TRANSFORMACE RELAČNÍHO DATOVÉHO MODELU NA OBJEKTOVÝ

TRANSFORMATION OF RELATIONAL TO OBJECT DATA MODEL

Vít Holub

Anotace
Článek poskytne čtenáři základní přehled v datových modelech, ukáže výhody a nevýhody současných databázových systémů spolu s typickými řešeními aplikační vrstvy a nastíní principy transformace konkrétní relační databáze na objektovou.

Annotation
This paper guides the reader through different data modeling approaches and reveals strong and weak areas of contemporary database systems along with typical application solutions. In addition, it provides an introductory ideas into the process of converting relational to object-oriented databases.

Klíčová slova
transformace, objektový datový model, relační datový model, databáze, impedanční nesoulad

Keywords
transformation, object data model, relational data model, database, impedance mismatch

Úvod

Pro vývoj i provoz informačního systému je nezbytné vytvoření lidsky čitelného formalizovaného „obrazu“ modelované reality. Tato abstrakce, která zachycuje strukturu uspořádání datových prvků a vazeb mezi nimi, se označuje pojmem datový model. Jedná se tedy o logické zobrazení, které více či méně úspěšně přemosťuje propast člověk – počítač.

Výběr datového modelu úzce souvisí s volbou databázového stroje. Platí, že každý databázový systém je vhodný pouze pro určitý datový model.

Pomineme-li dnes již archaické síťové a hierarchické datové modely, setkáváme se téměř výhradně se dvěma rivaly – modelem relačním (RDM) a objektovým (ODM). Relační model je mnohem starší, jeho vznik se datuje kolem roku 1970. Díky tomu je RDM pochopitelně zastaralejší a nadějnější budoucnost (doufejme) patří ODM.

Mnoho organizací ve svých relačních systémech nashromáždilo velké množství dat. Jednou z možností přechodu na objektové technologie je transformace relačního způsobu uložení dat do objektového. Protože se většinou jedná o rozsáhlé systémy, je manuální přepis neakceptovatelný.

Relační datový model

Základním prvkem RDM je tabulka.. Struktura tabulky je předem jasně definována. Každý řádek obsahuje jeden záznam tabulky, ve sloupcích jsou stejné typy hodnot každého záznamu. Pokud jsou data příliš složitá, než aby se dala reprezentovat v jedné tabulce, musí se vytvořit dodatečné tabulky, které budou uchovávat vztahové informace. Data logicky náležející jednomu záznamu jsou pak umístěna ve více tabulkách. Jednotlivé záznamy se na

sebe odkazují pomocí cizích klíčů, hierarchie a vztahy jsou realizovány pomocí joinů (viz níže).

Pro přístup k datům slouží speciální jazyk, nejčastěji SQL. Jeho dotazovací část umožňuje vybírat potřebné údaje pomocí operací selekce, projekce a spojení.

V souvislosti s RDM jsou nejčastěji zmiňovány tyto silné, resp. slabé stránky:

- k RDM existuje kvalitní matematický aparát, který výrazně zjednodušuje práci s daty
- jazyk SQL je i přes své četné mutace standardem
- databáze založené na RDM jsou stále velmi rozšířené a nic nenasvědčuje tomu, že by se tato skutečnost měla v nejbližší době výrazně změnit
- díky dlouhé době, po kterou se relační databáze vyvíjejí, jsou k dispozici kvalitní databázové stroje, které velmi dobře vyplnily prostor mezi mantinely danými principiálními technologickými omezeními
- komplexní datové struktury nemohou být modelovány přímo
- impedanční nesoulad mezi programovacími jazyky a databází
- při operacích nad více než několika tabulkami současně rapidně klesá výkonnost (v praxi není výjimkou join deseti a více tabulek)
- RDM je daleko více přizpůsoben architektuře počítače než lidskému pojetí problému

Objektový datový model

Uvedené nevýhody (zejména existence impedančního nesouladu) relačního modelu daly později vzniknout objektovým databázím.

Objektový model není omezen na uchovávání dat v řádcích a sloupcích. Návrhář vytvoří třídu (šablonu) objektů, která specifikuje strukturu každé svojí instance (ekvivalent záznamu v RDM). Díky vlastnosti skládání je veškerá informace o záznamu uložena v jediném objektu. To platí i pro metody, tedy programové kódy, které mohou pracovat s daty daného objektu. Až do příchodu uložených procedur, které měly tuto vlastnost v relačních databázích zprostředkovat, neměla tato funkčnost (spojení kódu s daty) obdoby. I nadále jsou však možnosti uložených procedur ve srovnání s metodami objektů ze své podstaty omezené.

K dotazování nad objektovými databázemi slouží standardizovaný ekvivalent SQL – jazyk OQL. Ten se od svého relačního kolegy na první pohled neliší, navíc ale poskytuje např. prostředky pro navigaci mezi objekty.

I zde předkládám výčet kladů a záporů:

- lidsky přirozená reprezentace modelované reality
- přehlednost, snadná udržovatelnost, rozšiřitelnost
- existence hierarchických struktur
- návaznost na trend OOP

negativa se týkají spíše OODBMS než datových modelů obecně:

- v porovnání s RDBMS jsou databáze založené na ODM méně vyzrálé
- bezkonkurenčně největší podíl na trhu mají RDBMS, a jejich výrobci nebudou ochotni jen tak vyklidit své pozice

Současná situace

Obchodní logika v moderních informačních systémech je dnes téměř výhradně řešena použitím technik objektově orientovaného programování. Nejrozšířenějším programovacím jazykem je dnes již patrně Java, následována jazyky C++, VisualBasic, Smalltalk... Všechny tyto jazyky lze s menšími či většími výhradami považovat za objektové. Jak již ale bylo řečeno, většina provozovaných databází je založena na RDM.

Přechod k objektům

Využití objektových technologií ve spojení s relačními databázemi může být chápáno několika způsoby:

1. Vystavěním objektového rozhraní nad relačním databázovým systémem

Tento způsob je bezesporu nejrozšířenější (viz předchozí odstavec). Je relativně levný a snadný, firma ovšem zůstává na půli cesty k objektovosti. Organizace vlastní framework, který mapuje objektové struktury do relačního schématu. Problémem podobných řešení je ztráta výkonu právě na úrovni frameworku - objekty a jejich struktura se při ukládání musí transformovat, takže např. v metodě „ulož“ nějakého objektu bude třeba k databázovému rozhraní přistupovat přes četné příkazy SQL „INSERT INTO VALUES“, které s objektovým pojetím nemají mnoho společného. Autor se také setkal s případem, kdy zobrazení každé řádky tabulky znamenalo vytvoření cca 30 objektů.

Další potíže se mohou objevit v průběhu návrhu - kromě možnosti bohatšího a intuitivnějšího způsobu modelování dat zahrnuje totiž objektová technologie řadu konceptů, které významně zvyšují programátorskou produktivitu. V relačním modelu neexistuje skládání, dědičnost, ani polymorfismus. Někteří prodejci velkých relačních databází, jako jsou společnosti Oracle, Microsoft a IBM, se pokoušeli implementovat koncepty objektově orientovaného návrhu, ale výsledky nesplnily očekávání programátorů. Řešení Oracle8 (viz dále) sice částečně překonává impedanční nesoulad, nicméně ostatní nevýhody RDM, zejména rychlost zpracování při složitějších operacích a problémy s perzistencí, přetrvávají.

Tento způsob je nejjednodušší, neboť nedochází k vlastní migraci dat. Systém se také během přechodu na objektový interface nemusí zastavit.

2. Použitím objektově-relační databáze

Objektově – relační databáze je v podstatě relační databáze, která již v sobě mapovací techniky obsahuje. Příkladem může být systém Oracle, který objektově – relační mapování umožňuje od verze 8.

Migrace dat je zde plně v režii databázového stroje (např. přechod z Oracle 7 na Oracle 8).

Oba uvedené způsoby však díky relačnímu způsobu uložení dat nedovolují plně využít možností, jaké objektově orientované programování přináší. Ani v rychlosti zpracování složitějších operací čistě objektovým databázím nemůže konkurovat.

3. Použití objektové databáze

Objekty v aplikaci zůstávají objekty i po přenosu do databáze, žádné mapování či jiné transformace dat při přístupu k databázi nejsou potřeba. To se projeví nejen výrazným zvýšením výkonu, ale i časovou úsporou při vývoji aplikací. Manipulace a navigace s objekty uloženými v databázi probíhá naprosto stejným způsobem, jako u dat uložených v paměti počítače – pro vývojáře to tedy znamená podstatné ulehčení práce. Tato vlastnost je v literatuře označována termínem „transparent persistence“.

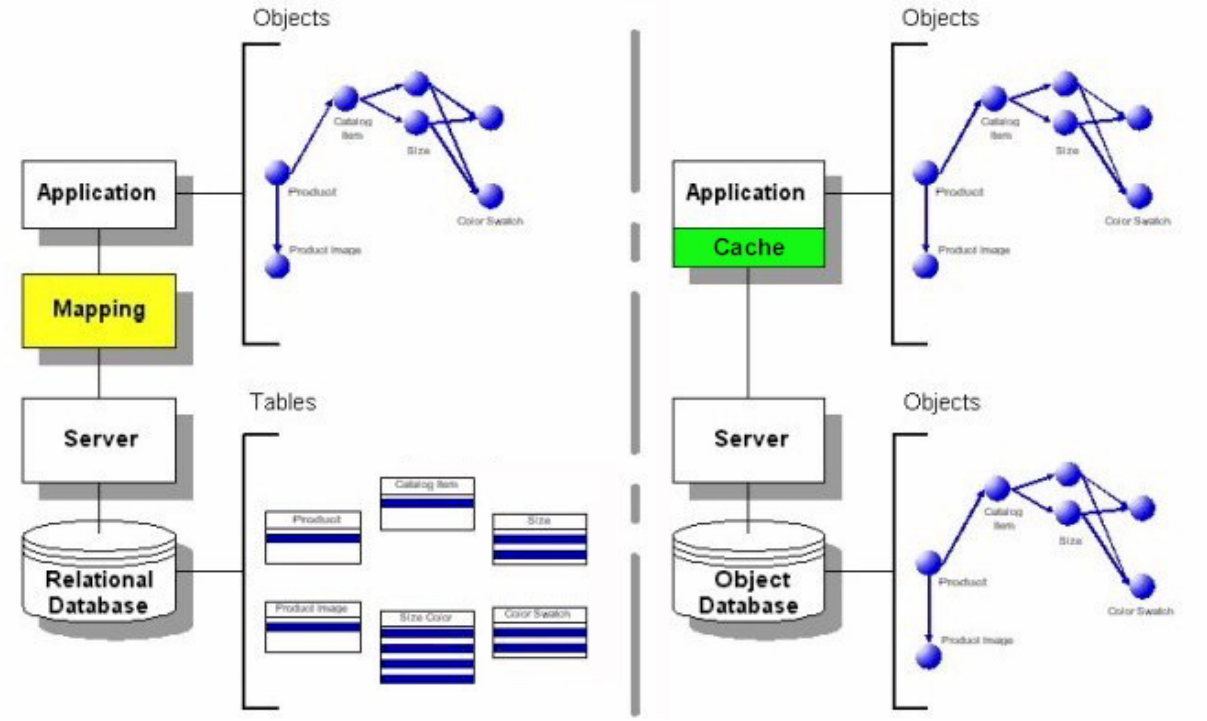
Aplikace může k objektům přistupovat pomocí jazyka OQL (Object Query Language), jakési obdoby SQL. Elegantněji však působí použití takového programovacího jazyka, který je pro danou databázi nativní. Příkladem je silný a oblíbený tandem Smalltalk – Gemstone. V tomto případě již vlastnost „transparent persistence“ není nic dlužna svému jménu.

Potíž může nastat v případě, že k databázi budou přistupovat i starší aplikace, jejichž komunikace s datovým úložištěm je založena na SQL dotazech – typicky tiskové aplikace a aplikace pro vytváření sestav. Zde se setkáváme s impedančním problémem „naruby“ - i

v tomto případě je řešením mapování (logicky nejvhodnějším krokem by samozřejmě bylo přepsání aplikací).

Migrace existující databáze je v tomto případě nejnáročnější. K jejímu bezpečnému provedení je třeba disponovat propracovanou metodikou a kvalitními migračními nástroji. Tomuto řešení bude věnována následující kapitola.

Uvedené způsoby názorně ilustruje následující schéma (mapovací vrstva je v případě objektově – relačních databází součástí databázového stroje).



Transformace RDM na ODM

Při rozhodování o migraci na ODM je třeba brát v úvahu několik aspektů:

- velikost investic do RDBMS, které firma v minulé době učinila
- jak velké náklady by přinesla transformace již fungujícího RDM do ODM
- jak složité a náročné na výpočetní výkon i na údržbu jsou již hotové aplikace nad RDBMS

Ve velkých systémech založených na relační technologii náklady na objektový re-engineering mohou narůst do výšin přesahujících zájmy i možnosti firmy. To může být příležitostí pro hledání efektivních technik, příp. vytvoření metodiky, pro co nejautomatizovanější a nejméně bolestnou transformaci RDM na ODM.

Sestavení uceleného návodu pro migraci je tématem disertační práce autora. Správný postup by měl pokrývat všechny fáze migrace a splňovat následující požadavky:

- poskytnout metodickou podporu pro proces migrace
- pomoci uživatelům transformovat relační schémata a databáze
- vyčerpat všechny důležité vlastnosti a možnosti objektových databází
- být správný a úplný

Kompletní migrace sestává ze tří hlavních fází:

1. Transformace datového modelu
2. Migrace dat
3. Migrace aplikací

Přenos aplikací neznamená pouhé nahrazení SQL příkazů, ale vyžaduje reinženýring imperativních programů do objektových. Proces migrace aplikací však stojí poněkud stranou – může být natolik různorodý, že nemá smysl ho zahrnovat do metodiky.

Transformace datového modelu

Vzhledem k tomu, že relační datový model je speciálním případem objektového, ztratil „během specializace“ některé metainformace. Proto je nutné provést analogii reverse-engineeringu. Principiálně nelze tento proces provést automaticky – je nutná interakce s uživatelem, který klasifikuje víceznačné konstrukce. Vhodným začátkem klasifikace je analýza DDL (data definition language) kódu. V literatuře se tato část transformace výstižně nazývá *sémantické obohacení*. Výstupem je tedy relační schéma obohacené správnými sémantickými informacemi.

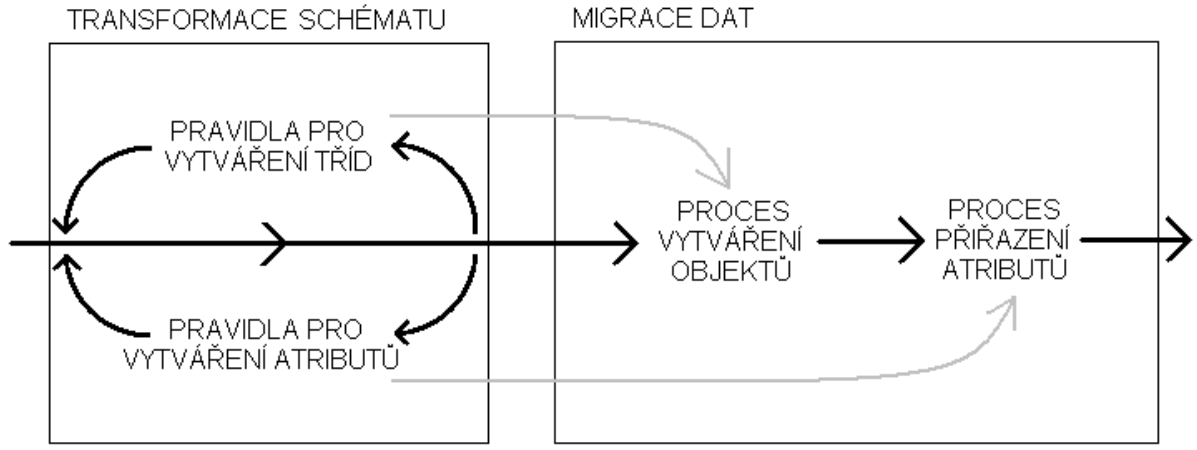
Druhá část může být již automatizována – k přesné a jednoznačné transformaci slouží *transformační pravidla*. Pomocí nich lze transformovat část relačního schématu na objektové. Tato pravidla mohou specifikovat, že např. cizí klíč bude transformován jako agregace, nebo že se relace přemění v třídu. Charakteristickým rysem je iterativní aplikace daných pravidel.

Pravidla mohou definovat, kdy dochází k vytvoření třídy a jak budou třídy hierarchicky uspořádány (pravidla pro vytváření tříd), a jakou budou mít tyto třídy strukturu (pravidla pro vytváření atributů).

Migrace dat

Rozdělení pravidel v předcházejícím odstavci napovídá, jakým způsobem bude probíhat migrace dat. V první fázi dojde k vytvoření tříd, v druhé k přiřazení atributů. Na fyzické úrovni bude migrace spočívat ve vylití (unload) relačních tabulek do sekvenčních souborů, a následném naplnění tříd, to vše podle transformačních pravidel.

Souvislost mezi transformací datového modelu a migrací dat ilustruje následující schéma:



Závěr

Jak relační, tak objektová databáze má dnes na trhu své místo. Relační technologie byla standardem po mnoho let, a mnoho současných aplikací a nástrojů stále vyžadují SQL přístup k datům. Vývojáři nicméně stále častěji sahají po čistě objektových řešeních, která jim přináší nesporné výhody mj. v oblastech náročnosti vývoje a údržby, rychlosti, rozšiřitelnosti, a snadnosti integrace s webovými aplikacemi. Nesnadným úkolem mnoha firem je nyní zvážení dalšího postupu v situaci, kdy mají svá cenná data uložena v relační podobě a potřebují vyvíjet další objektové aplikace. K usnadnění rozhodování by měla přispět možnost relativně snadné a levné migrace z relačního do objektového prostředí. Tu bude podporovat metodika, která je předmětem autorovy disertační práce.

Literatura

Merunka V.: Objektový přístup v databázových systémech. Skriptum ČZU 2002, ISBN 80-213-0882-6

Object-Oriented Databases, http://www.odbmsfacts.com/

Object-Relational Mapping, http://www.object-relational.com/

Object Data Management Group, http://www.odmg.org/

Ian Graham: Migrating to Object Technology. Addison-Wesley 1995

Merunka V., Polák J., Carda A.: Umění systémového návrhu, Grada 2003, ISBN 80-247-0424-2