Některé souvislosti mezi převodem objektově orientovaného modelu do modelu entitně-relačního.

Martin Molhanec

1. Úvod

Tento příspěvek navazuje na příspěvek z loňského roku, který byl zveřejněn ve sborníku PROGRAMOVÁNÍ '93 pod názvem "Souvislosti mezi objektově orientovanými a relačními metodologiemi návrhu programů" a snaží se myšlenky v něm uvedené dále rozvíjet.

Ve výše uvedeném příspěvku byly popsány některé zásadní úvahy o souvislosti mezi OOM (objektově orientovaný model) a ERM (entitně relační model). Z nich nejdůležitější si nyní stručně zopakujeme.

Počítačové systémy používají pro uschování proměnných a objektů, které existují během výpočtu paměť typu RAM. Takové objekty však existují pouze po dobu výpočtu (běhu programu). Skutečná data informačních systémů je nutné však uchovávat i po dobu, kdy informační systém není funkční (například po dobu, kdy je počítač vypnutý).

Z programovacího hlediska se ukládání takových údajů děje jejich zápisem do hromadných pamětí (pevný a pružný disk, magnetická páska ap.). Objekty a proměnné, které existují trvale ve výše uvedeném smyslu se nazývají **persistentní**.

Podle Coad-Yourdona existují dvě základní možnosti implementace persistentních objektů:

Relačním databázovým systémem

výhody:

Existuje celá řada výkonných a komerčně používaných RDBMS (Relational Database Management System). Jsou dostupné pro všechny používané počítačové systémy. Jsou dostupné v různých cenových kategoriích. Existuje pro ně standardní dotazovací jazyk SQL.

nevýhody:

Nejsou vhodné pro ukládání složitých strukturovaných dat. Jsou nositeli pouze datové části objektu (atributy objektu), nikoliv jeho části funkční (metody objektu). Integrita dat je zajištěna často pouze nepřímo.

Objektově orientovaným databázovým systémem

výhody:

OODBMS (Object Oriented DBMS) je přirozeným prostředkem pro uchování persistentních objektů. Je efektivnější pro uchování objektů nežli RDBMS.

nevýhody:

Nejsou běžně rozšířenými a standardizovanými prostředky. Jsou dostupné pouze na některých platformách a převážně ve vyšších cenových kategoriích.

Přirozeným prostředím pro uschování persistentních objektů jsou tedy **OODBMS (Object Oriented DBMS)**. Přes mnohé výhody OODBMS, například lepší uschování komplexních datových struktur a lepší zajištění datové integrity nejsou v současné době příliš rozšířeny a zdaleka se nedá o nich mluvit, jako o průmyslovém standardu. Proto se pro uschovávání objektů v současné době stále používá **RDBMS (Relation DBMS)**, které jsou v současné době průmyslovým standardem, existuje propracovaná teorie i praxe jejich používání a existuje standardní dotazovací jazyk **SQL (Structured Query Language)**, který se používá pro přístup k těmtom relačním databázovým systémům.

Vzniká tedy určité paradigma, kdy z hlediska analýzy je výhodné používat objektově orientované metody, zejména tam, kde je při implementaci použit **OOPL (Object Oriented Programming Language)**, zatímco ve fázi implementace persitence objektů jsou nutné znalosti datových metodologií, zejména relačních.

Podle [Daniels-93] existuje "konceptuální mezera" ("conceptual gap") mezi OOPL a RDBMS. Tamtéž jsou předloženy čtyři možnosti, jak výše zmíněný nedostatek řešit.

"Jump the gap"
Vytvořit mapování, mezi "objekty" a "entitami". Jedná se o metodu, kdy pro každý objekt je vytvořena jemu odpovídající položka v relační databázi. Nevýhodou této metody, je její určitá pracnost, protože je vesměs prováděna ručně.

"Build a bridge"
Je vytvořen OO front-end pro RDBMS, která se pro své okolí tváří jako OODBMS. Tento OO front-end je přirozeně dostupný z OOPL. Nevýhodou je malá optimalizace konstruktů a tím nízká výkonnost.

"Narrow the gap"
Tato strategie, využívaná výrobci RDBMS, spočívá v rozšíření možností RDBMS tak, aby zahrnovaly nebo imitovaly rysy OODBMS. Výhodou je kompatibilita a tradice z používanými RDBMS. Nevýhodou nižší výkonnost oproti skutečným OODBMS.

"Fill the gap"
Poslední strategie používá pro ukládání persistentních objektů skutečné OODBMS. Výhodou je vysoká optimalizace ukládání objektů. Nevýhodou jejich zatím malé komerční rozšíření, nekompatibilita a neexistence standardů. (Touto strategii vlastně náš rozpor zcela vymizí.)

V této své přednášce popíši základní aspekty **metody mapování**, jejíž cílem je učinit mapování objektů do entit **rutinní** záležitostí a tím zabránit vzniku chyb. Je však nutné si uvědomit, že se nebudeme zabývat pouze mapováním objektů do entit, ale i mapováním vztahů mezi objekty do vztahů mezi entitami. Vytvoření takové metodologie vede ke *zautomatizování* celého procesu: *analýza, design a implementace* a tedy k podstatnému zmenšení chyb, zvětšení stability systému, urychlení práce při programování systému i lepšímu využití již napsaných funkcí a procedur.

2.1. Základní pravidla pro mapování objektů do relačního prostředí.

Pro každou *třídu (class)* OO modelu je vytvořena jedna *tabulka (entity)* ER modelu. Jeden *objekt (instance)* dané třídy je tedy uložen do jedné řádky nebo *záznamu* (row nebo *record*) jemu příslušné tabulky. Každý *atribut* objektu se ukládá do *položky* (field) tabulky. Každá tabulka obsahuje jedinečný *primární klíč* sloužící pro vzájemné rozlišení (sebeidentifikaci) objektů.

2.2. Základní pravidla mapování vztahů OOM do vztahů ERM.

Problematika mapování vztahů mezi objekty (dle Coad-Yourdona) do vztahů mezi entitami je však poněkud složitější. Coad-Yourdon definují ve svých knihách [Coad/Yourdon-91a] a [Coad/Yourdon-91b] několik základních vztahů mezi objekty. Jedná se o vztahy *GEN_SPEC, WHOLE_PART a INST_CONNECTION*. Pro mapování každého vztahu platí specifická pravidla.

2.2.1. Mapování vztahu GEN-SPEC.

Vztah GEN-SPEC (generalizace-specializace) podle Coad-Yourdona je vztah dědičnosti (inheritance) mezi objekty. Existují tři základní způsoby mapování.

- Každá třída je mapována do jedné entity. Entita obsahuje nejen atributy dané třídy, ale i atributy třídy otce (přesněji řečeny všechny zděděné atributy). Tento způsob mapování však není výhodný. Přestože poskytuje snadnou práci s jednotlivými objekty, ztrácí se při tomto způsobu mapování informace o vztahu dědičnosti mezi jednotlivými objekty !

- Druhá metoda spočívá ve vytvoření jediné entity pro všechny třídy spojené vztahem dědičnosti, která obsahuje všechny atributy všech zúčastněných tříd. Tento způsob, je například zmiňován ve článku [Loomis-93], je však výhodný pouze tam, kde se jednotlivé specializace od sebe navzájem liší pouze malým počtem atributů.

- Třetí metoda, kterou považuji za metodicky správnou, je metoda, při které je každá třída zúčastňující se vztahu dědičnosti nahrazena jednou entitou, která obsahuje pouze atributy, definované pro danou třídu. Všimněme si však několika následujících problémů. Je především nutné pro všechny entity zúčastňující se vztahu dědičnosti definovat společný jedinečný primární klíč. Tento klíč bude v entitách potomcích současně i klíčem sekundárním. Navíc je výhodné v entitách otce definovat speciální atribut, tzv. category descriptor, který bude obsahovat název entity potomka.

2.2.2. Mapování vztahu WHOLE-PART

Vztah WHOLE-PART (celek-část) podle Coad-Yourdona zachycuje skutečnost, že nějaký objekt se skládá, nebo mu přináleží nějaké jiné objekty.

Vztah WHOLE-PART se mapuje do vztahu 0 : M nebo 1 : M mezi entitami. Při mapování se přenáší primární klíč celku do sekundárního klíče části.

2.2.3. Mapování vztahu INST-CONNECTION,

Podle Coad-Yourdona se jedná vztah mezi nezávislými objekty, který je založen na vzájemné souvislosti mezi nimi.

Pokud je vztah INST-CONNECTION typu 0 : M nebo 1 : M platí pro jeho mapování stejná pravidla, jako pro vztah WHOLE-PART. Pokud se jedná o vztah M : N je nutné vztah rozdělit do dvou vztahů typu 1 : M a N : 1 a vytvořit novou "vazební entitu".

2.3. Metoda pro převod OOM do ERM.

Převod OO modelu (analytického) do modelu ER (implementačního) se musí vyrovnat s následujícími problémy.

- Pro všechny entity ERM definovat primární klíče. Tady je dobré si uvědomit, že OOM na rozdíl od ERM existenci primárních klíčů nevyžaduje. Tato činnost není zcela triviální, pokud ji chceme zautomatizovat. Musíme totiž primární klíč přidělit nejprve všem entitám ze kterých se importují sekundární klíče.

  Je tedy nutné přidělit primární klíč nejprve všem otcům, všem celkům a všem ostatním entitám na straně 1 nebo 0. Tato úloha vede k zajímavým teoretickým úvahám o tom, za jakých podmínek je tato úloha vždy řešitelná a jakým algoritmem.
- Pro všechny otce definovat atribut "category descriptor", který bude obsahovat název třídy potomka.
- Importovat ve všech vztazích sekundární kliče. Pochopitelně entity ze kterých se kliče importují musí mít již přidělen klíč primární.
- Roztrhnout vztahy M : N na vztah 1 : M a vztah N : 1. Zároveň vytvořit novou spojovací entitu, importovat do ní sekundární klíče a definovat klíč primární.

Závěr

Všechny výše provedené úvahy nejsou pouze teoretickými hříčkami, ale mají velký praktický dopad. Uvažování v **objektech** je výhodnější nežli uvažování pouze v **relačních tabulkách**. A metoda, jak přejít z jednoho modelu do druhého usnadňuje implementaci.

V rámci své současné vědecké práce jsem navrhl pro objektově orientovaný a entitně relační model vhodnou **data dictionary** a algoritmus pro převod data dictionary objektového modelu do data dictionary entitně relačního modelu. Další částí mé práce je pak automatická generace příkazů v jazyce **SQL** či v jazyce **xBase** na základě informací obsažených ve výše uvedené data dictionary.

Celá práce pak může sloužit, jako základ objektově orientovaného **CASE** generujícího příkazy **relačních databázových systémů**. Cílem použití takové prostředku je však pochopitelně **snadné a bezpečné** programování databázových aplikací, které jsou jádrem všech velkých informačních systémů.

Literatura:

[Coad/Yourdon-91a]
Peter Coad and Edward Yourdon, Object Oriented Analysis, Yourdon Press, 1991
[Coad/Yourdon-91b]
Peter Coad and Edward Yourdon, Object Oriented Design, Yourdon Press, 1991
[Daniels-93]
John Daniels, Strategies for sharing objects in distributed systems, JOOP, January 1993
[Loomis-93]
Mary E S. Loomis, Object programming and database management, JOOP, May 1993

Autor :

Ing. Martin Molhanec
ČVUT-FEL
Technická 2
166 27 PRAHA 6, Dejvice
tel.: (+422) 332 2118