No SQL DB

Pojem NoSQL databáze

Termín NoSQL databáze byl zvolen pro volně specifikovanou třídu nerelačních datových úložišť. …

- Někdy se pro tato datová úložiště používá i termín postrelační.

ACID
Atomicity, Consistency, Isolation, Durability

Koncept ACID - základ transakčního zpracování relačních databází

Atomicity - znamená, že série modifikací databáze uskutečňující se během transakce je provedena celá nebo je databáze ponechána v původním stavu, jako začátkem transakce. Systém musí garantovat toto chováni v každé situaci včetně nečekaných poruch.

Consistency – zaručuje, že každá transakce převede databázi z jednoho konzistentního stavu do druhého. Do databáze budou zapsána pouze data odpovídající integritním omezením.

Isolation – data měněná transakcí jsou ostatním uživatelům až do úspěšného ukončení transakce viditelná v původní podobě, jako byla před začátkem transakce.

Durability – změny v databázi vyvolané úspěšně ukončenou transakcí jsou trvale uloženy na persistentním úložišti.

CAP teorém

- CAP teorém byl prezentován Eric A. Brewerem v roce 2000

- Dokázán S. Gilbertem a N. Lynch v „Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services" .

- Přijat v NoSQL komunitě - ovlivňuje návrh nerelačních databází.

CAP teorém

- **CAP teorém**: pro distribuovaný systém (konkrétně pro webové služby) je nemožné současně garantovat následující vlastnosti:

- Konzistence (**C**onsistency) – distribuovaný systém je považován za konzistentní v případě, že všechny operace čtení vrací verzi dat vloženou poslední operací zápisu do sdíleného úložiště.

- Dostupnost (**A**vailability) – systém by měl fungovat i když některé jeho servery havarují nebo jsou nedostupné kvůli poruchám sítě.

- Tolerance k rozdělení (**P**artition-tolerance) – systém je navržen tak, aby mohl pokračovat v práci i v případě, že se kvůli přerušení sítě rozdělí (dočasně nebo trvale) na několik samostatných částí.

CAP teorém

- Všechny tři vlastnosti jsou žádoucí, ale

pro jakýkoliv systém sdílení dat je možné dosáhnout maximálně dvou současně.

- Zvláště u webových aplikací založených

na horizontálním škálování je nutné se rozhodnout mezi konzistencí a dostupností.

CAP teorém

Rozdělení databází dle preference vlastností – I.

CA - Konzistence (C), Dostupnost (A)

- RDBMSs - Postgres, MySQL, apod. (relational)
- Vertica (column-oriented)
- Aster Data (relational)
- Greenplum (relational)

Rozdělení databází dle preference vlastností – II.

CP - Konzistence (C), Tolerance k rozdělení (P)

- BigTable (column-oriented/tabular)
- Hypertable (column-oriented/tabular)
- HBase (column-oriented/tabular)
- MongoDB (document-oriented)
- Terrastore (document-oriented)
- Redis (key-value)
- MemcacheDB (key-value)
- Berkeley DB (key-value)

Rozdělení databází dle preference vlastností – III.

AP - Dostupnost (A), Tolerance k rozdělení (P)

- Dynamo (key-value)
- Voldemort (key-value)
- Tokyo Cabinet (key-value)
- KAI (key-value)
- Cassandra (column-oriented/tabular)
- CouchDB (document-oriented)
- SimpleDB (document-oriented)
- Riak (document-oriented)

ACID a BASE

- Internetové aplikace, jako jsou sociální sítě, blogy, znalostní databáze a další, vytvářejí obrovské množství dat, která musí být dále zpracovávána.
- Společnosti provozující takové aplikace si musí stanovit preference týkající se výkonnosti, spolehlivosti, dostupnosti a konzistence. Jak již bylo zmíněno výše, CAP teorém říká, že ze tří žádoucích vlastností
- (konzistence, dostupnost, tolerance k rozdělení) je možné dosáhnout pouze dvou současně.
- Pro narůstající počet aplikací je dostupnost a tolerance k rozdělení důležitější než konzistence.

ACID a BASE

- Požadavek na internetové aplikace - musí být především výkonné a spolehlivé.
- Vysokého výkonu lze dosáhnout horizontálním škálováním.
- Spolehlivosti vyšší lze dosáhnout vyšší redundancí.
- Velmi obtížné při plném dodržení vlastností ACID.
- Proto je aplikován přístup BASE.

„Každý uzel v systému by měl být schopen rozhodování čistě na základě svého lokálního

stavu. Jakmile chcete něco udělat při velké zátěži, při výskytu chyb, a musíte čekat na

potvrzení změn, jste ztracení. Pokud se soustředíte na škálovatelnost, bude jakýkoli postup,

který vyžaduje potvrzování, vaším slabým místem. Berte to jako fakt.“

Werner Vogels, Amazon CTO and Vice President

BASE

Přístup označovaný jako BASE preferuje dostupnost, částečnou degradaci a výkon před konzistencí a izolací.

Zkratka BASE je složena z následujících termínů:

Basicaly **A**vailable – aplikace pracuje bez přerušení

Soft-state – aplikace nemusí být v každém okamžiku konzistentní

Eventual consistency – aplikace bude v blíže neurčené době opět konzistentní

Porovnání ACID a BASE

Zdroj:http://www.cs.berkeley.edu/

ACID

- Strong consistency
- Isolation
- Focus on „commit“
- Nested transactions
- Availability
- Conservative (pesimistic)
- Difficult evolution (e.g. schema)

BASE

- Weak consistency – stale data OK
- Availability first
- Best effort
- Approximate answers OK
- Aggressive (optimistic)
- Simpler !
- Faster
- Easier evolution

Konzistence

- **Strong Consistency** – všechny operace čtení musí poskytovat data z poslední operace zápisu nezávisle na replice na které se operace čtení odehrává.
- **Eventual Consistency** – operace čtení může poskytovat nekonzistentní data v průběhu synchronizace mezi replikami.

Konzistence databáze BASE

- BASE databáze je bez silné konzistence.
- Při aktualizaci není zaručeno, že každý, kdo čte z databáze, dostane aktualizovaná data.
- Vysoká dostupnost v BASE je dosahována povolením dílčích chyb tak, aby nedošlo k poruše celého systému.
- Ošetření případných chyb vzniklých nekonzistencí není prováděno na datové, ale na aplikační vrstvě.

Použití

- V praxi se ukazuje, že ACID transakce jsou nezbytné jen v některých případech.
- Dokonce jen v některých případech užití v rámci jedné aplikace.
- V internetových aplikacích se používají oba přístupy – ACID i BASE.
- V případech užití, kdy je nezbytná silná konzistence, jako jsou bankovní operace, se používá přístup ACID.
- V případech užití, kdy je upřednostňována dostupnost, se používá přístup BASE.

SQL a NoSQL databáze

- Pod pojmem SQL databáze –především relační a objektově-relační databáze používající jazyk SQL.
- Dlouhý vývoj - na velmi vysoké úrovni poskytovaných služeb i spolehlivosti.
- Transakční zpracování - silná konzistence dat.
- Je-li silná konzistence podmínkou zpracování – SQL databáze je správná volba, není důvod hledat jiné řešení.

Problém zpracování velkého objemu dat

- V případě velmi velkého množství dat nebude pravděpodobně možné dosáhnout potřebné kapacity a výkonu škálováním vertikálním - bude nutné použít škálování horizontální.
- Při horizontálním škálování je udržování silné konzistence databáze se vzrůstající zátěží stále náročnější a vede ke snižování výkonu i dostupnosti.
- Pro řadu aplikací nepřípustné:
  - Amazon uvádí, že prodloužení odezvy o 0,1s znamená snížení prodeje o 1%.
  - Google uvádí, že prodloužení odezvy o 0,5s znamená propad v provozu až 20% a s tím spojené finanční ztráty. Pro mnoho aplikací je přijatelnější nemít vždy databázi zcela konzistentní, ale zato neustále dostupnou s vysokou rychlostí odezvy uživateli.

Řešení

- NoSQL databáze byly vyvinuty podle požadavků odlišných od požadavků na relační (SQL) databáze v době jejich vzniku.
- Jejich použití je také jiné a nemá smysl dělat přímá srovnání.
- Oba přístupy jsou opodstatněné v určitých případech použití .
- Mnoho společností proto používá současně databáze relační (SQL) a NoSQL.

KATEGORIE NOSQL DATABÁZÍ

Možnosti katogorizace

- Velmi široká škála produktů, jsou často od sebe navzájem odvozovány
- Řada společných vlastností
- Rozdělení na základě datového modelu

Klíč-hodnota (Key-value)

- Inspirováno Amazons Dynamo
- Datový model: kolekce klíč-hodnota (key-value)
- Příklad:
  - Dynomite,
  - Voldemort,
  - Tokyo,
  - ……

Sloupcové databáze (Column-oriented)

- Inspirováno Google Bigtable
- Datový model: rodiny sloupců (Column family)
- Příklad:
  - Hbase,
  - Hypertable
  - ……..

Dokumentové databáze (Document databases)

- Inspirováno Lotus Notes
- Datový model: kolekce semistrukturovaných dokumentů
- Příklad:
  - MongoDB,
  - CouchDB,
  - …….

Grafové databáze (Graph databases)

- Inspirováno teorií grafů
- Datový model: vrcholy, hrany, key-value u obou
- Příklad:
  - Neo4j,
  - VertexDB,
  - DEX

ARCHITEKTURA NOSQL DATABÁZÍ

Existující projekty a řešení

- Jednotlivé implementace NoSQL databázových systémů se od sebe hodně liší,
- lze však vysledovat některá opakující se architektonická řešení,
- u jednotlivých implementací se potom vyskytují určité rozdíly.

Partitioning

1. Virtuální nódy

- Infrastruktura, na které jsou provozovány NoSQL databázové systémy, jako je například Bigtable, MongoDB nebo Cassandra, je tvořena velkým počtem serverů, které se od sebe liší svojí konfigurací.
- Z důvodu zohlednění a využití rozdílného výkon serverů se používají v konfiguracích databázových systémů takzvané virtuální nódy.
- Na jednom fyzickém serveru může běžet několik virtuálních nodů, v závislosti na jeho konfiguraci.
- Virtuální nódy používá MongoDB.

Partitioning

2. Dělení (Sharding)

- Sharding je metoda, která rozděluje velké soubory dat rozdělit na několik serverů.
- Metodu používá například MongoDB nebo Redis.
- Shard v pojetí MongoDB je skupina serverů, které udržují identické kopie určené části dat.

Partitioning

3. Hashing

- Nejjednodušší způsob rozdělení dat spočívá v rozdělení rozsahu primárního klíče
- pomocí hash funkce na stejné části, kdy každá část leží na jednom serveru. Klient pomocí hash funkce určí, na kterém serveru leží požadovaná data.
- Tento přístup využívá například systém Memcached.
- Nevýhoda: V případě výpadku jednoho ze serverů nebo při přidání dalšího serveru, musí být rozdělení dat provedeno znovu a data podle nového rozdělení opět rozdistribuována.

Poznámka: Toto nevadí u systému Memcached, který má relativně malý objem dat uložen pouze v operační paměti a redistribuce je snadno provedena přirozeným během databáze;

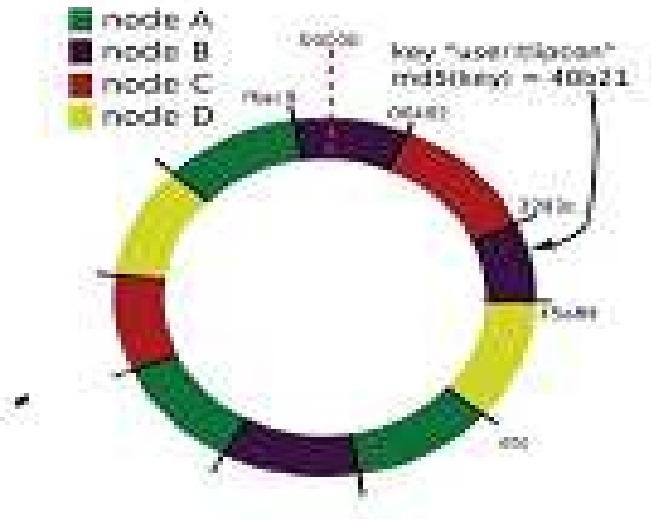
u databází, které mají data uložena na lokálním disku, by to znamenalo kopírování velkých objemů dat mezi servery.

Partitioning

4. Consistent Hashing

- Některé NoSQL databázové systémy, jako Cassandra nebo MongoDB, používají Consistent hashing.
- V systému Consistent hashing leží celý rozsah primárního klíče na pomyslném kruhu. Každá hodnota primárního klíče má svého správce, kterým je první nod na pomyslné kružnici ve směru hodinových ručiček.
- Výhodou tohoto řešení je, že přidání, odebrání nebo výpadek nodu se dotkne pouze nodů s ním sousedících.
- Redistribuce dat je nutná pouze mezi těmito sousedícími nody a ostatních nodů se nijak nedotkne.

Partitioning



Partitioning

5. Replikace

- K zajištění odolnosti proti výpadkům serverů (při vysokých počtech v infrastrukturách horizontálně škálovaných databází velmi časté) a k zajištění vysoké dostupnosti jsou data udržována ve více kopiích pomocí replikace.
- Replikace současně umožňuje rozkládání zátěže.