Princípy tvorby softvéru
Perzistencia, databázy

Robert Lukoťka
lukotka@dcs.fmph.uniba.sk
www.dcs.fmph.uniba.sk/~lukotka

M-255

Persistencia

Desktop aplikácia, ukladanie "Save" buttonom.

- Napíšeme k triedam serializačné a deserializačné metódy (PS: Tieto metódy asi chceme implementovať v podtriede, SRP)
- Všetko uložíme.
- What could possibly go wrong?

Persistencia

Desktop aplikácia, ukladanie “Save” buttonom.

- Napíšeme k triedam serializačné a deserializačné metódy (PS: Tieto metódy asi chceme implementovať v podtriede, SRP)
- Všetko uložíme.
- What could possibly go wrong?

Potrebujeme: trvacnosť, konzistencia.

Data exchange formats

BTW, ako serializovať objekty?

- Často je dobré použiť vhodný markup language: XML, JSON, YAML, ...
- Problém je so vzťahmi, najmä asociáciou a kompozíciou. Ako na to?
  - jasni vlastníci asociácií,
  - jasný vlastníci objektov,
  - hranice jednotiek pre serializáciu,
  - idčka objektov.

Persistencia

Predpokladajme, že chceme automaticky ukladať zmeny

- Kedy uložiť zmeny?
- Asi nechceme ukázať chorý medzistav.

Potrebujeme: atomickosť

Persistencia

Čo ak môže byť v systéme viacere "rozbabraných vecí naraz"

Persistencia

Čo ak môže byť v systéme viacere "rozbabraných vecí naraz"
Potrebujeme: tranzakcia, izolácia tranzakcií.

Problémy s perzistenciou

Všetko je krásne a jednoduché . . .

Problémy s perzistenciou

Všetko je krásne a jednoduché . . . kým k dátam pristupuje iba jeden proces.

- Duplikácia dát, pamäť vs úložisko, má vôbec zmysel si pamätať niečo v operačnej pamäti?
- Všetky problémy súvisiace s konkurentnosťou.

Problémy s perzistenciou

Všetko je krásne a jednoduché . . . kým k dátam pristupuje iba jeden proces.

- Duplikácia dát, pamäť vs úložisko, má vôbec zmysel si pamätať niečo v operačnej pamäti?
- Všetky problémy súvisiace s konkurentnosťou.

ACID

- atomickosť
- konzistentnosť
- izolácia
- trvácnosť

ACID

- atomickosť
- konzistentnosť
- izolácia
- trvácnosť

. . . toto predsa robia databázy.

OOP a persistencia

Čo to znamená pre OOP?

- Data a funkcionalita sú separované.
- Kód má charakter procedúr vykonávajúcich tranzakcie nad DB.
- jednoduché fixy (DAO, Active record) na tejto skutočnosti nič nemenia, resp. su príliš neefektívne.
- Problémom sa nedá vyhnúť, je nevyhnutné správne definovať hranice tranzakcií.
- Je možné postupovať aj OOP (k tomu sa dostaneme), často je ale procedurálny, resp. iný prístup praktickejší.

ACID Databázy

- Perzistencia a konkurencia: veľa komplexity
- Často najjednoduchšie riešenie je nechať to na ACID databázu (najjednoduchšie často = správne).
- ACID - silné grancie umožňujú jednoduché rozmýšľanie.

ACID Databázy

- Silné garancie prichadzaju za cenu priepustnosti systému
- Cena narastá v pripade distribuovaných riešení.
- Garancie sú často silnejšie ako je potreba business rules.
- Anyways, tu je svet krasny a ak nemusite nechodte z tadialto preč.

Čo ak tranzakcia trvá priliš dlho?

Web app

Browser, Server, Databáza

- Browsujúci vložil objekt do košíka. Kde to uložiť?

Web app

Browser, Server, Databáza

- Browsujúci vložil objekt do košíka. Kde to uložiť?

Browser, Load balancer, Server, Databáza

Web app

Browser, Server, Databáza

- Browsujúci vložil objekt do košíka. Kde to uložiť?

Browser, Load balancer, Server, Databáza

- Ukladanie v datazábe umožňuje serverom aby boli zameniteľné.

V prípade fakt veľa zápisov/čítaní je potrebné zvýšiť priepustnosť databázy / zvoliť distribuované riešenie.

Distribuované riešenie

- Replication
- Fragmentation(horizontal, vertical)

Pri distribuovaných DB je cena za ACID garancie vyššia.

< ako ACID - Atomickosť, Konzistencia

Atomickosť:

- Atomickosť môže byť napr. garantovaná iba v rámci menšiaho celku.
- Atomicky s inými zmenami nemožno vytvárať/mazať elementy.
- Systém síce umožňuje robiť atomicky hocičo, ale performance penalty je privysoká.

Consistency:

- Zapíšeme na server a potom prečitame niečo z repliky - vadí nám ak to tam ešte nie je?

< ako ACID - Izolácia

Problémy:

- Dirty reads - prečítame necommitnuté data
- Non-repeatable reads - ten istý prvok prečítame dvakrát s rôznymi výsledkami
- Phantom reads - počas vykonávania tranzakcie vznikli alebo zanikli riadky (najnáročnejšie zabezpečiť, kje potrebne zamkýnať časti tabuľky)
- Lost update - zrušíme zmenu vykonanú inou tranzakciou

Isolation levels:

- Serializable
- Repeatable reads
- Read committed
- Read uncommitted

< ako ACID - Durability

E.g. Mongo db - write concern (existujú aj read concerns - to skôr patrí pod consistency)

- Unacknowledged
- Acknowledged
- Journaled
- FSYNCED
- Replica Acknowledged (počet replík / majority)

ACID vs BASE

Basically Available, Soft state, Eventual consistency
Vieme s týmto existovať?

- Hlasovanie nodov o tom, koľko stojí daná vec?
- Hlasovanie nodov o tom, či táto vec má byť v košíku alebo nie?
- Hlasovanie nodov o tom, či má Fero na účte 0 alebo 1000 Eur?

Programátor môže usmerniť "hlasovanie".

ACID vs BASE

Basically Available, Soft state, Eventual consistency
Vieme s týmto existovať?

- Hlasovanie nodov o tom, koľko stojí daná vec?
- Hlasovanie nodov o tom, či táto vec má byť v košíku alebo nie?
- Hlasovanie nodov o tom, či má Fero na účte 0 alebo 1000 Eur?

Programátor môže usmerniť "hlasovanie".

- Nejde len o efektivitu ide aj o dostupnosť a rýchlosť.

Partitioning

Garancie sú často silnejšie ako je potreba business rules.

- Rezervácia izby, nie je však možné kontaktovať server zodpovedný za daný hotel.
- Môžem si rezervovať tuto izbu? Pozrel som sa pred hodinou, celý hotel bol voľný.

Nejde len o výpadok serveru spojenia / ide aj o latenciu.

Avialability vs consistency

CAP theorem - je aj skutočná veta, ale tento vyraz sa medzi developermi používa aj na vyjadrenie tohoto triviálneho pozorovania: Ak sa nemôžem spojiť s druhým serverom, musím si vybrať medzi konzistenciou a dostupnosťou.

- Je to business decision!!!
- Každopádne, pokiaľ je systém dostatočne malý, vyberte si ACID, rozhodnutí bude menej (Stále musíte riešiť zálohy a pod.).

Typy databáz - forma ukladania dat

- key-value páry/ dokumenty
- relačné tabuľky
- graf
- a iné: object, culumn-oriented, ...

Buzzwords: NoSQL, NewSQL

Forma ukladania dat

Relačné databázy:

- Dobre známe - programátori to vedia.
- Štandardizovaný dotazovací jazyk - SQL.
- R&D počas $>$ 40 rokov.

Správna default voľba.

Key-value páry / dokumenty

Aplikácia typicky chce špecifickú value/dokument.

- PS: typicky: v kóde vs runtime - rôzne veci
- Jednoduché query, zatiaľ, čo relačná databáza môže potrebovať nejaké joiny (mimochodom vďaka > 40 rokov R&D to zvládne dobre).
- Môze byť veľmi neefektívne queryovať niečo iné.
- Dokumenty sa ľahšie distrubuujú ako riadky tabuliek.

Grafové databázy

Vhodné na zachyenie asociacií.

- Skúste v SQL naqueryovať “Otca brata koňa predchádzajúceho majiteľa dcéra”, okrem toho, DB sa ujoinuje

ORM

Čo keď chceme robiť OOP s relačnými databázami? Čo by sme chceli?

- Určiť hranice tranzakcií musíme - z toho sa nevyvlečieme.
- Musíme naše triedy namapovať na tabuľky - (typicky nie super ťažké)
- Musíme rozhodnúť ktorá časť DB sa v rámci tranzakcie načíta (Lazy Loading, Eager Loading)
- Väčšinou si nemôžeme dovoliť robiť zmeny po jednej - príliš pomalé, treba toho urobiť viac naraz (Unit of Work).
- Keď z nejakého dôvodu načítame dvakrát, vznikne z toho iba jeden objekt (Identity map).

ORM

Bežný problém - je naň veľa, veľmi rôznych nástrojov:

ORM - Object Relational Mapping (A poväčšine touto skratkou označujeme nástroj na ORM)

ORM

ORMká sú veľmi rôzne.

- Niektoré vám v podstate iba nahrania SQL niečím "objektovejším".
- Niektoré sa vám postarajú aj o mená tabuliek (Trieda "Person", tabuľka "People").
- Niektoré majú veľmi lazy loading ak sčítate dve čísla z databázy, v skutočnosti sa nič neprečíta iba sa vytvorí objekt reprezentujúci súčet.
- Trieda + Mapper vs Trieda zviazaná s tabuľkou.
- ...

Zhrnutie

- S perzistenciou je kopec problémov (najmä ak sa k perzistentným dátam pristupuje konkurentne), najjednoduchšie je nechať to na databázu.
- ACID garancie sú super, pokiaľ nemusíte nevzdávajte sa ich.
- Na zvýšenie priepustnosti systému (najmä v prípade distribuovaných databáz) však môže byť niekedy žiadúce tieto garancie zúžiť.
- V prípade distribuovaných systémov rozhodujú o pomere medzi konzistenciou a dostupnosťou/latenciou business rules.
- Existuje viacero dátových modelov, v typickom prípade vyhráva relačný model pre svoju flexibilitu a R&D.
- Na mapovanie tried z OOP na relačné tabuľky je vyvinutých mnoho nástrojov - ORMká.

Ďalšie zdroje

- ACID - Wikipédia
- Data exchange formats - Wikipedia
- Video: M. Fowler: Introduction to NoSQL
- NoSQL
- Introduction to SQLAlchemy
- Understanding Durability & Write Safety in MongoDB