Souvislosti mezi objektově orientovanými a relačními metodologiemi návrhu programů

Martin Molhanec

1 Úvod

Tento příspěvek se zabývá, jak je již patrné z jeho názvu, problematikou souvislostí mezi objektově orientovanou analýzou systému a jeho implementací v relačním databázovém prostředí. Není to téma nikterak odtažité a v odborné literatuře je možné vyhledat na toto téma celou řadu článků.

Existuje celá řada důvodů pro používání objektově orientované analýzy. Objektově orientovaná analýza umožňuje daleko lepší pohled na zkoumaný systém, protože na rozdíl od datových nebo funkčních analýz zkoumá současně, jak datovou, tak funkční stránku věci.

V současné době existuje celá řada komerčně úspěšných objektově orientovaných jazyků, jako například Smalltalk, CLOS, C++, Object C nebo Object Pascal. A pro další jazyky se objektová rozšíření připravují, například Cobol, Modula, Ada nebo Fortran.

Ovšem většina těchto jazyků nemá v sobě zahrnuty možnosti existence *persistentních* objektů. Tedy možnost, aby takový objekt existoval i mimo dobu běhu programu, prostě většina těchto jazyků v sobě neobsahuje implementaci objektově orientovaných databází.

Přestože jsou v současné době objektově orientované databáze již komerčně dostupné, můžeme si uvést například OBJECTORY od Objective Systems nebo VERSANT od Object Technology, nejsou zatím běžně rozšířeným prostředkem.

Naopak se pro ukládání objektů stále udržuje konzervativní používání **relačních databází**, které jsou v současnosti průmyslovým standardem.

Mezi další výhody jejich použití dále patří používání strukturovaného dotazovacího jazyka SQL (**structured query language**), který je mezinárodně standardizován.

Navíc se stále více přes své četné nevýhody šíří používání databázových jazyků typu xBase, které postupně získávají vlastnosti vhodné i pro profesionální používání a jsou objektem zájmu významných softwareových firem, jako například Microsoft, Borland a Computer Associated.

Výsledkem všech výše uvedených trendů je potřeba řešit problematiku současného použití **objektově orientovaných (OO – object oriented)** programovacích jazyků a uživatelských prostředí a **relačně orientovaných** standardních databázových systémů. Tedy problematiku mezi objektově orientovanou analýzou a klasicky orientovanou implementací.

Jedná se tedy o použití objektově orientovaných metod ve fázi analýzy informačního systému a využití komerčních relačních databázových systémů při implementaci informačního systému.

2 Souvislost mezi entitně–relačním modelem a objektově orientovaným modelem.

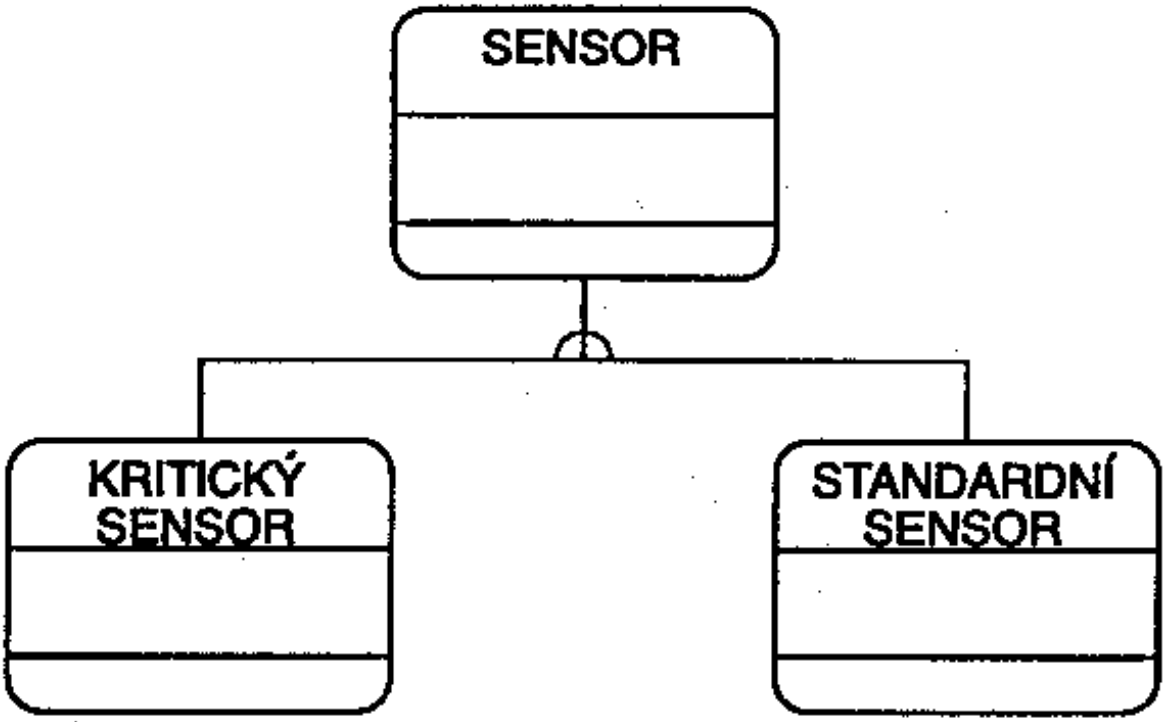
Přestože i v ER (entitně–relačním) modelu někteří autoři zavádějí nebo naznačují možnost hierarchických vztahů, jiní autoři se vyhýbají rozpoznávání takovýchto závislostí. Problém spočívá v tom, že samotné relační databáze nikterak neposkytují možnost rozlišení vztahu otec–potomek, například při vztahu mezi entitami ZÁKAZNÍK <—> NEPLATÍCÍ_ZÁKAZNÍK a nezávislého vztahu mezi dvěma entitami, například ZÁKAZNÍK <—> OBJEDNÁVKA.

Oproti tomu OO (objektově orientované) modely jsou postaveny na rozlišování různých vztahů a jejich významů mezi objekty. Navíc vztah otec–potomek je vztahem dědičnosti, která jednou ze základních vlastností objektově orientovaných systémů. Je tedy žádoucí přesně definovat souvislosti mezi objekty používanými v OO modelech a mezi entitami používanými v ER modelech. Naše metoda spočívá v použití OO notace podle Coad–Yourdona, a jejím převedení do odpovídajících ER diagramů, které budou sloužit pro faktickou definici relační databáze. Je nutné ovšem dodat, že tento postup při použití OO DBMS nemá smysl, protože OO DBMS je možné definovat přímo z výsledků OO analýzy a návrhu. Coad–Yourdon definují ve svých knihách [CYOOA1990 a CYOOD1990] několik základních vztahů mezi objekty.

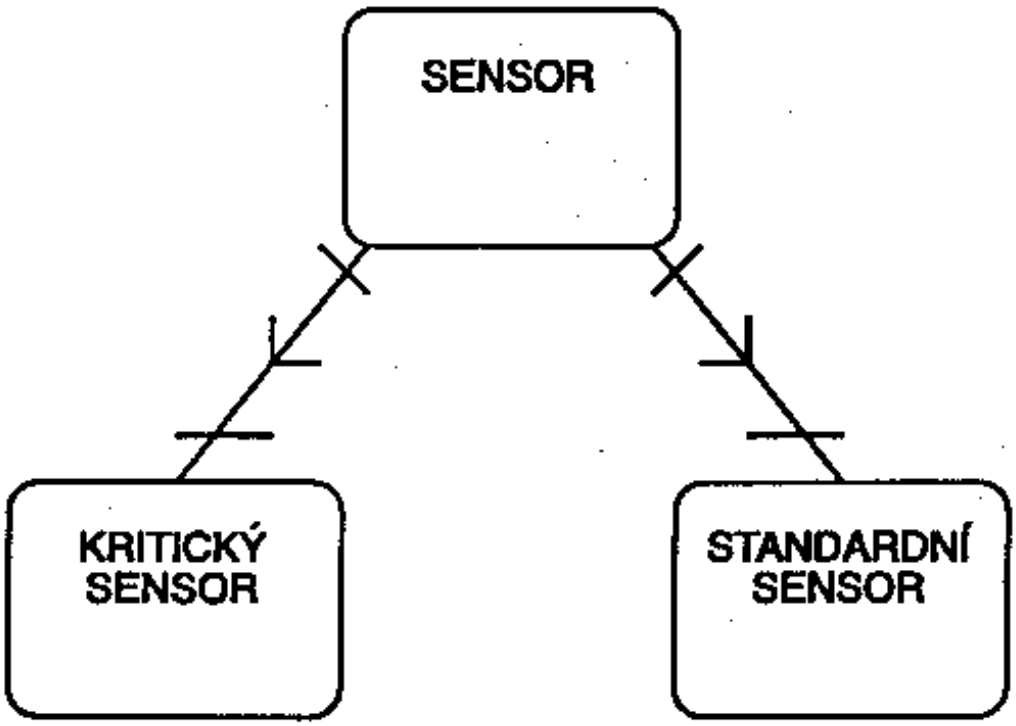
GEN–SPEC (generalizace – specializace)

Vztah GEN–SPEC mezi dvěma objekty je vyjádřením vztahu dědičnosti (otec–potomek) mezi těmito objekty. Mějme objekty SENSOR, KRITICKÝ_SENSOR a STANDARDNÍ_SENSOR a mějme je vyjádřeny ve standardní notaci dle Coad–Yourdona (obr. 1).

K tomuto vyjádření lze nalézt odpovídající vyjádření v notaci ER diagramů. Všimněme si však několika následujících problémů. Je především nutné definovat klíčový atribut (interní identifikátor relace). Na rozdíl od vyjádření v OO notaci dle Coad–Yourdona, kde vztah GEN–SPEC prostě existuje; je nutné, v notaci ER diagramů, kde je tento vztah vyjádřen relací, definovat který klíč je touto relací přenášen ! Směr relace je od entity, která odpovídá objektu otce k entitám, které odpovídají objektům potomek. Atribut, který je touto relací přenášen, je klíčovým atributem entity odpovídající objektu otec a externím



Obr. 1 Vztah GEN–SPEC v notaci dle Coad-Yourdona



Obr. 2 Vztah GEN–SPEC v notaci ERD

identifikátorem entit odpovídající objektům potomek. Vyjádření výše uvedeného příkladu v notaci ER diagramů je uvedeno na obr. 2.

Z výše uvedeného zobrazení je však hned vidět jeden nedostatek notace ER diagramů. Na rozdíl od OO notace dle Coad–Yourdona, kde je explicitně vyjádřeno, že jde o vztah

GEN–SPEC, není tato skutečnost v notaci ER diagramů nikterak zachycena. Přesto však pro implementaci vztahu GEN–SPEC v relačním prostředí platí důležitá pravidla, která je nutné brát při implementaci do úvahy. Tato pravidla se mění například v závislosti na tom, je-li otec v OO notaci dle Coad–Yourdona abstraktní třídou nebo nikoliv.

Pokud má objekt otec pouze jediného potomka, je odpovídající vyjádření v notaci ER diagramů velice jednoduché. Případ kdy je otec abstraktní třída a má pouze jediného potomka, je triviální a měl by upozornit na chybu v analýze. V takovémto případě je totiž možné otce a potomka sloučit v *jediný* objekt, čili v jedinou entitu.

Je patrné, že pokud není notace ER diagramů rozšířena tak, aby zachycovala vztahy, které jsou zachyceny v OO notaci dle Coad–Yourdona, je její vypovídající schopnost pro použití při skutečné implementaci objektů pomocí relačního databázového systému nedostatečná ! Navíc je patrné, že vztah GEN–SPEC nezaručuje referenční integritu mezi tabulkami databáze, tak jak by bylo žádoucí!

WHOLE–PART (celek – část)

Vztah WHOLE–PART podle Coad–Yourdona zachycuje skutečnost, že nějaký objekt se skládá, nebo mu přináleží nějaké jiné objekty. Například objektu LETADLO přináleží objekty MOTORY. Nejedná se zde již o vztah dědičnosti, jako v předešlém případě, ale je patrná jednak blízká sounáležitost objektů a nesymetrie jejich vztahu, vyjádřitelná pojmy CELEK (vlastník) – ČÁST (podřízený).

Všimněme si opět některých rozdílů ve vyjádření pomocí OO notace dle Coad–Yourdona a vyjádření v notace ER diagramů. Na rozdíl od OO notace dle Coad–Yourdona, která předpokládá, že vztah mezi objekty, jako takový existuje. Je opět nutné v notaci ER diagramů zavést a pracovat s interními a externími atributy entit ! Na rozdíl od vztahu GEN–SPEC jsou jednotlivé relace obecně typu M : N. Kde M a N mohou nabývat hodnot v rozsahu **0 – n**. Nicméně pro zjednodušení implementace budeme relace typu M : N, tak jak je obvyklé, rozkládat na dvě relace typu M : 1 a 1 : N ! Protože na rozdíl od předešlého případu GEN–SPEC každá část musí přináležet nanejvýš jednomu vlastníkovi, lze velice dobře zabezpečit **referenční integritu** systému!

INSTANCE CONNECTION (spojení instancí)

Podle Coad–Yourdona se jedná vztah mezi nezávislými objekty, který je založen na vzájemné souvislosti mezi jednotlivými objekty. Příkladem takovéhoto vztahu může být vztah mezi objekty OSOBA a AUTO. Je zřejmé, že mezi objektem OSOBA a objektem AUTO není vztah GEN–SPEC (objekt AUTO není specializací objektu OSOBA a ani naopak) ani vztah WHOLE–PART (objekt OSOBA se neskládá z objektů AUTO a ani naopak). Vztah mezi objektem OSOBA a objektem AUTO může, ale například být: „objekt OSOBA vlastní objekt AUTO", ale například také: „v objektu AUTO cestuje objekt OSOBA".

Na rozdíl od vztahů GEN–SPEC a WHOLE–PART, které nám popisují strukturu zkoumaného systému a při správné analýze bychom měli všechny vztahy těchto dvou typů objevit a definovat, je možné definovat velké množství vztahů typu INST–CONN (instance connection), které mohou mít pro řešení našeho problému smysl, ale i takové které smysl nemají. Takovým vztahem může být například vztah: „objekt OSOBA mluví o objektu AUTO". Je nutné zdůraznit, že smysluplnost vztahu INST–CONN závisí na námi řešeném problému (problem domain). Například vztah „objekt OSOBA vlastní objekt AUTO" má význam při řešení problematiky IS evidence vozidel, zatímco vztah „objektem AUTO cestuje objekt OSOBA" má význam při řešení IS cestovní kanceláře.

Opět při použití notace ER diagramů je nutné určit klíčové atributy a směr jejich přenosu. Tato záležitost nemusí být vždycky zcela triviální. Přechod od OO notace dle Coad–Yourdona k notaci ER diagramů není tedy zcela automatický. Při objektově orientovaném přístupu je možné se během analýzy výhodně oprostit od implementačních detailů, které však musíme brát na zřetel při vlastní implementaci v prostředí RDBMS.

3 Závěr

Tímto svým příspěvkem chci naznačit, že relační model je pro modelování skutečných vztahů mezi objekty reálného světa nedostatečný. Proto je přechod od relačních databázových systémů k objektově orientovaným databázovým systémům výhodný. Zvyšuje totiž bezpečnost a robustnost implementace programovaného systému. Nicméně vzhledem k tomu, že relační databázové systémy se v současné době stále používají a budou používat, je rozšíření notace ER diagramů výhodné a výše uvedené úvahy o vztahu mezi OO (objektově orientovaným) modelem a ER (entitně–relačním) modelem pomůckou při implementaci systému.

Literatura

[CYOOA1990] Coad,P. and Yourdon,E. (1990) *Object–Oriented Analysis.*

[CYOOD1990] Coad,P. and Yourdon,E. (1990) *Object–Oriented Design.*

Autor: Ing. Martin Molhanec
ČVUT–FEL, K–313
Technická 2
166 27 PRAHA 6, Dejvice
tel.: 332 2118, 332 2119