Algoritmus převodu schématu objektově orientovaného modelu do schématu entitně-relačního modelu

Martin Molhanec

ČVUT-FEL Praha, Katedra elektrotechnologie, Technická 2, 166 27 PRAHA 6, Česká republika

Abstrakt
Tématem tohoto příspěvku je popis algoritmu pro převod schématu objektově orientovaného modelu, který je výsledkem objektově orientované analýzy do schématu modelu entitně-relačního, který je vhodný pro standardní implementaci databázového systému prostřednictvím klasického relačního DBMS.

Úvod

Tento příspěvek navazuje na mé předchozí příspěvky [1] a [2], se kterými jsem měl možnost na této konferenci vystoupit v předchozích letech. Dále na můj příspěvek [3], který byl uveden na konferenci DATASEM'94. Podobně jako tyto příspěvky je tématem i tohoto problematika užití *objektově orientované analýzy* (dále jen OOA) při standardní *relační databázové implementaci* (dále jen RDI).

Ve výše uvedených příspěvcích, jsem vysvětlil souvislost mezi OOA a RDI a naznačil způsob přechodu - transformace schématu objektově orientovaného modelu do schématu modelu entitně-relačního. Dále jsem pro oba modely definoval datové slovníky, které jsou dostatečné pro jejich využití.

Algoritmus převodu objektově orientovaného schématu do schématu relačního.

Jak již bylo ve výše zmíněných předchozích příspěvcích řečeno, není algoritmus převodu *objektově orientovaného schématu* do *schématu entitně-relačního* zcela triviální. K tomu, aby naše ER schéma bylo pro praktické řešení použitelné, musí býti pro úspěšný převod přidány následující informace.

- Definice *PRIMÁRNÍCH KLÍČŮ* (dále jen PK), všude tam, kde je potřebné za primární klíč použít, některý reálný atribut již ve schématu obsažený. Přitom musí být dovoleno za součást PK definovat i některý z importovaných *CIZÍCH KLÍČŮ* (dále jen FK).
- Definice *ALTERNATIVNÍCH KLÍČŮ* (dále jen AK), včetně možnosti definovat za součást AK i některý z importovaných FK.

Dále jsem při důkladných rozborech celé problematiky dospěl k základní podmínce, kterou musí naše OO schéma splňovat, aby byl převod úspěšný[1].

- *Ve schématu se nesmí vyskytovat cyklické importování FK.*

Výše uvedená podmínka je v reálných modelech pravděpodobně splnitelná, pro lepší srozumitelnost je jí možné nahradit následujícími podmínkami.

- *Žádný ze předků libovolné entity nesmí být současně jejím potomkem.*
- *Celek nesmí být současně svojí částí.*
- *Potomek nemůže být celkem pro své předky.*

Je vidět, že výše uvedená tvrzení jsou zcela *triviální* a dobře odpovídají obrazu reálného světa, tak jak ho běžně chápeme. Dále zavedeme pro zjednodušení celého problému další důležitou podmínku.

- *Při vícenásobné dědičnosti mají všichni předci jednoho společného prapředka.*

Tato podmínka není nikterak omezující. Jednak dokonce existují některé objektově orientované přístupy, které vícenásobnou dědičnost vůbec nedovolují[2]. Navíc by bylo možné v případě, kdy bychom při vícenásobné dědičnosti společného prapředka neměli k dispozici, ho vždy vytvořit, jako jakéhosi praprapředka, který by ve skutečnosti neobsahoval žádné atributy a jeho účel by byl pouze formální[3].

Algoritmus převodu schématu objektově orientovaného modelu (dále jen OOM) do entitně-relačního modelu (dále jen ERM) bude sekvence následujících činností v uvedeném pořadí.

1. Všem předkům bude přidělen nový atribut s názvem *category descriptor*[4] (dále jen CD).
2. Všechny vazby M : N budou dekomponovány na dvě vazby 1 : M a N : 1 a bude vytvořena nová spojovací entita.
3. Bude provedena volba primárních klíčů a import cizích klíčů.
4. Budou zpracovány alternativní klíče.

Nejzávažnější částí postupu algoritmu je bod 3, ve kterém se provádí zpracování primárních klíčů a současně import klíčů sekundárních. Problém je následující.

- Protože součástí PK mohou být klíče FK, je třeba před přidělením PK provést import FK.
- Všechny entity ze kterých se importuje FK musí mít přiděleny PK.

Dvě výše uvedená tvrzení, která musí býti splněna, aby náš algoritmus měl řešení, vedou k již výše zmíněné větě o zákazu cyklického importu FK. Pro vyřešení výše uvedeného

---

[1] Podobné úvahy například též v [4].
[2] Například objektová implementace jazyka *Turbo Pascal* od firmy *Borland* nedovoluje vícenásobnou dědičnost.
[3] Například v jazyku *SmallTalk* se všechny třídy odvozují od jedné společné supertřídy.
[4] Zavedení tohoto atributu není nutné, ale je výhodné.

problému jsem zvolil rekurzivní algoritmus. Celý algoritmus je dále popsán v jednoduchém pseudojazyku[5].

```
// ALGORITMUS PŘEVODU OOM do ERM.
PROGRAM Gentbl

// I. Vytvoření CD pro všechny předky.
forall CLASS from CLATBL where exist any CHILD
        generate new ATTRIBUTE for CLASS with name CD
endfor

// II. Rozpojení všech vazeb M : N
forall CONN from CONNTBL where CONN
        is INST-CONN and CONN.LMAX > 1 and CONN.RMAX > 1
                generate new CLASS NEWCLASS with name build from
                        CONN.LCLAID -> CLASS.CLANAME and
                        CONN.RCLAID -> CLASS.CLANAME
                mark CONN.CC to True
                generate new CONNECTION between CONN.LCAID -> CLASS and
                NEWCLASS from CONN
                generate new CONNECTION between NEWCLASS and CONN.RCLAID
                        -> CLASS from CONN
endfor

// III. Generace PK a FK.
forall CLASS from CLATBL do CLASS.makePKFK()
endfor
// rekurzivní metoda makePKFK() je uvedena zvlášť

// IV. Generace AK
forall AKEY do
        forall AKMEMBER's do
        // případ, kdy je člen AK pouze atribut
        case AKMEMBER is ATTRIBUT_NAME
                makeAK()
        // případ, kdy je člen AK FK
        case AKMEMBER is FOREIGN_KEY_CLASS_NAME
                forall ATT from FOREIGN_KEY_CLASS_NAME
                        make AK()
                endfor // members of foreign key
        endcase // akmember
        endfor // members of akey
endfor // akeys

ENDPROGRAM
// konec programu - algoritmu
```

[5] Pro popis algoritmu byl zvolen jednoduchý pseudojazyk, srozumitelný každému čtenáři, který je obeznámen s jazyky C, C++ a SQL.

```
// Metoda make PKFK(), řešící  PK a FK
METHOD makePKFK()

// pokud je již PK vyřešen končíme
if SELF.PKEY == True
        Return
endif

// pokud jsem potomek musíme nejprve vyřešit všechny předky
if SELF:HaveAnyParent()
        forall CLASS from CLATBL where CLASS:IsMyParent() do
                CLASS:makePKFK()
                CLASS:ImportPKFK()
        endfor
        SELF:PKEY := True
endif

// pokud jsem částí musíme nejprve vyřešit své celky
if SELF:HaveAnyOwner()
        forall CLASS from CLATBL where CLASS:IsMyOwner() do
                CLASS:makePKFK()
                CLASS:ImportPKFK()
        endfor
endif

//  tato část se vykoná pokud je PK explicitní
if SELF:HaveExplicitPK()
        forall PKEY do
                where PKEY.CLASS == SELF
                        // zpracování explicitních PK
                        // je podobné již výše uvedené
                        // části pseudokódu pro zpracování
                        // alternativních klíčů
                SELF:PKEY := True
        endfor
endif

// pokud PK není explicitní a nebyl doposud vyřešen
// přidělí náš vlastní interní PK
if .not. SELF:PKEY
        SELF:GenerateUniqueID()
        SELF:PKEY := True
endif

ENDMETHOD
// konec metody makePKFK()

// Další metody, např. importPKFK() a další nejsou v tomto stručném popisu
// algoritmu uvedeny.
```

Popis způsobu prototypové algoritmizace převodu OO schématu do schématu ER.

Způsob realizace vychází z těchto předpokladů. Prostředí, ve kterém bude náš prototyp pracovat bude OS MS DOS. Důvodem byl požadavek použití co nejběžnějšího operačního prostředí. Vlastní datový slovník (dále jen DD) je realizován jako tabulky relačního databázového systému typu xBase. Jedná se tedy o soubory typu .DBF. Struktura DD byla popsána v mém předcházejícím příspěvku.

Pro ověření algoritmu je nutné mít k dispozici DD popisující schéma v OOM a doplňující informace o primárních a alternativních klíčích. Vzhledem k tomu, že nebylo k dispozici žádné vhodné grafické prostředí (CASE systém) pro vytvoření OO DD z diagramů dle Coad-Yourdonovy metodologie, rozhodl jsem se definovat jednoduchý objektově orientovaný prototypový jazyk (OOPJ a vytvořit jeho pseudopřekladač, pomocí kterého je možné zkonstruovat žádanou OO DD.

Pro realizaci pseudopřekladače OOPL byl zvolen programovací jazyk Clipper 5.2, který je součástí stejnojmenného relačního databázového systému typu xBase. Při implementaci bylo dále výhodně využito objektové rozšíření jazyka Clipper Class(y) 2.1. Podrobnější popis mnou navrženého OOPJ a realizace jeho pseudopřekladače není obsahem tohoto příspěvku.

Některá omezení navrženého algoritmu.

Vytvořený algoritmus transformující OO schéma do schématu ER má některá omezení.

Omezení vyplývající z použitého Coad-Yourdonova modelu:

- Nejsou dovoleny vztahy s atributy.
- Nejsou dovoleny vícenásobné vztahy.

A omezení vyplývající z navrženého algoritmu:

- U vícenásobné dědičnosti je předpokládán společný prapředek.

Další omezení pak vyplývají z konkrétní realizace výše uvedeného algoritmu a nejsou v tomto příspěvku uvedena.

Závěr

Doufám, že tento příspěvek poskytl základní informace o řešení algoritmu pro převod objektově orientovaného schématu do schématu entitně-relačního. Jsem si vědom určitých problémů, které bude mít čtenář, který neměl možnost si přečíst mé předchozí příspěvky zabývající se touto problematikou. Doufám, že tyto problémy se aspoň částečně odstraní při vlastní prezentaci tohoto příspěvku.

Možná může být také položena otázka, k čemu je možné výše uvedený algoritmus použít. Odpověď je jednoduchá - výše uvedený algoritmus je důležitou součástí každého systému CASE, který podporuje na straně jedné objektově orientované metodologie a na straně druhé je schopen generovat kód pro definici relačního databázového systému. Tento kód je totiž nutné generovat z informací uložených v ER DD, který je výsledkem právě zde uváděného algoritmu.

Algoritmus zde uvedený je pouze nástinem algoritmu skutečného. Nicméně o jeho funkčnosti svědčí jeho úspěšná prototypová realizace. Závěrem se chci též zmínit o skutečnosti, že problematika objektově orientované analýzy a entitně-relační implementace včetně zde diskutovaného algoritmu byla součástí mé úspěšně obhájené kandidátské disertační práce [5], kde jsou též uvedeny další podrobnosti k této, dle mého názoru, velice zajímavé a podnětné problematice.

Literatura

1. Mohlanec,M.: Souvislosti mezi objektově orientovanými a relačními metodologiemi návrhu programů. Programování'93, Ostrava 1993
2. Molhanec,M.: Některé souvislosti mezi převodem objektově orientovaného modelu do modelu entitně-relačního. Programování'94, Ostrava 1994
3. Molhanec,M.: Realizace datového slovníku objektově orientovaného modelu. Datasem'94, Brno 1994
4. Pokorný,J., Halaška,I.: Generování databázového schématu z E-R modelu do SQL. Datasem'93, Brno 1993
5. Molhanec,M.: Metodologie projektování rozsáhlých technologických systémů. Kandidátská disertační práce, ČVUT-FEL, Praha 1994