

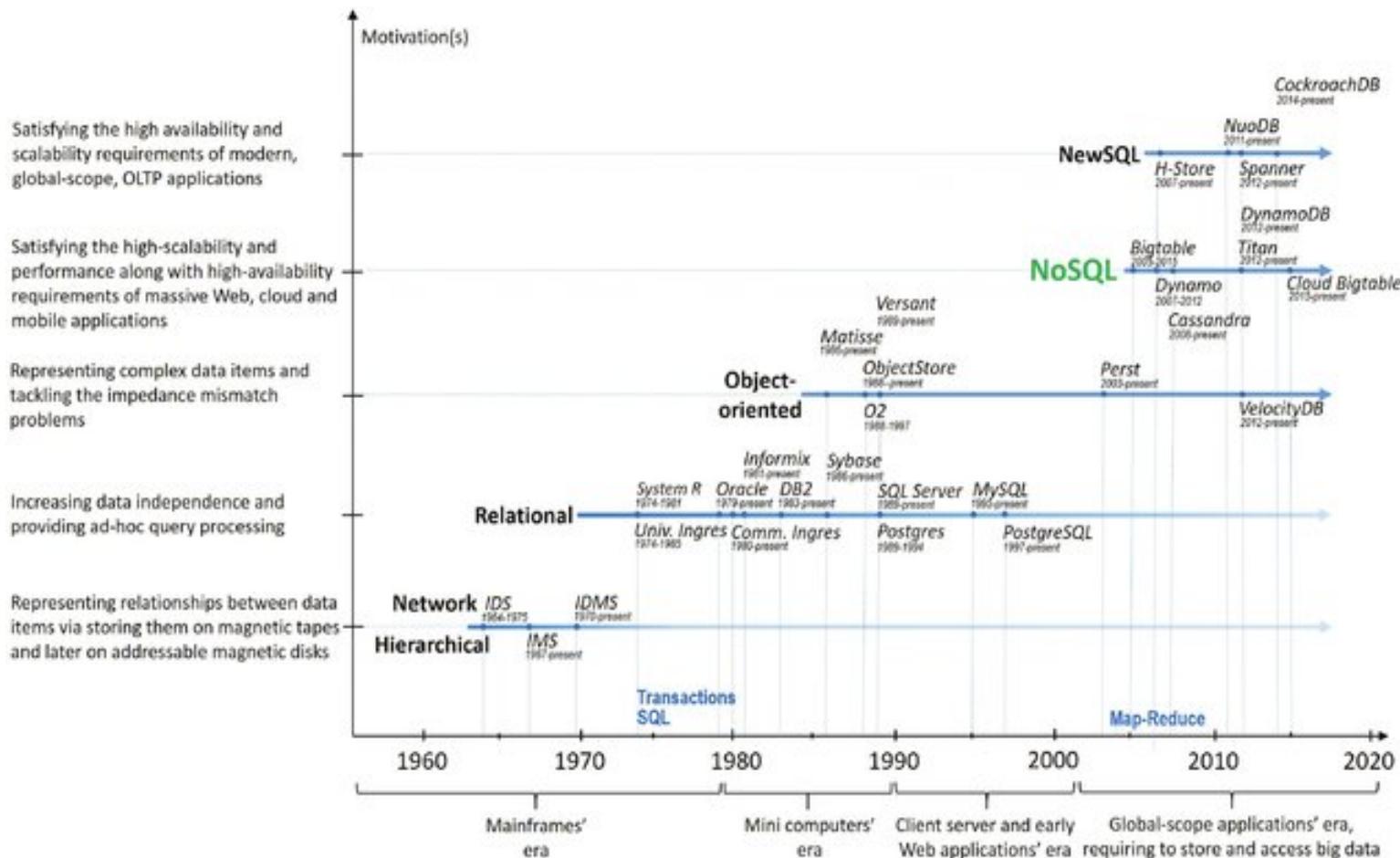
Računarstvo i informatika

*Katedra za računarstvo
Elektronski fakultet u Nišu*

Napredne baze podataka
NoSQL baze podataka

Zimski semestar 2020/2021

Uvod



Uvod

- Skladišta podataka pre pojave RDBMS
 - Mrežni i hijerarhijski modeli
 - Bazirani na hijerarhiji i multidimenzionalnim poljima
 - Skladištenje podataka na magnetnim trakama
- RDBMS
 - Generičko rešenje
 - Nezavisnost od podataka
 - Podrška za ad hoc upite
- Objektno orijentisane baze podataka
 - Predstavljanje kompleksnih podataka
 - Reševanje object-relational impedance mismatch problema

Uvod

- NoSQL baze podataka
 - Specifični zahtevi Web, mobilnih i cloud aplikacija
 - Distribuiranost
 - Skalabilnost
- NewSQL baze podataka
 - RDBMS sa karakteristikama NoSQL baza podataka

Uvod

- Relacioni model je predstavljen 1970 godine
- E. F. Codd, “**A Relational Model of Data for Large Shared Data Banks**”
- Relaciona algebra obezbeđuje deklarativne mehanizme za rad sa skupovima podataka.
- SQL se bazira na relacionoj algebri.

Ime	Prezime	Indeks	MBR
Petar	Petrović	1111	123456
Milan	Milanović	2222	654321
Jovan	Jovanović	3333	345612

Uvod

- Prednosti korišćenja RDBMS:
 - Efikasno skladištenje podataka
 - Podrška za ACID transakcije
 - Podrška za kompleksne SQL upite
 - Ogromna tehnološka baza (različiti DBMS-ovi, alati, programski interfejsi i sl.)

ACID

- ACID kao concept postoji nekoliko decenija
- Sve do nedavno je predstavljao osnovnu karakteristiku kojoj su težili svi DBMS proizvodi.
- Sistem koji nije zadovoljavao ACID karakteristike nije se mogao smatrati pouzdanim sistemom.
- Koncept je 1970 godine kreirao Jim Grey. Osnovne ideje su objavljene 1981 godine u radu “The Transaction Concept: Virtues and Limitations”.
- U startu je definisan ACD koncept.
- Transakcija je definisana kao određeni broj transformacija stanja sistema koje moraju da zadovoljavaju ACD svojstva odnosno ograničenja konzistentnosti sistema.

ACID

- Bruce Lindsey, 1979. godine, "Notes on Distributed Databases"
 - Oslanja se na ACD koncepte
 - Definisali osnove za ostvarivanje konzistentnosti sistema
 - Postavili standarde za replikaciju podataka
- Andreas Reuter i Theo Härdter, 1983, "Principles of Transaction-Oriented Database Recovery"
 - Uveli termin ACID

ACID

- **Atomicity (Atomičnost)**

- Princip sve ili ništa.
- Mora da se izvrši svaka operacija unutar transakcije ili se ne izvršava nijedna.
- Kada se deo transakcije ne izvrši, ne izvrši se čitava transakcija.

- **Consistency (Konzistentnost)**

- Transakcija mora da zadovolji sve definisane protokole i pravila u bilo kom trenutku.
- Transakcija ne sme da naruši ova pravila i sistem mora da bude u konzistentnom stanju na početku i na kraju transakcije.
- Ne postoje transakcije koje su poluzavršene.
- Transakcija prevodi DB iz jednog konzistentnog stanja u drugo.

ACID

- **Isolation (Izolacija)**
 - Nijedna transakcija nema pristup nekoj drugoj transakciji koja nije završena.
 - Transakcija ne vidi nekomitovane izmene iz drugih transakcija
 - Svaka transakcija je nezavisna za sebe.
- **Durability**
 - Kada se transakcija završi, pamti se da je završena i ne može se poništiti.
 - Završene transakcije moraju da prežive bilo koji pad sistema.
 - Komitovane izmene u bazi podataka su trajne.

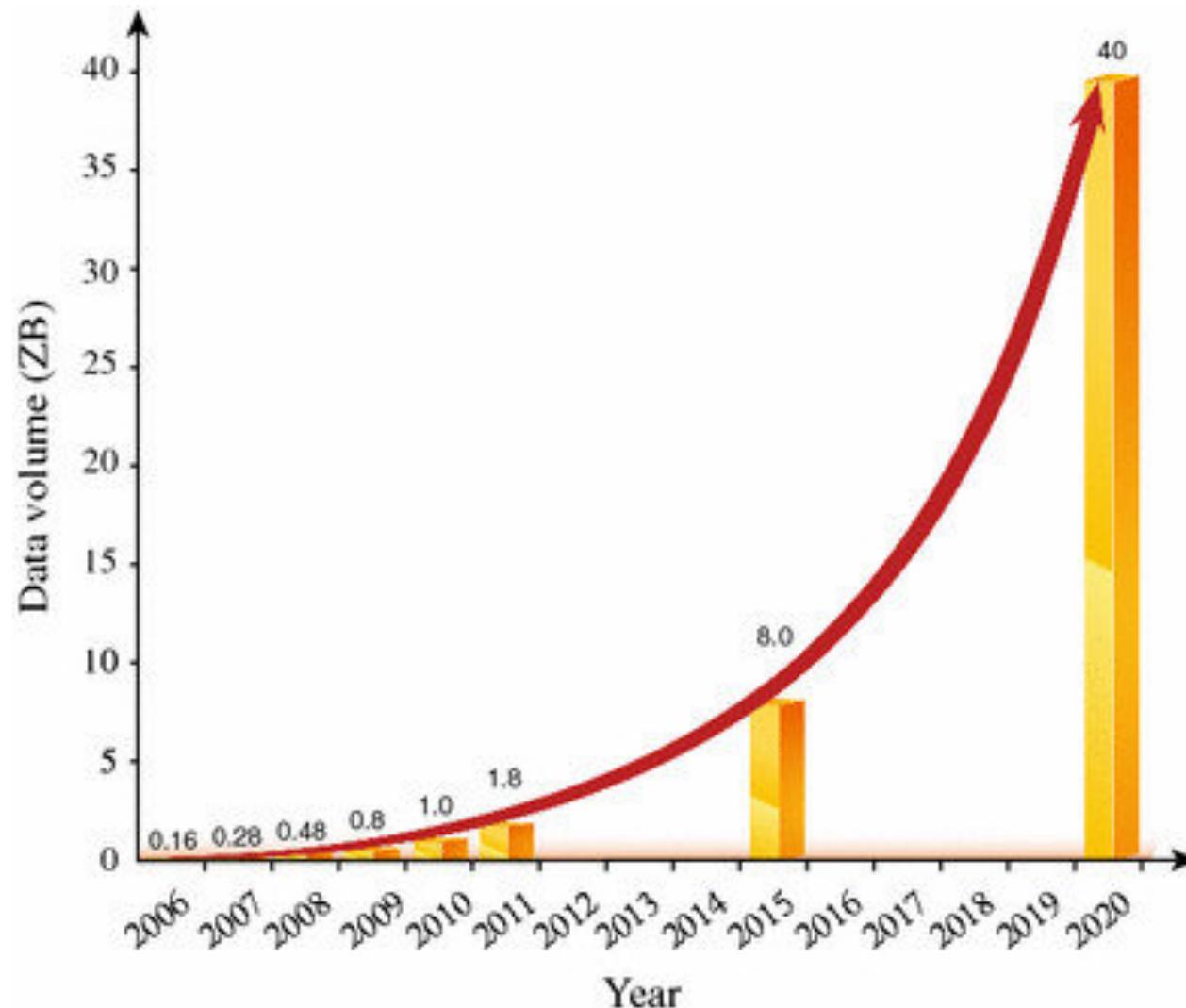
ACID

- Postoje razlike na koji način različiti proizvodi implementiraju ACID svojstva.
- Svaki RDBMS implementira ACID svojstva u cilju obezbeđivanja pouzdanosti.
- U svakom trenutku, DBMS koji implementira ACID svojstva, za svaki podataka se proverava da li zadovoljava definisana ograničenja.

Podaci na Web-u

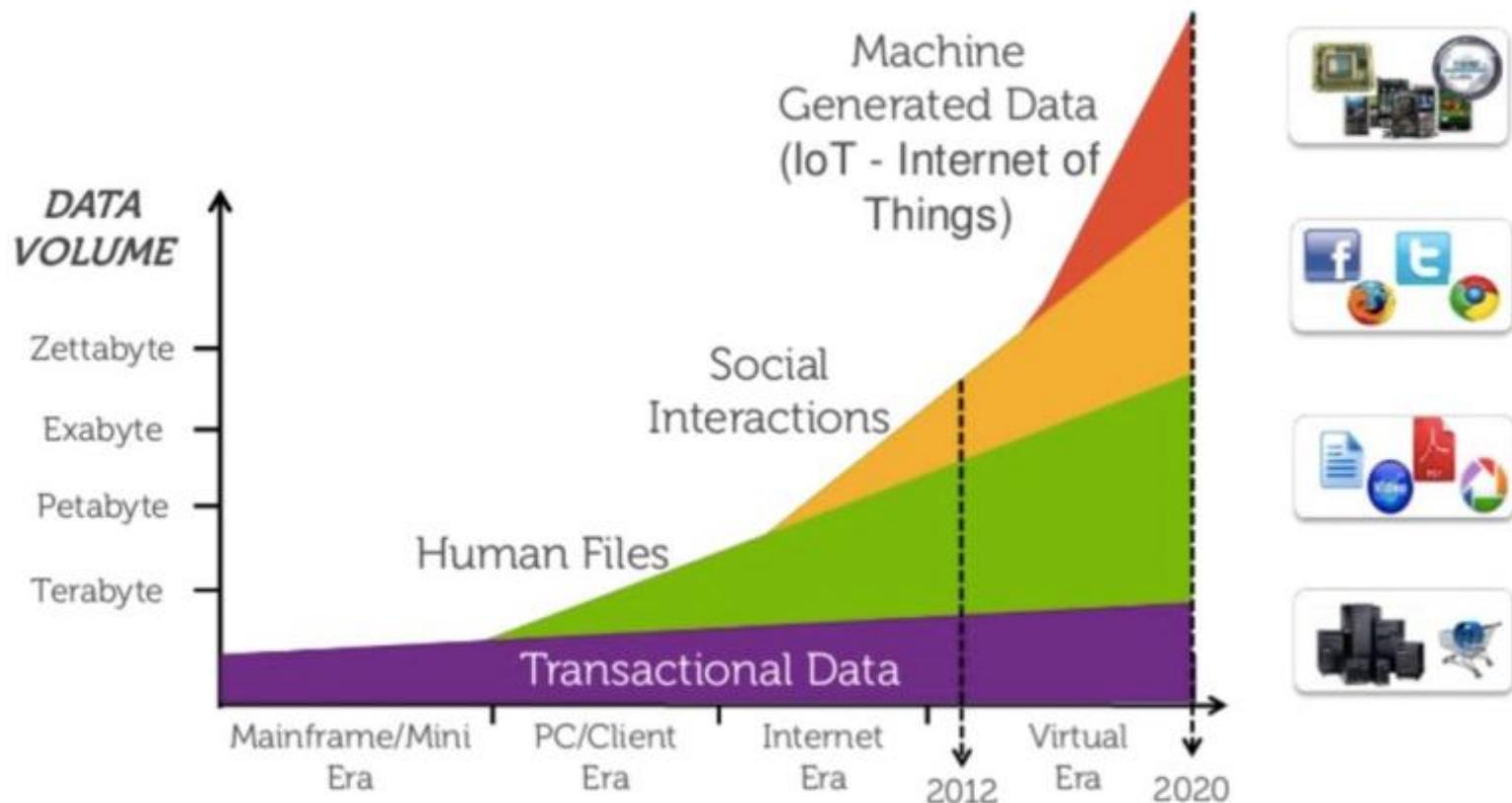
- Četiri osnovne karakteristike podataka na Web-u:
 - Velika količina podataka
 - Povezanost podataka (relacije)
 - Polustruktuiranost podataka
 - Arhitektura aplikacija koje koriste podatke

Podaci na Web-u

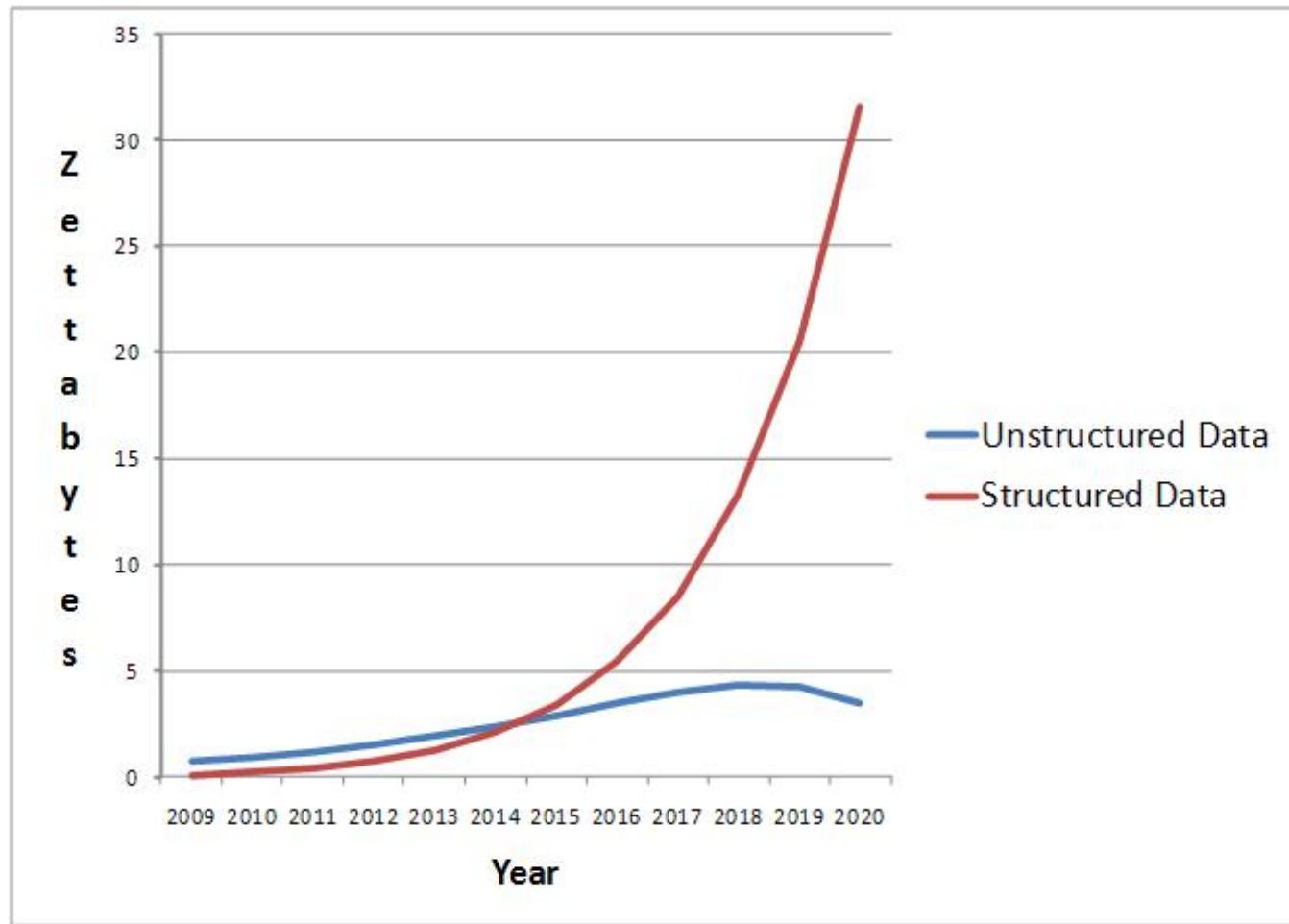


Podaci na Web-u

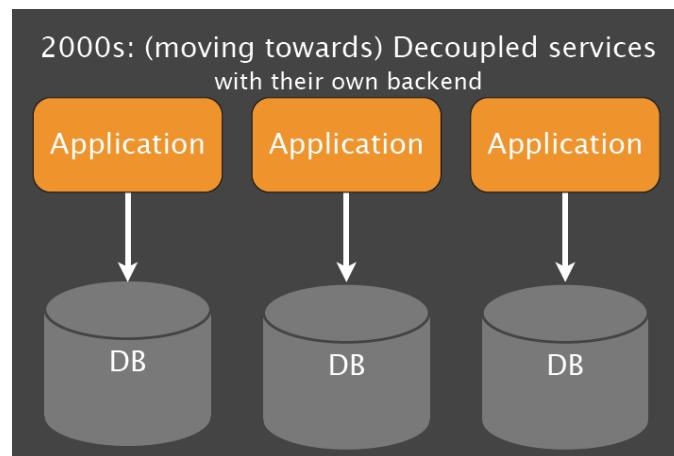
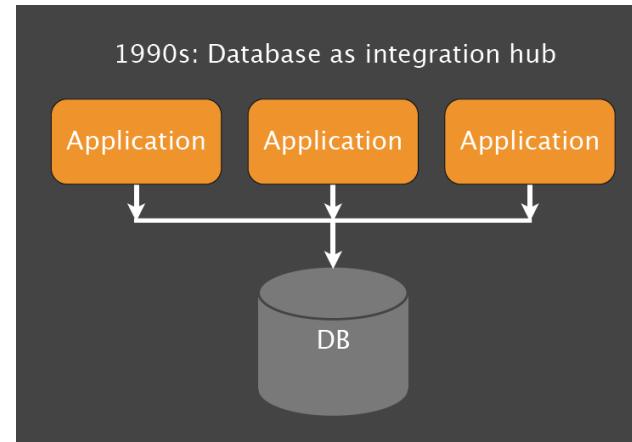
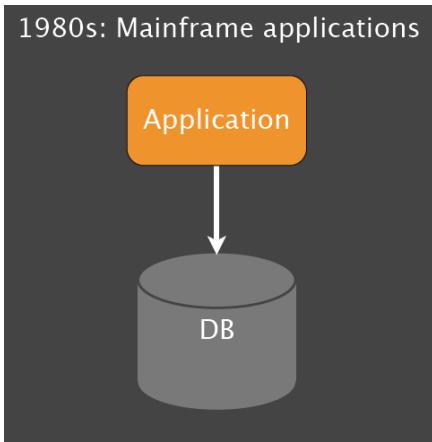
The Explosion of Data



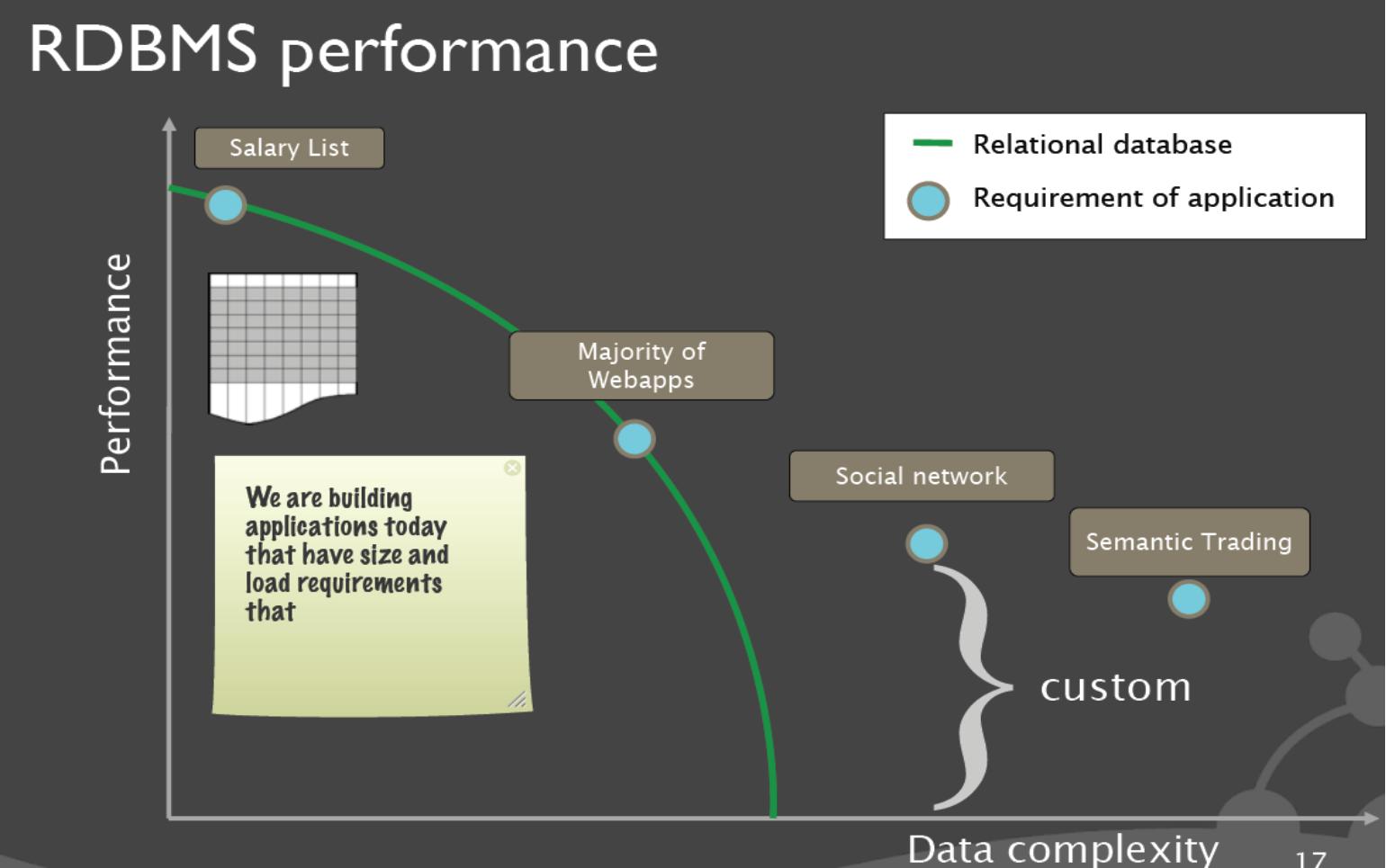
Podaci na Web-u



Podaci na Web-u



Podaci na Web-u



Podaci na Web-u

- Web aplikacije imaju drugačije potrebe u odnosu na aplikacije za koje su RDBMS razvijane.
- Web aplikacije zahtevaju:
 - Ekstremno veliki broj transakcija u jedinici vremena
 - Dinamička analiza velikih količina podataka
 - Kratko i predvidivo vreme odziva (latency)
 - Skalabilnost (po niskoj ceni)
 - Visok nivo dostupnosti (high availability)
 - Fleksibilnu šemu / polustruktuirane podatke
 - Geografska distribuiranost (veći broj čvorova u kojima se podaci obrađuju, mreža kao problem)

Podaci na Web-u

- Web aplikacijama nisu neophodne:
 - Transakcije
 - Kompleksni SQL upiti
 - Stroga konzistentost
 - Integritet podataka
- ACID svojstva su bila korisna kod manjih, horizontalno skalabilnih, normalizovanih relacionih baza podataka sa fiksnom šemom baze podataka.
- Nedostaci RDBMS
 - ACID transakcije nisu skalabilne
 - Horizontalno particionisanje
 - Neefikasni spojevi
 - Transakcije zahtevaju nepotrebnu obradu, odnosno unose dodatni “overhead”
 - Šema relationalnih baza podataka nije fleksibilna

Skalabilnost

- Sposobnost sistema da nastavi dobro da radi kada se dodaju novi resursi u cilju rešavanja problema:
 - Povećanog broja korisnika/zahteva koje treba obraditi
 - Povećena količina podataka koju treba obraditi ili uskladištiti
 - Povećanje kompleksnosti i broja funkcionalnosti koje se nude korisnicima

Skalabilnost

- **Scaling Up**



Skalabilnost

- **Scaling Up**
 - Dodavanje resursa jedinom čvoru u sistemu
 - Dodavanje CPU ili memorije
 - Migracija sistema na jaču platformu
 - Prednosti:
 - Brzo i jednostavno
 - Aplikacija se automatski skalira



Skaliranje

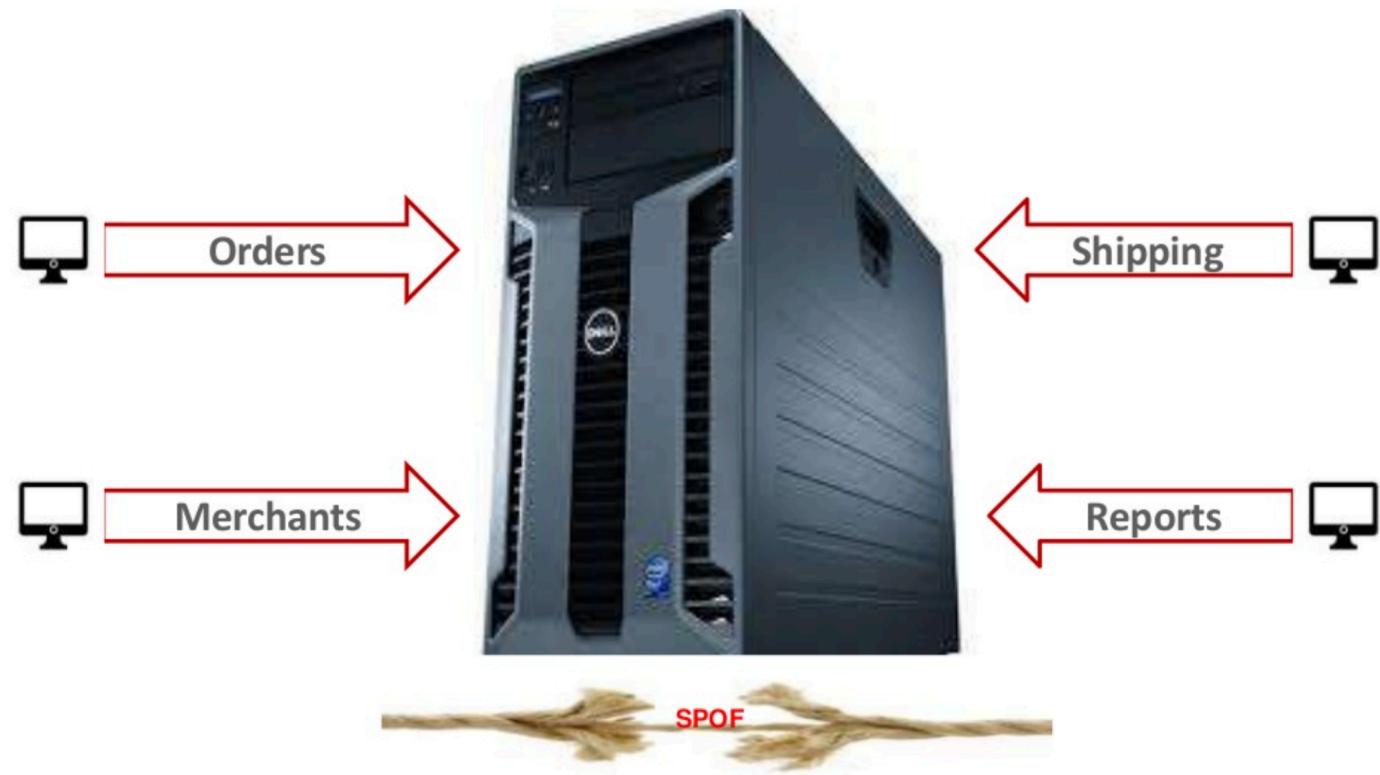
- **Scaling Up**

- **Nedostaci:**

- Kada se prevaziđu kapaciteti najačeg sistema
 - Cena
 - Zavisnost od samo jednog proizvođača
 - Single point of failure (SPOF)

Skaliranje

- Single point of failure



Skaliranje

- **Availability (dostupnost sistema)**
- Definiše se kao sposobnost korisnika da komuniciraju sa sistemom (slanje, ažuriranje ili preuzimanje podataka)
- **Downtime** – period kada sistem nije dostupan
 - Planirani – održavanje i nadogradnje sistema
 - Neplanirani – otkazi usled hardverskih i softverskih grešaka
- **High availability** = $24 \times 7 \times 365$
 - U praksi nastoji se da se minimizira downtime sistema i aplikacija (teži se 0)

Skaliranje

- High availability (NINES)

Availability %	Downtime per Year	Downtime per Month	Downtime per Week	Downtime per Day
90%	36.5 days	72 hours	16.8 hours	2.4 hours
99%	3.65 days	7.20 hours	1.68 hours	14.4 minutes
99.9%	8.76 hours	43.8 minutes	10.1 minutes	1.44 minutes
99.99%	52.56 minutes	4.38 minutes	1.01 minutes	8.66 seconds
99.999%	5.26 minutes	25.9 seconds	6.05 seconds	864.3 milliseconds

Skaliranje

- High availability principi
 - Redundantnost (Redundancy)
 - Detekcija grešaka (Fault detection)
 - Oporavak (Repair/Recovery)
 - Automatizovani i nenadgledani sistemi (Automated & Unattended)

Skaliranje

- Redundantnost



Both engines are always running.

In the event of an engine failure, the remaining engine must provide enough thrust to keep the airplane in flight, even if the failure occurs during take-off ...

Skaliranje

- High availability kod baza podataka
 - Redundantnost hardvera i softvera: više servera, više mrežnih linkova, kopije podataka, rezervno napajanje
 - RPO (recovery point objective) – količina podataka čiji se gubitak može tolerisati
 - RTO (recovery time objective) – downtime koji se može tolerisati
 - DB redundansa – softver i procesi

Skaliranje

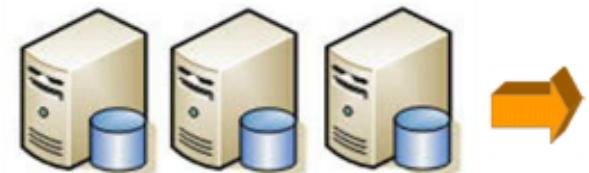
- **Scaling Out**



Skalabilnost

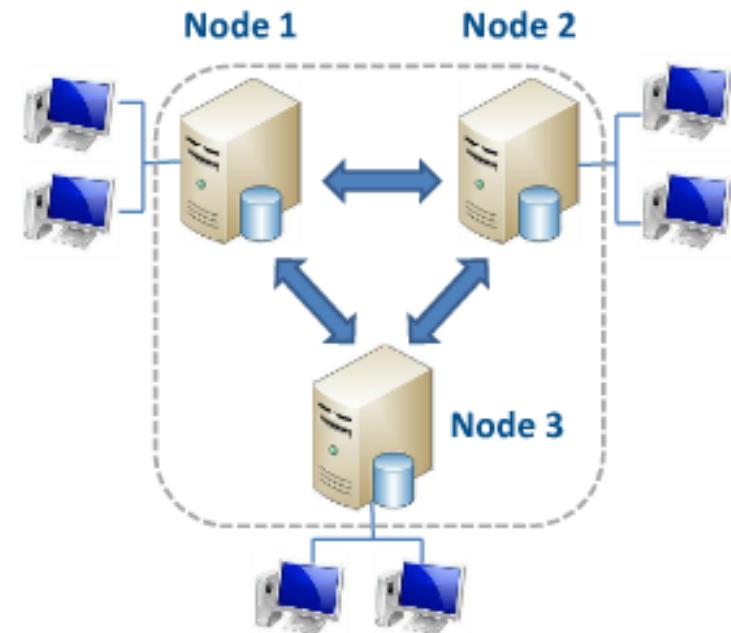
- **Scaling Out**

- **Dodavanje novih čvorova u sistem**
- **Funkcionalana (vertikalna) skalabilnost**
 - Grupisanje podataka po funkciji i distribuiranje funkcionalnih grupa u različitim bazama
- **Horizontalna skalabilnost**
 - Distribuiranje istih funkcionalnih grupa u različitim bazama
- **Prednosti:** fleksibilnost
- **Nedostaci:** kompleksnost



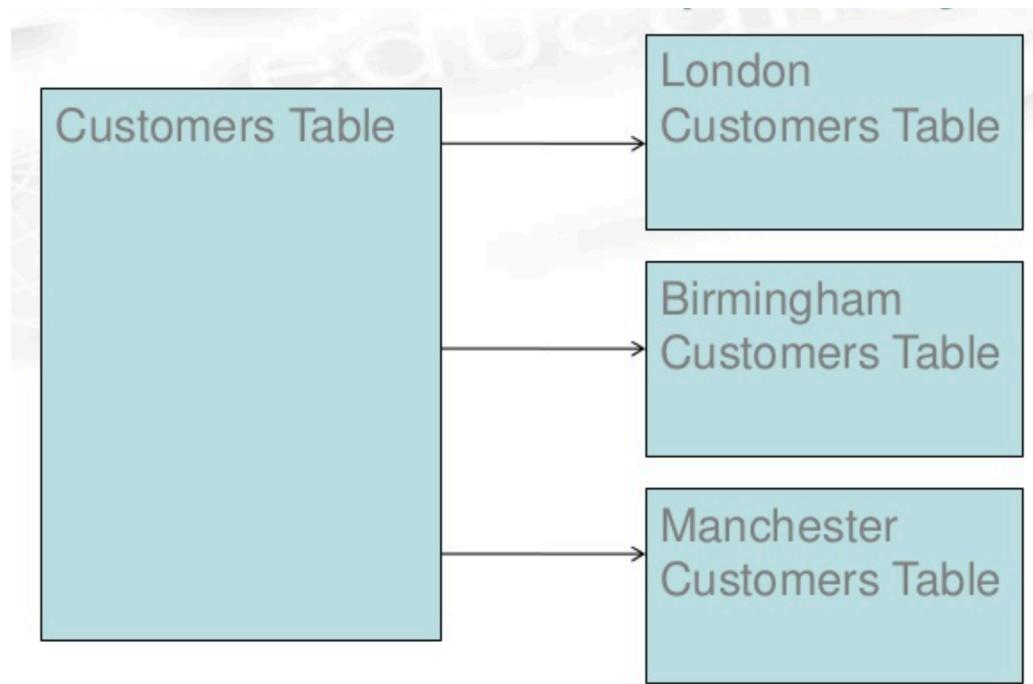
Distribuirane baze podataka

- Više čvorova
- Jedna baza podataka
- Fragmentacija
- Replikacija
- Skalabilnost
- Performanse
- High availability
- Otpornost na greške
- Sa stanovišta korisnika ponaša se kao centralizovana baza podataka



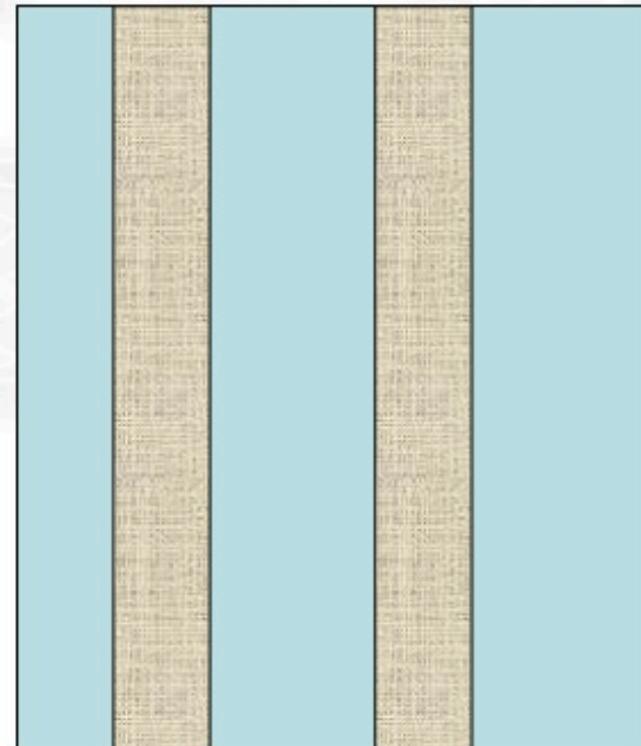
Distribuirane baze podataka

- Fragmentacija i transaprentnost
 - Podela objekta na manje delove
 - Može biti horizontalna i vertikalna



Distribuirane baze podataka

- Vertikalna fragmentacija



→ Create fragment with a
PROJECT with primary keys

← Reconstruct original table
with a JOIN



Distribuirane baze podataka

- Vertikalna fragmentacija

CustomerID	Name	Area	PaymentType	Sex
6	Smith	London	Cash	M
5	Patel	London	Card	F
8	Singh	Manchester	Card	F
9	Kodogo	Birmingham	Card	F
2	Rice	Manchester	Cash	M



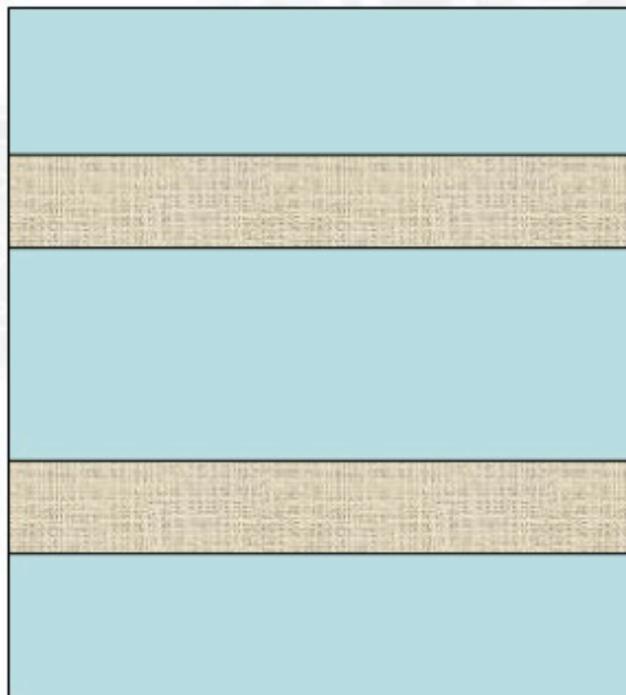
Vertical
Fragmentation

CustomerID, Name,
Sex

CustomerID	Name	Sex
6	Smith	M
5	Patel	F
8	Singh	F
9	Kodogo	F
2	Rice	M

Distribuirane baze podataka

- Horizontalna fragmentacija



→
Create fragment with
RESTRICT

←
Reconstruct original table
with UNION



Distribuirane baze podataka

- Horizontalna fragmentacija

CustomerID	Name	Area	PaymentType	Sex
6	Smith	London	Cash	M
5	Patel	London	Card	F
8	Singh	Manchester	Card	F
9	Kodogo	Birmingham	Card	F
2	Rice	Manchester	Cash	M



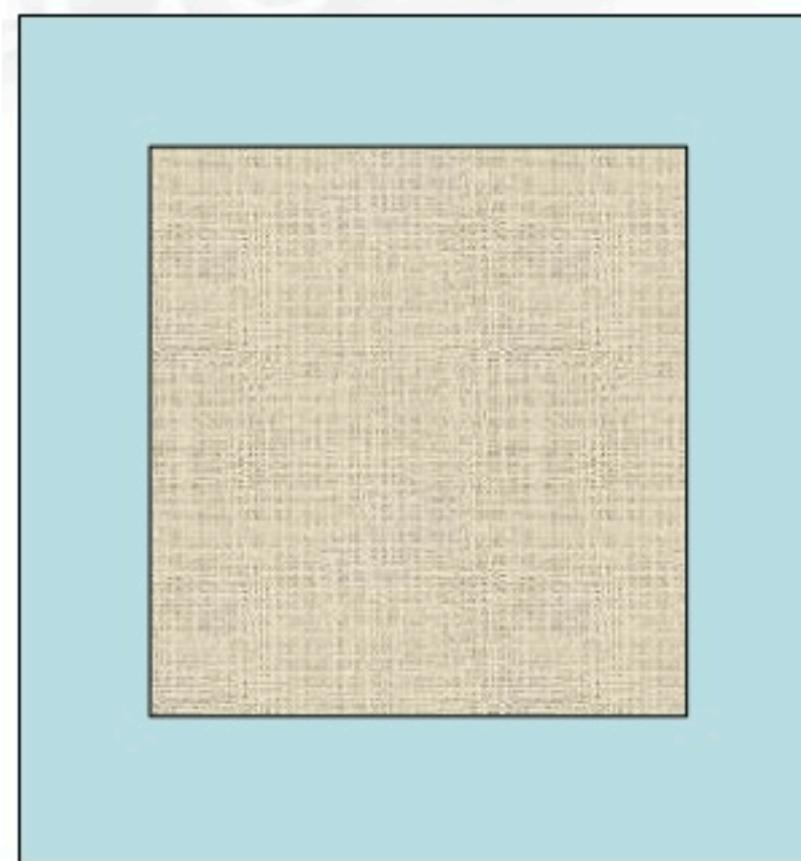
CustomerID	Name	Area	PaymentType	Sex
8	Singh	Manchester	Card	F
2	Rice	Manchester	Cash	M

Horizontal Fragmentation

Customers in Manchester

Distribuirane baze podataka

- Vertikalna i horizontalna fragmentacija





Distribuirane baze podataka

- Vertikalna i horizontalna fragmentacija

CustomerID	Name	Area	PaymentType	Sex
6	Smith	London	Cash	M
5	Patel	London	Card	F
8	Singh	Manchester	Card	F
9	Kodogo	Birmingham	Card	F
2	Rice	Manchester	Cash	M



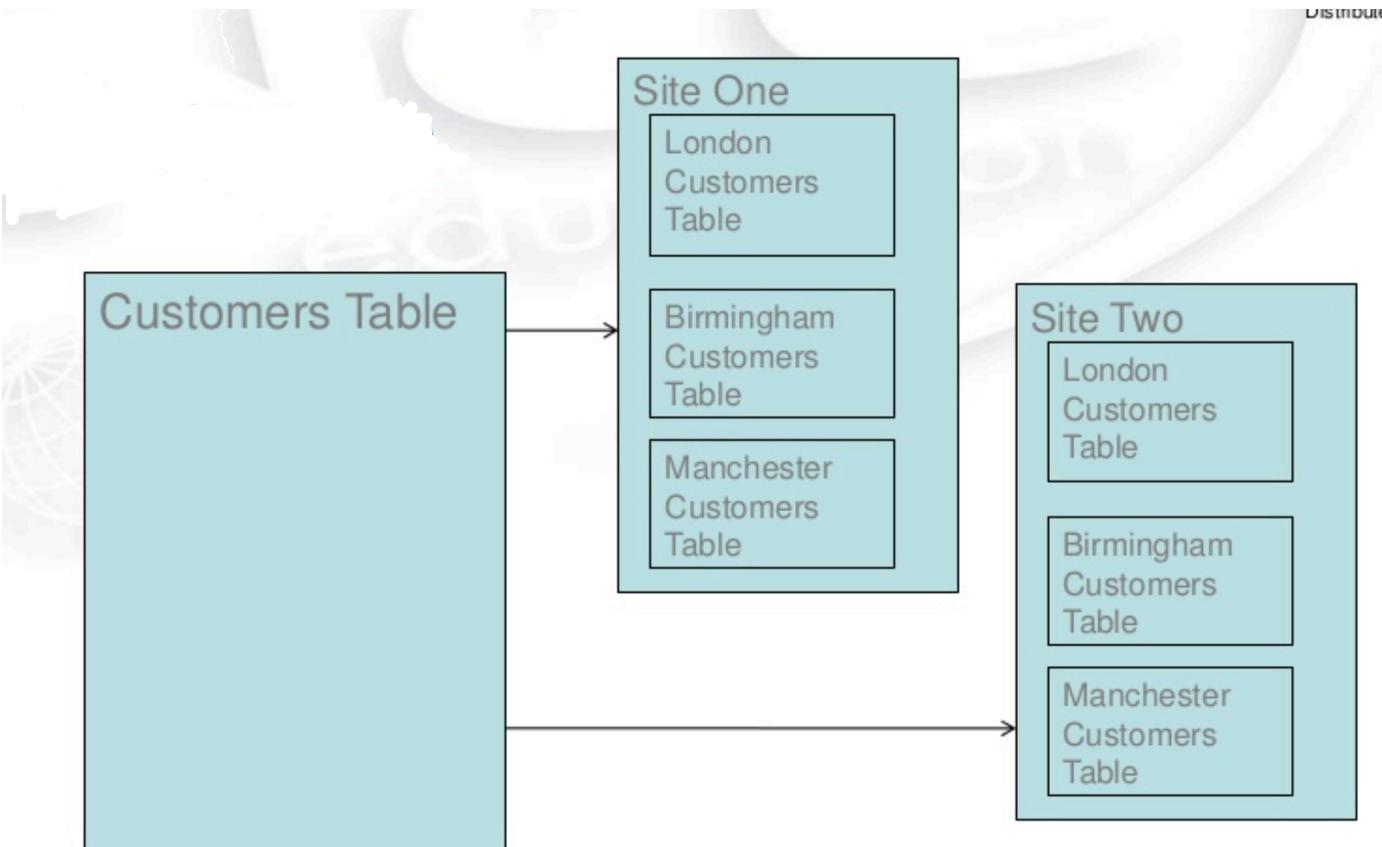
CustomerID	Name	PaymentType
8	Singh	Card
2	Rice	Cash

Vertical and Horizontal Fragmentation

Customers in Manchester and their
Payment Type

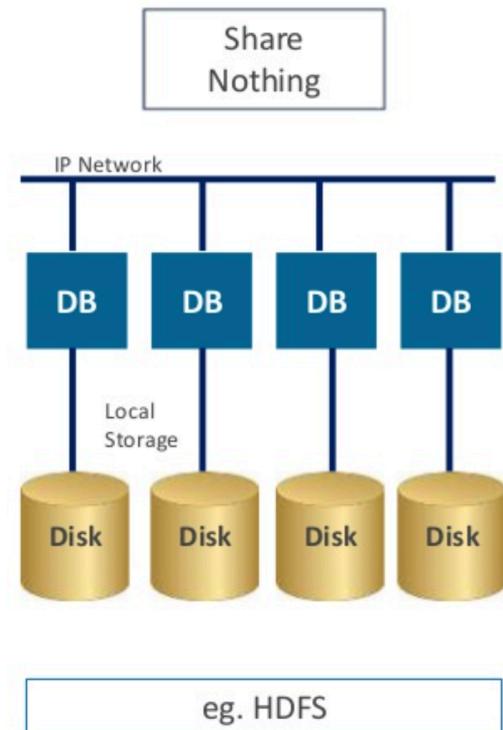
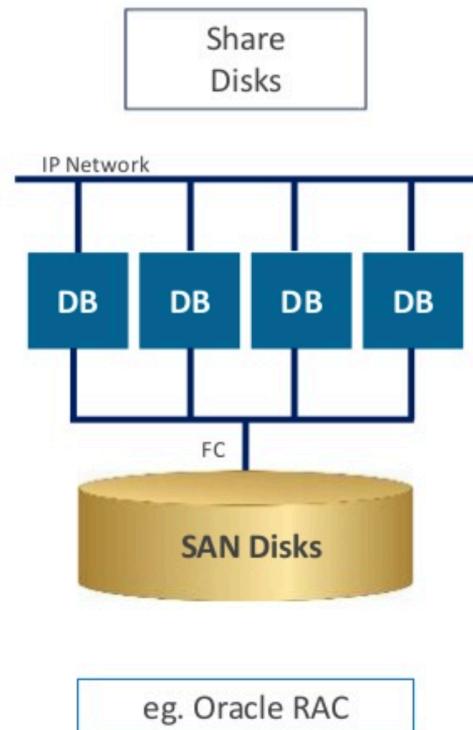
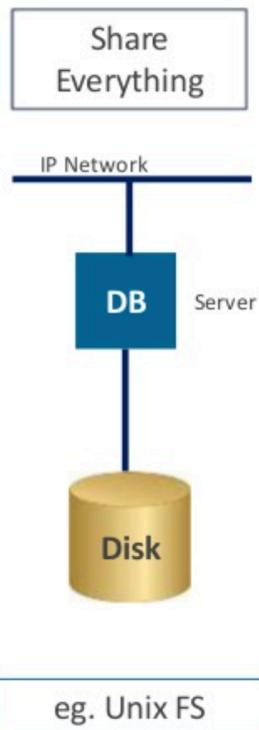
Distribuirane baze podataka

- Replikacija



Distribuirane baze podataka

- Shared nothing arhitektura





Distribuirane baze podataka

- Prednosti:
 - Bolja kontrola
 - Bolje performanse
 - Povećana dostupnost sistema
 - Lakše skaliranje sistema
- Nedostaci:
 - Kompleksnost
 - Cena
 - Sigurnost
 - Otežana kontrola integriteta



Distribuirane baze podataka

- Zahtevi koje moraju da ispune distribuirane baze podataka:
 - **Consistency** – sistem se nalazi u konzistentnom stanju posle svake operacije
 - Svi klijenti vide iste podatke
 - Sistem konzistentno prati sva definisana pravila i protokole
 - **Availability** – sistem je uvek dostupan (“*always on*”)
 - “*no downtime*”
 - Tolerancija na otkaz čvorova – klijenti uvek imaju pristup nekoj od kopija (replika)
 - Tolerancija na HW/SW promene
 - **Partition tolerance** – sistem funkcioniše čak i u slučaju da ne postoji konekcija između distribuiranih podskupova (pad mreže)
 - Ne samo za čitanje već i za upis



Distribuirane baze podataka

- **CAP Teorema (E. Brewer, N. Lynch)**
 - Brewer's teorema
 - **U potpunosti je moguće zadovoljiti samo 2 od 3 zahteva.**
 - Kompromis oko trećeg zahteva
 - **Odustaje se od pristupa “sve ili ništa”**
 - Biraju se različiti nivo konzistentnosti, dostupnosti ili particionisanja.
 - Treba prepoznati koja su od CAP pravila neophodna za funkcionisanje sistema.



Distribuirane baze podataka

- **CA: Consistency & Availability**
 - Kompromis oko Partition Tolerance
 - Karakteristična za single-site cluster rešenja
(lakše je obezbediti da su svi čvorovi u stalnom kontaktu)
 - Kada dođe do narušavanja topologije mreže, odnosno do particonisanja mreže, sistem se blokira.
 - Primer: dvofazni komit (2PC)



Distribuirane baze podataka

- **CP: Consistency & Partitioning**
 - Kompromis za Availability
 - Pristup pojedinim podacima može biti privremeno onemogućen ili ograničen
 - Ostatak sistema se nalazu u konzistentom/tačnom stanju
 - Primer: horizontalno particonisane baze podataka na većem broju servera (*sharded databases*)



Distribuirane baze podataka

- **AP: Availability & Partitioning**
 - Kompromis za Consistency
 - Sistem je dostupan i prilikom narušavanja mrežne topologije
 - Neki od podataka koje sistem vraća mogu biti privremeno neažurni (***temporarily not up-to-date***)
 - Zahteva strategiju za rešavanje konflikta (***conflict resolution strategy***)
 - Primer: DNS, keš, master/slave replikacija



Distribuirane baze podataka

- **CAE trade-off (Amazon)**
 - Cost-efficiency
 - High Availability
 - Elasticity
- Biraju se bilo koja dva (C,A, E)
 - Klijent čeka kada je sistem opterećen (C i E)
 - Ukoliko je moguće predvideti opterećenje, moguće je obezbediti A i C rezervisanjem resursa unapred
 - Nepotrebni resursi (over-provisioning) – A i E
- Svi žele A, problem je obezbediti C



Distribuirane baze podataka

• **BASE**

- CAP varijanta ACID svojstava
- **Basically Available**
- **Soft State**
- **Eventually Consistent**
- ACID forsira konzistentnost podataka dok BASE prihvata da će se konflikti desiti.



Distribuirane baze podataka

- Basically Available
 - Sistem garantuje dostupnost podataka u skladu sa CAP teoremom, odnosno sistem generiše odgovor na svaki zahtev
 - Odgovor može da vrati grešku ili da vrati podatke koji nisu konzistentni ili se menjaju
- Soft state
 - Stanje sistema se menja tokom vremena čak i kada nema ulaznih podataka zahvaljujući svojstvu eventual consistency.
- Eventual consistency
 - Sistem će dostići konzistentnost u određenom trenutku nakon što prestane da prima nove ulazne podatke.
 - Podaci se vremenom propagiraju u sve delove sistema
 - Sistem nastavlja da prima nove podatke, i ne proverava konzistentnost svake transakcije pre nego što pređe na narednu transakciju.

NoSQL baze podataka

- NoSQL baze podataka predstavljaju pokret a ne specifikaciju.
- Prvi put upotrebljen 1998. godine.
- **NoSQL != No SQL**
- **NoSQL == Not Only SQL**
- Termin se upotrebljava za sve nerelacione baze podataka (non-RDBMS)

NoSQL baze podataka

- Internet
- Google
- Bigtable whitepaper (Google) – 2006
- Dynamo whitepaper (Amazon) – 2007
- Cassandra release (Facebook) – 2008
- Voldemort release (LinkedIn) - 2009

NoSQL baze podataka

- Tipična primena:
 - Velike količine podataka (Massive data volumes)
 - Za skladištenje podataka se koristi distribuirana arhitektura
 - Google, Amazon, Facebook – 10K-100K servera
 - Veliki broj upita (Extreme query workload)
 - Nemogućnost efikasnog izvršavanja spojeva kod RDBMS u takvom okruženju
 - Schema evolution
 - Nije jednostavno obezbititi fleksibilnost šeme
 - Promene u šemi se mogu postepeno uvoditi kod NoSQL

NoSQL baze podataka

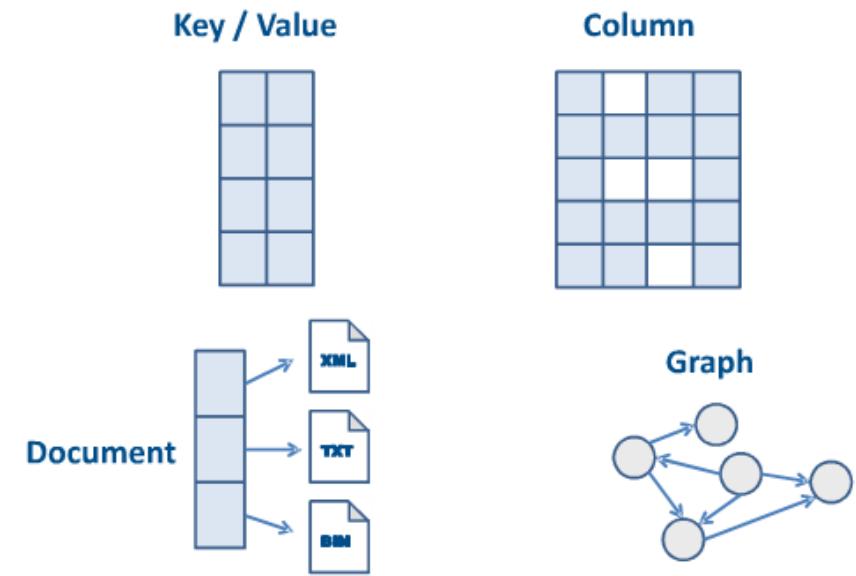
- Dobre strane:
 - Fleksibilnost
 - Skalabilnost
 - Eventually consistent
 - Jeftine (osnova, infrastruktura je skupa)
 - Prilagođene potrebama Web aplikacija

NoSQL baze podataka

- Loše strane:
 - Tehnologija još uvek nije stabilna
 - Ne postoje zajednički standardi
 - Loša podrška za transakcije
 - Loša podrška za pretraživanje podataka
 - Zahteva promenu načina razmišljanja
 - Vrlo je teško naći dva identična scenarija primene.

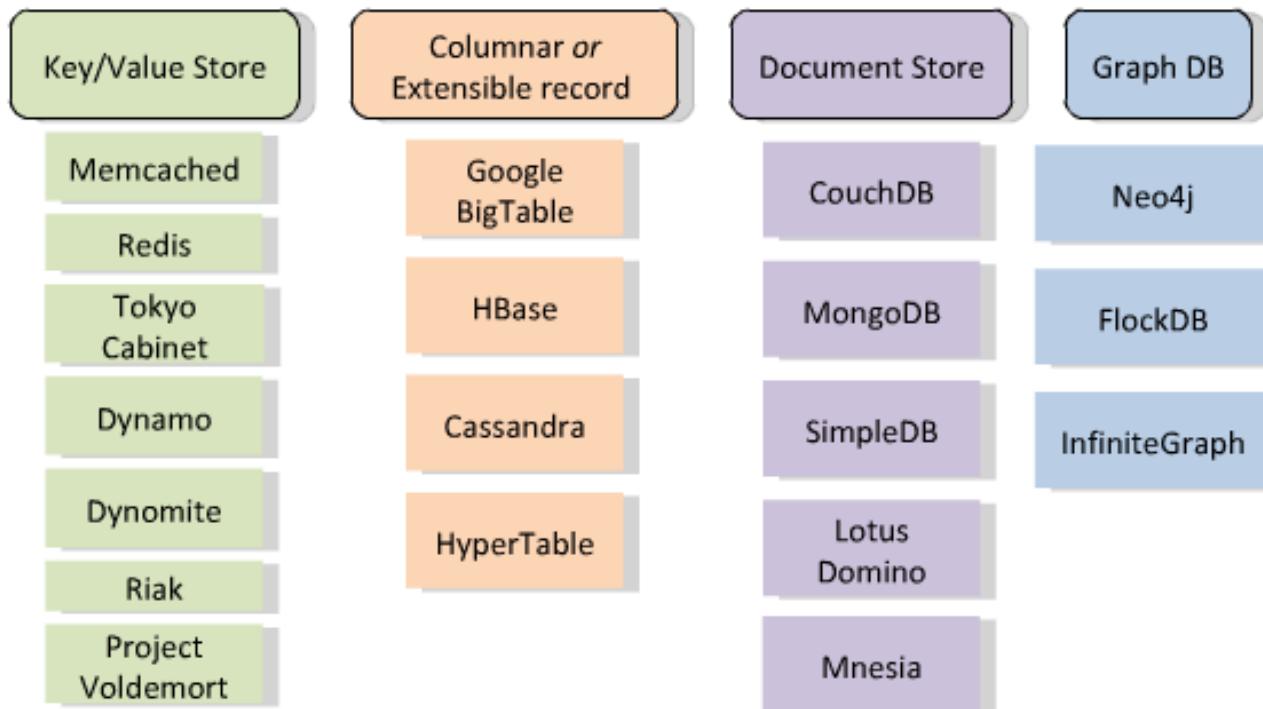
Taksonomija

- Key/Value stores
- Column stores (Extensible records)
- Document stores
- Graph databases



Taksonomija

Recent NOSQL database products

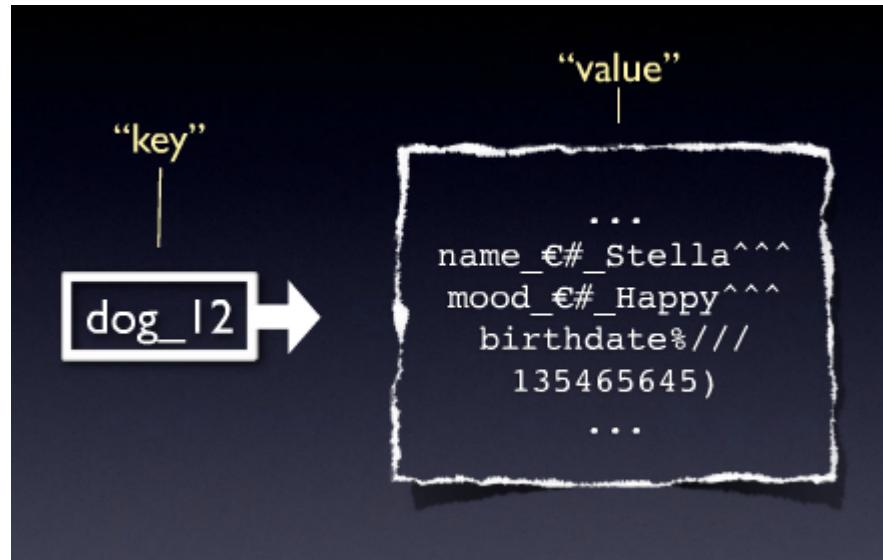


Taksonomija

- **Key/Value stores**
 - Key/Value lookups (DHT), Hash
 - Jedna vrednost, jedan ključ, nema duplikata, izuzetno brzo
 - Skaliranje ogromnih količina podataka
 - Projektovane da podnesu velika opterećenja
 - Podataka je obično BLOB, DB ne razume strukturu podatka.
 - Primer: Riak, Redis, Project Voldemort (Amazon Dynamo whitepaper)

Taksonomija

- Key/Value stores

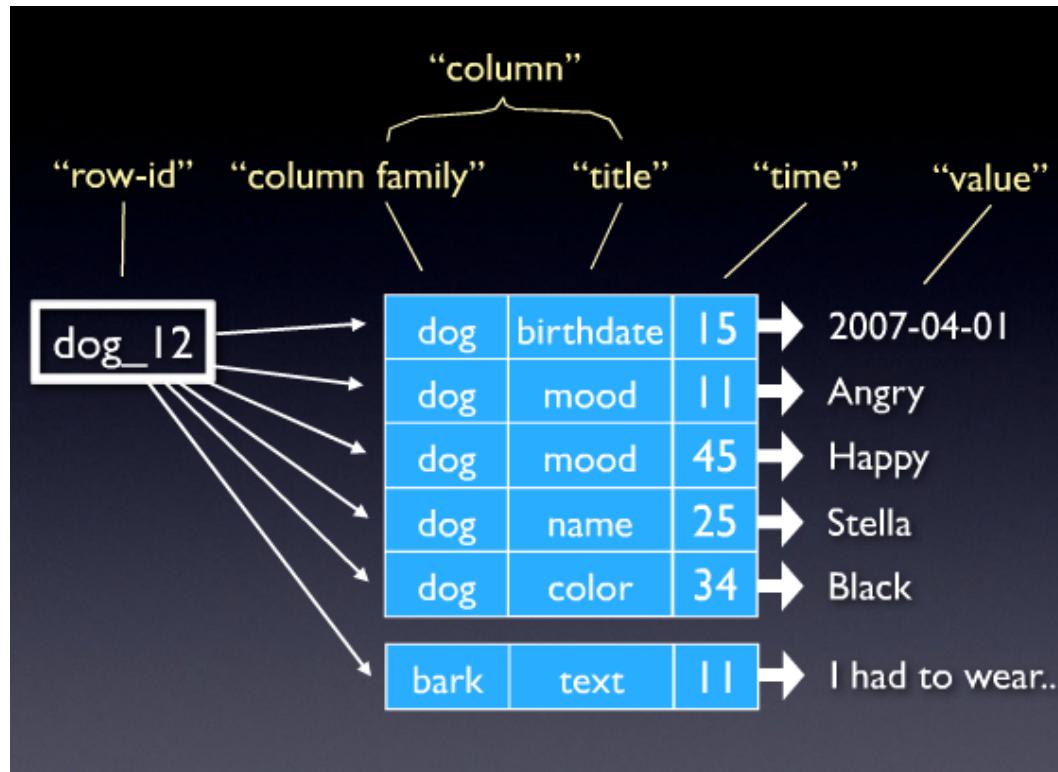


Taksonomija

- **Column stores**
 - BigTable kolonovi
 - Rasuta, distribuirana multi-dimenzionalan sortirana mapa
 - Konceptualno:
 - Jedna tabela, beskonačno velika
 - Svaka vrsta može imati različite kolone (po broju i tipu)
 - Tabela je retko posednuta: $|\text{rows}| * |\text{columns}| > |\text{values}|$
 - Primer: Hbase, Cassandra, Hypertable

Taksonomija

- **Column stores**



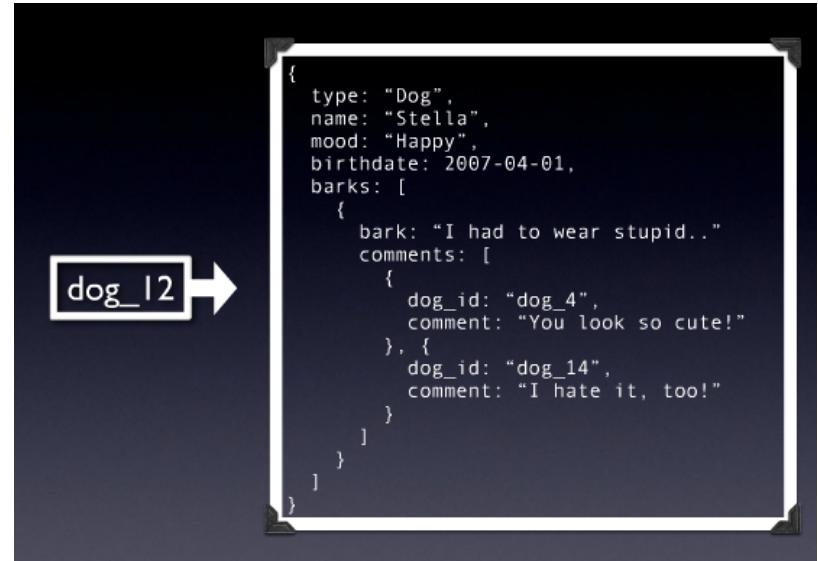
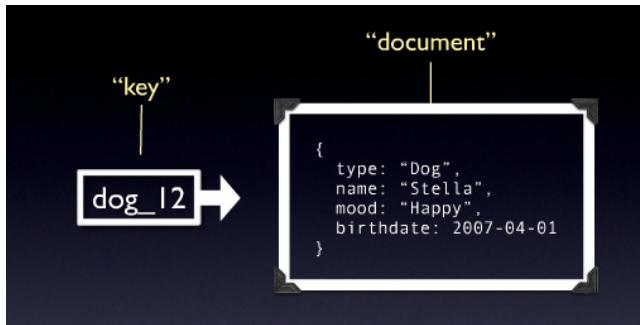
Taksonomija

- **Document stores**

- Key/Value store, value predstavlja polu-strukturani dokument čija je struktura razumljiva DB
- Podaci se mogu pretraživati ne samo po ključu
- Polu-strukturani dokumenti (XML, JSON)
- Primer: MongoDB, CouchDB, Amazon SimpleDB

Taksonomija

- **Document stores**



Taksonomija

• **Graph databases**

- Inspirisane matematičkom teorijom grafova: $G = (E, V)$
- Modelira se struktura podataka
- Navigacioni model podataka
- Skalabilnost / kompleksnost podataka
- Model: Key/Value parovi za Potege/Čvorove
- Relacije: Potezi između čvorova
- Primer: Neo4j, AllegroGraph, OWLIM

Taksonomija

- **Graph databases**

