32. NoSQL databáze- koncept, vlastnosti, dělení, srovnání s relačními databázemi. Pojmy: volné schéma, CAP teorém, indexování, agregace, replikace, škálování, sharding.

NoSQL databáze

- Nerelační databáze (žádné tabulky)
    - dokáží zpracovávat obrovské objemy rychle se měnících nestrukturovaných dat
    - pomáhají vývojářům rychle vytvářet databázové systémy pro ukládání nových informací a jejich přípravu pro vyhledávání a analýzu
    - umožňují
        - flexibilní vývoj
        - flexibilní zpracování dat
        - provoz v libovolném měřítku
    - ➢ cílem tedy ve výsledku není nahradit relační databáze
        - jedná se spíš o alternativu s jinou aplikací (kladivo vs. sekera)
        - ➢ vhodné např. pro big data a webové aplikace v reálném čase

Koncept

Vlastnosti

Dělení

Srovnání s relačními databázemi

Nejvhodnější pro:

- relační databáze
    - zpracování relačních dat majících logické a diskrétní požadavky určitelné předem
    - schéma potřebné udržovat a synchronizovat mezi aplikací a databází
    - starší systémy vytvořené pro relační struktury
    - aplikace vyžadující komplexní dotazování nebo transakce s více řádky
- NoSQL databáze
    - zpracování velkých objemů vzájemně nesouvisejících, nebo rychle se měnících dat
    - data nezávislá na schématu nebo schéma určené aplikací
    - aplikace preferující výkon a dostupnost než silnou konzistenci
    - neustále přístupné aplikace sloužící uživatelům po celém světě

Typ databáze:

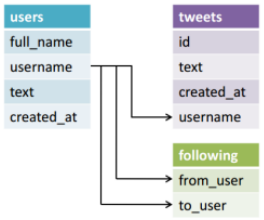
- relační databáze
  - definované relace (tabulky)
  - definované atributy (sloupce)
  - definované vazby mezi relacemi
  - ➢ data ukládána ve formě záznamů (řádků) v relacích
    - související data jsou uložena odděleně
    - v případě dotazů spojovány
  - ➢ umožňují snadno modelovat vztahy mezi daty

- NoSQL databáze
  - nerelační databáze
    - žádné relace
  - definovaný mechanismus pro ukládání a načítání dat
    - liší se databázi od databáze
  - ➢ vyhýbají se vazbám mezi daty
  - distribuované databáze

MarkLogic

Typy databází:

- relační databáze
  - databáze relací záznamů seskupených do vztahů



http://www.padakuu.com/article/295-relational-databases

- NoSQL databáze
  - key-value úložiště
  - dokumentové databáze
  - sloupcová úložiště
  - grafová úložiště
  - databáze časových řad
  - a další …

influxdb

Datové schéma:

- relační databáze
  - strukturovaná data
  - ➢ pevně předdefinované datové schéma
    - jasně definuje databázi
    - určuje relace, atributy, vztahy
    - ale také např. datové typy, indexy, pohledy, procedury, triggery a veškerá další omezení

- NoSQL databáze
  - umožňuje všechna data
    - strukturovaná data
    - částečně strukturovaná data
    - nestrukturovaná data
    - ➢ různé typy big data
  - ➢ volné schéma
    - žádné předdefinové nastavení
    - ➢ flexibilita

IBM DB2

redis

Hierarchické ukládání dat:

- relační databáze
  - obtížné hierarchické ukládání
  - data ukládána v relacích
    - s nárůstem počtu relací roste i náročnost na vytváření vazeb
    - udržitelné jen do určitého bodu

- NoSQL databáze
  - snadné hierarchické ukládání
  - klíč – hodnota
  - vhodné pro velké objemy dat
    - big data

ZÁVĚREČNÉ SROVNÁNÍ:

- výhody relačních databází
  - vztahy mezi daty – relace
    - velké množství vazeb a propojení
    - pevně dané schéma
    - komplexní dotazy
  - normalizace
    - databáze zabírá co nejméně místa
    - ideálně nulová redundance dat
  - známý dotazovací jazyk
  - datová integrita
    - smazání autora odstraní i jeho články
  - ACID vlastnosti
    - transakce se uskuteční celá, nebo vůbec

výhody NoSQL
- datový model
  - schéma není pevně definované
  - strukturovaná, částečně strukturovaná i nestrukturovaná data
    - flexibilní zpracování dat
- možnost umístění v distribuovaných cloudech
- horizontální škálování
- replikace
- práce s big data
  - analýza velkého množství dat
    - rychlost, výkon
- různé typy databází vyhovující různým aplikacím
- levnější na údržbu

- relační databáze
  - provázaná data
  - pevné schéma
  - strukturovaná data
  - vertikální škálování
  - ACID
  - SQL
  - jednoduché složité dotazy
  - lepší podpora
  - transakce

- NoSQL databáze
  - velké množství dat
  - flexibilní schéma
  - veškerá data
  - horizontální škálování
  - CAP
  - velké množství jazyků
  - komplikované složité dotazy
  - převážně open source
  - analýza

Pojmy:

Volné schéma

Pojem "volné schéma" (anglicky "schemaless") se v kontextu NoSQL databází používá k popisu databázového modelu, který neklade pevná omezení na strukturu uložených dat. Na rozdíl od tradičních relačních databází, které vyžadují pevně dané schéma a tabulky s přesně definovanými sloupci a jejich datovými typy, NoSQL databáze umožňují flexibilnější přístup k datům. Ve volném schématu NoSQL databáze není nutné předem definovat strukturu dat. Namísto toho můžete jednoduše vložit nebo upravit data v libovolném formátu, a databáze si s nimi poradí. To znamená, že jednotlivé záznamy mohou mít různé struktury a odlišné sady atributů.

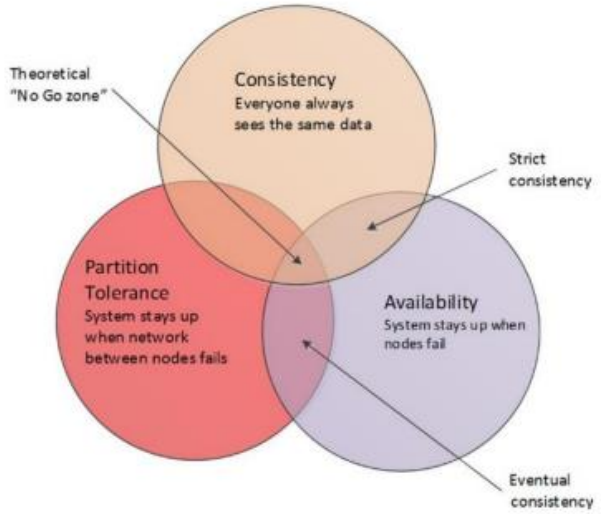
- volné schéma
  - dokumenty mohou být velmi komplexní
  - mohou obsahovat vnořené (embedded) dokumenty
  - nemusí obsahovat stejné oddíly, atributy, části nebo klíče
  - podobně jako objekty
  - vysoká míra flexibility

Tento přístup má několik výhod. Především je velmi flexibilní a umožňuje snadné přidávání a úpravy atributů bez potřeby migrovat celou databázi. To je užitečné v prostředích, kde se často mění požadavky na datový model nebo kde se pracuje s nestrukturovanými nebo polostrukturovanými daty.

Nicméně, volné schéma může také přinášet některé nevýhody. Při zpracování dat je nutné provést analýzu struktury každého záznamu, což může zpomalit některé operace. Navíc, nedostatek pevné struktury může vést k obtížím při zajištění konzistence dat, pokud jsou předpokládány určité vzorce nebo vazby mezi entitami.

CAP teorém

pro distribuovaný datový sklad nelze zajistit více jak 2 následující vlastnosti



- konzistence – consistency
  - všechny uzly mají stejná data
  - každé čtení vrací poslední výsledek, nebo chybu
- dostupnost – availability
  - na každý dotaz je vrácena (nechybová) odpověď
- odolnost k přerušení – partition tolerance
  - systém funguje dál i při zdržení či ztrátě zprávy
  - nezbytné pro big data

eventuálně konzistentní (AP)

- často volena škálovatelnost a vysoká dostupnost než 100% konzistence

NoSQL databáze nemohou dosáhnout absolutní konzistence (ACID) ve světě distribuovaných systémů.

C = *consistency* = každé čtení získá poslední zapsanou hodnotu nebo je signalizována chyba

A = *availability* (dostupnost) = všechny požadavky jsou čtení/zápis jsou vždy obslouženy (s určitou maximální latencí)

P = *partition tolerance* = systém funguje i při rozpadu na více částí (tj. i když jsou ztraceny či zpožděny zprávy mezi uzly)

Důsledek CAP teorému: **V distribuovaném systému lze dosáhnout vždy maximálně dvou vlastností z CAP.**

Klíčový důsledek: Pokud se předpokládá odolnost proti rozdělení (která je v distribuovaném světě nezbytná), pak je nutné si vybrat mezi (dobrou) dostupností a (dostatečnou) konzistencí.

- tj. např. lepší dostupnosti, pokud netrváme na dokonalé konzistenci
- vyšší odolnosti vůči rozpadu, za cenu nižší dostupnosti = vyšší latence (apod.)

Indexování

Indexování je proces vytváření indexů pro rychlé vyhledávání a filtrování dat v databázi. Indexy umožňují efektivní vyhledávání a zlepšují výkon dotazů tím, že předem strukturovaně ukládají informace o konkrétních sloupcích nebo hodnotách v databázi. To umožňuje rychlé nalezení a přístup k datům, což je zvláště užitečné při vykonávání dotazů, které obsahují podmínky a omezení.

Agregace

Agregace se v NoSQL databázích používá k zpracování a sumarizaci dat. Agregační operace umožňují **spojovat, skupinovat, filtrovat a provádět výpočty nad daty v databázi**. Tyto operace umožňují vykonávat komplexní analýzy, statistiky a operace nad velkým množstvím dat přímo v databázi, což eliminuje potřebu vývojářů přenášet a zpracovávat velká datová množství mimo databázi.

Škálování

Škálování se v NoSQL databázích **týká schopnosti systému zvládnout narůstající množství dat**, požadavků a provozu. Existují dvě základní metody škálování: vertikální a horizontální. Vertikální škálování zahrnuje zvětšování výpočetního výkonu a kapacity jednoho serveru (například přidání více paměti nebo procesorů), zatímco horizontální škálování zahrnuje přidávání dalších serverů

- škálovatelnost je vlastnost systému zvládat rostoucí množství požadavků přidáváním dalších prostředků

- relační databáze
  - vertikální škálování (nahoru)
    - změna vlastností stávajících prvků
    - ➢ přidání výpočetní síly
  - zpracování dat se nemění
    - je prováděno na výkonnějším stroji
  - zvýšení kapacity
    - nové komponenty jsou drahé
      - disky, paměti, CPU, síťování, atd.
    - ➢ nelze donekonečna
  - jednoduchost

- NoSQL databáze
  - horizontální škálování (ven)
    - přidání (nebo ubrání) prvků
    - ➢ přidání výpočetních uzlů
  - přesun k paralelnímu zpracování
    - rozdělení dat mezi uzly – sharding
  - zvýšení kapacity
    - nové komponenty jsou levné
      - základní HW
    - ➢ clustery
  - výkon, flexibilita, redundance

Replikace

kopie dat jsou umístěny na některých uzlech clusteru => zvýšení dostupnosti, propustnosti a odolnosti proti rozpadu clusteru

Sharding

různé části dat (shards = střep) **jsou rozmístěny na různé uzly v clusteru** => zvyšuje se kapacita a při vhodném rozmístění i dostupnost dat

• rovnoměrné rozdělení dat na uzlech (předpokládáme homogenní cluster)

• minimalizace počtu uzlů, na které musí uživatel přistoupit, aby data získal (jednoduchá distribuční funkce)

• optimalizace fyzického umístění serveru vzhledem k umístění klientů (topologie sítě, geografická lokalizace)