

Poglavje 1

Sodobne nerelacijske podatkovne baze

NoSQL (NoREL)

- Relacijski pristop: SUPB kot univerzalno orodje za reševanje vseh problemov hranjenja podatkov
- NoSQL: Not only SQL
 - Pristop k SUPB, različen od relacijskega
 - Prilagojene problemu, kjer jih uporabljamo
 - Deset do stokrat hitrejši od relacijskih SUPB
 - Neprilagodljivost na različne probleme (uporabimo izvedmo, najbližjo danemu problemu)
 - Ni standardov
 - Nezrelo področje in implementacije
 - Izdelava na ključ (za konkretne probleme)

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)

Zakaj NoSQL?

- Prve pojavitve v masivnih „webscale“ spletnih aplikacijah
- Twitter: dnevno 7 TB novih podatkov
- Facebook: 200 milijonov aktivnih uporabnikov, tedensko 10^9 novih zapisov
- Google, Yahoo ...: velikost baz v stotinah petabytov, masivna porazdeljenost
- Bogati podatkovni tipi: slike, zvok, video, tekst ...
- Velikost, bogati podatkovni tipi in masivna porazdeljenost grede v škodo relacijskim SUPB

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 (stiki; denormalizacija)
- Počasno procesiranje binarnih objektov (bogati tipi)
- 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, počasnost pisanja)
 - deljenje-razbijanje podatkov (partitioning, sharding): razdelitev podatkov v podmnožice, boljše skaliranje
 - izguba učinkovite izvedbe stikov med podatki
 - težavno preverjanje 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

Not
Only SQL

- **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 (veste, kaj pomeni?)
 - shema BASE (basically available, soft state, eventually consistent)
 - glavna prioriteta: masovna porazdeljivost
 - prednosti: cena, zmogljivost
 - slabosti: pomanjkanje standardov, potrebna specifična priučitev, omejena prenosljivost, nezrelost tehnologije
 - ne uporabljajo fiksne podatkovne sheme in stikov!



Primer nerelacijskega SUPB - MongoDB

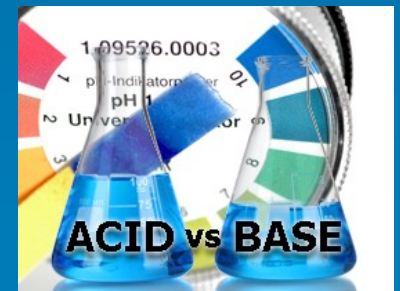
- Dokumentni podatkovni model $\Rightarrow$ JSON
- JavaScriptu podoben povpraševalni jezik v povezavi z JSON podatkovno sintakso
- Enostaven za uporabo
- Skalabilen
- Dostop iz Pythona: pyMongo



```
{  
  ...  
  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 – zaželenne lastnosti **vseh** porazdeljenih sistemov (tudi SUPB)

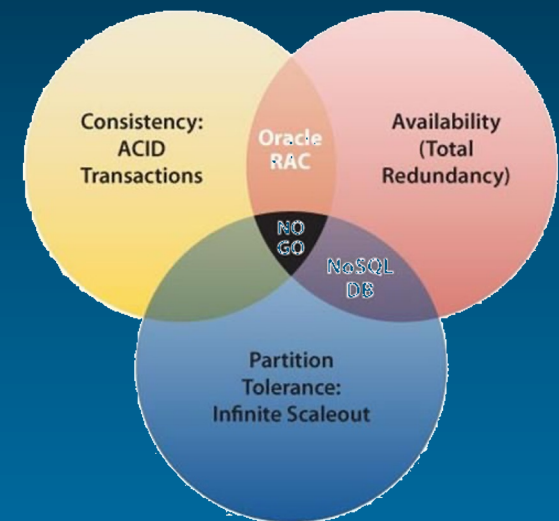
Not
Only SQL

- 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.



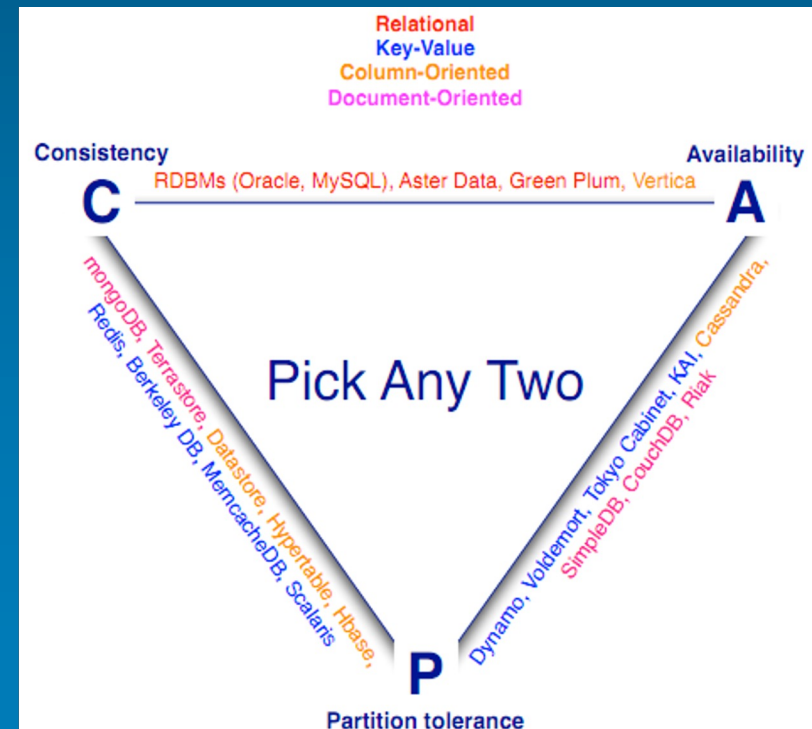
Teorem CAP

- 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: 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 consistency) z replikacijo podatkov in občasnim preverjanjem konsistentnosti v podatkovnih vozliščih.



Teorem CAP

Cena, ki jo plačamo za masovno porazdeljivost podatkov.

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, Dymomite, 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, Amazon S3 in kompatibilne



Težave NoSQL

- Ni standardnega/visokonivojskega povpraševalnega jezika
 - Sami delamo, kar sicer dela SQL prevajalnik (nizkonivojske operacije)
 - Navidezni absurdn: mnogi sodobni NoSQL sistemi implementirajo podmnožico SQL
- 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 (učinkovitih) shranjenih podprogramov
 - samo ena interakcija med aplikacijo in SUPB (namesto ena za vsak zapis, kot pri NoSQL)
 - premaknemo kodo k podatkom in ne obratno
- Konsistentnosti morda danes ne potrebujemo; kaj pa jutri?
 - premiki podatkov, nekomutativne posodobitve, večzapisne spremembe stanja
 - integritetne omejitve; eventuelna konsistentnost $\Rightarrow$ eventuelna 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 2026

Rank	Rank			DBMS	Database Model	Score		
	Oct 2026	Sep 2026	Oct 2025			Oct 2026	Sep 2026	Oct 2025
1.	1.	1.		Oracle	Relational, Multi-model ⓘ	1119.79	-2.48	-92.98
2.	2.	2.		MySQL	Relational, Multi-model ⓘ	837.96	-6.85	-41.70
3.	3.	3.		Microsoft SQL Server	Relational, Multi-model ⓘ	694.47	-4.22	-20.58
4.	4.	4.		PostgreSQL	Relational, Multi-model ⓘ	688.75	+5.50	+45.56
5.	5.	5.		MongoDB	Document, Multi-model ⓘ	374.64	-6.66	+6.63
6.	6.	6.		Snowflake	Relational	221.30	+9.69	+22.65
7.	7.	↑8.		Databricks	Multi-model ⓘ	171.24	+2.90	+42.44
8.	8.	↓7.		Redis	Key-value, Multi-model ⓘ	156.09	-1.97	+13.76
9.	9.	9.		IBM Db2	Relational, Multi-model ⓘ	106.29	-4.28	-16.08
10.	↑12.	↑11.		Apache Cassandra	Wide column, Multi-model ⓘ	98.86	+4.90	-6.30
11.	↓10.	↑12.		SQLite	Relational	95.73	-1.32	-8.82
12.	↓11.	↓10.		Elasticsearch	Multi-model ⓘ	94.11	-1.39	-22.55
13.	13.	13.		MariaDB	Relational, Multi-model ⓘ	75.60	+1.18	-12.16
14.	↑16.	↑17.		Apache Hive	Relational	69.58	-0.53	-5.64
15.	↓14.	↑18.		Splunk	Search engine	69.25	-2.43	-4.62
16.	↓15.	16.		Microsoft Azure SQL Database	Relational, Multi-model ⓘ	68.55	-2.29	-6.94
17.	17.	↓14.		Microsoft Access	Relational	63.32	+0.27	-17.47
18.	18.	↓15.		Amazon DynamoDB	Multi-model ⓘ	53.41	-2.94	-22.49
19.	19.	19.		Google BigQuery	Relational	49.78	-2.80	-13.43
20.	20.	20.		Neo4j	Graph	48.88	+1.72	-3.63
21.	21.	21.		Apache Solr	Search engine, Multi-model ⓘ	32.41	-1.62	-2.90

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 lastnosti gleda **binarno**
 - Ali je možen drugačen, bolj mehek pogled?

Splošni zaključki

- 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

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

Zaključek: nerelacijski SUPB – da ali ne?

- Ni standardnega povpraševalnega jezika. SQL je še vedno pomemben!
- 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