Objektové principy a SQL databáze

Helena Palovská[1]

[1]Katedra systémové analýzy, VŠE Praha, W. Churchilla 4, 130 00, Praha 3,
Katedra informatiky, VŠFS o.p.s., Estonská 500, 101 00, Praha 10
palovska@vse.cz

Abstrakt. Databázový přístup je založen na základní myšlence: aplikace jsou odděleny od správy dat, spravovaná data jsou sdílena. Každý datový objekt mívá svou zodpovědnou osobu, ale přísné zapouzdření objektů brání efektivnímu programování aplikací požadujících výběr dat. Manipulace s daty jsou na druhou stranu bezpečnější, pokud jsou prováděny pomocí zapouzdřených metod objektů. Návrh SQL databáze proto musí respektovat dvojí požadavky: zaprvé napomáhat efektivnímu vyhledávání dat, zadruhé nabízet interface v řeči objektů odrážejících reálné objekty aplikační domény. Tento článek předkládá některé myšlenky vhodné pro výuku informatiků v této problematice.

Klíčová slova: objekty, návrh SQL databáze, výuka

1 Úvod

Studenti informatiky jsou konfrontováni s nekonzistencí mezi objektovým paradigmatem tvorby aplikací a idejemi pro návrh a užití SQL databází. Tato nekonzistence je principielní. Myšlenka dělby zodpovědností a zapouzdření služeb je v konfliktu s potřebou znát a používat efektivní způsob práce s databázovým strojem, rozumět mechanismům jeho fungování.

Z hlediska databázového přístupu by bylo možné vidět databázi jako jediný objekt, nabízející své služby. Jenže těchto služeb není malá omezená množina, ale jsou jimi potenciálně všechny SQL příkazy. Specielně možných SQL SELECTů nad konkrétní databázovou strukturou je příliš mnoho na to, aby byly jednotlivě nabízeny jako služby. Samozřejmě je běžnou praxí to, že typové SELECTy navrhují zkušení databázoví programátoři, a pro ostatní jsou tyto zapouzdřeny do uložených procedur. Nicméně potřeba na ad-hoc dotazování tuto architekturu nabourává.

V následující kapitole jsou shrnuty základní principy efektivního vyžití databázového stroje. Specielně jsou vyjmenována pravidla pro správný návrh databázové struktury z tohoto hlediska.

Směrem k usnadnění objektově orientované tvorby aplikací přistupujících k SQL databázím je možno udělat některé kroky, zmíněné v kapitole další.

2 Efektivní využití databázového stroje

SQL databázové stroje byly optimalizovány hlavně v 80. letech, většina principů je dnes obecně známa. Zde je výčet těch základních:

- Databázový stroj je optimalizován pro efektivní vyhledávání záznamů podle hodnot v polích, nikoli podle jejich částí či podle výrazů, až na výjimky.
  - Výjimkou jsou začátky hodnot textových polí a některé SQL funkce, například YEAR, MONTH, DAY.
- Urychlení vyhledávání se provozuje pomocí indexů, ty jsou navrhovány pro konkrétní pole či jejich kombinace.
  - U specifických datových typů je nabízena možnost indexovat podle částí nebo funkcí.
- Vyhledávání podle hodnoty v poli, jež není indexováno, vyžaduje scan celé tabulky, jehož trvání je přímo úměrné délce záznamů a jejich počtu.
- Nejnáročnější operací je JOIN, k jehož urychlení je možno udržovat indexy pro spojovací pole, či denormalizovat – tedy vlastně se JOINu vyhnout.
- SELECTy a aktualizace jsou vzájemně konfliktní.
  - Operace UPDATE a DELETE vyžadují uzamčení alespoň nejmenší množiny fyzických bloků, obsahujících měněná data.
  - Operace INSERT nejméně zatěžuje provoz, její konflikt se SELECT je možno dobře řešit.

2.1 Důsledky pro návrh databáze

Vše, co bude samostatně vyhledáváno či porovnáváno, navrhněte jako pole nebo množinu polí. Jinými slovy, nalezněte atomy vyhledávání či porovnávání, a ty navrhněte jako samostatná pole. K některým funkcím nad určitými datovými typy lze přistupovat stejně, jako by jimi vracené hodnoty byly samostatnými poli tabulky.

Odlišujte „historická“ data, která po uložení zůstávají nadále beze změny, a „aktuální“ data, jež jsou měněna podle aktuálního stavu světa. U historických dat je možno zvážit jejich archivy či „sklady“, redundantní uložení sloužící výhradně k SELECTům. Také je u nich možno zvážit denormalizaci. Je třeba zvážit i dynamický návrh, vhodně oddělit historická data od aktuálních kvůli aktualizacím a nutností uzamykat odpovídající logické celky databázové struktury.

Vícehodnotové atributy objektů je nejlépe ukládat do samostatné tabulky se vztahem n:1 k tabulce ukládající objekty. Složené atributy je nejlépe ukládat ve složkách, pro každou složku samostatné pole. Je možno zvážit ukládání odvozených hodnot, pokud jsou často požadovány.

3 Objektové přístupy v návrhu databáze

Při návrhu databáze můžeme potřeby objektového přístupu zohlednit třemi způsoby: důsledně podporovat objektovou identitu, používat moderní logické struktury pro

ukládání komplexních objektů a navrhnout a nabízet metody pro přístup k objektům a pro manipulaci s datovými objekty.

3.1 Objektová identita, reference

Objekty lze do SQL databází ukládat jako řádky tabulek, a mohou mít objektovou identitu. Objektový identifikátor tak může sloužit místo primárního klíče, což má mnoho výhod:

Nehrozí, že by někdo omylem změnil hodnotu tohoto identifikátoru za života objektu-řádku tabulky. Toto však lze bohužel obejít, jestliže je smazán řádek a pak založen nový řádek se stejnými hodnotami atributů (a stejným významem řádku).

Zajišťování jedinečnosti objektových identifikátorů je prováděno databázovým strojem efektivněji, než u „klasických“ primárních klíčů.

Sémantické klíče objektů z aplikační oblasti mohou být nadále udržovány jako UNIQUE. Sémantické klíče jsou důležité, protože pomocí nich lidé vyhledávají záznamy.

Vztahy mezi objekty zaznamenávané v relačním modelu prostřednictvím cizích klíčů mohou být zachycovány objektovými referencemi. Výhody jsou následující:

Při moudrém návrhu konstruktorů event. destruktorů není v mnoha případech třeba hlídat referenční integritu. V ostatních případech stačí vhodný návrh metod pro aktualizaci odkazů. Hodnota referencí se zadává jako odkaz na konkrétní objektový řádek, což výrazně snižuje riziko chyby.

Indexy pro realizaci vazeb mezi nadřízenými a podřízenými řádky (vazby 1:n) jsou při použití objektových identifikátorů velmi efektivní.

Při použití konstruktu SET by bylo možno přirozeně realizovat i vazby n:m, tento konstrukt však zatím není v SQL databázích podporován.

3.2 Abstraktní datové typy, zahnízděné tabulky, strukturované hodnoty

V moderních SQL databázích je nabízena značná podpora pro návrh složité logické struktury objektových záznamů. Ta sahá od možnosti definovat různé datové typy pro daný aplikační kontext až k možnostem vytvářet složité logické struktury v organizaci databáze.

Abstraktní datové typy a tzv. odlišující typy slouží k respektování aplikační sémantiky a k ochraně před lidskými chybami silným typováním.

Konstrukce strukturovaných atributů a zahnízděných tabulek umožňují odrážet realitu složitých struktur objektů.

3.3 Metody

K zapouzdření manipulací s objektovými záznamy nabízejí SQL databáze možnost definovat metody příslušné jednotlivým řádkovým (tj. objektovým) typům i sloupcovým typům.

Důsledné užívání těchto metod může zajistit větší bezpečnost dat, na druhou stranu možnost manipulovat s daty přímo tuto výhodu narušuje. Nedat práva k přímým manipulacím je řešením v případě, že nemusíme udržovat funkčnost „legacy“ aplikací.

Protože SQL databáze umožňují používat typovou hierarchii v definici dat, v aplikacích je možno využívat i polymorfismu.

4 Požadavky na manipulace s daty

Pro aplikačního programátora je ideální následující přístup. K manipulací s daty používá metody, vytvořené designérem databáze. V ideálním případě ani jinou možnost nemá.

K požadavkům na výběr dat buď používá předdefinované SELECTy v uložených procedurách, nebo zadává SELECT příkaz sám a přitom respektuje mj. doporučení zmíněná v tomto článku v kapitole 2 k podpoře efektivity zpracování požadavku, a také se řídí zkušenostmi a znalostmi o konkrétním databázovém stroji a o tom, jak efektivně které formulace stejného požadavku zpracovává.

5 Závěr

Moderní SQL databáze nabízejí podporu objektových přístupů, bohužel však nabízejí také „staré“ možnosti práce s daty, takže designéři a programátoři nejsou vedeni ke správným postupům a volbám. Tento článek má ozřejmit jaká je cesta k objektovému návrhu SQL databáze a k objektovému přístupu při realizaci požadavků na data v aplikacích.

Reference

1. ORACLE. http://download-west.oracle.com/docs/cd/B14117_01/server.101/-b10743/objects.htm#i431766, Object Datatypes and Object Views.
2. Pokorný J. Objektově relační databáze. *Objekty ’1999*. Česká zemědělská univerzita Praha. 80-213-0552-54
3. Pokorný J. Objektově relační databáze. *Datakon 2002*. Masarykova universita v Brně. 80-210-2958-7.
4. Stonebraker M., Brown P. *Objektově relační SŘBD, analýza příští velké vlny*. Softwarové aplikace a systémy, BEN - technická literatura, Praha 2000. 80-860-5694-5 (BEN-technická literatura, Praha) 80-901-5074-8 (Softwarové aplikace a systémy, Praha).

Annotation

Object principles applicable in SQL databases are presented. Database approach is discussed with respect to encapsulation and responsibility division. Advices for teaching of informatics are proposed.