[P273]

Пројектовање база података

12а

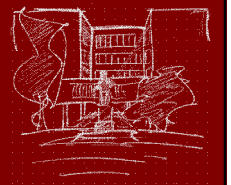
Саша Малков

Универзитет у Београду
Математички факултет

2023/2024

[P273]

Пројектовање база података



Саша Малков

Тема 13

Нерелационе базе података

(наставак)

Врсте нерелационих БП

- Основне врсте нерелационих база података:
  - парови кључева и вредности
  - складишта широких колона
  - складиште докумената
  - графовске базе података
  - објектне базе података
  - табеларне базе података
  - складиште торки
  - вишевредносне
  - мултимоделне
  - *XML* базе података
  - складишта садржаја (докумената, ресурса,...)
  - системи за претраживања
  - базе временских серија

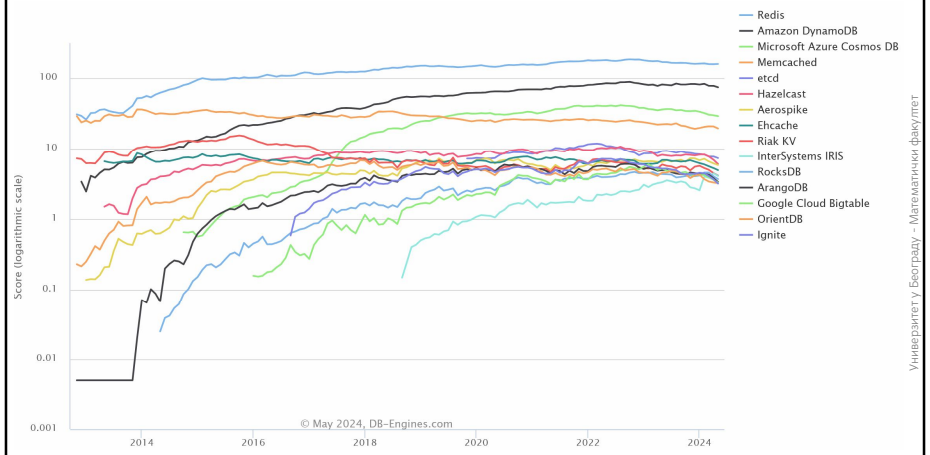
Базе парова кључева и вредности (1)

- Енгл. *key-value databases*
  - неки аутори одвајају посебно тзв. *базе торки* (енгл. *tupple store*)
- Структура:
  - свака колекција података је једна хеш табела
  - подацима се приступа ***искључиво*** на основу познатог кључа
    - или секвенцијално
  - вредности су у начелу једноставне или врло ниско стрруктуриране
    - ако вредности имају високу сложеност, обично се БП класификује као база са проширивим слоговима или база докумената

Базе парова кључева и вредности (2)

- Доприноси:
  - подржавају веома велике скупове података
  - веома су брзе
  - обично подржавају аутоматско реплицирање и хоризонтално партиционисање колекција
- Слабости:
  - подразумева се висок ниво редундантности
  - сложене структуре се имплементирају великим бројем колекција
  - ако су подаци “густо” повезани, ефикасност драстично опада
  - у основи немају механизме за очување интегритета
    - често не обезбеђују чак ни атомичност трансакција
  - не могу да се претражују по подацима
    - сваки индекс је нова колекција података...
  - услов тражења је искључиво фиксна вредност кључа или распон вредности кључа
- Примери:
  - *Redis*, *Memcached*, *Hayelcast*, *Riak*, *Oracle NoSQL*,...

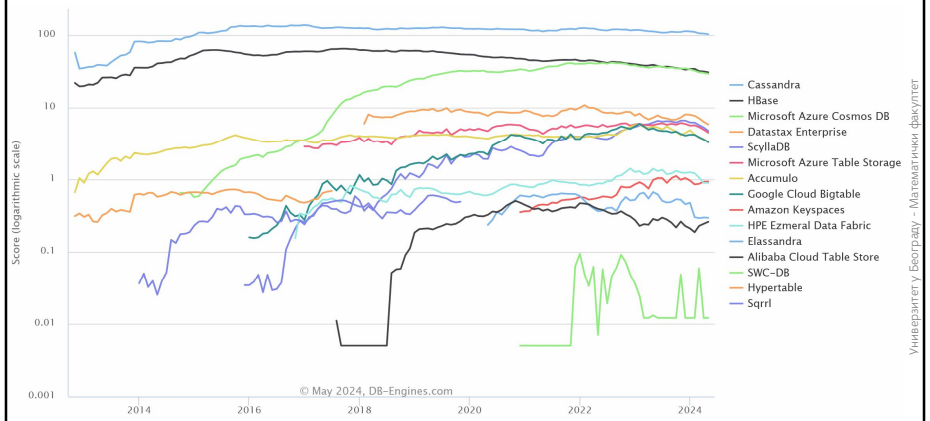
Базе парова кључева и вредности



Базе са проширивим слоговима

- Енгл. *wide-column databases*, *extensible record stores*
- Структура:
  - у основи слично као базе парова кључева и вредности
  - вредност представља колекцију великог броја парова имена и вредности
    - обично број парова бар начелно није ограничен
    - све скупа изгледа као база парова кључева и вредности са две димензије
- Доприноси:
  - подржавају веома велике скупове података
    - и велике податке и велики број података
  - веома су брзе
    - осим у неким случајевима вредности са веома сложеном структуром
  - већина подржава аутоматско реплицирање и хоризонтално партиционисање колекција
- Слабости:
  - све као за базе парова кључева и вредности
- Примери:
  - *Cassandra*, *BigTable*, *Druid*, *Accumulo*, *HDFS* (*Hadoop Distributed File System*), *HBase*

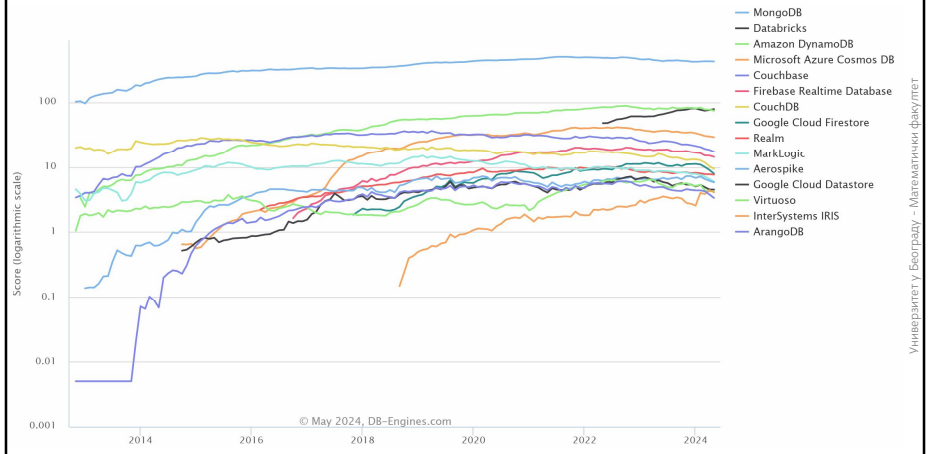
Базе са проширивим слоговима



Базе докумената

- Енгл. *document databases*
- Структура:
  - у основи личи на базу парова кључева и вредности
  - сваки документ представља вредност коме се додељује кључ
  - документи се записују у структурираним (*XML*, *YAML*, *JSON*, *BSON*,...) или неструктурираним облицима (нпр. *PDF*)
  - сваки документ има метаподатке
    - обично неструктурирање или са врло флексибилном структуром
  - тело документа (ако је структуриран) и метаподаци се аутоматски интерно структурирају
    - корисник не мора ништа да зна о томе
- Доприноси:
  - обично веома једноставан и ефикасан рад
  - обично подржавају бар полуаутоматско реплицирање и хоризонтално партиционисање колекција
    - често само репликација типа главни-подређени
- Слабости:
  - релативно ограничен домен примене
  - многе имплементације не омогућавају ад-хок упите и мењања података
  - неке имплементације не трпе велику учесталост мењања података
- Примери:
  - *MongoDB*, *CouchDB*, *MarkLogic*, *RavenDB*, *Amazon DynamoDB*, *Google Cloud Datastore*

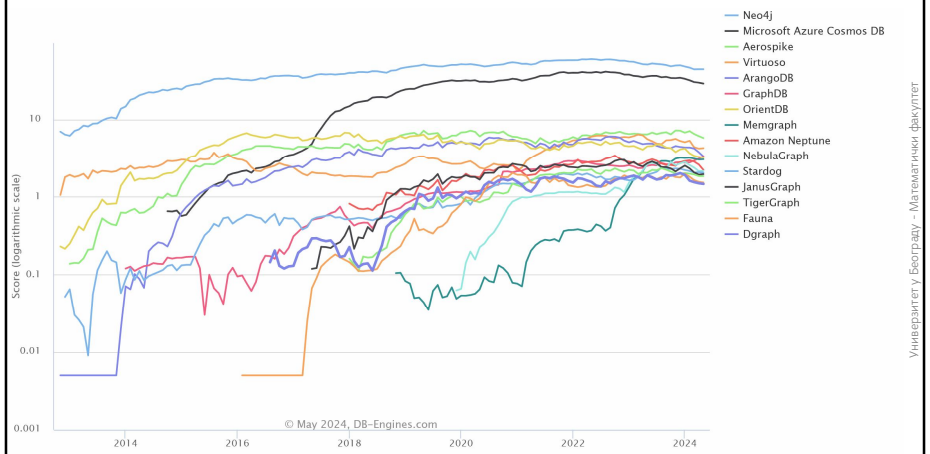
Базе докумената



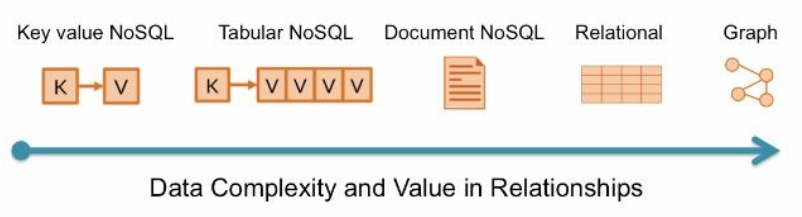
Графовске базе података

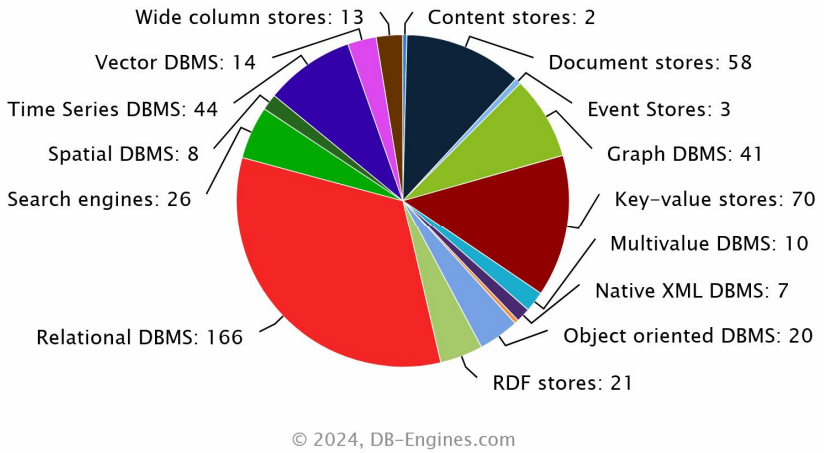
- Енгл. *graph databases*
- Структура:
  - базу чине чворови и везе међу њима
  - нема чврсте схеме, подаци имају веома слободну структуру
  - акценат је на односима између података, а не на структури података
- Доприноси:
  - насупрот другим нерелационим БП, веома су ефикасне када су у питању уобичајене операције са графовима
  - неке подржавају трансакције и *ACID* режим услова
  - обично само репликација типа главни-подређени
- Слабости:
  - релативно ограничен домен примене
  - нису погодне ван свог домена
- Примери:
  - *Neo4J*, *OrientDB*, *ArangoDB*, *Allegro*, *Virtuoso*, *MS Azure Cosmos DB*, *Titan*, *GraphDB*

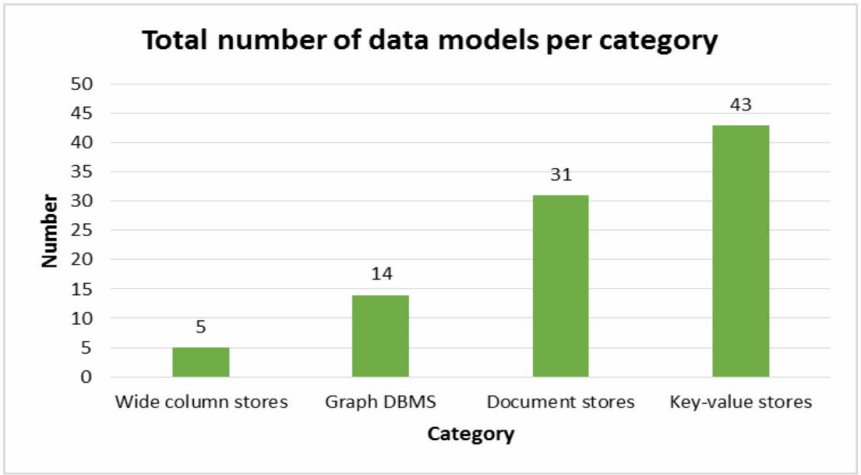
Графовске базе



Сложеност структуре података







(solidIT, 2014)

Разноврсност модела

- Разноврсност модела отежава стандардизацију и учење
- Код база докумената то представља нешто мањи проблем, зато што је већи део структуре базе података имплицитно одређен и корисник му не приступа непосредно
- Код база парова кључева и вредности имамо изузетно велику збрку
- Приметимо да су базе са проширивим слоговима најближе неком виду стандардизације
- Као и код претходног дијаграма, и овде има доста резерве
  - неки од модела су међусобно слични и деле концепте

Још?

- Препознају се још неке категорије база података
  - ***Вишедимензионе базе података***
  - ***Решеткасте базе података***
  - ***XML базе података***
  - ***Објектне базе података***
  - ***Научне базе података***
  - ***Векторске базе података***
  - ***Вишемоделне*** - представљају спој више модела
    - на пример: *ArangoDB*
  - и друге...
- Прилично исцрпан преглед нерелационих база података постоји на страници:
  - http://nosql-database.org/

Алтернативне релационе базе

- У последње време се као алтернатива нерелационим базама развијају релационе базе прилагођене дистрибуирању
  - претпоставља се употреба више чворова
  - чворови имају јасно раздвојене намене (на пример: контролни, процесни и складишни чворови)
  - погодне за облак
- Примери
  - *MS Azure Synapse*
  - *YDB*

Пројектовање нерелационих база података

- Поступак пројектовања има исте основне кораке као и у случају РБП, али се ти кораци разликују
  - Концептуално пројектовање
    - обрађени кораци су у основи исти
    - потребно је прикупити информације које ће помоћи при одабиру циљног модела података и СУБП
  - Логичко пројектовање
    - као први корак се обавља одабир модела података и СУБП
    - одвија се у односу на конкретан модел и СУБП
  - Физичко пројектовање
    - одвија се у односу на конкретан модел и СУБП

Одабир циљног модела података

- Одабир циљног модела података се одвија на основу
  - специфичности направљеног концептуалног модела базе података
  - на основу специфичних информација о планираној употреби
- Постоји велики број специфичних врста и имплементација нерелационих БП
  - потребно је проценити који од њих је концептуално и имплементационо најближи стварним потребама и конкретном концептуалном моделу

Погрешан избор?

- Веома често се прави погрешан избор модела и СУБП
- Пример:
  - прави се база докумената и мета-података о документима
  - изабере се *MongoDB*
  - затим се испостави да је примарна сложеност у обради мета-података или чак трансакцијама на мета-подацима
  - СУБП *MongoDB* је одличан за рад са документима и пружа велику флексибилност у моделирању и раду са мета-подацима
  - ***Али није добар за трансакциону обраду***
    - подржава *ACID* трансакције
    - али су релативно неефикасне, посебно ако су иоле сложеније (ако се једна трансакција односи на више докумената)
      - јединица обраде је документ, а не податак...

Погрешан избор? (2)

- Да би се избегао погрешан избор неопходно је да се пажљиво прикупе и анализирају информације о употреби, као и концептуални модел
- Пример:
  - веб-продавница, подаци о производима
  - свака врста производа има различите карактеристике и значајне атрибуте
    - фиксна схема РБП представља проблем
    - мада могу да се користе тзв. генерички атрибути
  - *MongoDB* је у овом случају сасвим оправдан и добар избор
    - подржава варијабилну схему мета-података
    - не захтевају се сложене трансакције над мета-подацима
- Потенцијални проблем:
  - ако се покуша да се *MongoDB* користи и за наруџбенице, плаћања и слично

Како избећи погрешан избор?

- Најпре размотрити могућност примене РБП
- Ако се не наиђе на проблеме, држати се РБП
- Ако се уочавају потенцијално озбиљни проблеми у случају употребе РБП, размотрити различите моделе НРБП
- За сваки модел (СУБП) који се разматра, пажљиво сагледати
  - Да ли успешно решава уочене проблеме РБП?
  - Да ли успешно решава оно што решава и РБП?
  - Да ли уводи неке нове проблеме?

Логичко и физичко пројектовање

- Логичко и физичко пројектовање имају сличне кораке као код РБП, али су они прилагођени одговарајућем моделу података и конкретном СУБП
  - зато их нећемо детаљније разматрати
- Имајући у виду да:
  - је модел података НРБП често тесно повезан са избором СУБП;
  - да се често ради о базама података које имају специфичну и локално сложену структуру, али која глобално није сложена
    - тј. односи међу подацима могу бити веома сложени, али се укупна структура БП састоји од релативно мало елемената (за разлику од неколико стотина табела код РБП)
- ...разлика између логичког и физичког пројектовања може да се “истопи”
- Ипак, остаје препорука као код РБП – не прескакати логичко пројектовање

Литература за ову тему

- *Lynch, Gilbert,* ***Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services****, ACM SIGACT News, Vol.33/2 (2002)*
- *Eric Brewer,* ***Towards Robust Distributed Systems****, PODC (2000)*
- *Dan Pritchett,* ***BASE: An Acid Alternative****, ACM Queue, 2008/06*

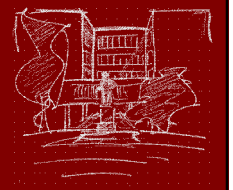
- *http://www.julianbrowne.com/article/viewer/brewers-cap-theorem*
- *http://danweinreb.org/blog/what-does-the-proof-of-the-cap-theorem-mean*
- *http://maniagnosis.crsr.net/2010/09/some-misconceptions-about-cap-theorem.html*

Универзитет у Београду - Математички факултет

[P273]

Пројектовање база података

Саша Малков



Тема 14

Apache Cassandra

Apache Cassandra

- Иницијално развијен од стране *Facebook*-а
  - Јавно објављен као пројекат отвореног кода (2008)
- Старање пренесено на фондацију *Apache* (2009)
  - Један од главних пројеката фондације *Apache* (2010)

- Међу најраспрострањенијим НРСУБП-овима
- Представља добар пример
  - значајно се разликује од РБП
  - релативно једноставна за разумевање и употребу

Универзитет у Београду - Математички факултет

Apache Cassandra

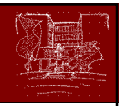
- Нерелациони СУБП
- Примарно намењен за дистрибуиране базе података
- Динамична схема
- Почива на проширеном моделу каталога
  - тј. база парова кључева и вредности
- Постепено је изграђен озбиљан упитни језик *CQL*

Универзитет у Београду - Математички факултет

Основни појмови

- *Cassandra* почива на моделу каталога
- Основни појмови су:
  - колона
  - фамилија колона
  - суперколона
  - фамилија суперколона
  - кључ
  - простор кључева
- *Опрез: ови појмови су другачији него код РБП*

Појмови (1)



- ***Колона***
  - једна вредност, праћена временом последње измене
  - појам колоне је близак појму атрибута код РБП
  - тројка (име, вредност, време измене)

```
{
        name:       "prezime",
        value:      "Petrović",
        timestamp:  123456789
}
```

  - поједностављен запис:

```
prezime: "Petrović"
```


Појмови (2)

- ***Суперколона***
  - сложена вредност, без времена последње измене
  - садржи једну или више колона
  - не постоји еквивалент код РБП, нешто као сложени атрибут

```
{ name: "licniPodaci",
  value: {
    ime: { name: "ime", value: "Goran",
           timestamp: 123456789 },
    prezime: { name: "prezime", value: "Petrović",
               timestamp: 123456789 },
    grad: { name: "grad", value: "Smederevo",
            timestamp: 123456789 }
  }
}
```


Појмови (3)

- ***Фамилија колона***
  - структура која може да садржи већи број ***редова*** (концептуално неограничено)
  - пресликава ***кључ*** у ***ред***
    - ред = скуп колона
    - ред је сличан суперколони
  - појам фамилије колона је близак појму табеле код РБП
  - концептуално представља каталог редова

```
Autori : {
    ivoAndric: {
        // pojednostavljen zapis, bez naziva i TS
        ime: "Ivo",
        prezime: "Andrić",
        ...
      },
    goranPetrovic: {
        ime: "Goran",
        prezime: "Petrović",
        ...
      },
}
```

Појмови (4)

- ***Простор кључева***
  - структура која садржи више фамилија колона или суперколона
  - концептуално одговара појму базе података или схеме код РБП

Проширени модел каталога

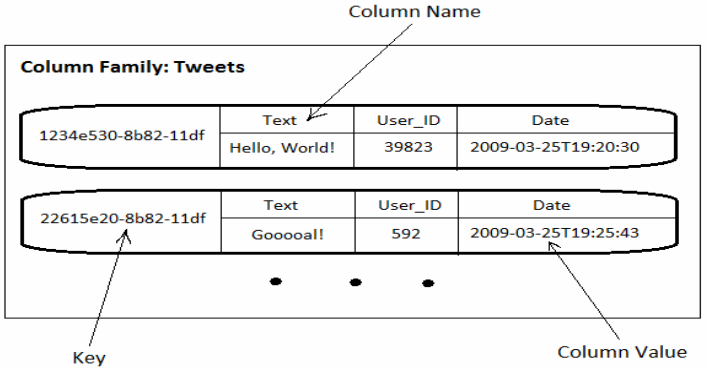
- *Cassandra* има два нивоа ***угнежденост*и**
  - први ниво је обавезан
    - чини га пар (име колоне, колона)
    - један ред (слог, *record*) се састоји од произвољног броја парова
    - парови су уређени само по имену колоне
    - ред има облик каталога
    - ред *MOPA* да има бар једну колону
  - други ниво је опцион
    - уместо да други елемент пара буде колона, то може бити колекција парова – суперколона

Имена колона

- Имена колона представљају истовремено
  - кључеве и
  - вредности
- Сваки ред (или суперколона) може да садржи произвољно много колона
  - називи колона могу да имају улогу вредности
  - вредности колона могу да буду празне

Пример: твитови

- Свака порука је посебан *ред*
  - кључ реда је кодирано време писања поруке
  - колоне одговарају атрибутима
    - представљају различите елементе описа поруке



Пример: твитови (2)

- Овај пример личи на једну табелу релационе базе:
  - један ред фамилије колона одговара реду табеле
  - једна колона одговара једном атрибуту реда
  - редови имају уједначен *формат*, тј. колоне са истим називима
    - (тако је у овом примеру, али НЕ МОРАЈУ да га имају)

- Један ред чине кључеви порука једног корисника
  - Свака порука представља посебну колону
    - означена је кодом времена писања
    - колоне су уређене по имену, па тиме и по времену писања
  - Вредности колона су празне
    - потребан је само списак кључева порука

Column Family: User_Timelines

39823	cef7be80-8b88-11df	1234e530-8b82-11df	•••
	—	—	•••
592	f0137940-8b8a-11df	22615e20-8b82-11df	•••
	—	—	•••

• • •

Пример: твитови једног корисника (2)

- Овај пример нимало не личи на табелу релационе базе података:
  - један ред фамилије колона представља колекцију података
  - једна колона одговара једном податку, а не атрибуту
  - редови немају уједначен *формат*
    - број колона није уједначен
    - називи колона нису уједначени
- Овај пример више личи на индекс

- Један ред чине подаци о порукама једног корисника
  - Једна *суперколона* се односи на једну веб локацију
    - назив означава веб локацију на коју се односи порука
    - вредност је колекција колона које идентификују поруке
      - називи колона су кључеви порука
      - вредности су празне

Column Family: User_URLs

Super Column Name

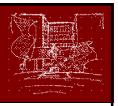
Column Name

98725	http://techcrunch.com/2010/07/09/...		http://cnn.com/world/...	•••	•••
	8fb7f240-8b91-11df	78f364e0-8b91-11df	cf128360-8b91-11df	•••	
	—	—	—	•••	

• • •

Column Value

Пример: сви твитови са неким *URI*-јем (2)



- Овај пример још мање личи на табелу релационе базе података:
  - један ред фамилије колона представља колекцију сложених података - суперколона
  - једна суперколона представља колекцију колона, па чак ни структура суперколона није уједначена
  - као и у претходном примеру, редови немају уједначен *формат*

Партиционисање

- Свака фамилија колона је хоризонтално партиционисана
- Партиционисање се врши сецкањем (*хеширање*) по кључу, на одговарајући број партиција

CQL

- У првим верзијама подаци су се користили са семантиком фамилија колона, искључиво кроз интерфејс каталога
- Нове верзије имају унапређен упитни језик *CQL3*
  - *Cassandra Query Language*
  - Намерно и именом и синтаксом личи на *SQL*
- Терминологија је приближена терминологији релационих база података
  - олакшано моделирање и учење
  - сада мало прикривена семантика

CQL

- простор кључева = база података
- табела = фамилија колона
- индекс = фамилија колона која садржи колекцију кључева неке табеле и служи за брзо претраживање

Прављење простора кључева

```
CREATE KEYSPACE Test1
WITH REPLICATION = {
   'class' : ‘SimpleStrategy',
   ‘replication_factor' : 3
};
```

- класа стратегије одређује начин реплицирања
- фактор репликације одређује колико реплика ће имати свака партиција

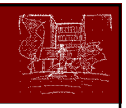
Стратегије репликације

- *SimpleStrategy*
  - подразумевана стратегија
  - униформна у односу на чворове
  - параметром репликације се одређује на колико чворова желимо да се понови свака реплика
  - добра за основне примене
- *NetworkTopologyStrategy*
  - напреднија стратегија
  - омогућава оптимизацију у односу на центре података
  - параметром репликације се одређује на колико чворова у сваком центру података желимо да се понови свака реплика
  - боља за примену у пракси

Брисање простора кључева

```
DROP KEYSPACE Test1
```


Промена простора кључева



```
ALTER KEYSPACE Test1
WITH REPLICATION = {
   'class' : 'SimpleStrategy',
   'replication_factor' : 3
};
```

Употреба простора кључева

- Еквивалент повезивању са базом података

  USE Test1

Прегледање простора кључева

- Списак свих простора кључева

  DECRIBE KEYSPACES

- Опис простора кључева (списак и структура табела)

  DECRIBE KEYSPACE [<*назив*>]

  - Ако се не наведе назив, описује се простор кључева који се тренутно користи

Типови података

- Текстуални
- Нумерички
- Логички
- Колекције
- Универзални јединствени идентификатори
- Остало

Типови података

- Текстуални
  - ascii – 8-битне ниске
  - inet – IP адреса у формату IPv4 или IPv6
  - text – ниска у формату UTF8
  - timestamp – форматирана ниска са датумом и временом
  - varchar – ниска у формату UTF8

Типови података

- Нумерички
  - bigint – 64-битни означени број
  - counter – 64-битни број посебне намене
  - decimal – децимални тип промењиве тачности (*Java*)
  - double – 64-битни покретни зарез IEEE-754 (*Java*)
  - float – 32-битни покретни зарез IEEE-754 (*Java*)
  - int – 32-битни означени број
  - timestamp – форматирана ниска са датумом и временом
  - varint – цео број произвољне прецизности/дужине (*Java*)
- Логички
  - boolean – логичка вредност, *true* или *false*

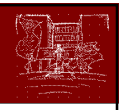
Типови података

- Колекције података
  - list – уређена листа са једним или више елемената
    - list<int>, list<text>
  - map – *JSON* каталог у облику: { literal : literal, literal : literal ... }
    - map<uuid,int>, map<int,text>
    - одговара листи колона
  - set – неуређен скуп са једним или више елемената
    - set<int>, set<text>
  - tuple – група од 2 или 3 податка
    - tuple<int,text,float>
  - није допуштено да елемент колекције буде колекција, попут: list<list<...>>

Типови података

- Остало
  - blob – произвољан низ бајтова
- Универзални јединствени идентификатори
  - uuid – јединствени ид. у стандардном формату *UUID*
  - timeuuid – јединствени ид. *типа 1*

Прављење табеле



- Намерно слично као у *SQL*-у

```
CREATE TABLE student (
   indeks text PRIMARY KEY,
   ime text,
   prezime text,
   smer text
);
```

Прављење табеле (2)

```
CREATE TABLE student (
    indeks text,
    ime text,
    prezime text,
    smer text,
    PRIMARY KEY( indeks )
);
```


Додавање података

```
INSERT INTO student ( indeks, ime, prezime, smer )
VALUES ('10442014', 'Petar', 'Petrovic', 'R1' );

INSERT INTO student ( indeks, ime, prezime, smer )
VALUES ('10432014', 'Ivan', 'Markovic', 'M1' );

INSERT INTO student ( indeks, ime, prezime, smer )
VALUES ('10452014', 'Tijana', 'Zivkovic', 'R1' );
```


Мењање података

```
UPDATE student
SET smer = 'V1'
WHERE indeks = '10432014'
```


Брисање података

```
DELETE ime
FROM student
WHERE indeks = '10432014';
```

- брише изабране колоне изабраних редова

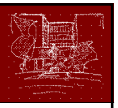
```
DELETE FROM student
WHERE indeks = '10432014';
```

- брише изабране редове

```
TRUNCATE student;
```

- брише све редове

Читање података



```
SELECT * FROM student
WHERE indeks='10452014';

SELECT * FROM student
WHERE ime='Tijana';
```

- зависно од верзије и конфигурације, претраживање по колонама за које не постоји индекс није дозвољено

Прављење и брисање индекса

```
CREATE INDEX student_ime ON student (ime);

CREATE INDEX student_prezime ON student (prezime);

CREATE INDEX student_smer ON student (smer);

DROP INDEX ...
```


Употреба колекција

```
ALTER TABLE student ADD napomene list<text>;
```

- постављање целе листе

```
UPDATE student SET napomene = [ ‘aaa', ‘bbb' ]
WHERE indeks = '10432014';
```

- додавање елемената на почетак и крај је брза операција, не чита листу

```
UPDATE student SET napomene = [ ‘xxx' ] + napomene
WHERE indeks = '10432014';

UPDATE student SET napomene = napomene + [‘yyy’]
WHERE indeks = '10432014';
```


Употреба колекција (2)

- мењање елемената је спорије, зато што чита листу

```
UPDATE student SET napomene[2] = ‘222'
WHERE indeks = '10432014';
```

- брисање елемената такође чита листу

```
DELETE napomene[2]
FROM student
WHERE indeks = '10432014';

UPDATE student
SET napomene = napomene - [‘aaa’]
WHERE indeks = '10432014';
```

За вежбу…

- Инсталирати и испробати СУБП *Cassandra*
- Размотрити могућности и посебно *CQL*
  - прикривена структура индекса
  - сличности и разлике у односу на *SQL*
  - …

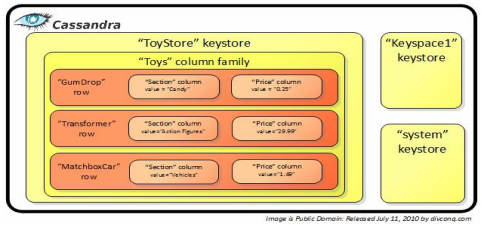


Image is Public Domain: Released July 11, 2010 by divconq.com

Литература за ову тему

- *Sugam Sharma*, ***An Extended Classification and Comparison of NoSQL Big Data Models***, *Arxiv*, *2015*.
- *Званична веб страна пројекта* ***Cassandra***
  - *http://cassandra.apache.org/*
- *Званична документација за* ***CQL***
  - *http://www.datastax.com/documentation/cql/3.1*
- ***DB-Engines***
  - *https://db-engines.com/en/*
- ***Интернет…***