Konzistence databáze v nekonzistentním světě

Radim Bača

Katedra informatiky
Fakulta elektrotechniky a informatiky
VŠB – Technická univerzita Ostrava

Obsah

Vysvětlíme si, co je

- transakce v databázích,
- konzistence,
- eventuální konzistence.

Transakce v databázích

- **Transakce** představuje jednotku práce nad databází, kterou je potřeba vykonat buďto celou anebo vůbec.
- Transakce se obvykle skládá z více operací.
- Příklad: převod peněz mezi dvěmi bankovními učty.

  Taková transakce se skládá ze dvou operací:

  1. odečtení požadované částky z jednoho účtu,
  2. přičtení této částky na druhý účet.

Transakce v databázích

- **Transakce** představuje jednotku práce nad databází, kterou je potřeba vykonat buďto celou anebo vůbec.
- Transakce se obvykle skládá z více operací.
- Příklad: převod peněz mezi dvěmi bankovními účty.

  Taková transakce se skládá ze dvou operací:

  1. odečtení požadované částky z jednoho účtu,
  2. přičtení této částky na druhý účet.

Transakce v databázích

- **Transakce** představuje jednotku práce nad databází, kterou je potřeba vykonat buďto celou anebo vůbec.
- Transakce se obvykle skládá z více operací.
- Příklad: převod peněz mezi dvěmi bankovními učty.

  Taková transakce se skládá ze dvou operací:

  1. odečtení požadované částky z jednoho účtu,
  2. přičtení této částky na druhý účet.

Transakce v databázích

- **Transakce** představuje jednotku práce nad databází, kterou je potřeba vykonat buďto celou anebo vůbec.
- Transakce se obvykle skládá z více operací.
- Příklad: převod peněz mezi dvěmi bankovními učty.

  Taková transakce se skládá ze dvou operací:

  1. odečtení požadované částky z jednoho účtu,
  2. přičtení této částky na druhý účet.

Transakce v databázích

- **Transakce** představuje jednotku práce nad databází, kterou je potřeba vykonat buďto celou anebo vůbec.
- Transakce se obvykle skládá z více operací.
- Příklad: převod peněz mezi dvěmi bankovními učty.

  Taková transakce se skládá ze dvou operací:

  1. odečtení požadované částky z jednoho účtu,
  2. přičtení této částky na druhý účet.

Výhody použití transakcí

- Hlavní výhody použití transakcí:
  - korektní souběžný přístup při vykonávání operací,
  - zotavení databáze při nějaké chybě.

ACID

`Atomicity -- Consistency -- Isolation -- Durability`

- Atomicita - jsou provedeny všechny operace transakce nebo vůbec žádná
- Konzistence - databáze je převedena z jednoho konzistentního stavu do druhého
- Izolace - transakce se během souběžného provádění nemohou nijak ovlivňovat
- Trvalost - jakmile je transakce korektně ukončena, pak změny v databázi zůstávají nehledě na chyby při běhu

ACID

`Atomicity -- Consistency -- Isolation -- Durability`

- Atomicita - jsou provedeny všechny operace transakce nebo vůbec žádná
- Konzistence - databáze je převedena z jednoho konzistentního stavu do druhého
- Izolace - transakce se během souběžného provádění nemohou nijak ovlivňovat
- Trvalost - jakmile je transakce korektně ukončena, pak změny v databázi zůstávají nehledě na chyby při běhu

ACID

`Atomicity -- Consistency -- Isolation -- Durability`

- Atomicita - jsou provedeny všechny operace transakce nebo vůbec žádná
- Konzistence - databáze je převedena z jednoho konzistentního stavu do druhého
- Izolace - transakce se během souběžného provádění nemohou nijak ovlivňovat
- Trvalost - jakmile je transakce korektně ukončena, pak změny v databázi zůstávají nehledě na chyby při běhu

ACID

`Atomicity -- Consistency -- Isolation -- Durability`

- Atomicita - jsou provedeny všechny operace transakce nebo vůbec žádná
- Konzistence - databáze je převedena z jednoho konzistentního stavu do druhého
- Izolace - transakce se během souběžného provádění nemohou nijak ovlivňovat
- Trvalost - jakmile je transakce korektně ukončena, pak změny v databázi zůstávají nehledě na chyby při běhu

ACID

`Atomicity -- Consistency -- Isolation -- Durability`

- Atomicita - jsou provedeny všechny operace transakce nebo vůbec žádná
- Konzistence - databáze je převedena z jednoho konzistentního stavu do druhého
- Izolace - transakce se během souběžného provádění nemohou nijak ovlivňovat
- Trvalost - jakmile je transakce korektně ukončena, pak změny v databázi zůstávají nehledě na chyby při běhu

Konzistence

- **Konzistencí** databáze rozumíme to, že v databázi máme korektní data, která odpovídají realitě.
- V našem příkladu s bankovním převodem to znamená, že celková suma peněz na účtech zůstává konstantní.
- Konzistence databáze většinou nemůže být vyjádřena takto jednoduše jednou větou a je obsažena v transakcích a operacích, které provádíme nad databází.

Konzistence

- **Konzistencí** databáze rozumíme to, že v databázi máme korektní data, která odpovídají realitě.
- V našem příkladu s bankovním převodem to znamená, že celková suma peněz na účtech zůstává konstantní.
- Konzistence databáze většinou nemůže být vyjádřena takto jednoduše jednou větou a je obsažena v transakcích a operacích, které provádíme nad databází.

Konzistence

- **Konzistencí** databáze rozumíme to, že v databázi máme korektní data, která odpovídají realitě.
- V našem příkladu s bankovním převodem to znamená, že celková suma peněz na účtech zůstává konstantní.
- Konzistence databáze většinou nemůže být vyjádřena takto jednoduše jednou větou a je obsažena v transakcích a operacích, které provádíme nad databází.

Další příklady konzistence

- Pokud hrajete nějakou on-line strategii, pak nechcete, aby jeden hráč viděl v určitém bodě krásného draka a druhý ošklivou princeznu.
- Počet kusů zboží v e-shopu by měl odrážet skutečný stav na skladu.

Konzistence v distribuovaných databázích (DD)

- Pokud data ukládáme na „jednom“ místě, pak je konzistnce obvykle nejméně zajímavý atribut transakce.
- Zcela jiná situace ale nastává, pokud chceme data rozložit na více databázových serverů (tzv. distribuovaná databáze).
- V distribuované databázi jsou data vždy replikována na více serverech, aby
  - bylo možné obsluhovat více paralelních čtení či dokonce zápisů stejných dat (čtení jednoho článku ze zpravodajského serveru) a
  - v případě výpadku jednoho serveru byl obsah stále dostupný.
- Při změně hodnoty na jednom serveru musíme pro udržení konzistence změnit hodnotu i na všech jeho replikách.

Konzistence v distribuovaných databázích (DD)

- Pokud data ukládáme na „jednom“ místě, pak je konzistnce obvykle nejméně zajímavý atribut transakce.
- Zcela jiná situace ale nastává, pokud chceme data rozložit na více databázových serverů (tzv. distribuovaná databáze).
- V distribuované databázi jsou data vždy replikována na více serverech, aby
  - bylo možné obsluhovat více paralelních čtení či dokonce zápisů stejných dat (čtení jednoho článku ze zpravodajského serveru) a
  - v případě výpadku jednoho serveru byl obsah stále dostupný.
- Při změně hodnoty na jednom serveru musíme pro udržení konzistence změnit hodnotu i na všech jeho replikách.

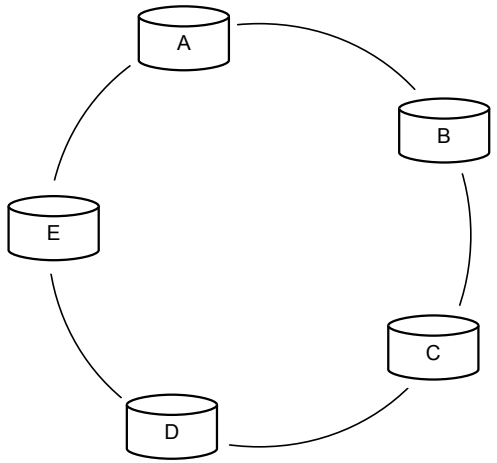
Konzistence v distribuovaných databázích (DD)

- Pokud data ukládáme na „jednom“ místě, pak je konzistnce obvykle nejméně zajímavý atribut transakce.
- Zcela jiná situace ale nastává, pokud chceme data rozložit na více databázových serverů (tzv. distribuovaná databáze).
- V distribuované databázi jsou data vždy replikována na více serverech, aby
  - bylo možné obsluhovat více paralelních čtení či dokonce zápisů stejných dat (čtení jednoho článku ze zpravodajského serveru) a
  - v případě výpadku jednoho serveru byl obsah stále dostupný.
- Při změně hodnoty na jednom serveru musíme pro udržení konzistence změnit hodnotu i na všech jeho replikách.

Konzistence v distribuovaných databázích (DD)

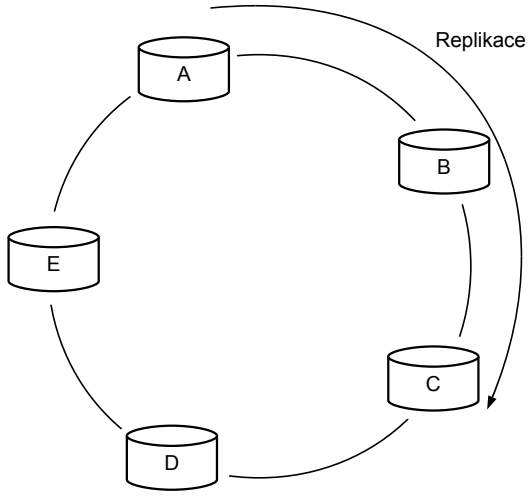
- Pokud data ukládáme na „jednom“ místě, pak je konzistnce obvykle nejméně zajímavý atribut transakce.
- Zcela jiná situace ale nastává, pokud chceme data rozložit na více databázových serverů (tzv. distribuovaná databáze).
- V distribuované databázi jsou data vždy replikována na více serverech, aby
  - bylo možné obsluhovat více paralelních čtení či dokonce zápisů stejných dat (čtení jednoho článku ze zpravodajského serveru) a
  - v případě výpadku jednoho serveru byl obsah stále dostupný.
- Při změně hodnoty na jednom serveru musíme pro udržení konzistence změnit hodnotu i na všech jeho replikách.

Příklad DD



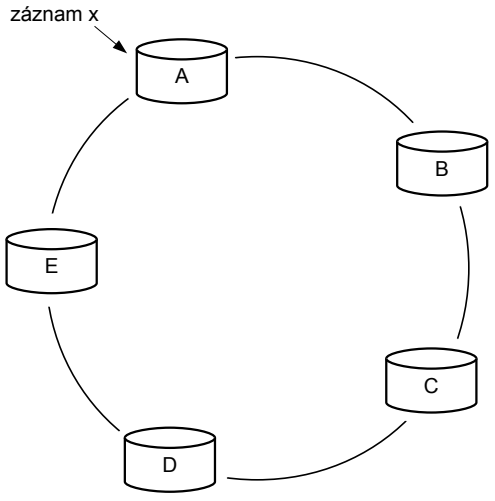
- Mějme distribuovanou databázi s 5 servery.

Příklad DD



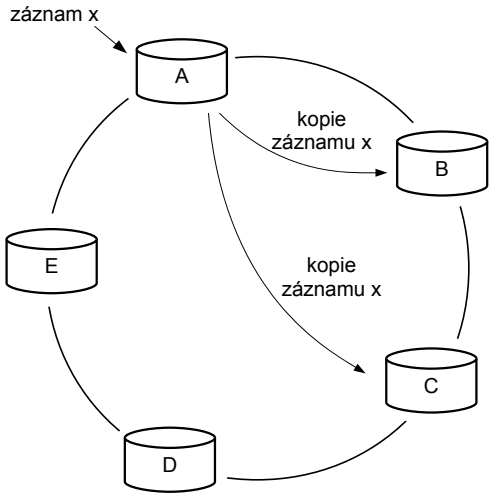
- Mějme distribuovanou databázi s pěti servery.
- Záznam v databázi vždy náleží jednomu serveru, ale zároveň jsou data replikována na 2 serverech po směru hodinových ručiček.

Příklad DD



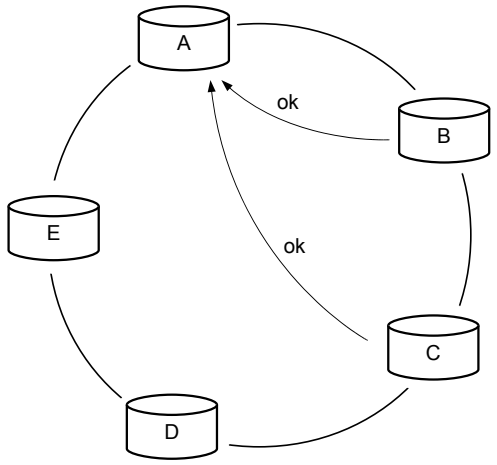
- Řekněme, že chceme zapsat záznam, který je uložen na serveru A.

Příklad DD



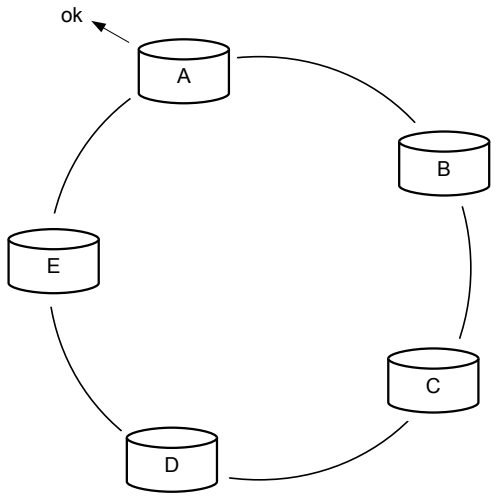
- Řekněme, že chceme zapsat záznam, který je uložen na serveru A.
- Záznam je nejprve poslán na repliky a čeká na potvrzení úspěšného zapsání.

Příklad DD



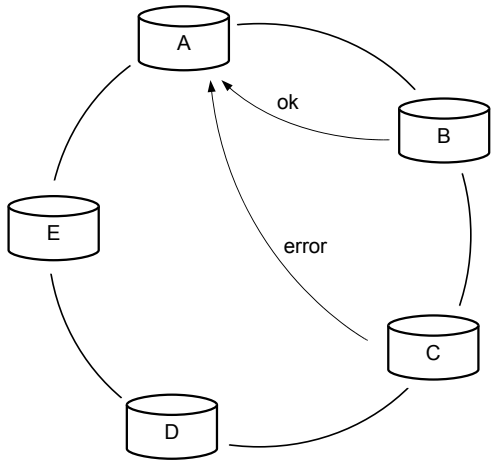
- Když dostaneme potvrzení, že všechny servery korektně zapsaly změny,

Příklad DD



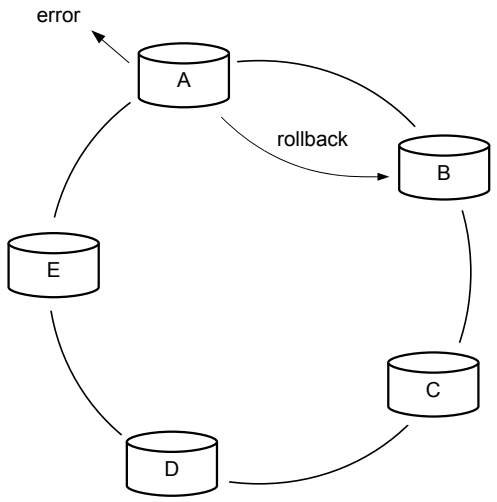
- Když dostaneme potvrzení, že všechny servery korektně zapsaly změny, pak potvrdíme provedení operace.

Příklad DD



- Pokud zápis na některém serveru selže,

Příklad DD



- Pokud zápis na některém serveru selže,

  musíme stornovat operace i na ostatních serverech a odeslat chybu při zápisu.

Problémy konzistence

- Základním problémem je, že komunikace i samotné servery mohou být nespolehlivé.
- Dále při zápisu dále dochází k zamykání záznamů.
- Servery musí znát topologie sítě a stav jednotlivých serverů, aby mohly spolehlivě vyloučit nekonzistenci při zápisu.
- Důsledkem je, že jednoduchá operace aktualizace článku může trvat neúměrně dlouho.
- Naštěstí existují řešení, takže mám o čem mít dnes přednášku.

Problémy konzistence

- Základním problémem je, že komunikace i samotné servery mohou být nespolehlivé.
- Dále při zápisu dále dochází k zamykání záznamů.
- Servery musí znát topologie sítě a stav jednotlivých serverů, aby mohly spolehlivě vyloučit nekonzistenci při zápisu.
- Důsledkem je, že jednoduchá operace aktualizace článku může trvat neúměrně dlouho.
- Naštěstí existují řešení, takže mám o čem mít dnes přednášku.

Problémy konzistence

- Základním problémem je, že komunikace i samotné servery mohou být nespolehlivé.
- Dále při zápisu dále dochází k zamykání záznamů.
- Servery musí znát topologie sítě a stav jednotlivých serverů, aby mohly spolehlivě vyloučit nekonzistenci při zápisu.
- Důsledkem je, že jednoduchá operace aktualizace článku může trvat neúměrně dlouho.
- Naštěstí existují řešení, takže mám o čem mít dnes přednášku.

Problémy konzistence

- Základním problémem je, že komunikace i samotné servery mohou být nespolehlivé.
- Dále při zápisu dále dochází k zamykání záznamů.
- Servery musí znát topologie sítě a stav jednotlivých serverů, aby mohly spolehlivě vyloučit nekonzistenci při zápisu.
- Důsledkem je, že jednoduchá operace aktualizace článku může trvat neúměrně dlouho.
- Naštěstí existují řešení, takže mám o čem mít dnes přednášku.

Problémy konzistence

- Základním problémem je, že komunikace i samotné servery mohou být nespolehlivé.
- Dále při zápisu dále dochází k zamykání záznamů.
- Servery musí znát topologie sítě a stav jednotlivých serverů, aby mohly spolehlivě vyloučit nekonzistenci při zápisu.
- Důsledkem je, že jednoduchá operace aktualizace článku může trvat neúměrně dlouho.
- Naštěstí existují řešení, takže mám o čem mít dnes přednášku.

Eventuální konzistence

- Když se vrátíme k příkladu se zpravodajským serverem: aktualizace článku na jednom serveru se propaguje na všechny repliky.
- Musíme skutečně okamžitě všichni vidět aktuální článek?
- Uživatel těžko postřehne, že má zrovna neaktuální článek, zejména pokud je jen „trochu“ neaktuální.
- Obecně platí, že u mnoha (zejména webových) aplikací na striktní konzistenci často vůbec nezáleží.
- Primární je dostat alespoň nějaká data, která se blíží skutečnosti.

Eventuální konzistence

- Když se vrátíme k příkladu se zpravodajským serverem: aktualizace článku na jednom serveru se propaguje na všechny repliky.
- Musíme skutečně okamžitě všichni vidět aktuální článek?
- Uživatel těžko postřehne, že má zrovna neaktuální článek, zejména pokud je jen „trochu“ neaktuální.
- Obecně platí, že u mnoha (zejména webových) aplikací na striktní konzistenci často vůbec nezáleží.
- Primární je dostat alespoň nějaká data, která se blíží skutečnosti.

Eventuální konzistence

- Když se vrátíme k příkladu se zpravodajským serverem: aktualizace článku na jednom serveru se propaguje na všechny repliky.
- Musíme skutečně okamžitě všichni vidět aktuální článek?
- Uživatel těžko postřehne, že má zrovna neaktuální článek, zejména pokud je jen „trochu“ neaktuální.
- Obecně platí, že u mnoha (zejména webových) aplikací na striktní konzistenci často vůbec nezáleží.
- Primární je dostat alespoň nějaká data, která se blíží skutečnosti.

Eventuální konzistence

- Když se vrátíme k příkladu se zpravodajským serverem: aktualizace článku na jednom serveru se propaguje na všechny repliky.
- Musíme skutečně okamžitě všichni vidět aktuální článek?
- Uživatel těžko postřehne, že má zrovna neaktuální článek, zejména pokud je jen „trochu“ neaktuální.

- Obecně platí, že u mnoha (zejména webových) aplikací na striktní konzistenci často vůbec nezáleží.
- Primární je dostat alespoň nějaká data, která se blíží skutečnosti.

Konzistence

- Striktní konzistence - klasická konzistence tak jak je chápána v ACID. Všechny změny provedené operací jsou uloženy v databázi a každé následné čtení vrací vždy novou hodnotu.
- Slabá konzistence - po provedení operace A existuje určitá prodleva, během které následné čtení může vracet nějakou hodnotu, která byla v poli před provedením změn operací A. Tato prodleva se nazývá *okno nekonzistnce*.
- Eventuální konzistence - jedná se o určitou formu slabé konzistence, kde je velikost okna nekonzistence dána parametry databáze a sítě.

Konzistence

- Striktní konzistence - klasická konzistence tak jak je chápána v ACID. Všechny změny provedené operací jsou uloženy v databázi a každé následné čtení vrací vždy novou hodnotu.
- Slabá konzistence - po provedení operace A existuje určitá prodleva, během které následné čtení může vracet nějakou hodnotu, která byla v poli před provedením změn operací A. Tato prodleva se nazývá *okno nekonzistnce*.
- Eventuální konzistence - jedná se o určitou formu slabé konzistence, kde je velikost okna nekonzistence dána parametry databáze a sítě.

Konzistence

- Striktní konzistence - klasická konzistence tak jak je chápána v ACID. Všechny změny provedené operací jsou uloženy v databázi a každé následné čtení vrací vždy novou hodnotu.
- Slabá konzistence - po provedení operace A existuje určitá prodleva, během které následné čtení může vracet nějakou hodnotu, která byla v poli před provedením změn operací A. Tato prodleva se nazývá *okno nekonzistnce*.
- Eventuální konzistence - jedná se o určitou formu slabé konzistence, kde je velikost okna nekonzistence dána parametry databáze a sítě.

Quorum systems

- Mějme **N** replik.
- Při zápisu rozešleme operaci na všechny repliky a operaci považujeme za úspěšnou, pokud se provede alespoň na **W** serverech.
- Při čtení pak musí odpovědět alespoň **R** serverů.
- Pokud nastavíme $\mathbf{R} + \mathbf{W} > \mathbf{N}$, tak můžeme mít striktní konzistenci operace.

Quorum systems

- Mějme **N** replik.
- Při zápisu rozešleme operaci na všechny repliky a operaci považujeme za úspěšnou, pokud se provede alespoň na **W** serverech.
- Při čtení pak musí odpovědět alespoň **R** serverů.
- Pokud nastavíme $\mathbf{R} + \mathbf{W} > \mathbf{N}$, tak můžeme mít striktní konzistenci operace.

Quorum systems

- Mějme **N** replik.
- Při zápisu rozešleme operaci na všechny repliky a operaci považujeme za úspěšnou, pokud se provede alespoň na **W** serverech.
- Při čtení pak musí odpovědět alespoň **R** serverů.
- Pokud nastavíme $\mathbf{R} + \mathbf{W} > \mathbf{N}$, tak můžeme mít striktní konzistenci operace.

Quorum systems

- Mějme **N** replik.
- Při zápisu rozešleme operaci na všechny repliky a operaci považujeme za úspěšnou, pokud se provede alespoň na **W** serverech.
- Při čtení pak musí odpovědět alespoň **R** serverů.
- Pokud nastavíme $\mathbf{R} + \mathbf{W} > \mathbf{N}$, tak můžeme mít striktní konzistenci operace.

Další používané techniky

- Dvoufázový protokol zamykání - zamykací protokol, který zajistí striktní konzistenci při zápisu.
- Paxos - další protokol, který zajišťuje striktní konzistenci při zápisu. Protokol je flexibilnejší než dvoufázový protokol, jelikož nedochází k explicitnímu zamykání, ale při konfliktech muže dojít k odmítnutí zápisu (i opakovanému).
- Vektorové hodiny - technika pro řešení nekonzistence během čtení dat.

Další používané techniky

- Dvoufázový protokol zamykání - zamykací protokol, který zajistí striktní konzistenci při zápisu.
- Paxos - další protokol, který zajišťuje striktní konzistenci při zápisu. Protokol je flexibilnejší než dvoufázový protokol, jelikož nedochází k explicitnímu zamykání, ale při konfliktech muže dojít k odmítnutí zápisu (i opakovanému).
- Vektorové hodiny - technika pro řešení nekonzistence během čtení dat.

Další používané techniky

- Dvoufázový protokol zamykání - zamykací protokol, který zajistí striktní konzistenci při zápisu.
- Paxos - další protokol, který zajišťuje striktní konzistenci při zápisu. Protokol je flexibilnejší než dvoufázový protokol, jelikož nedochází k explicitnímu zamykání, ale při konfliktech muže dojít k odmítnutí zápisu (i opakovanému).
- Vektorové hodiny - technika pro řešení nekonzistence během čtení dat.

Vektorové hodiny

- Vektorové hodiny jsou v podstatě *n*-tice dvojic (server, počítadlo).
- Každý objekt na serveru má přiřazeny jedny vektorové hodiny.
- Umožňují vyřešit následnost operací a detekovat, že dva objekty jsou v konfliktu.

Vektorové hodiny

- Vektorové hodiny jsou v podstatě n-tice dvojic (server, počítadlo).
- Každý objekt na serveru má přiřazeny jedny vektorové hodiny.
- Umožňují vyřešit následnost operací a detekovat, že dva objekty jsou v konfliktu.

Vektorové hodiny

- Vektorové hodiny jsou v podstatě n-tice dvojic (server, počítadlo).
- Každý objekt na serveru má přiřazeny jedny vektorové hodiny.
- Umožňují vyřešit následnost operací a detekovat, že dva objekty jsou v konfliktu.

Příklad

- U nějakého příspěvku XY máme počítadlo, které udává, kolika lidem se příspěvek líbil.
- Zároveň může kdokoli svůj zájem zrušit stiskem tlačítka 'Dislike'.
- Číslo 4 je uloženo na serveru A a je replikováno na serverech B a C.

Příklad

- U nějakého příspěvku XY máme počítadlo, které udává, kolika lidem se příspěvek líbil.
- Zároveň může kdokoli svůj zájem zrušit stiskem tlačítka 'Dislike'.
- Číslo 4 je uloženo na serveru A a je replikováno na serverech B a C.

Příklad

- U nějakého příspěvku XY máme počítadlo, které udává, kolika lidem se příspěvek líbil.
- Zároveň může kdokoli svůj zájem zrušit stiskem tlačítka 'Dislike'.
- Číslo 4 je uloženo na serveru A a je replikováno na serverech B a C.

Příklad

- Při kliknutí na tlačítko 'Like' se odešle na databázi událost 'inkrementace oblíbenosti XY'.
- Pokud požadavek zpracuje server A:
  - inkrementuje se číslo 4 na 5
  - a vytvoří se k číslu vektorové hodiny (A,1).
- Pokud po dalším kliknutí (jiným uživatelem) požadavek opět zpracuje server A:
  - inkrementuje se číslo 5 na 6
  - a vytvoří se k číslu vektorové hodiny (A,2).

Příklad

- Při kliknutí na tlačítko 'Like' se odešle na databázi událost 'inkrementace oblíbenosti XY'.
- Pokud požadavek zpracuje server A:
  - inkrementuje se číslo 4 na 5
  - a vytvoří se k číslu vektorové hodiny (A,1).
- Pokud po dalším kliknutí (jiným uživatelem) požadavek opět zpracuje server A:
  - inkrementuje se číslo 5 na 6
  - a vytvoří se k číslu vektorové hodiny (A,2).

Příklad

- Při kliknutí na tlačítko 'Like' se odešle na databázi událost 'inkrementace oblíbenosti XY'.
- Pokud požadavek zpracuje server A:
  - inkrementuje se číslo 4 na 5
  - a vytvoří se k číslu vektorové hodiny (A,1).
- Pokud po dalším kliknutí (jiným uživatelem) požadavek opět zpracuje server A:
  - inkrementuje se číslo 5 na 6
  - a vytvoří se k číslu vektorové hodiny (A,2).

Příklad

- Dochází k pravidelnému vyměňování informací mezi servery.
- Hodnota i s vektorovými hodinami se tedy rychle zpropaguje na další servery.
- Poté dojde k výpadku serveru A.

Příklad

- Dochází k pravidelnému vyměňování informací mezi servery.
- Hodnota i s vektorovými hodinami se tedy rychle zpropaguje na další servery.
- Poté dojde k výpadku serveru A.

Příklad

- Dochází k pravidelnému vyměňování informací mezi servery.
- Hodnota i s vektorovými hodinami se tedy rychle zpropaguje na další servery.
- Poté dojde k výpadku serveru A.

Příklad

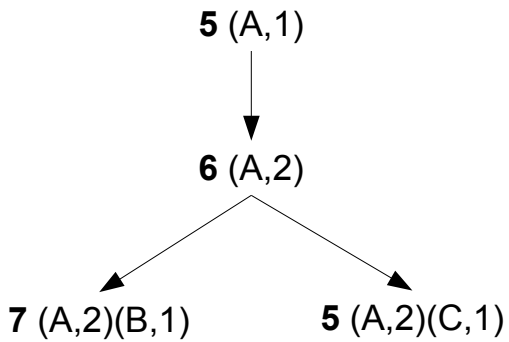
- Dojdou dva paralelní požadavky na změnu hodnoty oblíbenosti XY.
- Jeden požadavek zpracuje server B:
  - inkrementuje se číslo 6 na 7
  - a vytvoří se k číslu vektorové hodiny (A,2)(B,1).
- Další požadavek zpracuje server C (tento požadavek naopak ruší předchozí událost 'Like'):
  - dekrementuje se číslo 6 na 5
  - a vytvoří se k číslu vektorové hodiny (A,2)(C,1).

Příklad

- Dojdou dva paralelní požadavky na změnu hodnoty oblíbenosti XY.
- Jeden požadavek zpracuje server B:
  - inkrementuje se číslo 6 na 7
  - a vytvoří se k číslu vektorové hodiny (A,2)(B,1).
- Další požadavek zpracuje server C (tento požadavek naopak ruší předchozí událost 'Like'):
  - dekrementuje se číslo 6 na 5
  - a vytvoří se k číslu vektorové hodiny (A,2)(C,1).

Příklad

- Na základě hodnot vektorových hodin lze detekovat, že hodnoty ze serverů B a C jsou v paralelních větvích.



Příklad

- Když si servery B a C vzájemně vymění hodnoty oblíbenosti XY, pak musíme být schopni nějak sestavit skutečnou hodnotu.
- Stačí, když si zapamatujeme provedené operace nad hodnotou na serveru.
- Tyto operace se vždy zapisují do tzv. log souboru kvůli zotavení při pádu serveru.
- Oba servery tedy sestaví z těchto operací a původní společné hodnoty dvou paralelních verzí verzi novou

  a přiřadí výsledné hodnotě (6) vektorové hodiny (A,2)(B,1)(C,1).
- Když se server A zotaví, nejprve porovná své vektorové hodiny (A,2) s vektorovými hodinami (A,2)(B,1)(C,1),

  a protože jsou obsaženy v nových hodinách jeho sousedů, hodnota se jednoduše přepíše a vektorové hodiny se aktualizují.

Příklad

- Když si servery B a C vzájemně vymění hodnoty oblíbenosti XY, pak musíme být schopni nějak sestavit skutečnou hodnotu.
- Stačí, když si zapamatujeme provedené operace nad hodnotou na serveru.
- Tyto operace se vždy zapisují do tzv. log souboru kvůli zotavení při pádu serveru.
- Oba servery tedy sestaví z těchto operací a původní společné hodnoty dvou paralelních verzí verzi novou

  a přiřadí výsledné hodnotě (6) vektorové hodiny (A,2)(B,1)(C,1).
- Když se server A zotaví, nejprve porovná své vektorové hodiny (A,2) s vektorovými hodinami (A,2)(B,1)(C,1),

  a protože jsou obsaženy v nových hodinách jeho sousedů, hodnota se jednoduše přepíše a vektorové hodiny se aktualizují.

Příklad

- Když si servery B a C vzájemně vymění hodnoty oblíbenosti XY, pak musíme být schopni nějak sestavit skutečnou hodnotu.
- Stačí, když si zapamatujeme provedené operace nad hodnotou na serveru.
- Tyto operace se vždy zapisují do tzv. log souboru kvůli zotavení při pádu serveru.
- Oba servery tedy sestaví z těchto operací a původní společné hodnoty dvou paralelních verzí verzi novou

  a přiřadí výsledné hodnotě (6) vektorové hodiny (A,2)(B,1)(C,1).
- Když se server A zotaví, nejprve porovná své vektorové hodiny (A,2) s vektorovými hodinami (A,2)(B,1)(C,1),

  a protože jsou obsaženy v nových hodinách jeho sousedů, hodnota se jednoduše přepíše a vektorové hodiny se aktualizují.

Příklad

- Když si servery B a C vzájemně vymění hodnoty oblíbenosti XY, pak musíme být schopni nějak sestavit skutečnou hodnotu.
- Stačí, když si zapamatujeme provedené operace nad hodnotou na serveru.
- Tyto operace se vždy zapisují do tzv. log souboru kvůli zotavení při pádu serveru.
- Oba servery tedy sestaví z těchto operací a původní společné hodnoty dvou paralelních verzí verzi novou

  a přiřadí výsledné hodnotě (6) vektorové hodiny (A,2)(B,1)(C,1).
- Když se server A zotaví, nejprve porovná své vektorové hodiny (A,2) s vektorovými hodinami (A,2)(B,1)(C,1),

  a protože jsou obsaženy v nových hodinách jeho sousedů, hodnota se jednoduše přepíše a vektorové hodiny se aktualizují.

Příklad

- Když si servery B a C vzájemně vymění hodnoty oblíbenosti XY, pak musíme být schopni nějak sestavit skutečnou hodnotu.
- Stačí, když si zapamatujeme provedené operace nad hodnotou na serveru.
- Tyto operace se vždy zapisují do tzv. log souboru kvůli zotavení při pádu serveru.
- Oba servery tedy sestaví z těchto operací a původní společné hodnoty dvou paralelních verzí verzi novou

  a přiřadí výsledné hodnotě (6) vektorové hodiny (A,2)(B,1)(C,1).
- Když se server A zotaví, nejprve porovná své vektorové hodiny (A,2) s vektorovými hodinami (A,2)(B,1)(C,1),

  a protože jsou obsaženy v nových hodinách jeho sousedů, hodnota se jednoduše přepíše a vektorové hodiny se aktualizují.

Příklad

- Když si servery B a C vzájemně vymění hodnoty oblíbenosti XY, pak musíme být schopni nějak sestavit skutečnou hodnotu.
- Stačí, když si zapamatujeme provedené operace nad hodnotou na serveru.
- Tyto operace se vždy zapisují do tzv. log souboru kvůli zotavení při pádu serveru.
- Oba servery tedy sestaví z těchto operací a původní společné hodnoty dvou paralelních verzí verzi novou

  a přiřadí výsledné hodnotě (6) vektorové hodiny (A,2)(B,1)(C,1).
- Když se server A zotaví, nejprve porovná své vektorové hodiny (A,2) s vektorovými hodinami (A,2)(B,1)(C,1),

  a protože jsou obsaženy v nových hodinách jeho sousedů, hodnota se jednoduše přepíše a vektorové hodiny se aktualizují.

Příklad

- Když si servery B a C vzájemně vymění hodnoty oblíbenosti XY, pak musíme být schopni nějak sestavit skutečnou hodnotu.
- Stačí, když si zapamatujeme provedené operace nad hodnotou na serveru.
- Tyto operace se vždy zapisují do tzv. log souboru kvůli zotavení při pádu serveru.
- Oba servery tedy sestaví z těchto operací a původní společné hodnoty dvou paralelních verzí verzi novou

  a přiřadí výsledné hodnotě (6) vektorové hodiny (A,2)(B,1)(C,1).
- Když se server A zotaví, nejprve porovná své vektorové hodiny (A,2) s vektorovými hodinami (A,2)(B,1)(C,1),

  a protože jsou obsaženy v nových hodinách jeho sousedů, hodnota se jednoduše přepíše a vektorové hodiny se aktualizují.

Vektorové hodiny

- Překvapivě se tato metoda dá aplikovat na mnoho operací ve webovém prostředí:
  - komentáře,
  - přiřazování do skupin (nákupní košík).

Vektorové hodiny - nevýhody

- Vektorové hodiny mohou nepříjemě narůstat, což není žádoucí.
- Proto dochází k odstraňování „starých“ částí hodin.
- K tomu se používá časové razítko pro jednotlivé části.
- Je nutné stanovit, jak bude probíhat slučování paralelních verzí.
- V uvedém příkladu může slučování probíhat automaticky; někdy ale mohou být konflikty komplikovanější, takže je nutný zásah uživatele.

Vektorové hodiny - nevýhody

- Vektorové hodiny mohou nepříjemě narůstat, což není žádoucí.
- Proto dochází k odstraňování „starých“ částí hodin.
- K tomu se používá časové razítko pro jednotlivé části.
- Je nutné stanovit, jak bude probíhat slučování paralelních verzí.
- V uvedém příkladu může slučování probíhat automaticky; někdy ale mohou být konflikty komplikovanější, takže je nutný zásah uživatele.

Závěr

- NoSQL
- Motivací pro databázové systémy využívající eventuální konzistenci jsou jednoduchá škálovatelnost a dostupnost.
- Tyto databázové systémy jsou základem mnoha velkých webových aplikací (sociální sítě, on-line hry, e-shopy, zpravodajské weby).

Závěr

- NoSQL
- Motivací pro databázové systémy využívající eventuální konzistenci jsou jednoduchá škálovatelnost a dostupnost.
- Tyto databázové systémy jsou základem mnoha velkých webových aplikací (sociální sítě, on-line hry, e-shopy, zpravodajské weby).

Závěr

- NoSQL
- Motivací pro databázové systémy využívající eventuální konzistenci jsou jednoduchá škálovatelnost a dostupnost.
- Tyto databázové systémy jsou základem mnoha velkých webových aplikací (sociální sítě, on-line hry, e-shopy, zpravodajské weby).

Díky za pozornost!

- Google (BigTable)
- Amazon (Amazon DynamoDB)
- Facebook
- Last.fm
- Twitter
- BBC (CouchDB)
- Wiley, Springer, Oxford press, Elsevier (MarkLogic)
- SourceForge, Guardian (MongoDB)
- The New York Times (MongoDB)