NoSQL

Pojem NoSQL

- Termín NoSQL databáze označuje velmi početnou širokou skupinu nerelačních databází.
- NoSQL databáze na rozdíl od relačních databází většinou nepoužívají dotazovací jazyk SQL.
- Termín NoSQL bývá databázovou komunitou vysvětlován jako „not only SQL“ („nejen SQL“). Alternativní označení *postrelační databáze*.
- Do skupiny NoSQL databází patří mnoho datových úložišť.
  - Mimo jiné sem patří i představitelé těchto druhů databází: grafové, dokumentové, objektové, XML databáze a okrajově také některý software, který zcela nesplňuje parametry pro databázi.
- Nejčastější uplatnění:
  - Cloud computing,
  - Aplikace Web. 2.0 (webové stránky obsahující prostor pro sdílení a společnou tvorbu obsahu),
  - Sociální sítě (nutné horizontální škálování zahrnující obrovské množství uzlů).

Historie

- První společnost, která musela řešit problém s velkým objemem strukturovaných dat – Google (převažující charakter uchovávaných dat - obsahy stránek, propojení mezi stránkami a textové odkazy).
- Problém Google řešen pomocí vytvoření projektu BigTable, - počátek velkého rozvoje databází s pozdějším společným označením NoSQL databáze.
- Databáze jsou většinou založené na principu klíč/hodnota.
- BigTable znamená zvrat v databázovém světě - není řádkově orientovanou databází, jako tomu je u tradičních databází, ale databází sloupcově orientovanou.

Historie

- Další společností, které se naskytl podobný úkol k vyřešení jako společnosti Google, je Amazon. Největší internetový prodejce se inspiroval projektem BigTable.
- Amazon použil stejné ukládání dat s využitím principu klíč/hodnota (key/value) - konkrétní řešení v distribuované databázi s názvem Dynamo (ukládá data o prodávaných produktech, všechny dotazy na stránkách Amazonu jsou prováděna nad touto databází).
- Oba projekty, BigTable i Dynamo, se staly inspirací pro vznik a vývoj několika dalších NoSQL databází.

Oracle Berkeley DB

- Historicky nejstarší databáze Oracle Berkeley DB se na rozdíl od většiny NoSQL databází se neinspirovala projektem BigTable.
- Její vznik se odhaduje na dobu cca před 16 roky, přičemž se její škála vlastností postupně rozšiřovala. Dnes se dělí do tří jedinečných verzí: Berkeley DB, Berkeley DB Java Edition, Berkeley DB XML.
- Jedná se o produkt firmy Oracle, přestože byla vytvořena na univerzitě v Berkeley. Všechny zmíněné rysy NoSQL databází spočívající v používání různých dotazovacích jazyků či rozhraní (nejen SQL) lze najít i u Oracle Berkeley DB umožňující ukládání a čtení dat s využitím:
  - SQL syntaxe,
  - XQuery,
  - Java objektů,
  - párů klíč/hodnota.

Datový model

- V databázové světě se tradičně k popisu databáze využívá datový (logický) model.
- NoSQL se naproti tomu uznává jiný přístup, intuitivní a bez využívání předepsaných norem, podle kterého se pak různí i terminologie.
- Mizí zde odlišnosti mezi konceptuálním a databázovým pohledem na data. Lze určit několik druhů datových modelů, ale přesto jsou v nich u jednotlivých databází velké odlišnosti. Společné rysy NoSQL databází jsou sepsány níže a další vlastnosti jejich datových modelů budou vysvětleny v dalších kapitolách u jejich představitelů.

Druhy datových modelů

- Nejvíce NoSQL databází ukládá data jako kombinaci dvojic klíč/hodnota.
- Klíč se shoduje se jménem atributu v relačních databázích a také se jménem sloupce v jazyku SQL.
- Zkráceně lze použít výrazy atribut a sloupec. Z toho vyplývá označení sloupcové NoSQL databáze.
- Další NoSQL databáze jsou tvořeny kolekcemi dvojic klíč/hodnota - obecněji je možno hovořit o semistrukturovaných dokumentech vybavených indexy.
- Občas bývají označeny (ne zcela správně) jako dokumentově orientované NoSQL databáze. Mnohdy se k zápisu jejich datové struktury používá jazyk JSON (JavaScript Object Notation). JSON je odlehčený formát pro výměnu dat.

NoSQL databáze - úložiště

- Některé NoSQL databáze jsou nazývány úložiště typu klíč/hodnota. Jejich model je jednodušší, protože obsahují jen množinu dvojic klíč/hodnota, jde o množinu pojmenovaných hodnot.
- Klíč s hodnotou je ve vztahu, kde klíč je jednoznačným identifikátorem hodnoty (může být charakterizována řetězcem nebo odkazem na tento řetězec).
- Ukládány jsou zde dva typy hodnot:
  - strukturované
  - nestrukturované (typicky BLOB).

NoSQL databáze - úložiště

- Princip klíč/hodnota bývá přirovnáván k jednoduchým abstrakcím, jako např. souborové systémy a distribuované hašovací tabulky (DHT), s výhodou rychlého vyhledávání.
- Dvojice klíč/hodnota mohou být různých typů. Naproti tomu z pohledu relačního modelu dat by nemusely pocházet ze stejné tabulky.
- NoSQL databáze mají výhodu:
  - v rychlosti,
  - škálovatelnosti
  - nepotřebují hodnoty NULL (neřídí se dle schématu).
- Za nevýhodu je považován až příliš jednoduchý datový model.

NoSQL– grafové databáze

- Dalším datovým modelem NoSQL databází je shledávána kategorie s označením grafové databáze.
- Pracuje na obdobném principu jako síťové databázové systémy.
- Pouze jejich uzly a hrany představují data strukturovaná jako množiny dvojic klíč/hodnota.

Dotazování

- Nejméně propracovanou komponentou v NoSQL databázích je dotazování.
- Nad některými NoSQL databázemi se lze dotazovat jazykem SQL, avšak pouze jeho omezenou formou. Ta nenabízí některé jeho vlastnosti, kterými jsou operace spojení, agregace2 a zanořování poddotazů. Příklady dotazovacích jazyků :
  - GQL (Google Query Language)
  - HQL (Hypertext Query Language)
  - GQL je používán společnosti Google, konkrétně v produktu s názvem AppEngine3 (AppEngine je cloud řešení pro vytváření a správu aplikačních programů na Google platformě).
  - Jazyk HQL obsahující i aktualizační a jiné příkazy se využívá v NoSQL databázi Hypertable.
- Oba jazyky jsou podmnožinou jazyka SQL.

Procedurální přístup

- Další možnost dotazování je pomocí procedurálního přístupu.
- *Klasické API4 pro NoSQL databáze většinou nabízí operace jako:*
  - *get(klíč), tj. extrakce hodnoty daného klíče,*
  - *put(klíč, hodnota) - (vytvoření nebo aktualizace hodnoty daného klíčem),*
  - *delete(klíč) - (odstranění klíče a jeho hodnoty),*
  - *execute(klíč, operace, parametry) - (vyvolá operaci na hodnotě dané klíčem, která je speciální datovou strukturou, např. seznam, množina apod.).*

JOIN, ORDER

- Operace JOIN (spojení) a ORDER BY nejsou v NoSQL podporovány z důvodu horizontálního škálování dat.
  - Stejný argument platí i v situaci, kde se využívá plně funkčního SŘBD na každém uzlu.
  - Možným řešením problému s operací spojení může být její implementace na straně klienta.
  - Další potřebné funkce týkající se dotazování jsou ponechány na klientovi.
  - Tento přístup často znamená ruční programování dotazů, což je ideální pro lehké úlohy a naproti tomu velice pracné pro komplikovanější úlohy.
  - Příkladem tohoto rozšíření může být přidání vyhledávání podle klíčových slov nebo využití relační databáze k ukládání metadat o objektech v NoSQL databázích.

Standardizace

- Z důvodu rozdílnosti jednotlivých NoSQL databází je signifikatní obtížná standardizace (unifikovaný dotazovací standard).

Transakční zpracování

- Vlastnosti transakčního zpracování:
  - atomicita (atomicity – A) – vykoná se buď celá transakce, nebo nic (transakce je ne-rozdělitelná),
  - konzistence (consistency – C) – výsledkem transakce jsou správná data,
  - izolace (isolation – I) – transakce jsou na sobě vzájemně nezávislé,
  - trvanlivost (durability – D) – data potvrzené transakce jsou trvale uložena.

ACID

- Systém splňující plně všechny čtyři vlastnosti ACID se nazývá „silně konzistentní“.
- „Silně konzistentní“ databázové systémy jsou potřeba pouze v některých případech (např. při použití v bankách, obchodu).
- Ve většině případů použití databáze je zapotřebí ACID transakcí pouze v určitých situacích.

ACID

- Označením „případně konzistentní“ se mohou nazývat databáze, které nezcela podporují ACID vlastnosti.
- Při úmyslném zanedbání „silné konzistence“ se naskýtá možnost získání více dostupnosti, a tím pádem i výhodu v lepší škálovatelnosti.
- Přístup bez „silné konzistence“ vyhovuje NoSQL databázím, které ho mnohdy využívají.
  - Poznámka: Podobný princip se již objevil u datových úložišť z 90. let, kde docházelo k občasným konfliktům při aktualizacích dat a hlavní prioritou zde nebyla konzistence.

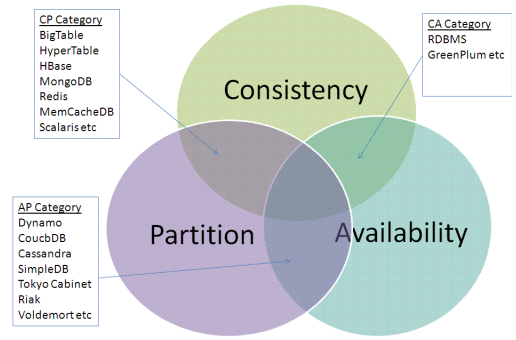
CAP teorém

- Problém je možno řešit i jiným způsobem než na základě vlastností ACID.
- CAP teorém obsahující tři požadavky:
  - *konzistenci (consistency – C),*
  - *dostupnost (availability – A),*
  - *toleranci k rozdělení (partitioning tolerance – P).*
- CAP teorém nazývaný též podle jeho tvůrce jako Brewerův teorém říká, že neexistuje distribuovaný systém, který by splňoval všechny tři požadavky plně a najednou.

CAP

- *„Jednotlivé požadavky lze vysvětlit podle Brewera a zároveň z hlediska NoSQL databází:*
  - *Consistency znamená, že v určitém čase všechny uzly distribuovaného systému vidí stejná data.*
  - *Availability znamená, že každý klient po svém dotazu dostane informaci o tom, či operace byla úspěšná, anebo nebyla (někdy se používá i vysvětlení, že každý klient vždy může číst i zapisovat data).*
  - *Partition tolerance je vlastnost, která hovoří o tom, že když část sítě vypadne (přeruší se spojení, anebo dojde ke ztrátě přenosu mezi uzly na minimálně 2 disjunktní množiny uzlů) systém bude stále schopný odpovídat na dotazy.“*

Znázornění CAP Brewerova teorému a zařazení jednotlivých databázových systémů



Návrh DDBS a CAP Brewerův teorém

- Ve skutečnosti lze vytvořit návrh distribuovaného systému, který splňuje dvě z vlastností CAP zároveň na 100% a pouze částečně zbývající vlastnost.

Návrh DDBS a CAP Brewerův teorém

Existují dvě možnosti řešení:

1. Je kladen výrazný důraz na silnou konzistenci a zkouší se maximalizovat dostupnost.
2. Hlavní priorita je v dostupnosti a vytváří se snaha o co nejsilnější konzistenci.

Ad 1. Výhodou této možnosti je snadnější tvorba aplikací, díky ACID vlastnostem, a řízení datových služeb.

Ad 2.Tato možnost řeší ekonomickou stránku, protože jakákoliv nedostupnost databáze (služby) by mohla zapříčinit vznik finančních ztrát. Naproti tomu obsahuje nevýhodu v nutnosti implementace komplikované aplikační logiky do vytvářených aplikací zkvalitňující konzistenci.

Základní charakteristika - BASE

- U databáze bez silné konzistence platí, že když některý uživatel data zapisuje, tak jiný uživatel, který je čte, nemusí nová data vidět vždycky správně. To je způsobeno replikací (překopírováním) jakékoliv změny do celé databáze až s časovou prodlevou.
- Tento transakční model zahrnuje vlastnosti BASE (Basically Available, Soft-state, Eventually Consistent). Systém s BASE díky umožněním částečných chyb zlepšuje dostupnost. Propouští pouze takové chyby, které nezpůsobí chybu celého systému.

- Pokud relační databáze vždy dodržují požadavky ACID, jsou silně konzistentní.
- Někdy je však uvolňován požadavek izolace a jsou pak případně konzistentní.
- U NoSQL databází existují dvě výše zmíněné možnosti návrhu, (plná tolerance k rozdělení (P)).
- Vysvětlení: Důvodem je časté využití NoSQL databází v podnikatelské oblasti s intenzivním využitím sítě internet pro velký počet zákazníků. Zákazníci jsou pro firmy využívající NoSQL databázi klíčoví a proto musí být zajištěna přístupnost, např. webové stránky nebo její některé části či funkčnosti. Ze stejného důvodu se zajišťuje nízká latence, pomocí dostupnosti, a vysoká průchodnost, což ve výsledku dělá databázi (službu) více responzivní pro uživatele.

Škálovatelnost

- Relační databáze ve většině případů jsou umístěny na jednom serveru.
- Možné škálování relačních databází se realizuje přidáním více procesorů nebo zvyšováním vnitřní či vnější paměti.
- Tento způsob zlepšování jednoho velkého serveru se nazývá vertikální škálování (scale-up).
- Při instalaci relačních databází na vícenásobný server se k synchronizaci využívá replikace.
- U relačních databáze, které disponují velkou vyjadřovací silou (Oracle),nastává problém - je ji obtížné navyšovat prostřednictvím využití více počítačů místo jednoho databázového serveru.

Řešení

- Vytvoření nové skupiny lépe škálovatelných SŘBD - NoSQL databáze.
- Tato skupina SŘBD je díky rozdělení dat škálovatelná téměř lineárně při zapojování vyššího počtu použitých serverů.
- Ve většině případů je využit princip distribuovaných hašovacích tabulek (DHT - Distributed Hash Table).
- Podstata: hašují se dvojice (klíč/hodnota) do kapes (buckets), což jsou dílčí paměťové bloky, které jsou uloženy na odlišných uzlech.

Výhody a nevýhody

- Horizontální uložení dat přináší výhodu rozdělení výpočtů na paralelní úlohy mezi servery.
- Paralelní úlohy se však stávají obtížně zpracovatelnými pro algoritmus či programovací jazyk.
- Některé specializované programovací jazyky řeší a do jisté míry snižují složitost úloh.
- Příkladem může být programovací jazyk vytvořený společností Google s názvem MapReduce
- Nevýhodou jeho použití se jeví velmi obtížná implementace relační operace spojení (JOIN) – musí být zavedena na straně klienta.

Přínos technologie

- Dlouhou dobu byla omezením technická stránka paměťových disků.
- Řešení se objevilo s příchodem technologie disků SSD (solid-state drive) – maximální zvýšení rychlosti čtení i zápisu.
- Zároveň zlepšují architekturu sdílení disků, což může ve výsledku způsobit omezování potřeby rozdělování dat.

Vertikální fragmentace

- Data v NoSQL databázích mohou být rozdělena i vertikálně, což spočívá například v rozdělení záznamu na části, přičemž je každá z nich uložena na jiný uzel – to napomáhá horizontálnímu škálování.
- Naproti tomu horizontální škálování zároveň komplikuje dosažení ACID vlastností.

Hlavní představitelé

- Do skupiny NoSQL databází patří velký počet databázových systémů
- Seznam nejvýznamnějších představitelů NoSQL databází na následujícím obrázku ve formě tabulky obsahující u jednotlivých databází:
  - status, jestli se jedná o projekt, produkt či službu, jakou společností byla databáze vytvořena,
  - kdo je jejím prodejcem (není- li uveden, tak se jedná buď o interní projekt (neprodejný) nebo open-source projekt),
  - u jednotlivých zástupců je uveden i typ, který je určen datovým modelem SŘBD daného databázového systému,
  - upřednostňované vlastnosti CAP teorému,
  - a společnosti, kterého používají.

Database	Status	Created by	Vendor	Type	Consistency/Availat tolerance	Users
HBase	Project	Yahoo	No	Column store	CP	Yahoo, Adobe, Trend Micro, Veoh Networks, Stumpleupon
Cassandra	Project	Facebook	No	Column store	AP	Facebook, Digg, Twitter, Reddit, Rackspace, Cisco, Openx
Hypertable	Project	Zvents	Hypertable Inc	Column store	CP	Zvents, Baidu, Rediff, Inepex, Endgame Systems
BigTable	Internal project	Google	No	Column store	CP	Google
CouchDB	Product	Couchio	Couchio	Document store	AP	BBC, Assay Depot, Engine Yard, Sauce Labs, Flintbox
CouchDB	Service	Couchio	Cloudant	Document store	AP	Scoopler, Cellecta, Meteor
Riak	Product	Basho	Basho	Document store	AP	Comcast, Mochi, Cedexis, Collecta, TeleSkele
MongoDB	Product	10gen	10gen	Document store	CP	EA, New York Times, Disqus, Github, Sourceforge
Dynomite	Project	Cliff Moon	No	Graph	AP	?????
Neo4J	Product	Neo Technologies	Neo Technologies	Graph	??	Swami, The Swedish Defence Forces, Windh Technologies, Flextoll
InfoGrid	Product	NetMesh	NetMesh	Graph	??	?????
HypergraphDB	Project	Kobrix	Kobrix	Graph	??	?????
Dynamo	Internal project	Amazon	No	Key value store	AP	Amazon
SimpleDB	Service	Amazon	Amazon	Key value store	AP	Alexa, Livemocha, AdaptiveBlue, 37 Signals, Sonian
Redis	Project	Salvatore Sanfilippo	VMware	Key value store	CP	Engine Yard, Github, Craigslist, The Guardian
Tokyo Cabinet	Project	Mikio Hirabayashi	No	Key value store	AP	?????
MemcacheDB	Project	Steve Chu		Key value store	CP	?????
Membase	Product	NorthScale	NorthScale	Key value store	CP	Zynga, Heroku, NHN
Voldemort	Project	Linkedin	No	Key value store	AP	Linkedin

Rozdělení (užší pojetí)

- Existují různá dělení NoSQL databází.
- Nejčastějším rozdělením bývá do skupin podle modelu dat:
  - sloupcové databáze,
  - semistrukturované dokumenty,
  - databáze typu klíč/hodnota,
  - grafové databáze,
  - více-modelové databáze (z angl. Multimodel Databases).

Rozdělení (širší pojetí)

- Skupiny NoSQL databází v širším pojetí:
  - objektové databáze,
  - XML databáze,
  - multidimenzionální databáze,
  - více-hodnotové databáze,
  - grid a cloud řešení.

Apache Cassandra

- Apache Cassandra je distribuovaný databázový systém určený k správě velmi velkého množství strukturovaných dat rozmístěných na více uzlech.
- Cassandra je podle svého datového modelu, z kterého přebírá základ od Google BigTable, řazena ke sloupcovým databázím.

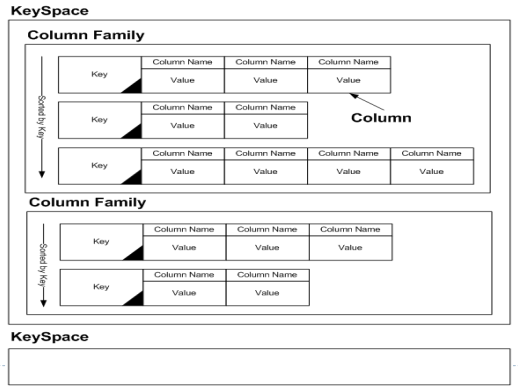
Apache Cassandra

- Apache Cassandra převzala od systému Dynamo způsob řízení clusteru (množiny pro-pojených serverových uzlů), replikaci dat a mechanismy zajišťující odolnost vůči chybám.
- Apache Cassandra je uzpůsobena pro používání v clusteru tvořeného obyčejnými počítači, jejichž počet bývá řádově ve stovkách.
- Cassandra vznikla ve společnosti Facebook, převážně pro služby vyžadující nízkou latenci (odezvu), např. pro službu Inbox Search (Inbox Search umožňuje vyhledávat v historii zpráv posílaných na sociální síti Facebook podle jména autora zprávy nebo klíčových slov nacházejících se ve zprávě).
- Cassandra byla vydána jako open-source projekt roku 2008 společností Facebook (používána dodnes). Mezi společnosti využívající databázový systém Cassandra patří i společnosti Twitter, Cisco a další velké firmy.

Datový model

- Uložená data si lze představit jako pětici množin, které určují přesnou pozici.
- Jednotlivé úrovně (podle pětice) se nazývají:
  - keyspace,
  - column family,
  - row (řádek),
  - column (sloupec),
  - value (hodnota).

Datový model



Datový model

- Keyspace může obsahovat více column family,
- column family může obsahovat více řádků.
- Pětice určuje pozici hodnoty a sloupec obsahuje jednu hodnotu.
- Různé řádky nemohou obsahovat stejný sloupec, což ukazuje velkou odlišnost od tradičních databází.
- Každému sloupci se přiřazuje časové razítko (timestamp), které odpovídá době vzniku záznamu.
- Existuje možnost vytvoření časového razítka zániku záznamu TTL (time to life) (po vypršení stanovené doby se záznam smaže).
- Existuje možnost sdružení sloupců do „rodičovského sloupce“ - aplikace nastavením column family přidáním další dimenze supersloupce.

Konzistence dat

- Vycházíme-li z vlastností CAP teorému:
  - Cassandra je zastáncem AP systému (zabezpečuje eventuální konzistenci).
  - Při replikaci je část aktuálních dat uložena na uzlu a na jiných uzlech jsou starší data, přesto je zabezpečeno, že všechny uzly uvidí aktuální data.
  - V Cassandře lze nastavovat poměr mezi konzistencí dat a latencí, což znamená zhoršení latence při zvýšení konzistence a naopak.