POROVNÁNÍ RELAČNÍHO A OBJEKTOVÉHO DATOVÉHO MODELU V KONSTRUKCI DATABÁZOVÝCH SYSTÉMŮ

COMPARISON OF THE RELATIONAL AND OBJECT-ORIENTED DATA MODEL FOR DATABASE SYSTEMS DEVELOPMENT

Tomáš Doskočil, Vojtěch Merunka

Anotace
Článek na praktickém příkladě databáze ekonomických subjektů porovná vlastnosti relačního a objektového datového modelu ve dvou implementačních prostředích databázových systémů Oracle a Gemstone.

Summary
The paper will compare features of the relational and object-oriented data model using a practical example of economical subjects in two implementation environments of database systems Oracle and Gemstone.

Key words
relational and object-oriented data model, database system, Oracle, Gemstone

Klíčová slova
relační a objektový datový model, databázový systém, Oracle, Gemstone

Úvod

O objektových databázích a objektově orientovaném přístupu při návrhu informačních systémů se v poslední době mnoho mluví. Ve školách se ale praktické práci s objektovou technologií nevěnuje dostatečná pozornost. Proto jsme se rozhodli naše zkušenosti předat prostřednictvím tohoto článku.

Cíl a metodika

Cílem je pomocí praktického příkladu databáze ekonomických subjektů demonstrovat rozdíly mezi relačním a objektovým přístupem k modelování databázových systémů. Tyto odlišnosti budou ukázány ve dvou konkrétních prostředích velkých databázových systémů, a to Oracle 9i Server a Gemstone 6.0.1 for Smalltalk Server. Jedná se o systémy, které byly v letošním školním roce prakticky používány ve výuce. Téma příspěvku je také v souladu s výzkumným záměrem, jehož předmětem je aplikovaný výzkum v oblasti informačního managementu a využití moderních softwarových technologií. Závěry práce a přínosy z očekávané diskuse budou použity k inovaci vyučovaných odborných informatických předmětů v následném školním roce.

Podkladem pro sepsání tohoto článku byla především několikaletá zkušenost s výukou předmětu „Databázové systémy II.“ ve 3. ročníku oboru „Informatika“ naší fakulty a práce na již jmenovaném výzkumném záměru.

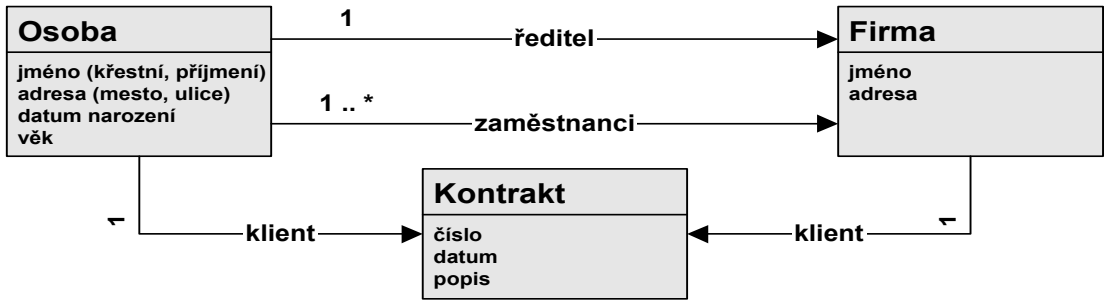
V článku bude postupováno následujícím způsobem:

1. Nejprve bude nalezena jednoduchá, ale dostatečně strukturálně bohatá úloha včetně modelových dat.
2. Pro tuto úlohu bude sestaven její konceptuální model podle standardu UML.
3. Poté bude tato stejná úloha naprogramována ve dvou různých implementačních databázových prostředích. Tj. jako relační databáze v Oracle a objektová databáze v Gemstone.
4. Budou vzájemně porovnány oba rozdílné přístupy k implementaci.
5. Obě dvě aplikace budou naplněna stejnými daty.
6. Obě aplikace budou porovnávány pomocí odlišností v implementaci stejných dotazů do obou prostředí.

Výsledky

popis řešené úlohy

Pro názornou demonstraci jsme si zvolili následující jednoduchou databázovou úlohu. Mějme databázi klientů, u kterých budeme sledovat kontrakty. Nechť platí, že každému jednomu kontraktu přísluší jeden klient, ale jeden klient může mít více kontraktů. Klienti budou dvojího typu: jednotlivé osoby a firmy. U každého kontraktu budeme sledovat následující atributy: číslo, datum a popis. Každá osoba bude mít atributy: jméno, příjmení, adresu (město a ulici) trvalého bydliště, datum narození a věk. Každá firma bude mít atributy název, adresu, a také ředitele a zaměstance, což budou osoby shodného typu jako někteří klienti. Schéma si lze znázornit na obrázku č. 1:

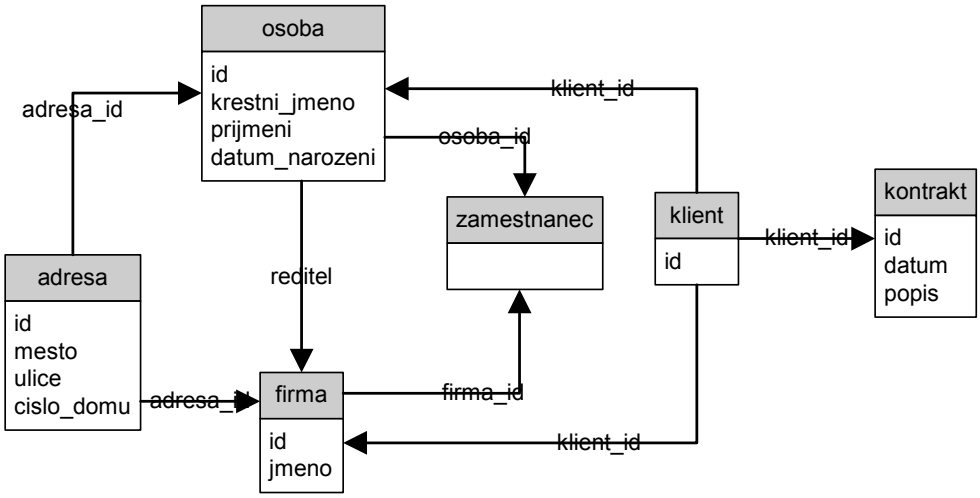


obr. č. 1. - konceptuální schéma úlohy

relační implementace

V relačním prostředí jsou entity reálného světa zobrazovány relačními tabulkami. Proto v návrhu musí existovat tabulky pro osoby, firmy a kontrakty vyplývající přímo z konceptuálního schéma úlohy.

Adresy osob i firem je možné uložit do jediné tabulky a odkazovat se na ně prostřednictvím vzdálených klíčů. Zobrazení ředitele lze realizovat pomocí vzdáleného klíče od firem na osoby. Pro zaměstnance je nutno vytvořit vazební tabulku mezi osobou a firmou. Vazba od osob ke kontraktům může být vytvořena prostřednictvím dvojice vazebních tabulek osoba-kontrakt a firma-kontrakt nebo tabulkou klientů, která sjednocuje osoby i firmy a umožní vytvořit vazbu klient-kontrakt. V návrhu bude použito druhé řešení, které ve skutečnosti napodobuje způsob, kterým se pracuje v objektovém návrhu. Přesto povede zápis dotazu v obou případech shodně na použití klausule *union all*, protože v relačním schématu neexistuje polymorfismus.



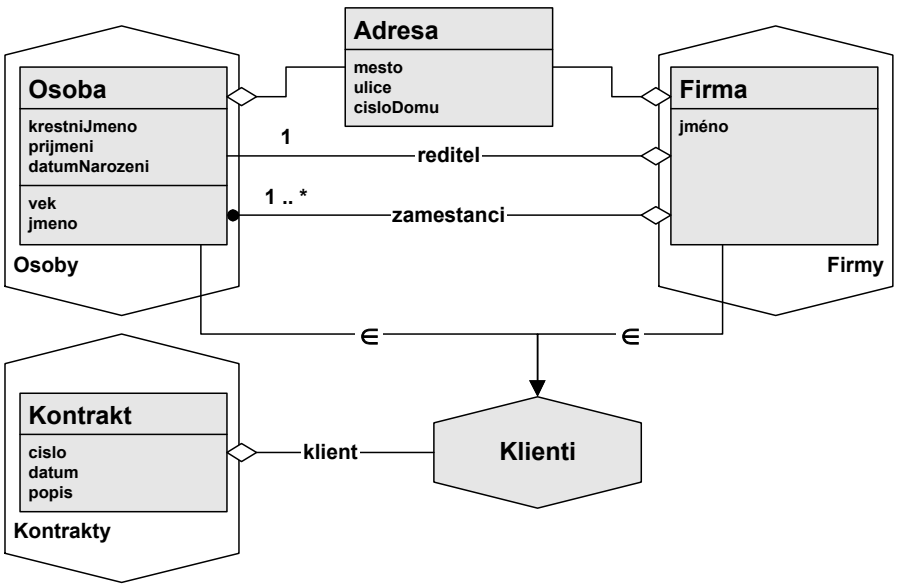
obr. č. 2. - relační implementace databáze

objektová implementace

Objektová implementace na rozdíl od relační dovoluje přímo zobrazit modelovanou realitu, aniž bychom museli úlohu nějak uměle strukturovat do soustavy provázaných tabulek. Na druhou stranu to je ale složitější proces, protože můžeme navrhovat nejen *třídy objektů*, které do značné míry odpovídají konceptu relační tabulky, ale také *množiny objektů*, které mohou obsahovat jako prvky objekty z různých tříd.

Kromě toho budeme také objekty mezi sebou skládat - například objekt *Kontrakt* může přímo obsahovat jako svoji datovou složku objekt *Klient* - tedy ne, že by obsahoval hodnotu nějakého primárního klíče odkazujícího se do množiny *Klientů*, ale přímé skládání dat. Dále můžeme některé atributy implementovat jako metody, které svoji hodnotu vypočítají. *Věk osoby* bude počítán z jejího data narození a nebude třeba do databáze někam ukládat jeho číselnou hodnotu. Rovněž tak *jméno osoby* bude poskytovat metoda, která jej složí z hodnot *křestního jména* a *příjmení*.

Jeden z možných výsledných návrhů ukazuje obrázek č. 3. Diagram dodržuje standard UML, ale navíc je zde zaveden symbol pro množinu objektů (šestihran) rozlišený od symbolu pro třídu objektů (obdélník). Ke třídám budou ještě zavedena množiny všech instancí s názvy *Osoby*, *Firmy*, *Kontrakty* a dále také množina *Klienti* obsahující objekty z obou tříd *Osoba* i *Firma*.



obr. č. 3. - objektová implementace databáze

dotazy nad implementovanými databázemi

K porovnání zápisu dotazů poslouží následující dotazy:

- najdi kontrakty firem nad 25 zaměstnanců,
- najdi zákazníky z Prahy,
- najdi firmy jejichž ředitel se jmenuje Novák,
- najdi firmy, jejichž nějaký zaměstnanec se jmenuje Novák,
- najdi klienty starší 50 let.

Nyní si rozdílnost obou implementací porovnejme v následující tabulce č. 1:

dotaz v SQL v relační databázi	dotaz v Gemstone Smalltalku v objektové databázi
1. Najdi kontrakty firem majících více než 25 zaměstanců select kon.* from kontrakt kon, firma f where kon.klient_id = f.klient_id and f.id in (select firma_id from zamestnanec group by firma_id having count (firma_id)>25)	Kontrakty select: {:k | k.klient.zamestanci.size > 25}.
2. Najdi klienty (osoby i firmy) z Prahy select kli.*, f.jmeno name from klient kli, firma f, adresa a where kli.id = f.klient_id and a.id = f.adresa_id and a.mesto = 'praha' union all select kli.*, o.prijmeni name from klient kli, osoba o, adresa a where kli.id = o.klient_id and a.id = o.adresa_id and a.mesto = 'praha'	Klienti select: {:k | k.adresa.mesto = 'Praha'}.
3. Najdi firmy, jejichž ředitel se jmenuje Novák. select f.* from firma f, osoba o where f.reditel = o.id and o.prijmeni = 'novak'	Firmy select: {:f | f.reditel.prijmeni = 'Novak'}.
4. Najdi firmy, jejichž nějaký zaměstnanec se jmenuje Novák. select distinct f.* from firma f, zamestnanec z, osoba o where z.firma_id = f.id and z.osoba_id = o.id and o.prijmeni = 'novak'	Firmy select: {:f | f.zamestanci.*.prijmeni = 'Novak'}.
5. Najdi klienty (jen osoby) starší 50 let. select * from osoba where klient_id is not null and sysdate-datum_narozeni>(50*365.25)	Osoby select: {:o | o.vek > 50}.

tab. č. 1. - porovnání dotazů

Jak názorně vyplývá ze zápisu dotazů, tak přednosti objektového datového modelu – kromě kratšího a přehlednějšího zápisu – jsou:

a) Díky skládání objektů nemusíme používat operaci relačního spojení pomocí klíčů[1] (dotazy 1., 2., 3. a 4.).

b) Díky polymorfismu objektů, který dovoluje držet všechny klienty v jedné množině, nemusíme v dotazech rozlišovat, s jakým typem objektů pracujeme. (dotaz č. 2.)

c) Díky implementaci atributů pomocí metod přirozenější dotazování (dotaz č. 5. na věk a dotaz č. 1. na velikost podmnožiny).

Diskuse

Na základě posouzení rozdílů datového modelu relační databáze a objektové databáze bylo prokázáno, že nelze jednoduše konvertovat relační databázi do objektového prostředí. Při takovém převodu se nemohou projevit přínosy objektového paradigmatu. Výhodné vlastnosti objektové databázové aplikace, kterou ocení její uživatelé, jsou totiž vyváženy potřebou specifických znalostí nezbytných ke tvorbě takové aplikace.

Problematika konverze relačních databází a relačních datových modelů do objektových není zatím příliš prozkoumána. V praxi se používá přístup, kdy je databáze navržena celá znovu. To je jedna z oblastí výzkumu, které se budeme na katedře rozhodně dále věnovat.

Závěry

Problematika objektových databází je složitá a její pochopení nezbytně vyžaduje praktické zkušenosti s nástrojem. Je samozřejmě třeba mít čistě objektově orientovanou databázi a ne hybridní databázi, které jsou často označovány jako „objektově relační“. Je tomu tak proto, že u takových systémů zůstává základním médiem provázaná soustava relačních tabulek pomocí klíčů a objektové paradigma je tu omezeno jen na vybrané datové typy uvnitř tabulek.

Velkým problémem je tu také cena a dostupnost objektových nástrojů, protože jejich dodavatelé na rozdíl například od firmy Oracle téměř vůbec neposkytují zvýhodněné speciální licence pro účely výuky a výzkumu na univerzitách.

Z obou uvedených důvodů není snadné na vysokých školách zavádět prakticky orientovanou výuku objektových databázových technologií.

Objektové paradigma má blíže k uvažování člověka než relační. Objektový návrh přirozeněji popisuje svůj vzor v realitě a umožní lépe pochopit a zpracovat modelovaný problém. Na výuce na vysokých školách má podle nás přes všechny problémy nezastupitelnou úlohu, protože právě zde vzniká nová generace budoucích kvalifikovaných tvůrců softwaru – absolventů informatických oborů.

Literatura

[1] Abadi M., Cardelli L.: *A Theory of Objects*, Springer 1996, ISBN 0-387-94775-2

[2] Ambler, Scott W.: *More Process Patterns - Delivering Large-Scale Systems Using OO Technology*, Cambridge University Press - Managing Object Technology Series 1999, ISBN 0-521-65262-6

[3] Ambler, Scott W.: *Process Patterns - Buiding Large-Scale Systems Using OO Technology*, Cambridge University Press - Managing Object Technology Series 1998, ISBN 0-521-64568-9

[1] Pozorný čtenář jistě zaregistroval, že zde uvedená objektová implementace nepoužívá žádné primární klíče. To, že nemusíme používat primární klíče, pokud je nepotřebujeme, je také jeden z důležitých rysů čistě objektových databází.

[4] Blaha M., Premerlani W.: *Object Oriented Modeling and Design for Database Applications*, Prentice Hall 1998, ISBN 0-13-123829-9

[5] Catell R.G.G. et al.: *The Object Database Standard – ODMG2.0*, Morgan Kaufman 1997, ISBN 1-55860-463-4

[6] Date C.J., *An Introduction to Database Systems (6th Edition)*, 1995, Addison-Wesley, ISBN: 0-201-82458-2

[7] Kroha P.: *Objects and Databases*, McGraw Hill 1993 ISBN 0-07-707790-3

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

[9] Taylor, David A.: *Business Engineering with Object Technology*, John Wiley 1995 ISBN: 0471045217

[10] Oracle Corporation [online]. [cit. 2003-07-05]. Dostupný z www: < http://www.oracle.com >.

[11] GemStone Systems [online]. [cit. 2003-07-05]. Dostupný z www: < http://www.gemstone.com >.

Adresa autorů:
Tomáš Doskočil, Vojtěch Merunka, Katedra informačního inženýrství,
email: {doskocil,merunka}@pef.czu.cz, PEF ČZU v Praze