Sodobne nerelacijske podatkovne baze (NoSQL)

- problemi relacijskih podatkovnih baz (teoretični in praktični)
- nerelacijske podatkovne baze, NoSQL
  - MongoDB (dokumentni SUPB)
  - Neo4j (grafni SUPB)
  - Chroma ali Milvus (vektorski SUPB)

Sodobne nerelacijske podatkovne baze (NoSQL)

- *"Facebook now has more than 1.8 billion monthly active users. Its data storage is more than 300 petabytes. Every 60 seconds on Facebook:*
  - *510 comments are posted*
  - *293,000 statuses are updated*
  - *136,000 photos are uploaded"*

- *"Twitter now has more than 320 million active users, sending > 500 million tweets every day"*
- "Webscale" aplikacije
  - veliko podatkov

    veliko istočasnih uporabnikov
  - veliko zahtevanih obdelav
  - časovno pogojene obremenitve (Facebook torek in četrtek +20%)

Ali so relacijske baze primerne za takšne obremenitve?

Problemi relacijskih SUPB

- Počasnost stičnih operacij $R \bowtie_\theta S = \sigma_\theta(R \times S)$
- Pogosto po nepotrebnem prekompleksen in preveč restriktiven podatkovni model
- Normalna oblika upočasnjuje procesiranje podatkov (denormalizacija)
- Počasno procesiranje binarnih objektov
- Začasni podatki, ki niso povezani z glavnimi (vsebine nakupovalnih vozičkov, ...)
- Vzdrževanje močne konsistentnosti (ACID) tudi, ko ni potrebna (všečkanje na FB)

- Relacijski SUPB zasnovani za poslovne in ne "webscale" aplikacije!
- Posledice
  - počasnost pri veliki količini podatkov
  - slabe zmožnosti skaliranja

Možne rešitve

- **vertikalno skaliranje (scaling up):** nadgradnja virov z zmogljivejšimi
  - kje je meja? cena?
- **horizontalno skaliranje (scaling out):** porazdelitev podatkov po več strežnikih
  - master-slave: pisanje se izvaja na glavnem strežniku, branja se lahko izvajajo iz sekundarnih baz (problem - konsistentnost)
  - deljenje podatkov (partitioning, sharding): razdelitev podatkov v podmnožice, boljše skaliranje, vendar izguba možnosti izvedbe stikov med podatki in referenčne integritete

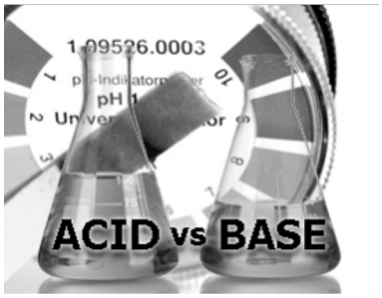
Možne rešitve

- **Sprememba logike in izvedbe**
  - replikacija cele baze v več "master" baz,
  - izvedba brez stikov – denormalizacija (stiki so eno od ozkih grl),
  - hranjenje baz v primarnem spominu,
  - zamenjava podatkovnega modela (nerelacijski)

Nerelacijski SUPB – NoSQL

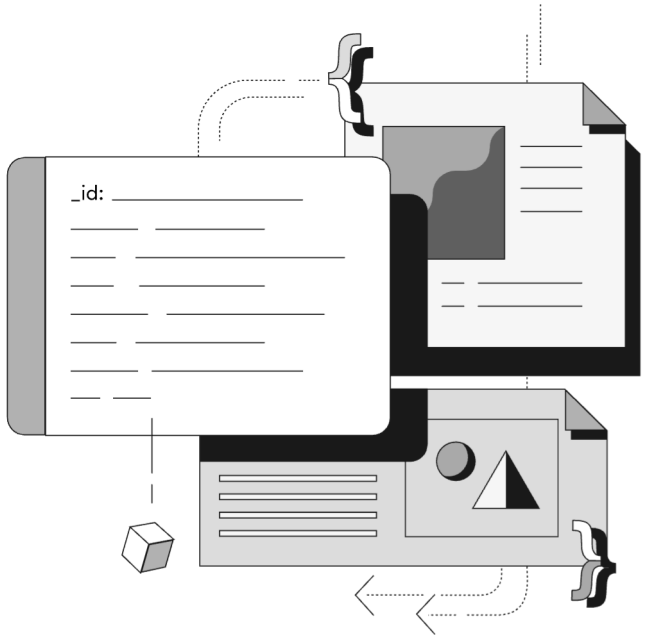
- **relacijske baze**: *one size fits all*
- **ozka grla**
  - Stične operacije
  - Konsistentnost podatkov v PB pri sočasni uporabi - ACID

- **nerelacijske baze**: *prilagojene zbirke podatkov za specifične aplikacije*
  - večinoma žrtvujejo shemo ACID
  - shema BASE (basically available, soft state, eventually consistent)
  - prednosti: cena, zmogljivost
  - slabosti: pomanjkanje standardov, potrebna specifična priučitev, omejena prenosljivost, nezrelost tehnologije
  - ne uporabljajo fiksne podatkovne sheme in stikov!



Primer nerelacijskega SUPB - MongoDB

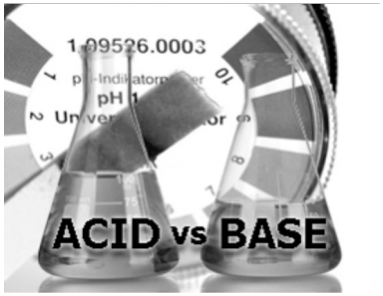
- Dokumentni podatkovni model $\Rightarrow$ JSON
- JavaScriptu podoben povpraševalni jezik v povezavi z JSON podatkovno sintakso
- Enostaven za uporabo
- Skalabilen
- Dostop iz Pythona: pyMongo




```
{
   ...
   name: { first: "Alan", last: "Turing" },
   contact: { phone: { type: "cell", number: "111-222-3333" } },
   ...
}
```

ACID vs BASE

- ***ACID – močna konsistentnost – relacijski SUPB***
- ***BASE – šibka konsistentnost***
- ***Basic Availability***
  - Podatki v PB so dostopni "večino časa"
- ***Soft-state***
  - Konsistentnost pisanja ni zagotovljena, ravno tako ne konsistentnost med replikami podatkov (v vsakem trenutku)
- ***Eventual consistency***
  - Sčasoma se podatek "stabilizira" v konsistentno stanje

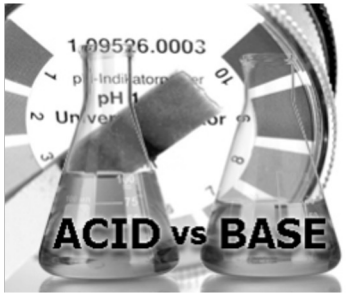


CAP – zaželene lastnosti **vseh** porazdeljenih sistemov (tudi SUPB)

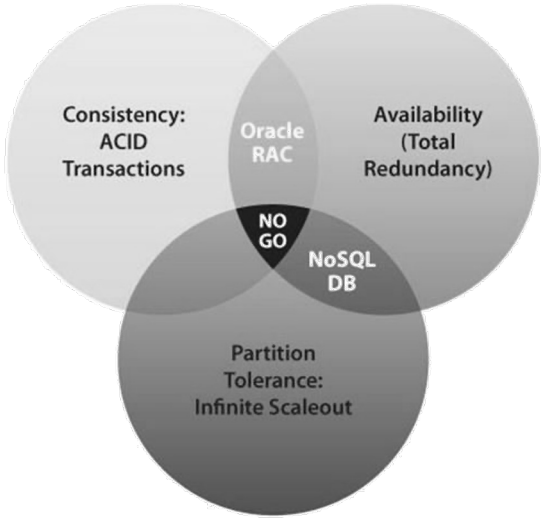
- Consistency – (močna) konsistentnost: ACID
  Vsako branje prebere zadnjo veljavno vrednost podatka.

- Availability – razpoložljivost:
  Vsaka bralna zahteva dobi vrnjeno vrednost (v principu brez garancije, da gre za zadnjo verzijo podatka).

- Partition tolerance – odpornost na delitev podatkov:
  Sistem deluje kljub težavam (zakasnitve ali izgube podatkov) na poljubno velikem delu (<100%) izpadlih povezav med vozlišči.



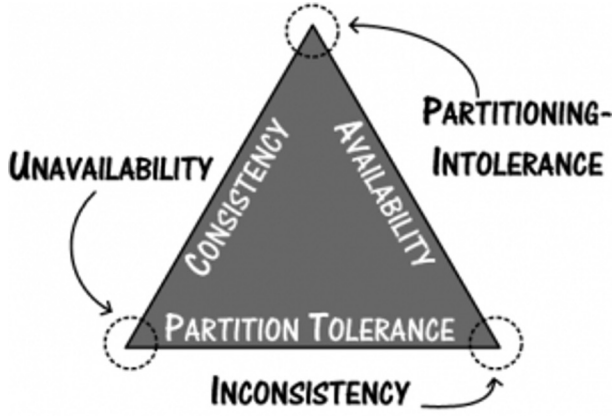
Teorem CAP



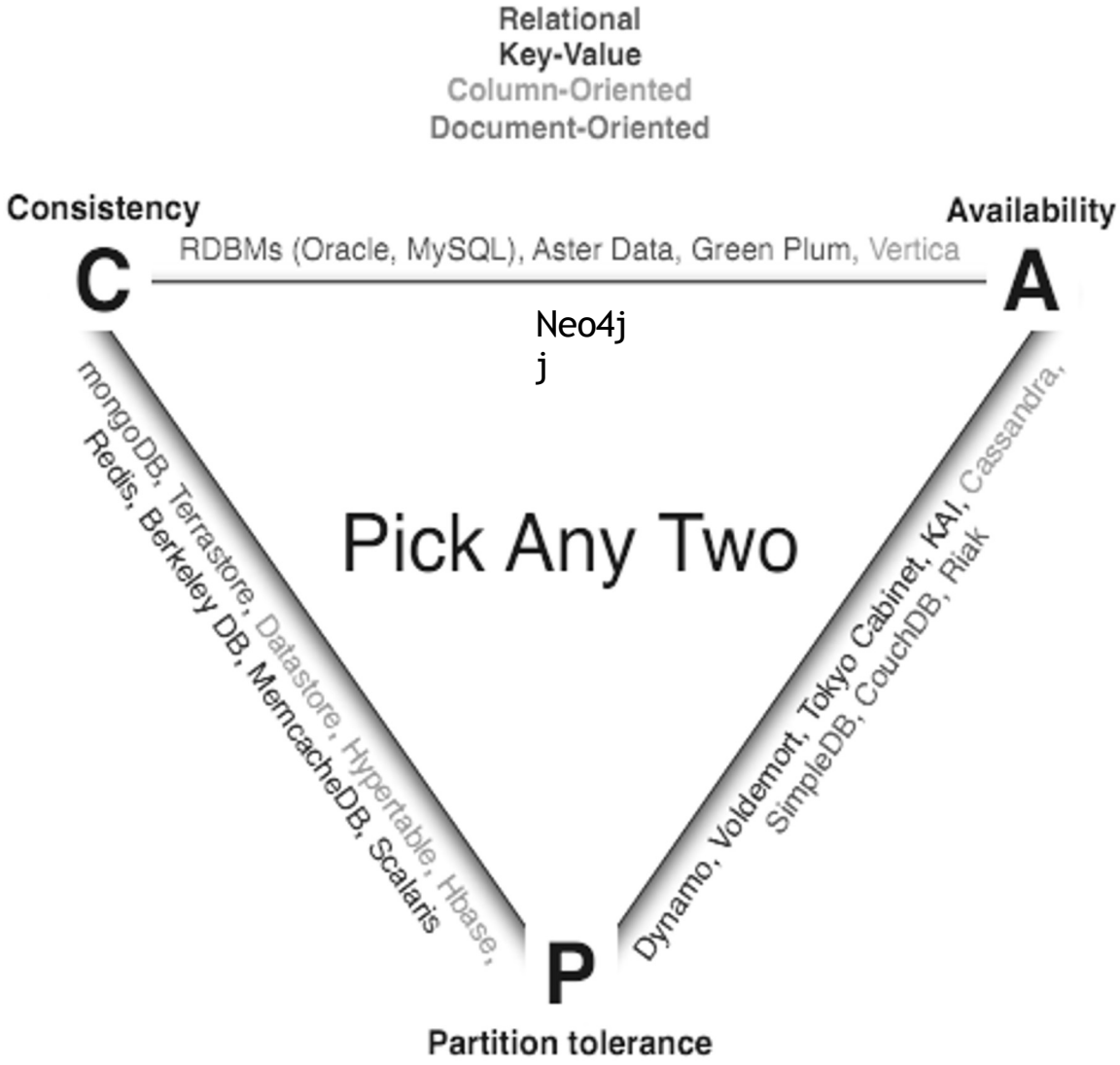
- Eric Brewer, 2000; dokaz na MIT 2002
- Vsak sistem za delo s podatki ima tri lastnosti:
  - konsistentnost (Consistency),
  - razpoložljivost (Availability) in
  - zmožnost delitve podatkov (Partitions)
- Teorem pravi: pri deljenju podatkov lahko zagotovimo **največ dve** od teh treh lastnosti
- Primer: podatke razdelimo na več računalnikov. Kasnejša posodobitev podatka zahteva posodobitev vseh kopij. Scenarija:
  - zaklenemo vse kopije, da zagotovimo konsistentnost (zmanjšana razpoložljivost), ali
  - žrtvujemo konsistentnost na račun večje razpoložljivosti (sčasoma podatki postanejo konsistentni)

Teorem CAP

- Torej: žrtvovati je potrebno enega od kriterijev: C, A ali P
  - **konsistentni in razpoložljivi (CA)** sistemi imajo težavo z delitvijo podatkov, težave običajno obvladujejo z replikacijo,
  - **konsistentni, porazdeljeni sistemi (CP)** imajo težavo z razpoložljivostjo podatkov ob naporu zagotavljanja njihove konsistentnosti,
  - **razpoložljivi, porazdeljeni sistemi (AP)** dosegajo sčasno konsistentnost (eventual consistentcy) z replikacijo podatkov in občasnim preverjanjem konsistentnosti v podatkovnih vozliščih.



Teorem CAP

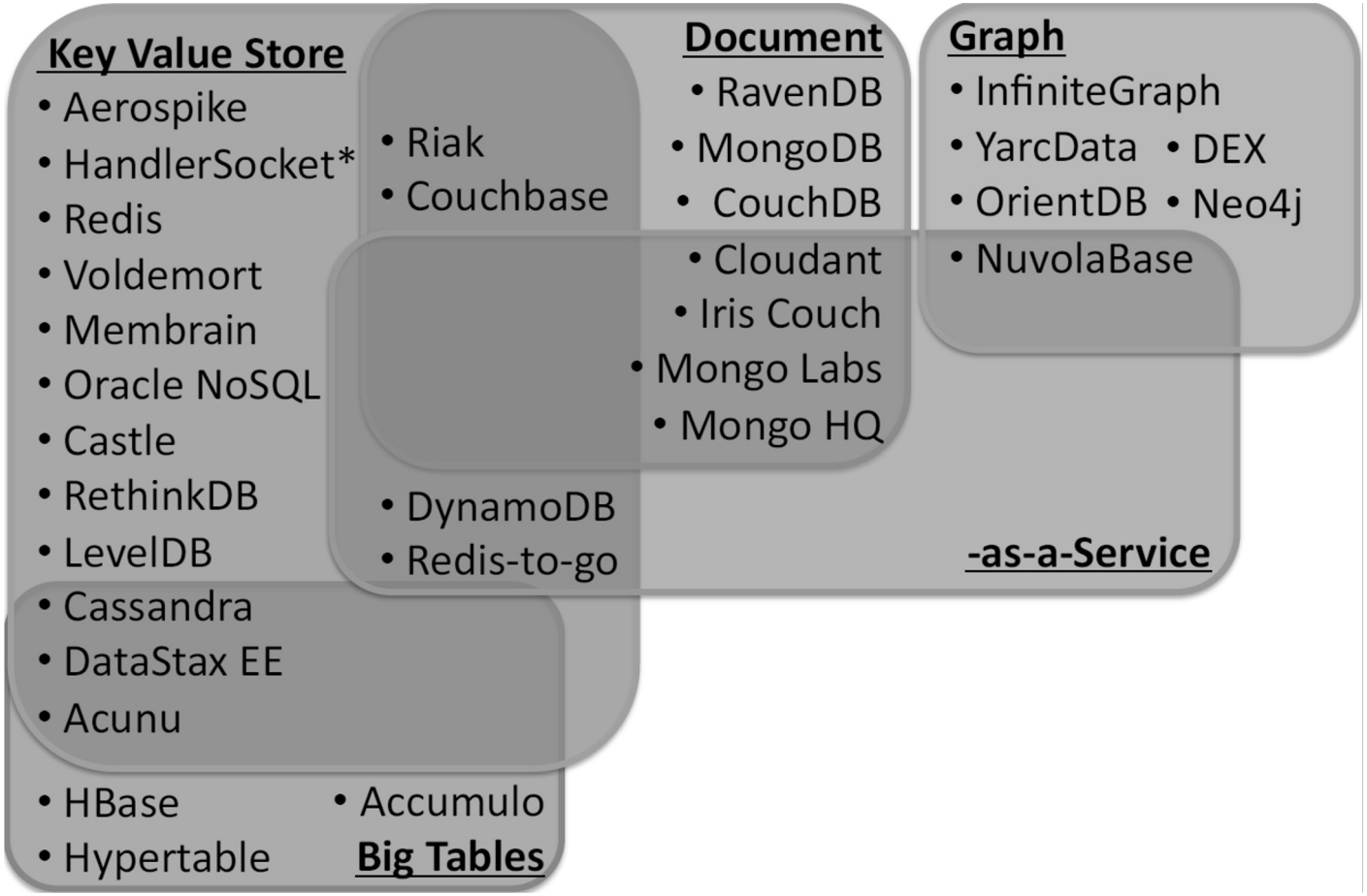


https://mwhittaker.github.io/blog/an_illustrated_proof_of_the_cap_theorem/

Izvedbe NoSQL

- Prilagojene specifičnim načinom uporabe in vrstam aplikacij
  - **shrambe s ključi (key-value pairs):** BerkleyDB, Keyspace, Dynomite, Voldemort, MemcachedDB in Tokyo Cabinet.
  - **dokumentne zbirke**: MongoDB (Foursquare, SourceForge, Fotopedia, Joomla Ads), CouchDB (CERN, BBC), Redis
  - **zbirke grafov:** Neo4j, AllegroGraph, InfoGrid, Sones, graphDB in FlockDB
  - **stolpično usmerjene zbirke (column-oriented databases):** Hypertable in Cassandra (Digg, Facebook, Twitter, Rackspace), dodatki rel. bazam
  - **objektne shrambe-v zadnjem času dvig popularnosti (podatkovna jezera):** Oracle Coherence, db4o, ObjectStore, GemStone, Polar

NoSQL SUPB


Key Value Store
• Aerospike
• HandlerSocket*
• Redis
• Voldemort
• Membrain
• Oracle NoSQL
• Castle
• RethinkDB
• LevelDB
• Riak
• Couchbase
Document
• RavenDB
• MongoDB
• CouchDB
• Cloudant
• Iris Couch
• Mongo Labs
• Mongo HQ
Graph
• InfiniteGraph
• YarcData • DEX
• OrientDB • Neo4j
• NuvolaBase
• DynamoDB
• Redis-to-go
-as-a-Service
• Cassandra
• DataStax EE
• Acunu
• HBase
• Hypertable
• Accumulo
Big Tables

Težave NoSQL

- Ni standardnega povpraševalnega jezika
  - Sami delamo, kar sicel dela SQL prevajalnik (nizkonivojske operacije)
- Zavržemo > 40 let izkušenj relacijskih SUPB
  - Težko smo boljši od prevajalnika/optimizatorja SQL poizvedb
  - Visokonivojski jeziki so boljši po več kriterijih (neodvisnost od podatkov, manj kode)
- Ni shranjenih podprogramov
  - samo ena interakcija med aplikacijo in SUPB (namesto ena za vsak zapis, kot pri NoSQL)
  - premaknemo kodo k podatkom in ne obratno
- Če ACID danes ne potrebujemo, ali lahko zagotovimo to za jutri?
  - premiki podatkov, nekomutativne posodobitve, večzapisna stanja
  - če potrebujemo integritetne omejitve
  - eventuelna konsistentnost $\rightarrow$ zmeda v podatkih

Za kaj oz. kdaj je torej primeren NoSQL

- Netransakcijski sistemi
- Enozapisne, komutativne transakcije
- Neprimerno za sodobne OLTP sisteme
- Ne potrebujemo enovitega povpraševalnega jezika
  - programska koda
  - CQL, UnQL (po zgledu SQL, nestandardno, nekompatibilno)

SUPB v letu 2024

Rank Oct 2024	Rank Sep 2024	Rank Oct 2023	DBMS	Database Model	Score Oct 2024	Score Sep 2024	Score Oct 2023
1.	1.	1.	Oracle	Relational, Multi-model	1309.45	+22.85	+48.03
2.	2.	2.	MySQL	Relational, Multi-model	1022.76	-6.73	-110.56
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	802.09	-5.67	-94.79
4.	4.	4.	PostgreSQL	Relational, Multi-model	652.16	+7.80	+13.34
5.	5.	5.	MongoDB	Document, Multi-model	405.21	-5.02	-26.21
6.	6.	6.	Redis	Key-value, Multi-model	149.63	+0.20	-13.33
7.	7.	↑ 11.	Snowflake	Relational	140.60	+6.88	+17.36
8.	8.	↓ 7.	Elasticsearch	Multi-model	131.85	+3.06	-5.30
9.	9.	↓ 8.	IBM Db2	Relational, Multi-model	122.77	-0.28	-12.10
10.	10.	↓ 9.	SQLite	Relational	101.91	-1.43	-23.23
11.	11.	↑ 12.	Apache Cassandra	Wide column, Multi-model	97.61	-1.34	-11.21
12.	12.	↓ 10.	Microsoft Access	Relational	92.15	-1.61	-32.16
13.	13.	↑ 14.	Splunk	Search engine	91.27	-1.75	-1.10
14.	14.	↑ 17.	Databricks	Multi-model	85.60	+1.35	+9.78
15.	15.	↓ 13.	MariaDB	Relational, Multi-model	84.89	+1.45	-14.77
16.	16.	↓ 15.	Microsoft Azure SQL Database	Relational, Multi-model	74.53	+1.58	-6.40
17.	17.	↓ 16.	Amazon DynamoDB	Multi-model	71.85	+1.78	-9.07
18.	18.	18.	Apache Hive	Relational	52.57	-0.50	-16.61
19.	19.	↑ 20.	Google BigQuery	Relational	51.18	-1.48	-5.39
20.	20.	↑ 21.	FileMaker	Relational	44.40	-0.80	-8.92
21.	21.	↑ 23.	Neo4j	Graph	42.51	-0.17	-5.93

SQL + NoSQL = NewSQL?

- NoSQL:
  - Nova, moderna zvrst nerelacijskih SUPB
  - Zavračanje pojmov fiksnih shem tabel in stičnih operacij
  - Načrtovan z namenom omogočanja zahtev po masivnem horizontalnem skaliranju
  - Omogoča upravljanje podatkov brez definirane sheme
  - Bye, bye, SQL
- NewSQL
  - Zadnji trend na področju relacijskih SUPB
  - Ohranja SQL in ACID
  - Načrtovan z namenom omogočanja zahtev po masivnem horizontalnem skaliranju (Google Spanner, CockroachDB: CP, namesto A imamo "best effort" HA=99.999%) ali
  - Tako velik performančni napredek, da horizontalno skaliranje ni več potrebno (VoltDB, in-memory database: CA, P ni potreben)

Splošni zaključki

- Zelo specializirana namembnost večine NoSQL SUPB
- Žrtvovanje konsistentnosti za boljše porazdeljeno delovanje
- Vendar v praksi:
  - V porazdeljenih sistemih je lastnost P nepogrešljiva
  - Konsistentnost (C) se vedno bolj kaže kot nepogrešljiva
- Teorem CAP na lastnosi gleda **binarno**
  - Ali je možen drugačen, bolj mehek pogled?

Splošni zaključki

An airline reservation system:
- When most of seats are available: it is ok to rely on somewhat out-of-date data, availability is more critical
- When the plane is close to be filled: it needs more accurate data to ensure the plane is not overbooked, consistency is more critical

- Moderna "mehka" interpretacija teorema CAP:
  - Teorem CAP v resnici prepoveduje le perfektno razpoložljivost (A) in konsistentnost (C) ob prisotnosti razpada/particioniranja omrežja (kar je zelo redek dogodek)
  - Kompromis zagotoviti perfektno A ali C je v resnici dinamičen - NoSQL SUPB samo nekaj časa vztraja v konkretnem (CP ali AP) načinu: CAP -> PACELC (kompromisi)
  - Primer: CockroachDB - žrtvuje 0.001% A
- Zanimivo branje
  - *http://www.infoq.com/articles/cap-twelve-years-later-how-the-rules-have-changed*
  - Abadi, Daniel J. "Consistency tradeoffs in modern distributed database system design", *IEEE Computer Magazine* 45.2 (2012): 37

Zaključek: nerelacijski SUPB – da ali ne?

- Ni standardnega povpraševalnega jezika.
- V primerjavi z relacijskimi PB problematično pisanje (nižjenivojski jeziki) in optimizacija poizvedb.
- Podpora transakcijam ACID. Ali jih res ne potrebujemo na dolgi rok?
- BASE zahteva trezen razmislek, sicer lahko vodi v zmedo.
- Kdaj torej uporabiti nerelacijske SUPB?
  - Zahtevana masivna (webscale) paralelizacija
  - Zahtevana večja **fleksibilnost** ali **specifične lastnosti** podatkovnega modela
  - Ni potrebe po transakcijah
  - Ne potrebujemo enovitega, standardnega povpraševalnega jezika