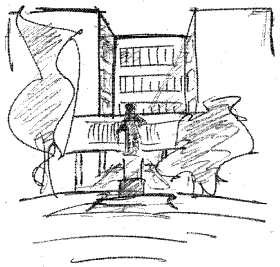
[P273]

Пројектовање база података

12



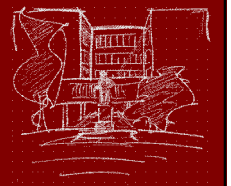
Саша Малков

Универзитет у Београду
Математички факултет

2023/2024

[P273]

Пројектовање база података



Саша Малков

Тема 12

Теорема *CAP*

Својства (услови) *CAP*

- При прављењу дистрибуираних система (не само база података) као најважнији циљеви се истичу три важна својства:
  - конзистентност (*consistency*)
  - расположивост (*availability*)
  - прихватање раздвојености (*partition tolerance*)
- Ова три својства се скраћено означавају као “*CAP*”

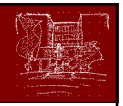
Конзистентност

- “Конзистентан систем функционише као целина или не функционише уопште”
- “Сва читања, на свим чворовима, морају да дају исти резултат”
- **“Резултат операције (читања) *никада не зависи* од чвора на коме се извршава”**
- Разликује се од истог термина у контексту особина трансакција (*ACID*)
  - “... без обзира на било каква дешавања са базом или системом, ако систем ради онда увек мора да врати исправан резултат...”

Расположивост

- **“Систем је *увек* расположив”**
  - расположивост се практично дефинише као одзивност система у неким гарантованим границама
  - (у овом контексту се разматра само време одзива)
- Парадокс је да систем обично није расположив управо онда када је најпотребнији
  - у време када је расположивост најпотребнија, онда је и најтеже остварива, зато што је систем тада обично највише оптерећен
  - ако је систем расположив када није потребан, то нема значаја

Толеранција раздвојености



- **“Ниједан скуп проблема, осим потпуног отказивања, не сме да произведе неисправан одзив система”**
  - тј. систем мора да прихвата делимичне отказе комуникације и да наставља исправно функционисање
  - (у овом контексту се разматра исправност одзива)
- Повремени прекиди комуникације међу чворовима су неизбежни
- Раздвојеност (*partition*) је стање комуникационе мреже у коме су делови система (обично услед квара) подељени на партиције (два или више раздвојених скупова чворова) између којих не постоји комуникација
- Очекује се да систем функционише и даје исправне резултате чак и у условима раздвојености

Хипотеза *CAP*

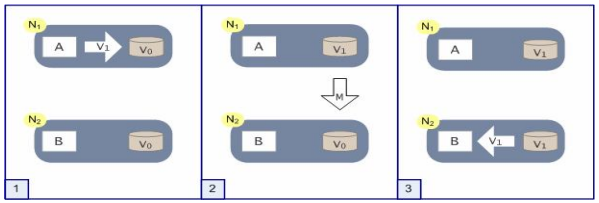
- *Eric Brewer, Berkley* (2000)
- “Није могуће дефинисати систем који задовољава све *CAP* услове (конзистентност, расположивост и толеранцију раздвојености).”
- Могуће је дефинисати систем који задовољава изабрана два од ових услова.

Теорема *CAP*

- Доказали *Gilbert, Lynch, MIT* (2002)
  - Доказ наилази на оспоравања, уз тврдње да је “проблематичан” због “спорног” начина формалног дефинисања *CAP* услова
- Представићемо само идеју доказа ради разумевања идеје

Теорема *CAP* (2)

- Претпоставке:
  - нека имамо два чвора и на њима реплициран податак *V:*
    1) нека на чвору 1 трансакција *A* ажурира *V*
    2) ажурирање се поруком *M* пропагира на чвор 2
    3) нека на чвору 2 ***нешто касније*** трансакција *B* чита *V*
  - на слици је представљено жељено понашање



Теорема *CAP* (3)

- Претпоставимо да порука *M* није стигла на одредиште
  - због *раздвојености* делова система
- У основи имамо три могућа понашања система:
  а) трансакција A се поништава
    - то значи да систем ***не прихвата раздвојеност*** својих делова, зато што због раздвојености онемогућава операције ажурирања
  б) трансакција A се успешно наставља (и затим завршава)
    - то значи да систем ***није конзистентан***, зато што ће читање података V на чворовима 1 и 2 давати различите резултате
  в) трансакција A чека на успешно слање поруке *M*
    - то значи да систем ***није расположив*** (непознато време одзива), зато што не можемо да гарантујемо време у коме ће се трансакција завршити
- У сваком од случајева није испуњен по један од услова

Компромиси

- Због претходног проблема, при пројектовању (или конфигурисању) дистрибуираног система неопходно је направити неки компромис:
  - одбацивање толеранције раздвојености
  - одбацивање расположивости
  - одбацивање конзистентности
  - ублажавање услова (тј. одбацивање више строго дефинисаних услова)
  - заснивање система на другачијем скупу услова
  - пројектовање заобилазних путева

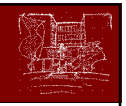
Компромиси (2)

- ***Одбацивање толеранције раздвојености***
  - "пристајемо да систем не ради у случају раздвојености"
  - један начин превазилажења је да све буде на једној машини
    - али ако већ морамо да правимо ДБП онда то вероватно није прихватљиво
  - алтернатива је да се раздвојеност (а тиме и отказ система) сведе на најмању могућу меру помоћу вишеструког умрежавања
  - обе алтернативе су веома скупе
- Имајући у виду да се дистрибурана решења користе само онда када су неопходна због обима података и послова, овај вид компромиса се ретко прави

Компромиси (3)

- ***Одбацивање расположивости***
  - "у случају раздвојености не гарантује се време одзива"
  - последице проблема се умањују пажљивим пројектовањем система, тј. успостављањем што ниже спреге међу чворовима
- Систем који није расположив је практично неупотребљив
  - због тога се овај приступ користи релативно ретко
    - ако је конзистентност примарна
    - ако је толеранција раздвојености незаобилазна
    - тежи се ниској спрегнутости међу чворовима

Компромиси (4)



- ***Одбацивање конзистентности***
  - "допуштамо да исти упит даје различите резултате на различитим чворовима"
  - ако су разлике прихватљивије него ниска расположивост
  - ако је учесталост појављивања сведена на прихватљиву меру
  - тј. ако је цена неконзистентности прихватљива
- Конзистентност је један од основних услова за успешан рад база података
- Ипак, постоје случајеви када није примарна
  - нпр. веб претраживачи
    - ако се нешто не проналази на сваком чвору, то није велики проблем
  - чак и продавнице
    - ако се мали број производа прода по старој цени, неће пропасти фирма

Измењен скуп услова – *BASE*

- Појмови из теореме почивају на уобичајеним концептима рада са базама података, тј. постоји зависност са особинама трансакција – *ACID*
- Али, може да се дефинише и другачији скуп услова, мање оштар, у ком случају све одговарајуће особине могу да се ускладе
  - *BASE* уместо *ACID*

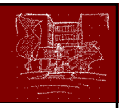
Измењен скуп услова – *BASE* (2)

- ***BASE***
  - ***B****asically* ***A****vailable*
    - не гарантује се расположивост одговора, већ само система
    - ако не може да се добије одговор, бар ће се добити обавештење о томе
    - и поред тога, висока расположивост на штету преосталих услова
  - ***S****oft-state*
    - стање система може да се мења чак и када није у току ниједна трансакција (пре свега ради асинхроног остваривања конзистентности)
  - ***E****ventually consistent*
    - жртвују се гарантована стална конзистентност и изолованост трансакција зарад расположивости
    - систем ће *у неком тренутку* (*eventually*) постати конзистентан
    - али ће и пре тог тренутка радити и давати одговоре (иако потенцијално различите на различитим чворовима)

Измењен скуп услова - *BASE* (3)

- Суштина концепта је у ***одложеном усклађивању*** садржаја на различитим чворовима
  - конзистентност се остварује, али не обавезно у оквиру исте трансакције (*eventual consistency*)
  - садржај чворова се мења и ван одвијања трансакција, а у циљу њиховог усклађивања (*soft state*)
  - ако систем није у потпуности расположив, чвор може да покуша да пружи мање поуздан или само делимичан одговор, уз одговарајућу информацију о томе (*basically available*)

Измењен скуп услова - *BASE* (4)



- Основна разлика је у тренутку усклађивања података између чворова, тј. у начину обављања репликације података
  - Код *ACID* услова
    - репликација би требало да се одвија у оквиру трансакције
  - Код *BASE* услова
    - репликације се одвија асинхроно, у неком тренутку по завршетку трансакције
    - тј. трансакција се одвија локално или на мањем броју чворова

Компромиси (5)

- Пројектовање заобилазних путева
  - пројектом система може да се предвиди да се у зависности од потреба и тренутних услова жртвују различите ствари
    - у зависности од врсте трансакција бира се шта се жртвује
  - нпр. у случају продавнице
    - није велики проблем ако цена производа *привремено* није уједначена
      - осим ако се ради о веома скупом производу
      - или је учесталост одступања веома висока
    - није проблем ако евидентирано стање у магацину *привремено* не одговара стварном стању
      - ако је купац унапред обавештен о таквој могућности
      - ако се то не дешава често и не траје дуго
    - али може да буде проблем ако се наплати производ који није на стању
      - ако није могуће обезбедити нову количину у прихватљивом временском року
- У суштини се своди на *BASE*

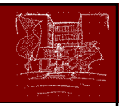
Закључак

- Чињеница да није могуће направити савршен дистрибуиран систем (у односу на скуп услова *ACID*) не значи да није могуће направити систем који задовољава реално прихватљиве захтеве (скуп услова *BASE* или други компромиси)
- Многи велики пословни системи данас користе дистрибуиране базе података које почивају на скупу услова *BASE*, што потврђује да су релаксирани услови често прихватљиви
  - ако се води рачуна при пројектовању и
  - прецизније приступи дефинисању захтева

Пројектовање са *BASE* условима

- У основи се своди на две нове активности:
  - Функционална декомпозиција података у фрагменте
    - ако претпостављамо да се већ пројектује дистрибуирана база података, овде само мало прецизније одређујемо услове за дефинисање фрагмената
  - Имплементација трансакција према *BASE* условима
    - операције се не мењају, али се мења време њиховог извршавања
    - а тиме и начин имплементирања
  - Одређивање услова репликације
- Постоје и другачији приступи
  - овај је међу једноставнијим
  - и вероватно међу најзаступљенијим

Пројектовање са *BASE* условима (2)



- Декомпозиција
  - скуп података се дели на фрагменте
    - који представљају најмање могуће функционалне целине
    - унутар којих морају да важе *ACID* услови
    - међу фрагментима је довољно да важе *BASE* услови

Пројектовање са *BASE* условима (3)

- Подела трансакција на синхрони и асинхрони део
  - свака трансакција се *лоцира* у једном *матичном фрагменту*
  - промене података у матичном фрагменту се имплементирају синхроно, тј. на уобичајен начин у оквиру трансакције
  - промене података у другим фрагментима се у оквиру трансакције само *евидентирају*
    - евиденција мора да задовољава *ACID* услове
  - евидентиране потребне измене се у другим фрагментима спроводе асинхроно

Пројектовање са *BASE* условима (4)

- Одређивање услова репликације
  - претпоставља се да међу репликама важе *BASE* услови
  - синхронизација реплика се одвија асинхроно, ван трансакција

Пројектовање са условима *BASE* - пример

- Фрагмент А: подаци о производима
- Фрагмент Б: подаци о продаји

- Трансакција ажурирања цене производа у режиму *ACID*:

```
BEGIN TRANSACTION
    UPDATE Product SET Price = :new_price
        WHERE ProductID = :product_id
    UPDATE CartItem SET Price = :new_price
        WHERE ProductID = :product_id
        AND CartID in (SELECT CartID FROM Cart WHERE Status = 'OPEN')
END TRANSACTION
```

- Иста трансакција у режиму *BASE*, лоцирана у фрагменту А:

```
BEGIN TRANSACTION
    UPDATE Product SET Price = :new_price
        WHERE ProductID = :product_id
    QUEUE ADD
        UPDATE CartItem SET Price = :new_price
        WHERE ProductID = :product_id
        AND CartID in (SELECT CartID FROM Cart WHERE Status = 'OPEN')
END TRANSACTION
```


Последице описаног поступка

- Сви упити у оквиру једног фрагмента су ***локално конзистентни***
  - у оквиру фрагмената су испоштована сва правила интегритета
- Резултати упита на различитим репликама су ***потенцијално*** неконзистентни
  - више реплика може да врати различите резултате
- Резултати логички повезаних упита на различитим фрагментима могу да буду неконзистентни
  - чак и ако се извршавају локално (!!!)
- Фрагмент је у основи расположив и у случају партиционисања
  - функционалан је и може да одговара на ***неке*** упите
- Свака промена података се у неком тренутку пропагира и на друге реплике и на друге повезане фрагменте
- База података испуњава *BASE* услове

Конфликти

- Представљају основни вид проблема
  - код свих видова имплементације *BASE* скупа услова
- Различите врсте конфликата
- Основни облик
  - ако се две реплике истог податка независно мењају, питање је која верзија је "исправна" и како успешно извести усклађивање
    - слично као са верзијама програмског кода...
    - ...али мануелно разрешавање обично није прихватљива опција
- Сложенији облик
  - ако се користе и редундантни подаци а не само реплике

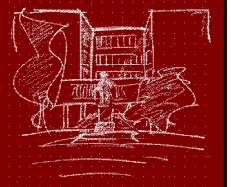
Разрешавање конфликата

- Више начина решавања:
  - забрана мењања података у случају партиционисања
    - умањена расположивост
  - дефинисање хијерархија међу редундантним копијама конкретних врста података
    - дефинисање примарних и секундарних фрагмената за све податке
      - само на примарним се изводе трансакције у односу на те податке
      - на секундарним се само пропагирају (одложене) измене
      - у примеру са ценама, за цену је примарни фрагмент А
    - не умањује употребљивост, али компликује имплементацију
  - вишеструке верзије података (*MVCC – multiversion concurrency control*)
    - алтернатива закључавању
    - за сваку копију података се води верзија
    - може да постоји више верзија "истог" података
    - повезано са концептима ***оптимистичких трансакција*** и ***оптимистичке репликације***
    - конфликти могу аутоматски да се препознају, али не и да се отклоне
  - ...

[P273]

Пројектовање база података

Саша Малков



Тема 13

Нерелационе базе података

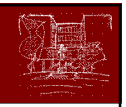
Појам нерелационих база података

- Релационим базама података су претходиле базе података засноване на другим моделима података
  - хијерархијске
  - мрежне
- ... али се данас термин ***нерелационе базе података*** (углавном) не односи на њих

Појам нерелационих база података

- У савременом развоју софтвера термин ***нерелационе базе података*** се односи на системе за управљање колекцијама података које:
  - немају строгу статичку структуру података
  - немају исцрпну проверу услова интегритета
  - не користе упитни језик *SQL*
  - (некада се подразумева и дистрибуираност)
- ... каква је онда корист од њих?

Слабости релационих база података



- Релационе базе података имају многе квалитете
  - (нећемо их сада набрајати)
- Али ти квалитети имају цену:
  - релативно висока цена читања
    - због нередундантности и строге структуре података
      - тј. због честог спајања података
  - отежано дистрибуирање
    - пре свега због “ултимативне” конзистентности података
  - скупа промена структуре
    - због повезаности структуре са употребом и оптимизацијама

Пример: Глобалне веб апликације

- Данашње глобално доступне апликације (веб, мобилни уређаји, интернет ствари) постављају специфичне изазове пред базе података:
  - конкурентни корисници (трансакције или читања)
    - милиони
  - количина података
    - дневна производња терабајта и петабајта података
  - обрада
    - свака активност корисника захтева неку обраду података
      - макар и само због праћења навика корисника...
  - оптерећење
    - непредвидив пораст оптерећења
    - неравномерно оптерећење
  - велика динамичност
    - непрекидно додавање нових могућности
      - тј. проширивање структуре базе података
    - промена постојећих компоненти система
      - тј. мењање постојеће структуре базе података

...

- При дистрибуирању РБП перформансе не расту довољно брзо
  - не расту ни близу линеарно
  - а оптерећење расте чак и експоненцијално
  - није једноставно додавати и склањати чворове
- Нормализована схема
  - захтева споре операције спајања
  - захтева врло пажљиво фрагментисање и реплицирање
  - отежава мењање

...

- Олакшица долази из пословног модела:
  - Што је апликација већа, уобичајено је и да већи део те апликације буде бесплатан
  - Ако је апликација бесплатна, онда нешто *може* да се жртвује
  - Обично се релативно безболно подноси умерено жртвовање конзистентности

РБП и теорема *CAP*

- Основне карактеристике РБП су настале ради обезбеђивања чврсте и стабилне структуре базе података
- Шта ако се нека од основних карактеристика релационих база података ***ослаби*** да би се превазишли проблеми које описује теорема *CAP*?
  - Да ли се и остале онда доводе у питање...
- Питања која се природно постављају:
  - Да ли је исплативо плаћати цену РБП, ако се при томе ипак одустаје од неких њихових квалитета?
  - Да ли је исплативије одустати од још неких особина РБП?

Компромиси

- Свет РБП је бескомпромисно оријентисан према поузданости и конзистентности
  - ...добро, не баш сасвим...
    - имамо компромисе у виду одабира нивоа изолованости
- У свету ДБП компромиси су неопходни
  - ...теорема *CAP*...
- У домену база података данас једну од најактивнијих области истраживања и развоја представљају управо различити видови нерелационих СУБП (НРСУБП, НРБП)
  - ...тражи се *права мера* компромиса...

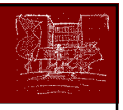
Корак уназад?

- НРБП су постојале пре релационих...
  - ...да ли је, онда, њихово оживљавање корак уназад?
- Неки од савремених НРСУБП имају сличности са старим нерелационим моделима података
  - ...можда аутори нису читали старе текстове...
- Већина значајних ипак нема много сличности

Када се размишља нерелационо?

- Неко може размишљати нерелационо зато што:
  - ...не познаје добро РБП, па мисли да РБП нису решење за неке проблеме чак и када оне то јесу
  - ...покушава да пронађе рупу на саксији
  - ***...има проблем који не може да реши помоћу РБП***

Шта то значи у пракси?



- Разматрају се ефикасна нерелациона решења
  - ако је потребно да се оствари
    - једноставније и ефикасније дистрибуирање података
    - лако додавање чворова
    - висока расположивост
    - слободна (или бар флексибилнија) структура података
  - ако је прихватљиво да се жртвује
    - одређен ниво конзистентности
    - аутоматска провера интегритета
  - ако “циљеви” и “жртве” могу да се ускладе

Дефиниција?

- Нема јединствене дефиниције, али може се рећи:
- НРБП је БП која:
  - не почива на релационом моделу података (или га се бар не држи чврсто)
  - лако се дистрибуира
  - хоризонтално је скалабилна (тј. лако подноси значајне промене схеме)
- Друге честе карактеристике:
  - без статичке схеме
  - лака репликација
  - једноставан *API*
  - евентуална конзистентност
    - почива на скупу услова *BASE* а не *ACID*
  - претпоставља изузетно велике количине података

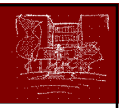
NoSQL

- Често се НРБП означавају као *NoSQL* базе података
  - изворно, одступање од *SQL*-а, као симбола РБП
    - *"no SQL"*
  - данас углавном теже да имају језик налик на *SQL*...
    - ...па се често тумачи као "*not only SQL*"

Типични проблеми и алт. решења

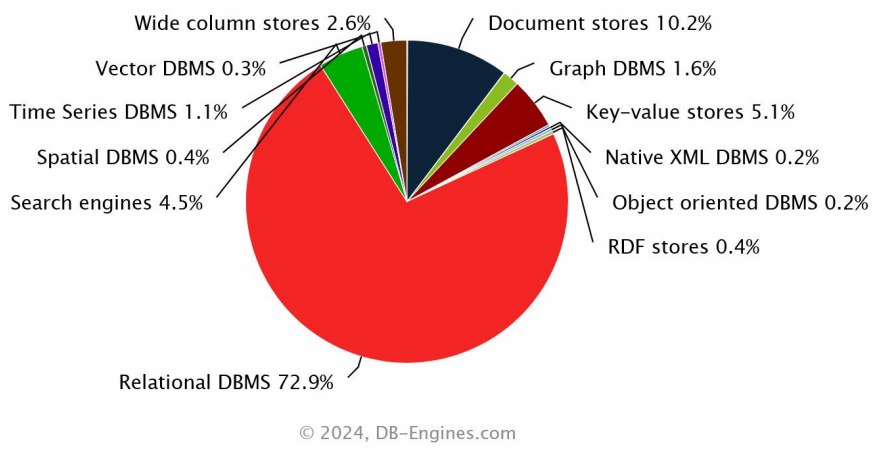
- Сложене структуре података у специфичним доменима
  - објектне базе података
- Висок ниво дистрибуирања
  - различите нерелационе базе података
- Слободна или флексибилна структура података
  - различите нерелационе базе података
- Неопходне изузетно високе перформансе
  - меморијске базе података
- Огромне количине података ниске сложености
  - различите врсте матричних база података

Врсте нерелационих БП

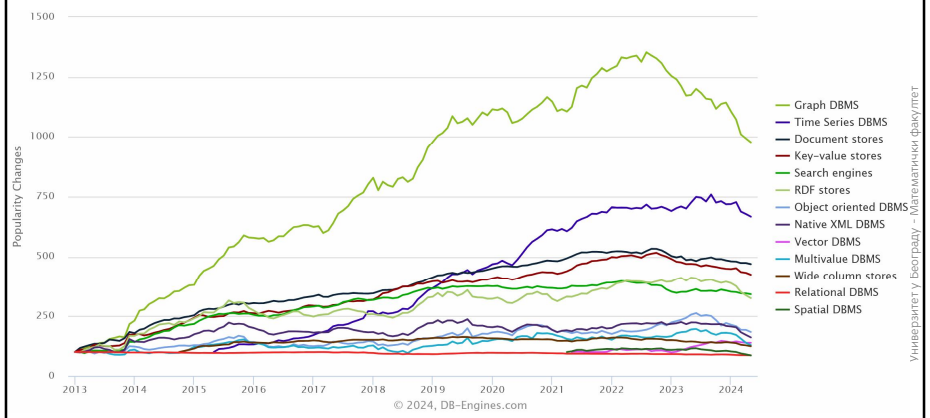


- Основне врсте нерелационих база података:
  - парови кључева и вредности
  - складишта широких колона
  - складиште докумената
  - графовске базе података
  - објектне базе података
  - табеларне базе података
  - складиште торки
  - вишевредносне
  - мултимоделне
  - *XML* базе података
  - складишта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

Популарност према моделима

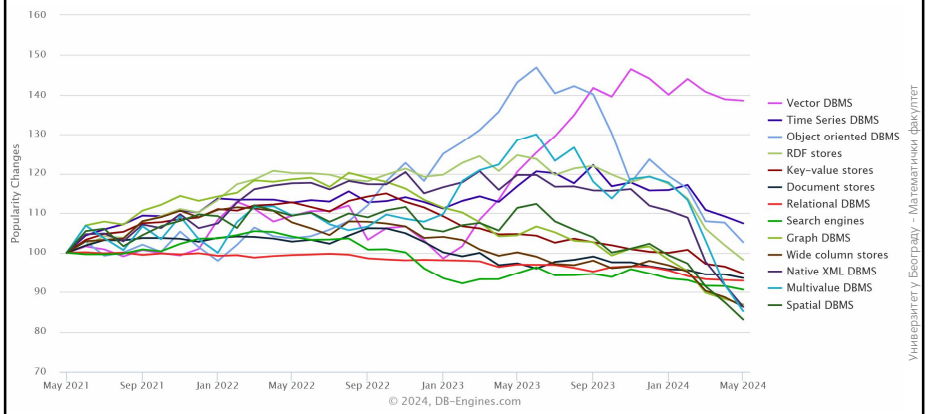


Популарност – тренд последњих година



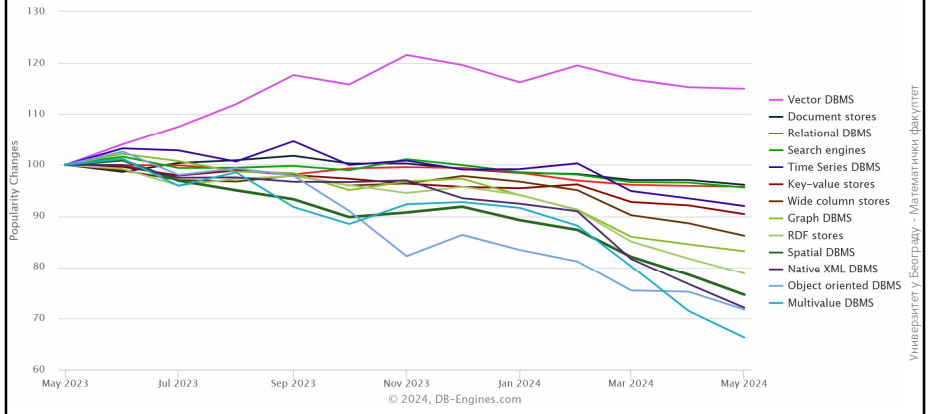
Напомена: Дијаграм приказује релативну промену популарности у процентима

Популарност – тренд у 36 месеци



Напомена: Дијаграм приказује релативну промену популарности

Популарност – тренд у 12 месеци



Напомена: Дијаграм приказује релативну промену популарности

Популарност – пар коментара...

- Претходне дијаграме би требало посматрати са резервом
  - Представљају тренд, тј. промену популарности од почетка периода
  - Популарност је процењена на основу нових реферисања на вебу, броја тражења на Гуглу, огласа за посао,...
  - Од краја 2013. има много текстова о значајној улози графовских база у специфичним проблемима, али број примена вероватно није много већи од тога, па видимо да у последње време значајно опадају
  - Базе докумената и базе са проширивим слоговима су већ довољно добро документоване и постоји веома велики број примена и мимо тога
  - "Класичне" базе парова кључева и вредности се полако повлаче пред базама докумената и базама са проширивим слоговима

Наставак следи...

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...***