Princípy tvorby softvéru
Perzistencia, databázy

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

M-255

OOP a 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?

OOP a 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?
  - hierarchický design,
  - jasný vlastníci objektov,
  - idčka objektov.

OOP a persistencia

Desktop aplikácia, ukladanie "Save" buttonom.

- What could possibly go wrong?

OOP a persistencia

Desktop aplikácia, ukladanie "Save" buttonom.

- What could possibly go wrong?

Potrebujeme: atomickost

OOP a persistencia

Všetko je krásne a jednoduché . . .

OOP a persistencia

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

- Duplikácia dát, pamäť vs úložisko, má vôbec zmysel si pamätať čo je na úložisku?
- Kontrola prístupu k zdieľaným datam, konzistentnosť, atomickosť a trvácnosť je ťažšie zabezpečiť.

OOP a persistencia

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

- Duplikácia dát, pamäť vs úložisko, má vôbec zmysel si pamätať čo je na úložisku?
- Kontrola prístupu k zdieľaným datam, konzistentnosť, atomickosť a trvácnosť je ťažšie zabezpečiť.

Potrebujeme: izolácia

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.
- Problémom sa nedá vyhnúť, je nevyhnutné správne definovať hranice tranzakcií.
- Je možné postupovať aj OOP (proxies, identity maps, connection/transaction objects), často je ale procedurálny, resp. iný prístup praktickejší.

BTW: Aké problémy prinášajú procedúry pracujúce priamo nad databázou?

ACID Databázy

- Perzistencia a konkurencia: veľa komplexity
- Často najjednoduchšie riešenie je nechať to na ACID databázu (mimochodom, najjednoduchšie často = to 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

- Ukladanie v databábe umožňuje serverom aby boli zameniteľné.

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

< ako ACID

- Isolation levels - Keď začneme poľavovať z ACID, veci sa stávajú komplikovanejšími.

ACID vs BASE

Basically Available, Soft state, Eventual consistency

- Hlasovanie nodov o tom, koľko stojí daná vec?
- Hlasovanie nodov o tom, či táto vec má byť v košíku alebo nie?
- Alebo to môžeme nechať na programatora: Čo je v košíku? - union....
- Distribuované riešenia: Nejde len o efektivitu ide aj o dostupnosť.

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 veľný.

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

Avialability vs consistency

CAP theorem - je aj skutocna veta, ale tento vyraz sa medzi developermi pouziva aj na vyjadrenie tohoto trivialneho pozorovania: Ak sa nemozem spojit s druhym serverom, musim si vybrat medzi konzistenciou a dostupnostou.

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

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

Zhrnutie

- S perzistenciou je kopec problémov (najmä ak sa k perzistentným datám 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 rozhoduju o pomere medzi konzistenciou a dostupnosťou/latenciou business rules.
- Existuje viacero datových modelov, v typickom prípade vyhráva relačný model pre svoju flexibilitu a past R&D.

Ďalšie zdroje

- ACID - Wikipédia
- Data exchange formats - Wikipedia
- Video: M. Fowler: Introduction to NoSQL
- NoSQL