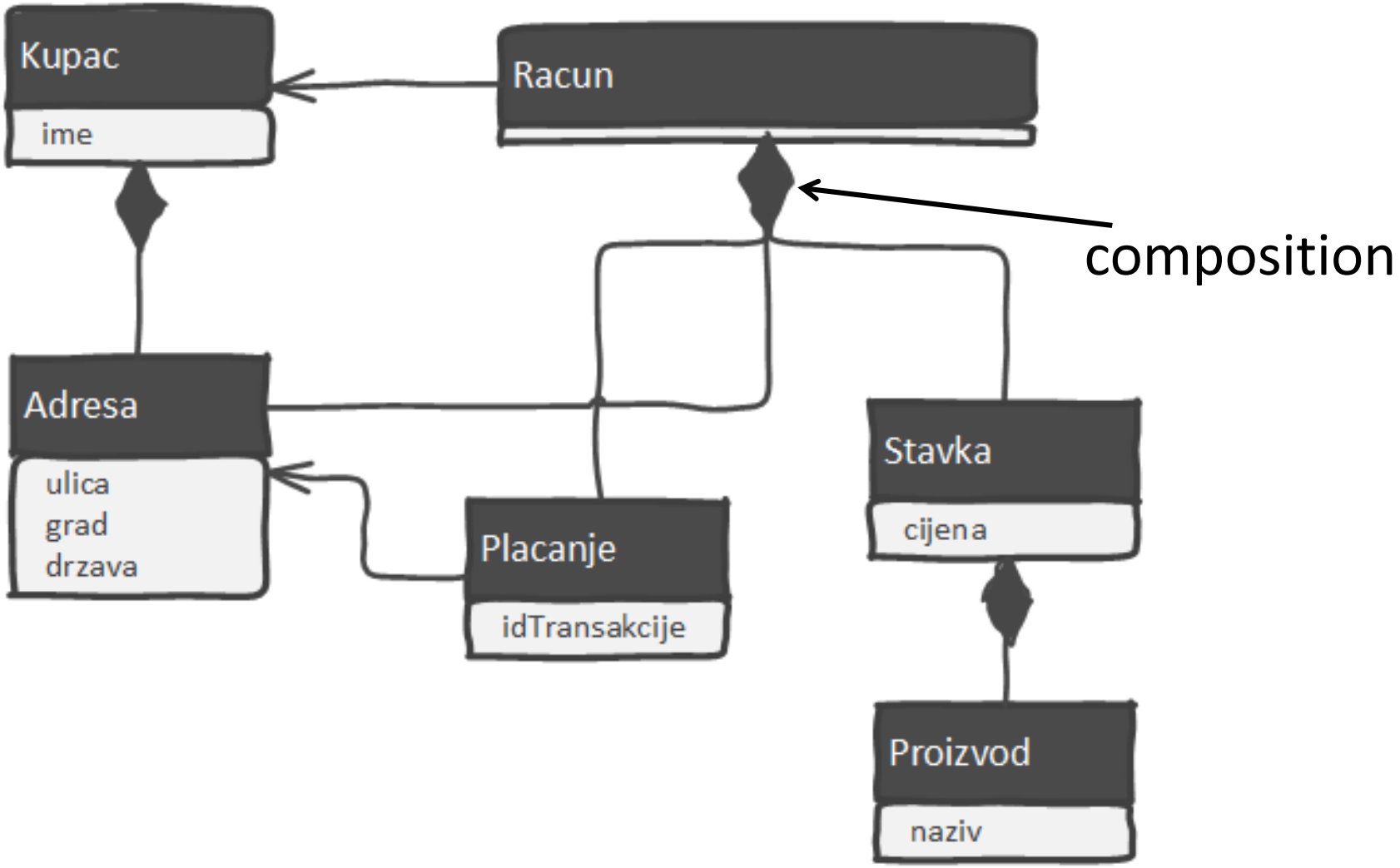
Predavanja
prosinac 2015.

NoSQL
2/3

Agregatni model

- Pojam dolazi iz DDD-a
- Agregat je:
  - Složeni zapis koji dozvoljava
    - Liste
    - Gniježđenje drugih zapisa
  - Skup objekata koji se obrađuju kao jedan zapis (npr. narudžba i stavke narudžbe)
- Jedan korijen, po njemu ga se:
  - referencira
  - osigurava integritet
- Osnovne jedinice podataka - dohvaća se i/ili sprema cijeli agregat
  jedan agregat ~ jedna "transakcija"

Agregat - primjer (1)

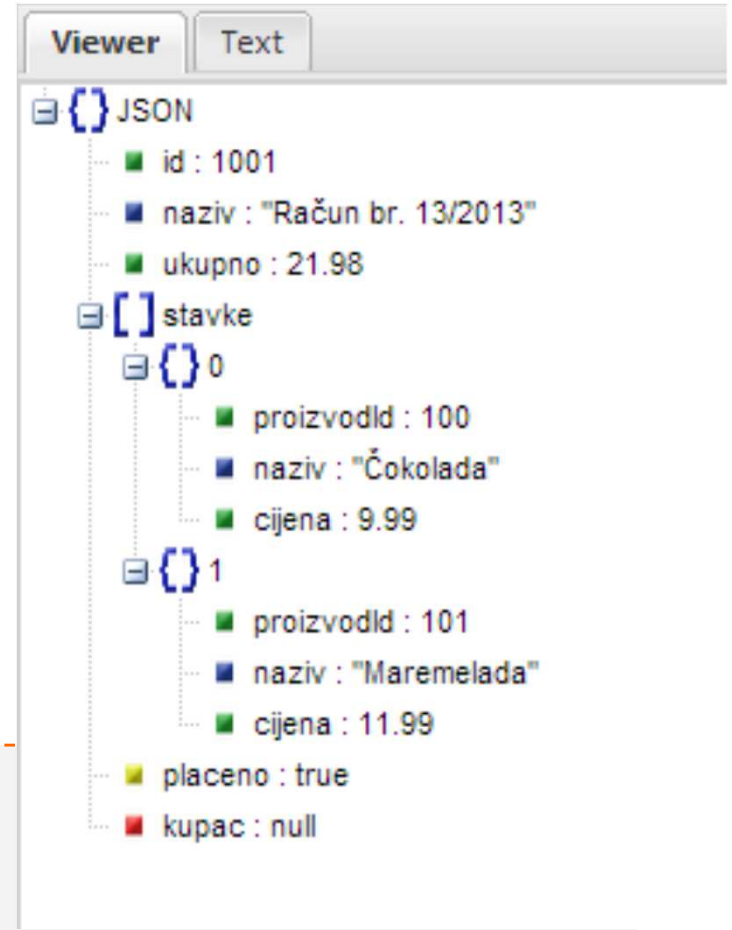


Digresija: JSON podsjetnik (1)

- JSON - **J**ava**S**cript **O**bject **N**otation
  - *Plain-text*, čitljiv
  - Ne ovisi o programskom jeziku
  - Hijerarhijski (gniježđenje)
- JSON vs XML:
  - js.eval () (ali *"eval is evil"*, koristiti JSON parser)
  - lakši rad
  - Kraći od XML-a
  - Ne treba zatvarati oznake
  - Polja
- ime:vrijednost parovi, npr. **`"ime":"Krešo"`**

Digresija: JSON podsjetnik (2)

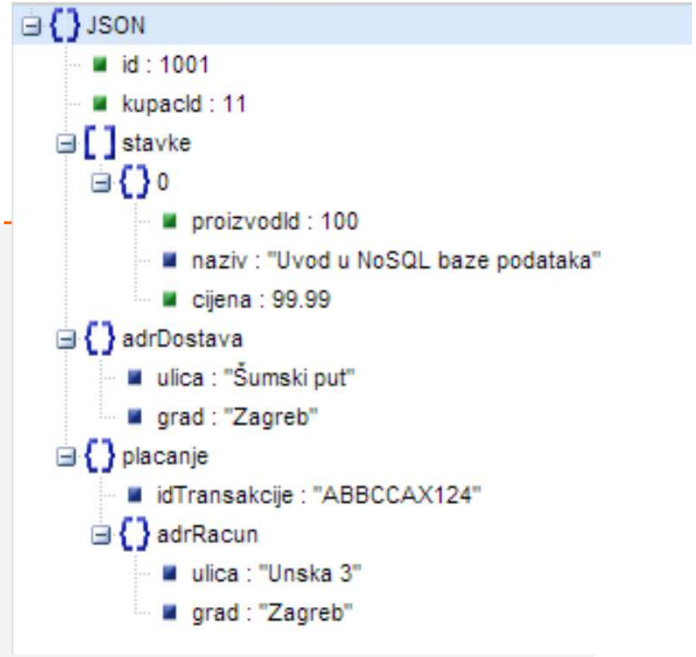
- Vrijednost može biti:
  - Broj (int, real)
  - String ("")
  - Boolean (true/false)
  - Polje (u [])
  - Objekt (u {})
  - null




```
{
    "id": 1001,
    "naziv": "Račun br. 13/2013",
    "ukupno": 21.98,
    "stavke": [
        {"proizvodId": 100, "naziv": "Čokolada", "cijena": 9.99},
        {"proizvodId": 101, "naziv": "Maremelada", "cijena": 11.99}
    ],
    "placeno": true,
    "kupac": null
}
```

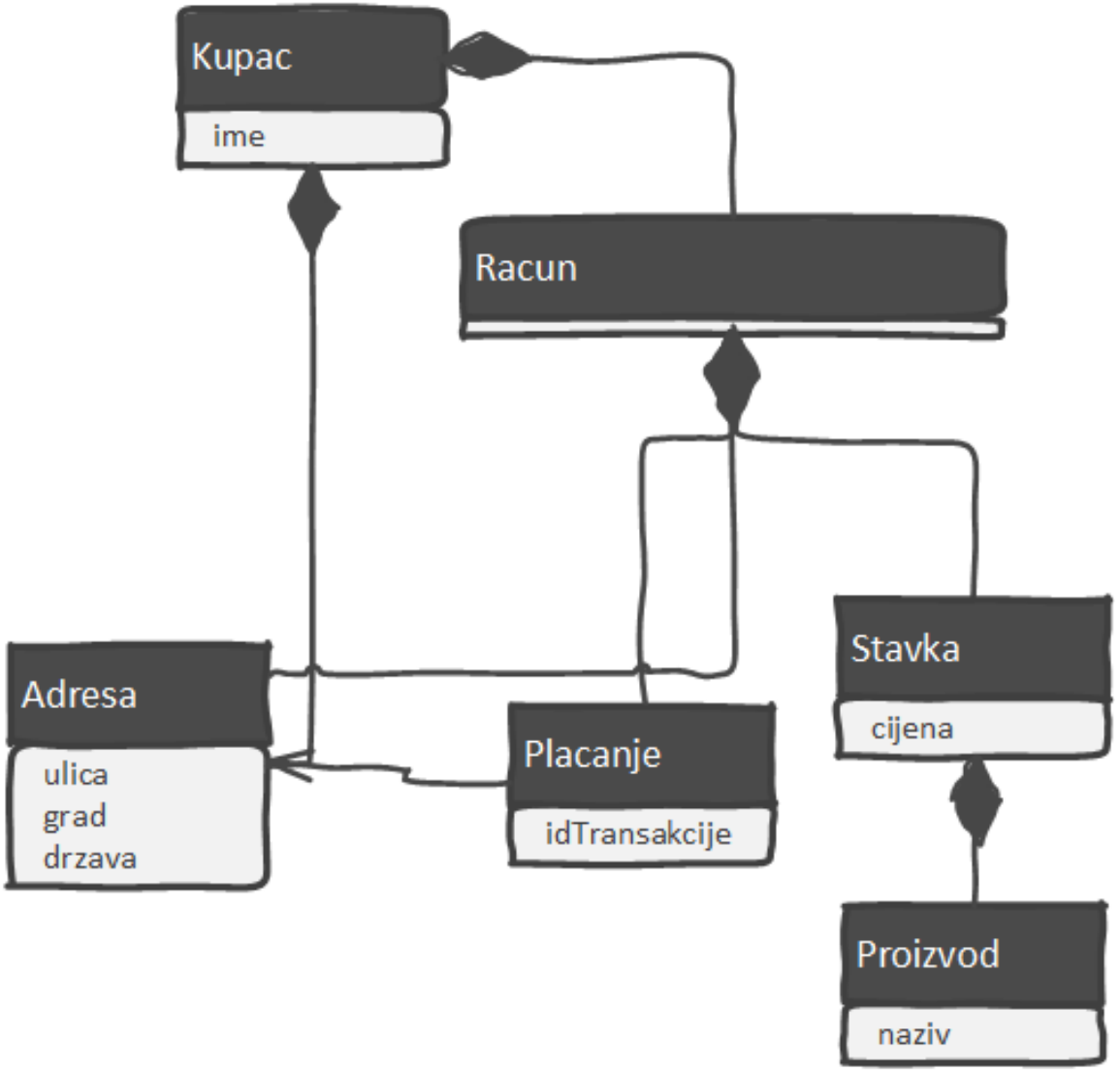
Agregat - primjer (1) - sadržaj

- JSON:




```
// kupac
{
    "id": 11,
    "ime": "Krešo",
    "adrRacun": {"ulica":"Unska 3", "grad": "Zagreb"}
}

// racun
{
    "id": 1001,
    "kupacId": 11,
    "stavke": [
        {"proizvodId": 100, "naziv": "Uvod u NoSQL baze podataka", "cijena": 99.99}
    ],
    "adrDostava": {"ulica":"Šumski put", "grad": "Zagreb"},
    "placanje": {
        "idTransakcije": "ABBCCAX124",
        "adrRacun": {"ulica":"Unska 3", "grad": "Zagreb"}
    }
}
```

Agregat - primjer (2)



Agregatni model - komentar

- Nema općenitog naputka gdje postaviti granice agregata - ovisi u uporabi
- **Zgodno za distribucije podataka**, agregati su smislene cjeline
- Relacijski model nema tu informaciju - "*aggregate ignorant*", isto kao i grafovski model
- *Aggregate ignorant* <> loše svojstvo
- Agregatni model može i pomoći i odmoći, ovisno o kontekstu:
  - ✓ Dohvat, spremanje, distribucija računa
  - ✗ Analiza prodaje u protekla dva mjeseca
- **Glavni razlog**: uporaba u **distribuiranoj** okolini, želimo **minimizirati broj čvorova** kojima pristupamo kod prikupljanja podataka

Ključ-vrijednost baze podataka

- **Model podataka: (ključ, vrijednost) parovi**
- Operacije:
  - Unos(k, v)
  - Dohvat(k)
  - Ažuriranje(k, v)
  - Brisanje(k)
  - Neki podražavaju određenu **strukturu** vrijednosti, **atribute** vrijednosti
  - Neki podržavaju dohvat na temelju raspona ključeva
- Neki primjeri: *Riak, Dynamo,…*

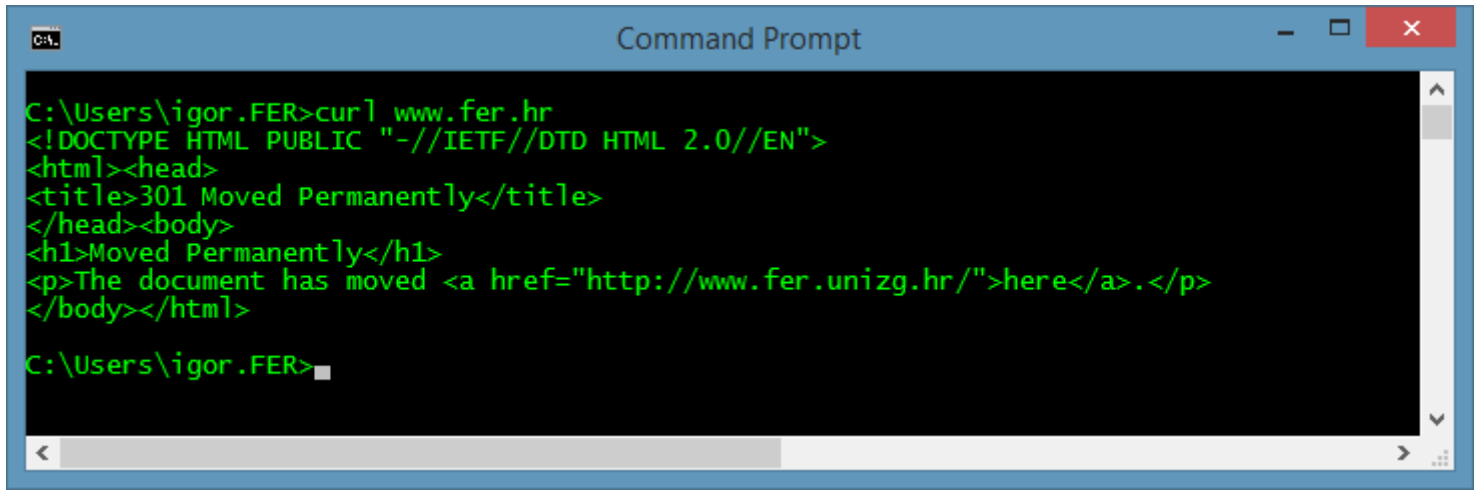
Primjer: Riak

- Inspiriran s *Amazon Dynamo*
- Distribuirana KV baza podataka
- V - bilo što (*plain text*, JSON, slika, video, …)
- Jednostavno HTTP sučelje (*„Riak speaks web”*)
- Fault tolerant
- Skalabilnost
  - Lako je dodati novi čvor u klaster
  - Automatska distribucija podataka
  - *„a near-linear performance increase as you add capacity”*
- Nedostatci:
  - Ad-hoc upiti
  - Povezivanje zapisa (ref.int.)

curl (1)

- *„**curl** is a command line tool and library for transferring data with URL syntax, supporting DICT, FILE, FTP, FTPS, Gopher, HTTP, HTTPS, IMAP, IMAPS, LDAP, LDAPS, POP3, POP3S, RTMP, RTSP, SCP, SFTP, SMTP, SMTPS, Telnet and TFTP. curl supports SSL certificates, HTTP POST, HTTP PUT, FTP uploading, HTTP form based upload, proxies, HTTP/2, cookies, user+password authentication (Basic, Digest, NTLM, Negotiate, kerberos...), file transfer resume, proxy tunneling and more.”*

- Npr.



curl (2)

- Neke često korištene opcije:

```
-d, --data    DATA    HTTP POST data (H)
--data-binary DATA    HTTP POST binary data (H)
-o, --output  FILE    Write output to <file> instead of stdout
-v, --verbose         Make the operation more talkative
-X, --request COMMAND Specify request command to use
-H, --header LINE     Custom header to pass to server (H)
```

Key Value repozitorij

- REST:
  - `/riak/bucket/`**`key`** `# Old format`
  - `/buckets/bucket/keys/`**`key`** `# New format`
- Put (pohranjivanje)

```
$ curl -v -X PUT http://localhost:8091/riak/test/helloworld \
-H "Content-Type: text/html" \
-d "<html><body>hello world</body></html>"
```

- Get (dohvat):

```
$ curl http://localhost:8091/riak/test/helloworld
<html><body>hello world</body></html>
$ curl http://localhost:8092/riak/test/helloworld
<html><body>hello world</body></html>
$ curl http://localhost:8093/riak/test/helloworld
<html><body>hello world</body></html>
```

Primjer: spremamo sliku

- Npr., stavimo sliku:

```
$ curl -X PUT http://localhost:8091/riak/slike/igor.png \
-H "Content-type: image/png" \
--data-binary @homer.png
```

Dokument baze podataka

- Nalik ključ-vrijednost, pri čemu je vrijednost dokument
- **Model podataka: (ključ, dokument)**
- Dokument: JSON, BSON, XML, YAML, neki drugi polustrukturirani format, binarni podaci
- Osnovne operacije:
  - Unos(k, d)
  - Dohvat(k)
  - Ažuriranje(k, d)
  - Brisanje(k)
  - **Dohvat na temelju sadržaja dokumenta**! (nije standardizirano, nema upitnog jezika)
- Neki sustavi omogućuju i indeksiranje
- Neki primjeri: *CouchDB, MongoDB, SimpleDB,…*

Primjer: MongoDB dokumenti

Relacijska baza podataka: **relacija**

Ime	Prezime	DatRodj	MjestoRodj
Ivan	Car	11.11.1971.	
Iva	Kralj		Šibenik

MongoDB: **kolekcija**

```
{
    "_id": ObjectID("4efa8d2b7d284dad101e4bc9"),
    "Ime" : "Ivan",
    "Prezime“ : "Car",
    "DatRodj" : "11.11.1971."
},
{
    "_id" : ObjectID("4efa8d2b7d284dad101e4bc7"),
    "Ime" : "Iva",
    "Prezime" : "Kralj",
    "MjestoRodj" : "Šibenik"
}
```

Primjer: MongoDB upiti

- Mongo upiti se formiraju kao JSON (BSON) objekti

SQL	MongoDB

`CREATE TABLE student(`
`  mbr INT,`
`  prezime CHAR(50)`
`)`
	Implicitno - unosom prvog dokumenta u kolekciju.
	Može i eksplicitno:
	`db.createCollection(„student")`

`ALTER TABLE student ADD…`	Implicitno, svaki dokument je moguće promijeniti bez mijenjanja sheme

`INSERT INTO student (`
`  100, 'Šostakovič'`
`);`
	`db.student.insert(`
	`  {mbr:100, prezime: 'Šostakovič'}`
	`)`

`SELECT * FROM student;`	`db.student.find();`

`SELECT prezime`
`  FROM student`
`WHERE mbr = 200`
`ORDER BY prezime;`
	`db.student.find(`
	`  {mbr:100},`
	`  {prezime:1}`
	`).sort({prezime:1});`

`UPDATE student SET prezime =`
`'Shostakovich'`
`WHERE mbr = 100;`
	`db.student.update(`
	`{ mbr: 100 },`
	`{$set : { prezime : 'Shostakovich' } }`
	`);`

`DELETE FROM student WHERE mbr = 100;`	`db.student.remove(`
	`{ mbr: 100 }`
	`);`

Agregatni model - KV i Dokument baze podataka

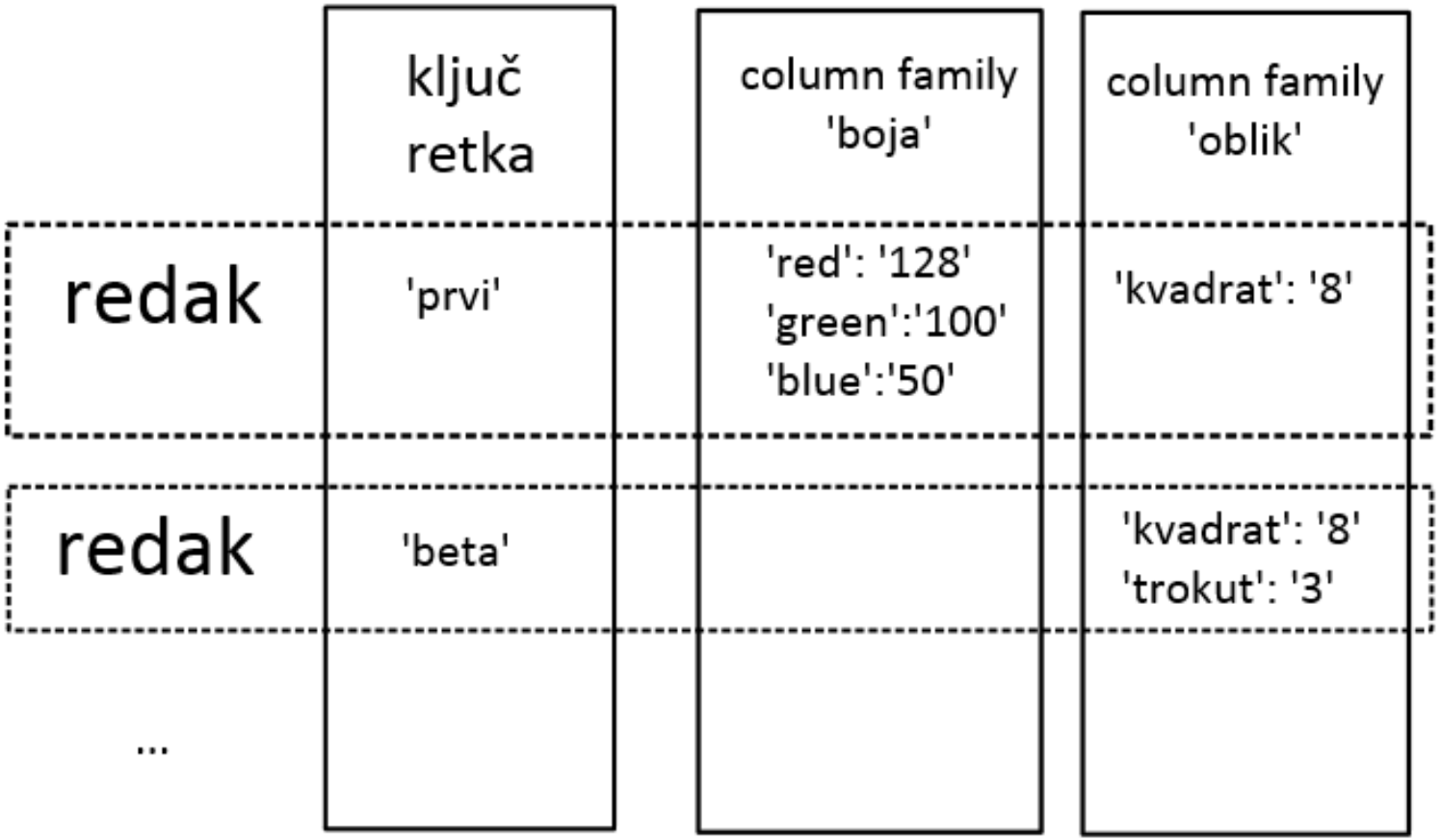
- Ključ-vrijednost i Dokument BP zasnovane na agregatnom tipu podataka
- Ključ vrijednost
  - ✓✗ Dohvat na temelju ključa
  - ✓✗ Sadržaj je "blob"
- Dokument baze podataka
  - ✓ Dohvat na temelju upita
  - ✓ Dohvat dijela dokumenta
  - ✓ Indeksiranje
  - ✗ Ograničenja na sadržaj
- U praksi, granica između KV i D baza podataka je nejasna

CF baze podataka

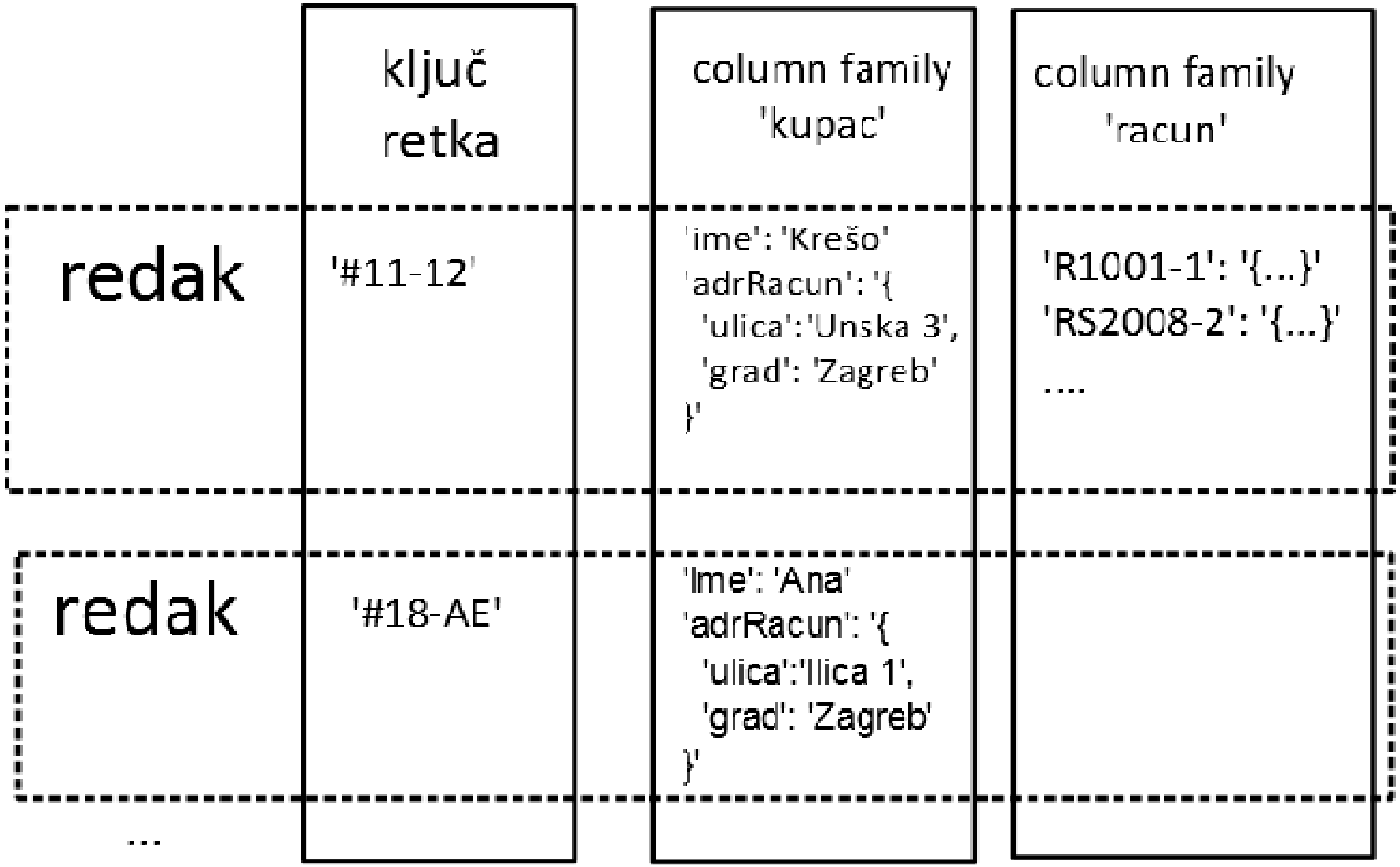
- *Chang et al. [2006], Bigtable: A Distributed Storage System for Structured Data*
- **Model podatka: *column family***
- **Nije tablica**
- **Dvorazinska mapa, dvorazinski agregat**
- Prvorazinski ključ: ključ retka
- Drugorazinski ključ: ključ stupca
- Svaki stupac je član neke familije stupaca

CF primjer (1)

- `get('prvi', ' boja:green')`

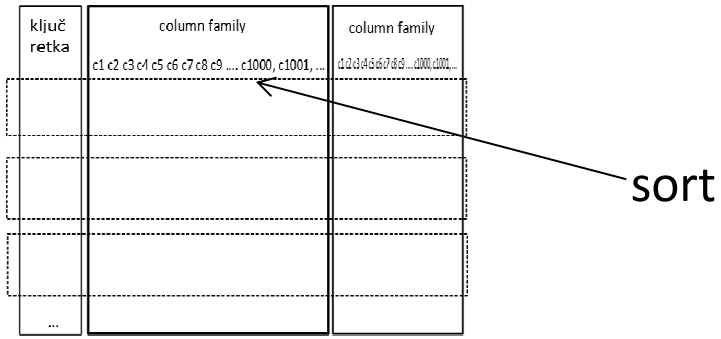


CF primjer (2)

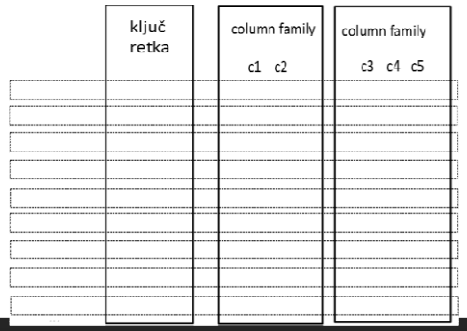

ključ
retka
column family
'kupac'
column family
'racun'
redak
'#11-12'
'ime': 'Krešo'
'adrRacun': '{
'ulica':'Unska 3',
'grad': 'Zagreb'
}'
'R1001-1': '{...}'
'RS2008-2': '{...}'
....
redak
'#18-AE'
'Ime': 'Ana'
'adrRacun': '{
'ulica':'Ilica 1',
'grad': 'Zagreb'
}'
...

CF komentar

- Dvostruki pogled na podatke:
  - Po **retcima**: svaki redak možemo smatrati agregatom (npr. kupac sa šifrom #11-12)
  - Po **stupcima**: svaka CF je tip zapisa, s retcima za svaki zapis (npr. narudžbe)
- Redak = spoj (JOIN) svih zapisa u svim CF-ovima
- Različiti načini ustroja:
  - *Wide row*



  - *Skinny row*



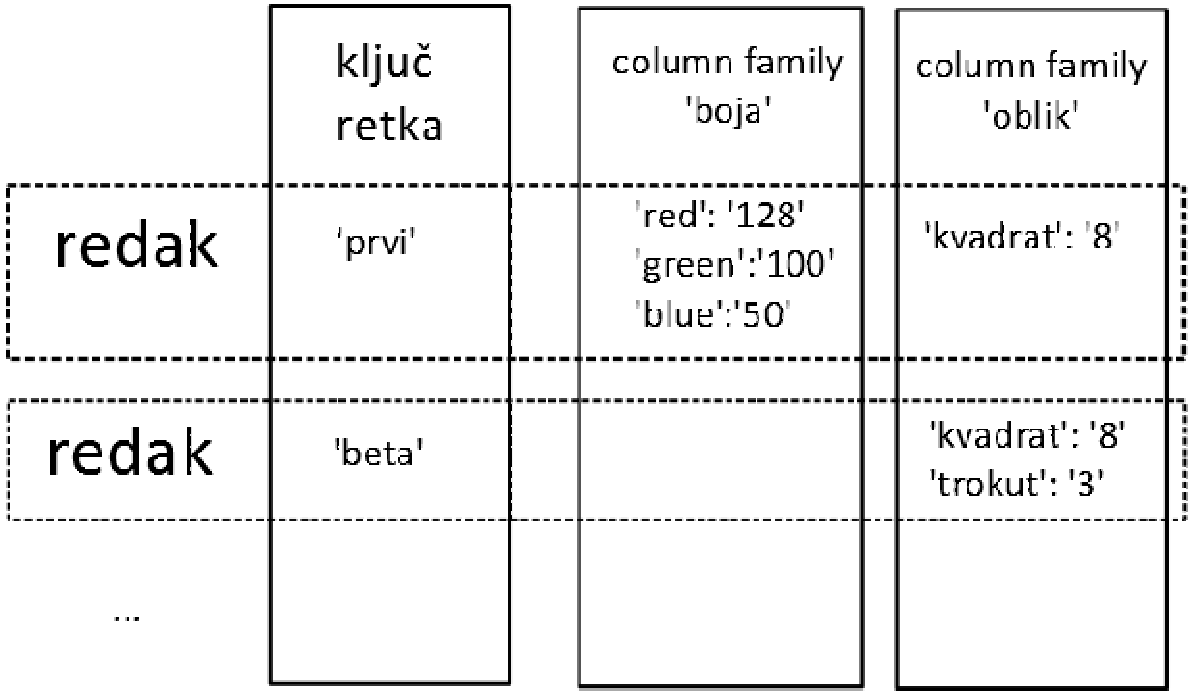
Primjer: HBase

- Stupčasta baza podataka, inspirirana s Google Big Table*
- *“Sparse, distributed, persistent multidimensional sorted map.”*
- Koristi Hadoop (= HDFS+MR)
- Samo ako se podaci broje u GB+:
  "This project's goal is the hosting of very large tables -- billions of rows X millions of columns -- atop clusters of commodity hardware."
- Dobra svojstva:
  - Skalabilnost
  - Održavanje verzija
  - Kompresija
  - Memorijski rezidentne tablice
  - *Fault tolerant*

**Chang et al. [2006], Bigtable: A Distributed Storage System for Structured Data*

Struktura

- "table"* - ključ vrijednost - mapa mapa (*map of maps*)
- "row" (sortirano)
- "column family"
- "column"
- "value", npr.
  prvi/boja:green
  je '100'

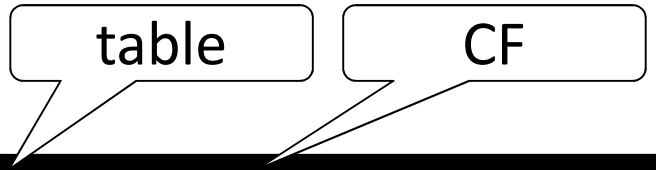


Terminologija podsjeća na relacijske baze podataka, ali su pojmovi suštinski **vrlo različiti**. Kako bi se to naglasilo, nazivi su ovdje pisani u navodnicima.

Column families

- Sve HBase operacije su atomarne na razini retka - konzistentni retci
- Zašto ne staviti sve atribute u istu obitelj?
  - Zasebna konfiguracija svake CF

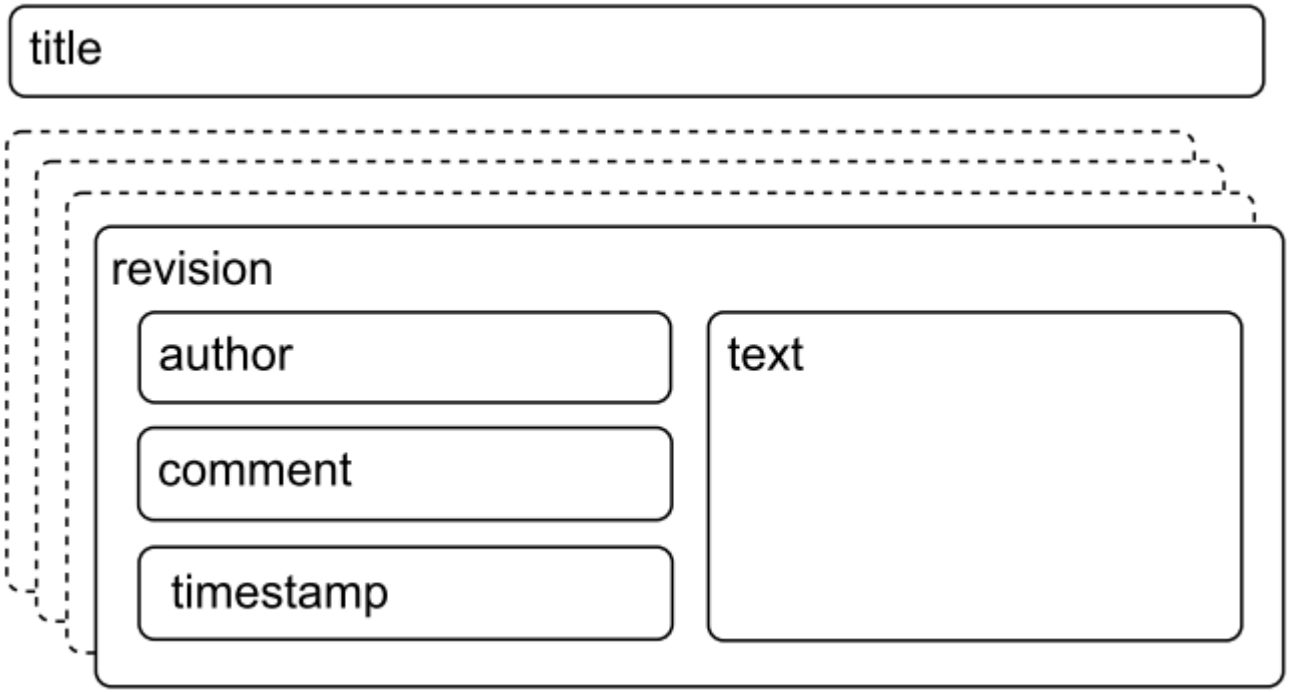
Verzije




```
hbase(main):002:0> create 'test', 'semafor'

hbase(main):004:0> put 'test', '1', 'semafor:', 'crvena'

hbase(main):005:0> put 'test', '1', 'semafor:', 'zuta'

hbase(main):006:0> put 'test', '1', 'semafor:', 'zelena'
0 row(s) in 0.0040 seconds

hbase(main):007:0> get 'test', '1'
COLUMN                          CELL
 semafor:                       timestamp=1372341842883, value=zelena

hbase(main):010:0> get 'test', '1', {COLUMN => 'semafor:', VERSIONS => 4}
COLUMN                          CELL
 semafor:                       timestamp=1372341842883, value=zelena
 semafor:                       timestamp=1372341838768, value=zuta
 semafor:                       timestamp=1372341829568, value=crvena
```

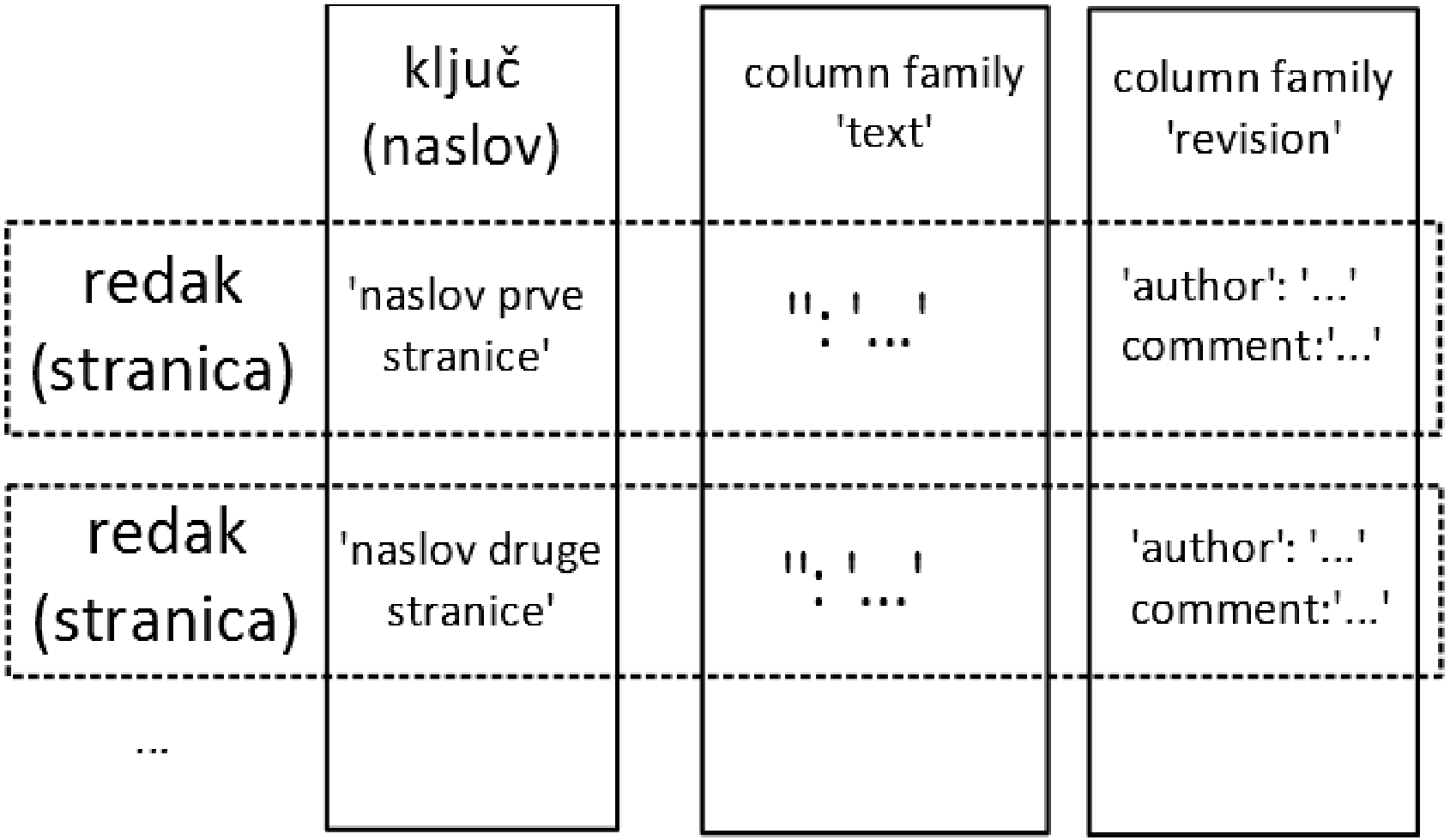
Primjer - wiki*

```
hbase(main):007:0> create 'wiki',
{NAME => 'text', VERSIONS => org.apache.hadoop.hbase.HConstants::ALL_VERSIONS },
{NAME => 'revision', VERSIONS => org.apache.hadoop.hbase.HConstants::ALL_VERSIONS}
0 row(s) in 0.2040 seconds
```



**Preuzet iz knjige Seven databases in seven weeks*

Primjer - wiki (2)



Primjer - wiki (3) – unos stranice

Unesimo prvi redak:

```
hbase(main):002:0> put 'wiki', 'Pocetna', 'text:', 'Dobrodosli'
0 row(s) in 0.6750 seconds

hbase(main):003:0> put 'wiki', 'Pocetna', 'revision:author', 'igor'
0 row(s) in 0.0050 seconds

hbase(main):004:0> put 'wiki', 'Pocetna', 'revision:comment', 'Prvi zapis u mom
wikiju'
0 row(s) in 0.0130 seconds

hbase(main):005:0> get 'wiki', 'Pocetna'
COLUMN                          CELL
 revision:author                timestamp=1372332177394, value=igor
 revision:comment               timestamp=1372332182860, value=Prvi zapis u mom
wikiju
 text:                          timestamp=1372332177312, value=Dobrodosli
3 row(s) in 0.0340 seconds
```

Nije idealno - ts je različit. Nažalost, shell ne dopušta unos više od jedne vrijednosti u jednoj naredbi, pa je za to potrebno koristiti API.

Primjer - wiki (4) - import wikipedije

- S Interneta dohvaćamo "dump" wikipedije i odmah punimo hbase tablicu (koristeći ruby skriptu koja parsira xml)*

```
hbase(main):004:0> curl
http://dumps.wikimedia.org/enwikibooks/20130626/enwikibooks-20130626-pages-
articles-multistream.xml.bz2 | bzcat | ./bin/hbase shell
./ruby/import_from_wikipedia.rb
```

```
…
 95  116M   95  110M    0     0   765k      0  0:02:35  0:02:28  0:00:07  800k 100500 records
inserted (Ba Zi/Date Selection in 1927)
101000 records inserted (Development Cooperation Handbook/The video resources linked to this
handbook/The Documentary Story/Searching for the questions to ask)
 97  116M   97  113M    0     0   756k      0  0:02:37  0:02:33  0:00:04  489k 101500 records
inserted (Template:DPL)
 97  116M   97  113M    0     0   753k      0  0:02:38  0:02:34  0:00:04  380k 102000 records
inserted (Ba Zi/HP H6)
 98  116M   98  114M    0     0   748k      0  0:02:39  0:02:36  0:00:03  380k 102500 records
inserted (Managing your business with anyPiece/Inventory system for recycle business/Storage
Packaging/Storage packaging - History)
103000 records inserted (How to Ace FYLSE/October 2012 Exam)
 99  116M   99  115M    0     0   750k      0  0:02:38  0:02:37  0:00:01  480k 103500 records
inserted (ElementarySpanish/Meeting people/Lesson 1)
100  116M  100  116M    0     0   752k      0  0:02:38  0:02:38 --:--:--  712k
```

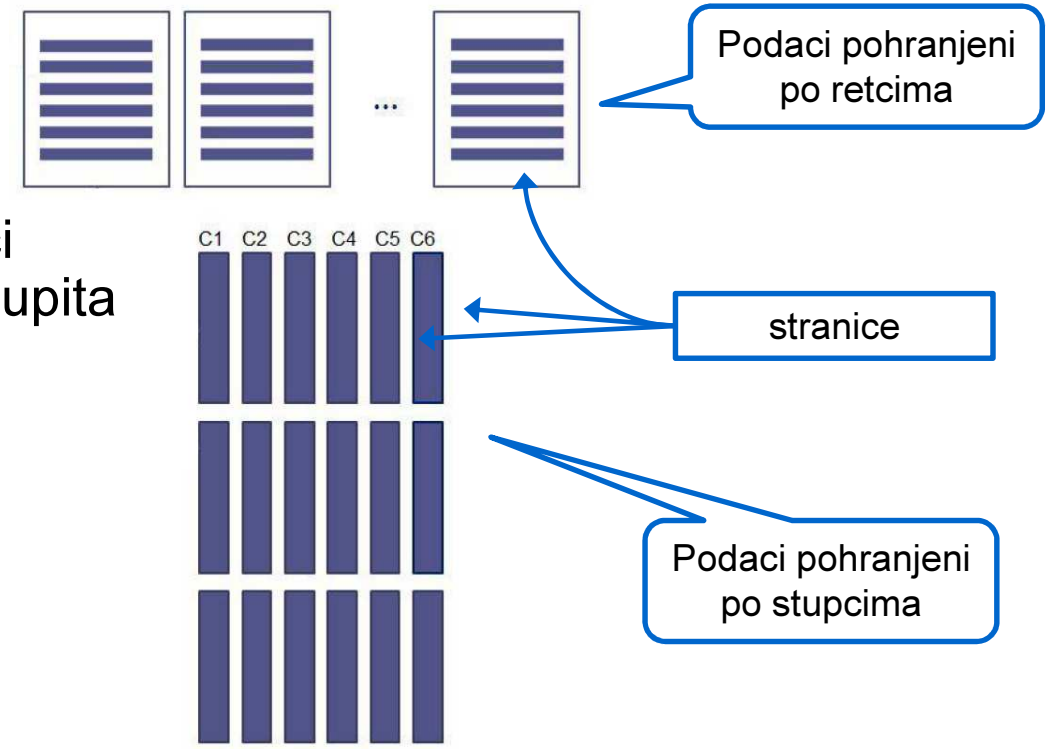
*Detaljnije opisano u knjizi *Seven databases in seven weeks*

Primjer - wiki (5) - import wikipedije

```
hbase(main):004:0> count 'wiki', INTERVAL => 10000
Current count: 10000, row: Blender 3D: Noob to Pro/Basic Animation/Rendering
Current count: 20000, row: Chemical engineering
Current count: 30000, row: Development Cooperation Handbook/The video resources linked to
this handbook/The Documentary
Story/The KFI story
Current count: 40000, row: File:NotesCornell.png
Current count: 50000, row: Hebrew Roots/The Law and the Covenants/Covenants:The Covenant
of Israeli Sovereignty
Current count: 60000, row: Mandarin Chinese/Sentences/He is a student
Current count: 70000, row: Programming:Game Maker/Intro
Current count: 80000, row: Structural Biochemistry/Protein function/Heme
group/Hemoglobin/Affinity Constant
Current count: 90000, row: Template:User language/sah
Current count: 100000, row: Wikijunior Languages/Finnish
103927 row(s) in 8.5280 seconds

hbase(main):005:0> get 'wiki', 'Chemical engineering'
COLUMN                          CELL
 revision:author                timestamp=1277299531, value=Adrignola
 revision:comment               timestamp=1277299531, value=Redirected page to
[[Subject:Chemical engineering]]
 text:                          timestamp=1277299531, value=#REDIRECT [[Subject:Chemical
engineering]]
3 row(s) in 0.0470 seconds
```

Digresija: CF nisu stupčaste baze podataka (1)

- Podaci **pohranjeni** po stupcima, npr. *C-Store*
- *Column oriented* DBMS http://en.wikipedia.org/wiki/Columnar_database
- Prednosti takvih sustava neki proizvođači (*Oracle, Informix, Microsoft, …*) uvode u obliku indeksa u klasične, *row-oriented*, baze podataka



- ✓ Dohvaćaju se samo stupci potrebni za razrješavanje upita (u tipičnoj ČT, ispod 15%)
- ✓ Lakše sažimanje
- ✓ Veća iskoristivost međuspremnika (bolje sažimanje, često korišteni stupci)

Digresija: CF nisu stupčaste baze podataka (2)

- ✓ Za red veličine (nekad i nekoliko redova veličine) brže obavljanje upita
- Korisne kad su rijetka pisanja, a česta čitanja
- Npr. Microsoft SQL Server 2012 Vertipaq*:
  - Test: 1 TB zvjezdasti spoj (1,44 milijarde redaka), 32procesora, 256 GB RAM:

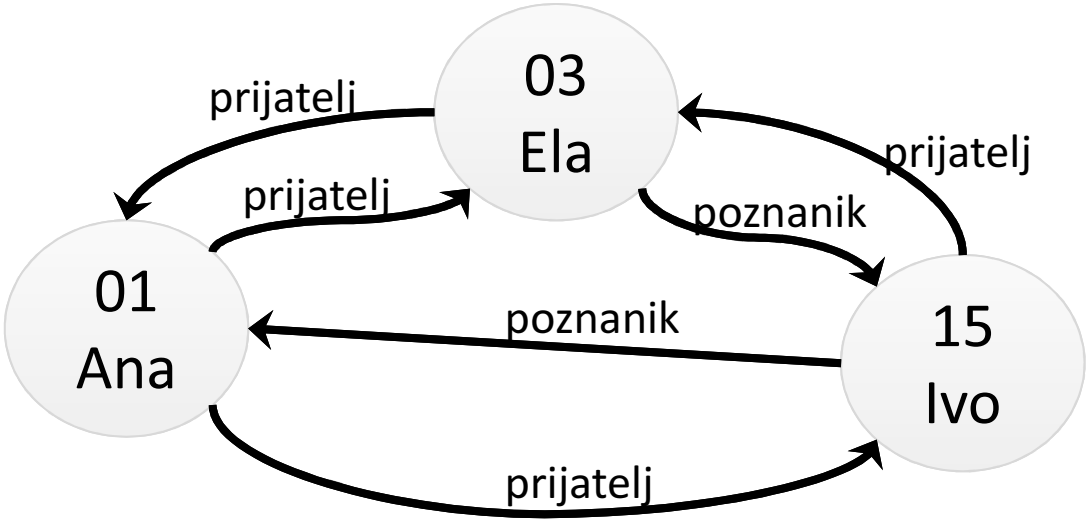
	Total CPU time (seconds)	Elapsed time (seconds)
Columnstore	31.0	1.10
No columnstore	502	501
Speedup	16X	455X

  - ✓ Može dati ubrzanja od stotine do tisuću puta, barem desetorostruko
  - ✓ Faktor sažimanje 4-20 na realnim podacima
  - ✗ Ne može se raditi INSERT
  - ✗ Sporija izgradnja indeksa 2-3 puta u odnosu na B-stablo

* http://download.microsoft.com/download/8/C/1/8C1CE06B-DE2F-40D1-9C5C-3EE521C25CE9/Columnstore%20Indexes%20for%20Fast%20DW%20QP%20SQL%20Server%2011.pdf

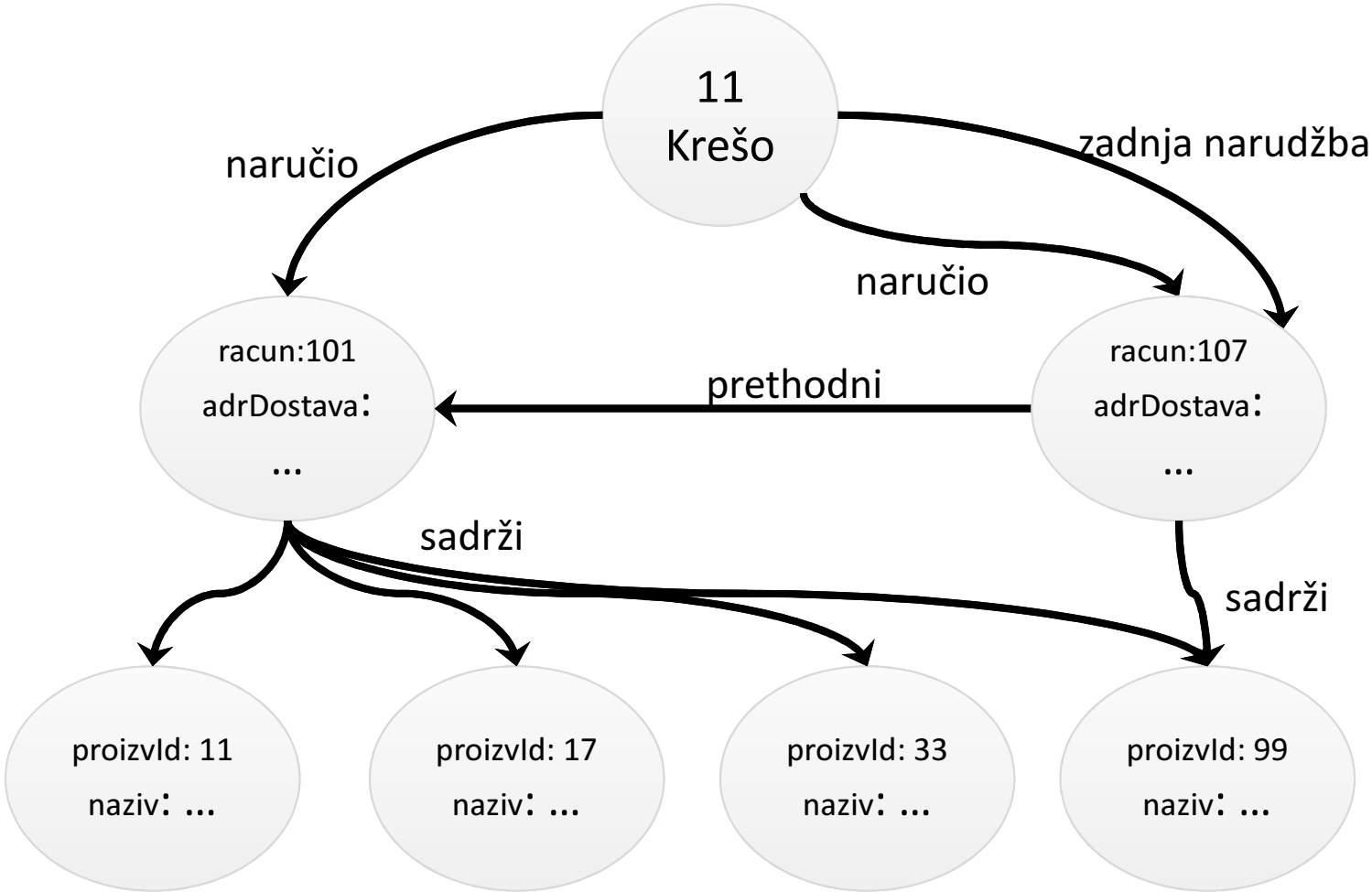
Graf baze podataka

- **Model podataka: čvorovi, bridovi, svojstva:**
  - Čvorovi mogu imati svojstva (KV parovi)
  - Bridovi imaju oznaku, smjer, početni i odredišni čvor
  - Bridovi također mogu imati svojstva
- Sučelja i upitni jezici nisu standardizirani (*Cypher, SPARQL, Gremlin*)
- Primjer:



- Neki primjeri: *Neo4j, GraphDB, DEX, FlockDB, InfoGrid, OrientDB, Pregel, …*

Graf baze podatka - primjer



Primjer: Neo4j

- “whiteboard friendly,”
- All about relationships
- U raznim „veličinama”:
  - *Emedded*
  - *Cluster support*, MS replikacija
- Može pohraniti desetke milijardi čvorova i bridova

- Upitni jezici:
  - Cypher
  - ~~Gremlin~~

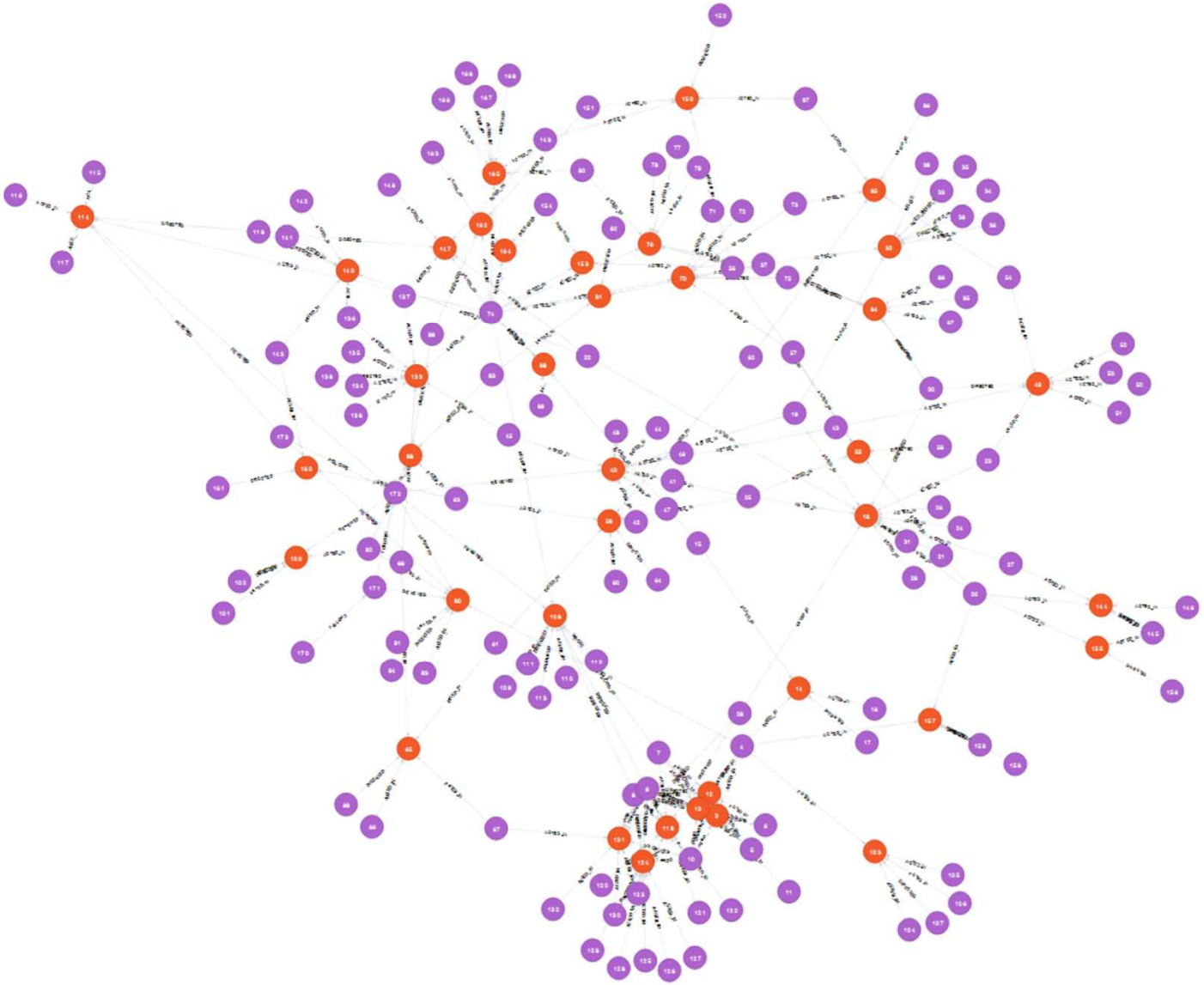
Cypher upitni jezik

```
START
[MATCH]
[WHERE]
RETURN [ORDER BY] [SKIP] [LIMIT]
```

Primjer: unos zapisa

```
CREATE (TheMatrix:Movie {title:'The Matrix', released:1999, tagline:'Welcome to the
Real World'})
CREATE (Keanu:Person {name:'Keanu Reeves', born:1964})
CREATE (Carrie:Person {name:'Carrie-Anne Moss', born:1967})
CREATE (Laurence:Person {name:'Laurence Fishburne', born:1961})
CREATE (Hugo:Person {name:'Hugo Weaving', born:1960})
CREATE (AndyW:Person {name:'Andy Wachowski', born:1967})
CREATE (LanaW:Person {name:'Lana Wachowski', born:1965})
CREATE (JoelS:Person {name:'Joel Silver', born:1952})
CREATE
  (Keanu)-[:ACTED_IN {roles:['Neo']}]->(TheMatrix),
  (Carrie)-[:ACTED_IN {roles:['Trinity']}]->(TheMatrix),
  (Laurence)-[:ACTED_IN {roles:['Morpheus']}]->(TheMatrix),
  (Hugo)-[:ACTED_IN {roles:['Agent Smith']}]->(TheMatrix),
  (AndyW)-[:DIRECTED]->(TheMatrix),
  (LanaW)-[:DIRECTED]->(TheMatrix),
  (JoelS)-[:PRODUCED]->(TheMatrix)

CREATE (Emil:Person {name:"Emil Eifrem", born:1978})
CREATE (Emil)-[:ACTED_IN {roles:["Emil"]}]->(TheMatrix)

CREATE (TheMatrixReloaded:Movie {title:'The Matrix Reloaded', released:2003,
tagline:'Free your mind'})
```

Primjer: filmovi i glumci



Cypher - primjeri

- Dohvati Toma Hanksa:

```
MATCH (tom {name: "Tom Hanks"})
RETURN tom
```

- Dohvati čvora naslova „Cloud Atlas”:

```
MATCH (cloudAtlas {title: "Cloud Atlas"})
RETURN cloudAtlas
```

- Dohvati imena deset osoba (čvorova tipa Person):

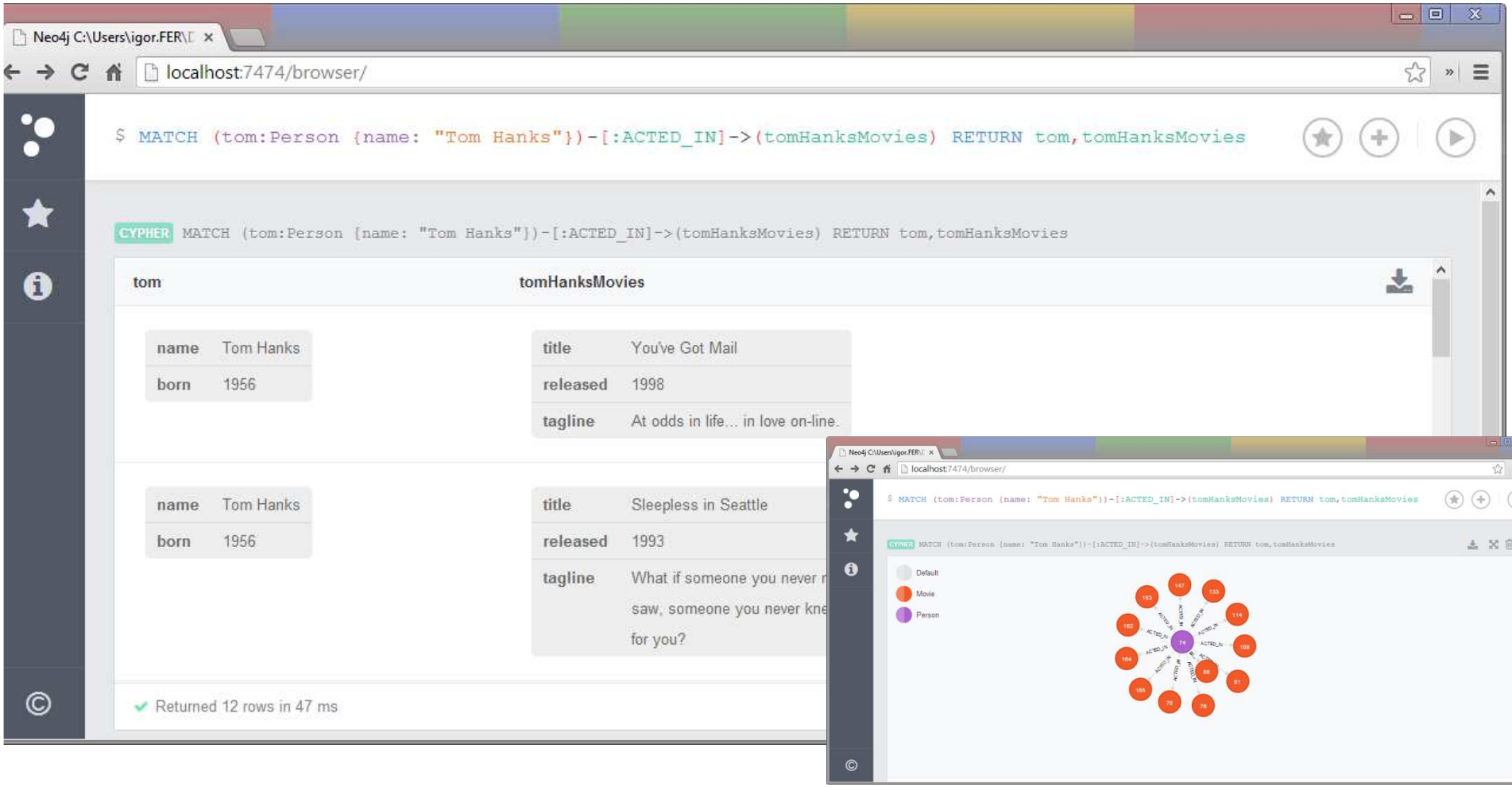
```
MATCH (people:Person)
RETURN people.name LIMIT 10
```

- Dohvati filmove iz devedesetih:

```
MATCH (nineties:Movie)
WHERE nineties.released > 1990 AND nineties.released < 2000
RETURN nineties.title
```

Primjer: filmovi u kojima je glumio TH

```
MATCH (tom:Person {name: "Tom Hanks"})-[:ACTED_IN]->(tomHanksMovies)
RETURN tom,tomHanksMovie
```

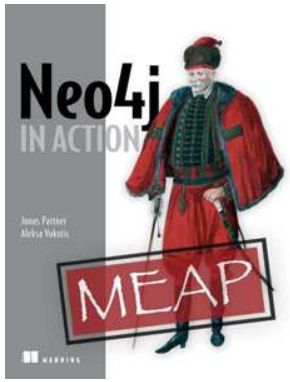


Relacijske baze i veze

- *"relational databases deal poorly with relationships"* ☺
- Prijatelji od prijatelja mojih prijatelja? (FOAF, napredni SQL)

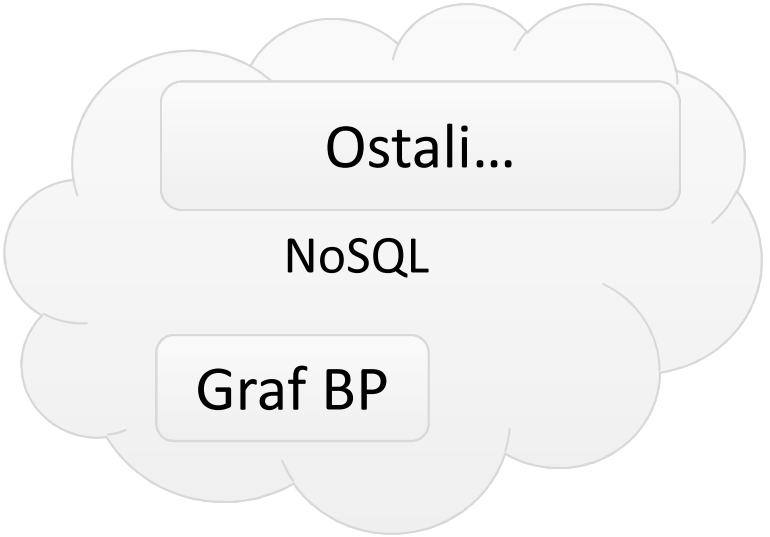
Depth	Execution Time – MySQL	Execution Time – Neo4j
2	0.016	0.010
3	30.267	0.168
4	1,543.505	1.359
5	Not Finished in 1 Hour	2.132

http://www.neotechnology.com/how-much-faster-is-a-graph-database-really/



Graf baza podataka

- *"Strange fish in SQL pond"*



- Razgrađuje podatke u još manje jedinice od RBP-a

- Nisu pogodne za distribuciju
- Upitni jezik
- ACID
- Zajedničko s ostalima: ne-relacijski model, popularnost
- Pogodni za složene, polu-strukturirane, jako povezane podatke

Baze podataka bez sheme podataka

- *Schemaless*
- Kod RBP je potrebno **unaprijed** definirati **fiksnu** shemu
- NoSQL baze podataka nemaju shemu podataka:
  - ✓ Moguće je pohraniti "bilo što"
  - ✓ Lako je mijenjati "što se sprema", dodati nove podatke
  - ✓ Lako je raditi s heterogenim podacima
    - **Implicitna shema** ipak postoji - u aplikaciji ☹
  - ✗ BP ne poznaje implicitnu shemu - ne može iskoristiti to znanje za efikasniju pohranu i dohvat
  - ✗ BP ne može provoditi validaciju (integritet)
- Što kada više različitih aplikacija pristupa BP-u?
- Fleksibilnost zbog nedostatka sheme vrijedi samo unutar granica agregata (~ što ako treba promijeniti granice agregata?)

Migracija sheme

- Npr. mijenamo ime atributa, dodajemo atribut, tablicu, ...
- **RDB**
  - SQL delta skripte, DBDiff
  - Automatizacija pomoću alata: *DBDeploy, Liquibase, DBMantain*
  - Problem kada imamo više aplikacija, posebice legacy
    - *Transition phase* (npr. koristimo trigger koji održava staru/novu vrijednost atributa, dok se ne poprave sve legacy aplikacije)
- **NoSQL**
  - Sitnice trivijalno (npr. dodamo atribut), ali inače ništa lakše
  - Inkrementalna migracija:
    - *schema version*
    - Postupno mijenjanje kod snimanja ☹
  - Što ako kod GraphDB promijenimo oznaku brida?
    - Možemo održavati paralelene bridove (stari+novi), dodati metapodatke (verzija, vremenska oznaka i sl.)

Materijalizirane virtualne relacije

- *Materialized views*
- Inspirirano virtualnim relacijama u RBP-ovima
- NoSQL nemaju virtualne relacije
- Materijalizirane virtualne relacije - upiti izračunati unaprijed i pohranjeni na *disku*
- Npr. prodaja određene grupe proizvoda u tekućem mjesecu
- Pogodno za podatke s puno čitanja i malo pisanja
- Dva pristupa:
  - *Eager* - ažuriranje kod unosa
  - *Stale* - ažuriranje nakon unosa, periodički, asinkrono. Može se napraviti i izvan BP i snimiti (ali češće se to radi kroz raspoložive mehanizme NoSQL baze podataka)

Model podataka – zaključak (1)

- Kod modeliranja podataka, generalno je pravilo da se podaci denormaliziraju kod pisanja, kako bi bili pogodniji za ciljane vrste čitanja
- BP koje sadrže agregate (KV,D,CF) se teže nose s odnosima koji prelaze granice agregata
- Grafovske BP razlažu podatke na "sitne" dijelove. Prikladne su za podatke koji imaju složene međuovisnosti
- BP koje nemaju shemu su fleksibilnije kod unosa i omogućuju lako promjenu atributa koji se unose, ali **implicitna shema** ipak postoji - "negdje" u aplikacijskom kodu.

Model podataka – zaključak (2)

Aggregete oriented

Key-Value

Document

Column-family

Graph

Schemaless