Databáze: relační nebo objektové?

Helena Palovská[1]

[1]Katedra informatiky, Vysoká škola finanční a správní, Praha
Katedra informačních technologií, Vysoká škola ekonomická, Praha
palovska@vsfs.cz,palovska@vse.cz

Abstrakt. Je připomenuta historie relačního modelu dat a je konstatován odklon jazyka SQL od původních představ autora tohoto modelu. Jsou přineseny argumenty pro setrvání u relačního modelu dat jakožto modelu umožňující nejširší varietu výběru dat. Takzvaný impedance mismatch je nahlížen jako mírně zmatený a v některých představách pomýlený. Je navrženo, že objektově orientovaný přístup je možno podpořit dalšími jazyky poskytovanými relační databází, a to podle normy ODMG 3.0 nebo následující. Jsou konstatovány nedostatky současné praxe vůči této představě.

Klíčová slova: databáze, objektové, relační

1 Úvod

Ve svém článku [15] jsem se zabývala možností aplikovat objektové principy v SQL databázích, tj. tím, co umožňuje jazyk SQL. Důvodem k tomu byl fakt, že v dnešní výuce i praxi se jako databáze používají SQL databáze a k tvorbě aplikací se používají objektové principy a objektově orientované jazyky. Mezi relačními principy a objektovými je skok v paradigmatu, který je často interpretován jako historická nutnost opustit relační databáze a přejít k objektovým. Některé argumenty pro tento přechod diskutuji jako z mého pohledu domnělé, a přináším argumenty pro používání relačních databází. Někteří autoři (např. [1]) vidí koexistenci relačních databází a objektového prostředí pro vývoj aplikací jako nutnost danou praxí, já argumentuji pro racionálnost takového přístupu.

2 Relační databáze

S myšlenkou logicky organizovat data v databázi do matematických relací přišel Edgar Frank Codd na konci šedesátých let, komplexním způsobem shrnul svoje myšlenky poprvé v článku [7]. Po dvou desítkách let plných výzkumu a praxe relačních databází publikoval E.F.Codd druhou verzi, tentokrát obsáhlejší a vydanou jako knihu [8]. Relační databáze adoptovaly podporu jazyka SQL, ačkoli E.F.Codd sám nebyl ani jeho autorem ani zastáncem. Spolu s C.J.Datem ([10]) i jinými upozorňovali na nedostatky jazyka SQL i na nedostatky implementace relačních principů v produktech SŘBD.
Pro podporu vhodnosti použití relací v databázích E.F.Codd uvedl různé argumenty. Nejznámější je asi tzv. „informační rys“, požadavek aby všechny informace byly

přístupné jednotným způsobem. Druhým argumentem je požadavek, aby jazyk komunikace s databází nezatěžoval uživatele tohoto jazyka specifiky strukturního uspořádání dat. Dvě a půl desítky let úspěšného používání relačních databází dávají E.F.Coddovi zapravdu o vhodnosti relačního modelu, ačkoli možná nejpodstatnější důvod úspěchu není ani v [7] ani v [8] zmíněn. Tímto důvodem je schopnost relačního uspořádání dat poskytovat nejširší varietu možností deklarovat požadavky na výběr dat – tzv. dotazovat databázi. Toto šíře vysvětlím v kapitole 5.

Poslední jedna a půl desítky let přinášejí požadavky, pokusy i úspěšné komerční produkty SŘBD postavené na objektových principech. Důvodem k tomu je hlavně to, že objektově orientovaný přístup se prokázal jako velmi efektivní při tvorbě aplikací a tvůrci těchto aplikací shledávají obtížným formulovat svoje požadavky na perzistenci objektů v jazyce SQL, stejně tak jako „lámat“ své objekty do databázových relačních tabulek. V tomto článku se pokusím argumentovat pro to, že takové požadavky nedávají důvod k opuštění relačního modelu.

3 Objektové principy

O objektových principech je možno psát obšírně či stručně je jednotlivě vyjmenovávat, pokud jde o jejich seznam, ten se časem se rozšiřuje. Za nejpodstatnější myšlenku považuji dělbu zodpovědností, která technicky vede k autonomii, k zapouzdření programových objektů.

Z hlediska datových struktur přináší objektová orientace dvě nové myšlenky: abstraktní datové typy – tj. možnost definovat nové typy podle potřeb aplikace nad základními typy poskytovanými standardně, a tzv. komplexní objekty, složité struktury jako jednotky manipulace s daty. V obojím vychází SQL vstříc, i když co se týče složitých struktur chybí dobrá podpora konstrukce množin (SET). Navíc, tak jak je tomu v programovacích jazycích, nabízí SQL konstrukci ukazatelů, a provázání záznamů takto překonává nutnost konstrukcí primární klíč-cizí klíč. Tím je sice porušen E.F.Coddův „informační rys“ jak ho E.F.Codd vysvětloval, proti tomuto vysvětlení je však možné argumentovat, protože aplikace ukazatelů v záznamech umožňuje komunikovat i nadále jednotným způsobem, i když jiným[2].

Další často zdůrazňovaný rozdíl objektově orientovaného přístupu proti starším je sloučení datové struktury – záznamu o stavu objektu – s definicí chování objektu. Toto nemá s relačním modelem nic společného, tedy nehovoří to pro ani proti němu. To je otázka rozdělení funkcionality do částí sytému jimž se svěří zodpovědnost za tuto funkcionalitu. Databáze může nést zodpovědnost za určitou část funkcionality, pokud se to jeví účelné. SQL pro tuto možnost nabízí uložené procedury a hybridní konstrukce – triggery.

[2] K „traverzování“ mezi datovými objekty stačí jediný jazykový prvek, použitelný k vyjádření požadavku na související data. Zatím je tento přístup realizován v dotazech. V jazyce pro manipulaci s daty SQL v tomto ohledu pokulhává.

Na tomto místě chci zdůraznit, že poznámky o schopnostech SQL nejsou poznámkami o relačním modelu. Jak E.F.Codd zdůrazňoval, SQL a relační datový model nejsou jedno a totéž.

4 Impedance mismatch

Často se hovoří o tzv. neshodě v „impedancích" relačních databází a objektově orientovaného vývoje aplikací, pro vysvětlení viz [3],[4] nebo [16].

V mnoha ohledech jde o rozdíl v softwarově inženýrských přístupech, strukturovaném a objektově orientovaném, ten je zásadní. Technologický problém kombinace relačních a objektově orientovaných technologií je teoreticky řešen algoritmy mapování objektů do relačních tabulek. Opodstatněná potíž vzniká z nepohodlí pro objektově orientované vývojáře při nutnosti použít jazyk SQL, tím se snižuje výkonnost vývojářů. Zde se dnes nabízejí různá řešení, spočívající ve vložení další vrstvy mezi relační databázi a aplikační program, middleware zprostředkovávající přístup k databázi, viz. [2].

Další důvod neshod spočívá v rozdílném chápání databázových služeb, bez ohledu na použitý databázový model, ať relační či objektový. V databázovém přístupu je zodpovědnost za perzistenci dat oddělena od zbytku informačního systému a svěřena (jediné) databázi, takže databázové služby jsou zapouzdřeny do databáze zajišťující sdílení společných dat. Právě toto poslední se jeví jako největší problém při spolupráci správců dat a objektových vývojářů, neboť zásadní rozdíl je v nazírání privátního a veřejného. Co správce dat vidí jako veřejné, objektový vývojář má tendenci vidět jako privátní. Protože správci dat je svěřen úkol dlouhodobě udržovat databázi v konzistentním stavu a poskytovat datové služby nad sdílenými daty všem autorizovaným aplikacím, nemůže připustit privátnost těchto dat. Častá představa objektových vývojářů o datových službách jsou jakési objektové brokery poskytující zároveň perzistenci jejich privátních objektů.

5 Proč je relační model nejlepší

Navzdory úsilí vytvořit a dodávat objektové databáze současná praxe převážně používá relační databáze a některou z technik vyjmenovaných ve [2]. Důvody je možno nacházet různé, od setrvačnosti zákazníků a síly tradičních dodavatelů relačních databází po nutnost udržovat legacy systémy. Výkonnost objektových databází je také zpochybňována, přesto jsou i aplikační oblasti, kde se užití objektových databází prosazuje, pro výkonnost provozu i údržby specifického informačního systému. Povšimněme si, že jde o typové aplikační oblasti, tedy oblasti s podobnými business objekty.

V dalším se pokusím vysvětlit, v čem spočívá výjimečnost relačního modelu a co způsobuje jeho přednost před ostatními modely. To by pak bylo možno vidět jako další důvod k setrvávání u relačních databází.

5.1 Relace je vztah

V databázi máme data a souvislosti mezi nimi. Tyto souvislosti jsou nějaké vztahy mezi daty. Vztahy, které nejsou odvoditelné (matematicky, logicky…), je třeba zaznamenat, například vztah mezi rodným číslem pana Nováka a jeho bydlištěm, nebo vztah mezi číslem faktury a jednou z položek této faktury. Relace v relačním databázovém modelu je abstraktní konstrukce pro typ vztahu, každý typ vztahu je modelován jednou relací. Když do databáze zaznamenáváme konkrétní vztah mezi konkrétními daty, ukládáme instanci nějaké relace. Relační databázový model nespecifikuje implementaci relací, technologie tabulek není totožná s relačním databázovým modelem. (Viz také [11], [9].)

5.2 Fakta jsou složena z elementárních faktů

Databáze má sloužit k zjišťování fakt, což znamená vyhledávání a získávání dat na základě jejich hodnot a souvislostí. V tomto vyhledávání chceme být co nejméně omezeni.

Nejméně omezeni budeme, když budeme mít přístup k elementárním faktům (např. že pan Novák se narodil 1.1.1900, nebo že pan Novák pracuje v oddělení ABX). Diskusi nad tím, co je elementární fakt, lze nalézt například v [13]; jednoduše to lze vyjádřit tak, že relace reprezentující takový typ faktu je atomická, tj. že ji nelze rozložit do méně-árních relací bez ztráty informace. Metodu datové analýzy a návrhu databáze na základě elementárních faktů lze nalézt v [12], mnoho dostupného je také v [14]. Z elementárních faktů, resp. z jim odpovídajících atomických relací, lze složit jakoukoli konstrukci na základě logiky, porovnávání a skládání. Předpokládáme, že logiku a porovnávání databázový stroj umí, souvislosti mezi daty má zaznamenány. Zbývá vytvářet konstrukce podle své potřeby.

Pokud jsou v databázi některé konstrukce, například některé složité objekty, pevně vytvořené (tzn. jsou uloženy nikoli elementární fakta, ale fakta složená) a jejich rozložení na atomy (atomické relace odpovídající elementární faktům) je za použití jazyka poskytovaného databází nemožné nebo obtížné, pak jsme omezeni v dotazování.

Toto nemusí být důvod proti použití speciálních databázových modelů – hierarchického, síťového, objektového, XML. Pokud jsou předpokládány jen dotazy určitého typu, pak může být účelné uspořádat data tak, aby tyto dotazy byly přednostně efektivně zpracovatelné, lépe než skládáním z elementárních faktů.

Pokud však chceme mít nejširší varietu v možnostech získávání informací z dat, dolování, není nic lepšího než relační model.

6 Závěr – návrh architektury

Co vidím jako budoucnost databází, je inkorporace dnes používané vrstvy, vložené mezi relační databázi a objektově orientované aplikace, do SŘBD. To znamená, že databáze by měla být schopna komunikovat v objektově orientovaném jazyce, tak jak to požaduje norma ODMG (současná verze je zveřejněna v [6]), a tuto komunikaci překládat z a do relačního jazyka. Podstatné je, že relační jazyk by zůstal, kvůli možnosti konstruovat nové složité objekty, dolování dat i kvůli kontrolovatelnému udržování konzistence databáze.

Reference

1. AMBLER Scott W.. URL: www.agiledata.org AgileData website
2. AMBLER Scott W.. URL: http://www.agiledata.org/essays/implementationStrategies.html . Encapsulating Database Access: An Agile "Best" Practice
3. AMBLER Scott W.. URL: http://www.agiledata.org/essays/impedanceMismatch.html . The Object-Relational Impedance Mismatch
4. AMBLER Scott W.. URL: http://www.agiledata.org/essays/culturalImpedanceMismatch.html . The Cultural Impedance Mismatch Between Data Professionals and Application Developers
5. AMBLER Scott W.. *Agile Database Techniques: Effective Strategies for the Agile Software Developer*. Wiley 2003. ISBN 978-0471202837
6. CATTEL R.G.G., BARRY Douglas K. (eds.). *The Object Data Standard: ODMG 3.0* . Morgan Kaufmann Publishers 2000. ISBN 1-55860-647-5
7. CODD E.F.. A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM 13(6) 1970*
8. CODD E.F.. *The Relational Model for Database Management, Version 2*. Addison Wesley Publishing Company 1990. ISBN 0-201-14192-2 . URL: http://portal.acm.org/toc.cfm?id=SERIES11430&type=series&coll=ACM&dl=ACM
9. DARWEN Hugh, DATE C. J. . The third manifesto. *ACM SIGMOD Record 24 (1): 39-49. 1995*, DOI:10.1145/202660.202667. ISSN 0163-5808
10. DATE C. J. . *Where SQL Falls Short* (zkrácená verze). Datamation 1987. (nezkrácená verze) *What Is Wrong with SQL*, dostupné z The Relational InstitUte, San Jose.

11. DATE C. J. . Preview of The Third Manifesto, *Database Programming & Design 11(8) 1998*, ISSN 0895-4518. OCLC 89297479, URL: http://ww.dbpd.com/vault/9808date.html
12. HALPIN Terry. *Information Modeling and Relatonal Databases, From Conceptual Analysis to Logical Design*. Morgan Kaufmann Publishers 2001. ISBN 1-55860-672-6
13. HALPIN Terry. *What IS An Elementary Fact.* NIAM-ISDM 1993. URL: http://www.orm.net/pdf/ElemFact.pdf
14. ORM. URL: www.orm.net Object Role Modelling
15. PALOVSKÁ Helena. Objektové principy a SQL databáze., *Objekty 2005*. VŠB TU Ostrava 2005. ISBN 80-248-0595-2.
16. Wikipedia. http://en.wikipedia.org/wiki/Object-relational_impedance_mismatch. Object-relational impedance mismatch

Annotation

History of relational data model is reminded and a diversion of the language SQL to the original model is stated. Arguments to stay with relational model being a model enabling the widest range for data selection are brought up. So called impedance mismatch is regarded confusing and in some ways mistaken. It is suggested that object-oriented approach can be supported by other languages granted by relational databases, according to e.g. norm ODMG. A drawback of contemporary praxis to this is stated.
Using ideas of C.J. Date [11] and Terry Halpin [13],[14] is shown that relational data model is the best if data retrieval is to be less confined. It is made a proposal to built DBMS upon relational model with object-oriented or other models supported in upper layers for communication.
It is also noted that long term data administration would be better arranged this way. Only in cases when only one specific type of usage is ever to be supported other data organization could be preferable.