ROZDÍLY V NÁVRZÍCH RELAČNÍCH A OBJEKTOVÝCH DATABÁZÍ A JEJICH DŮSLEDKY PRO TRANSFORMACI MODELŮ

RELATIONAL AND OBJECT DATABASES DESIGN DIFFERENCES AND IT'S IMPLICATIONS TO MODEL TRANSFORMATION

Vít Holub

Anotace:
Článek na praktickém příkladu ukazuje vliv některých postupů specifických pro relační a objektový návrh na výslednou strukturu relačního, resp. objektového statického modelu. Zjištěné poznatky dokládají nezbytnost doplnění sémantických informací a zpětného sestavení konceptuálního modelu pro účely transformace schémat.

Klíčová slova:
databáze, relační design, objektový design, transformace schémat

Abstract:
The purpose of this article is to present the implications of design-specific approaches to the static structure of relational and object models. Finally the conclusion is taken that semantic enrichment and the conceptual model reverse-engineering are neccesary in the schema transformation process.

Keywords:
database, relational design, object-oriented design, schema transformation

ÚVOD

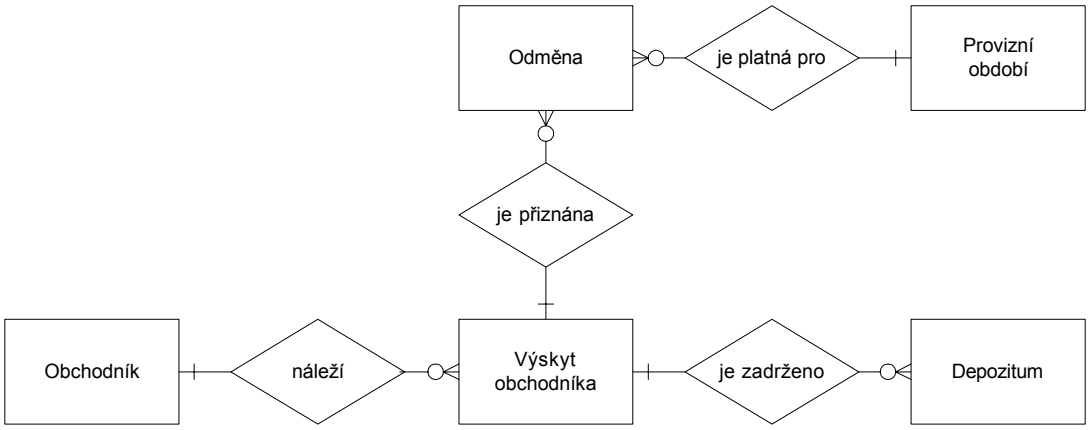
Použití relačních databází (RDBMS) pro uložení dat může být v některých případech nevýhodné. Týká se to zejména případů složitých datových struktur, kdy omezené prostředky RDBMS nemohou zachytit kompletní sémantické informace.
Pokud je rozhodnuto o přechodu na objektovou databázi (ODBMS), je nutné provést migraci datové základny. Transformace datového modelu je jejím klíčovým prvkem.

VYBRANÉ ODLIŠNOSTI VE STRATEGIÍCH NÁVRHU S VLIVEM NA STRUKTURU MODELU

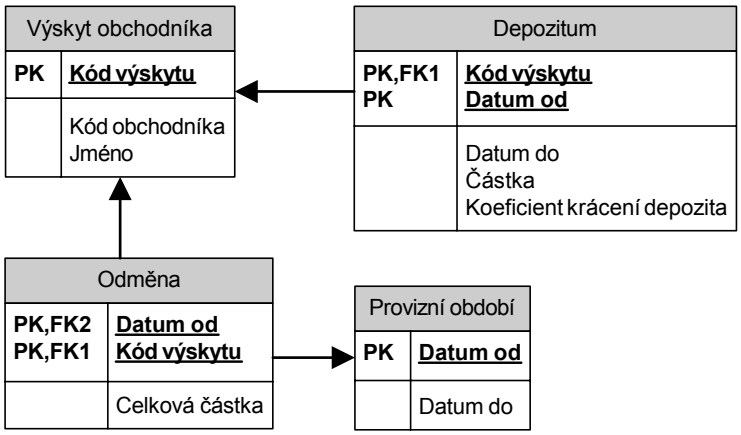
Transformace datových modelů může být na první pohled velmi snadná a přímočará. Třídy by jednoduše odpovídaly tabulkám, objekty záznamům. Informační systém implementovaný podle takového návrhu by pravděpodobně fungoval, jistě by ale nebyl sestaven optimálně a uživateli by nepřinesl žádnou výhodu.
Uvedený postup by byl v pořádku jen v případě, kdyby objektový datový model byl analogií relačního, kdyby techniky návrhu nabízely shodné nebo podobné prostředky.
Objektové paradigma a je však od relačního značně odlišné. Porovnáme-li dva takto různě vytvořené, ale jinak ekvivalentní modely, zjistíme, že představa o prostém nahrazení tabulek třídami je mylná. V čem se tedy oba přístupy liší a čím je způsoben očividný rozdíl ve struktuře modelů?

Zohlednění dynamické a funkční složky ve statické struktuře objektového modelu

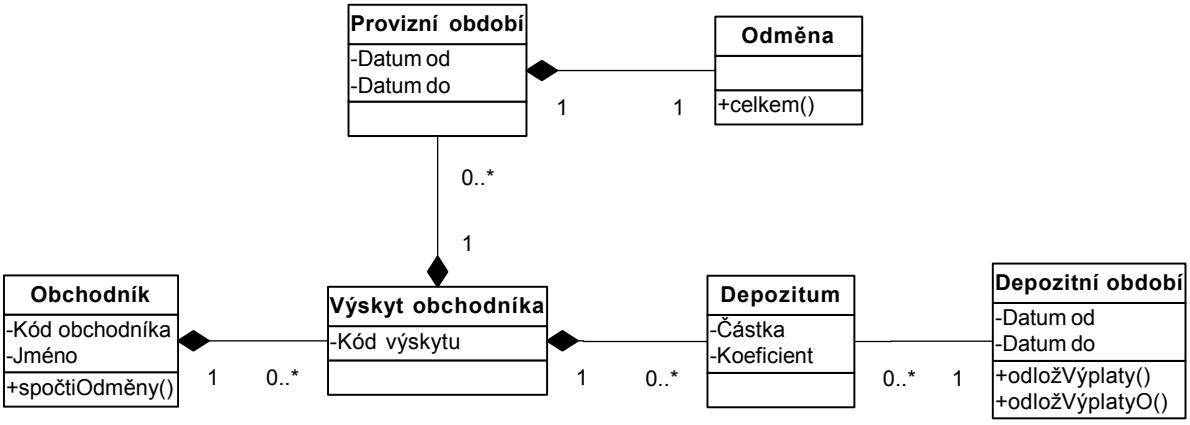
Podstatným rysem objektového návrhu je začlenění aplikační logiky přímo do databáze. Každá třída má definované operace, které může po obdržení zprávy vykonat. V RDBMS není tato integrace možná - existují sice uložené procedury, ale mezi nimi a tabulkami neexistuje žádný vztah.
Podrobně je vše vidět na následujícím příkladu:



Model znázorňuje část informačního systému pro výpočet a správu odměn obchodníkům, kteří pro pojišťovnu sjednávají pojistné smlouvy. Obchodník jako jeden výplatní subjekt může pracovat ve více rolích - výskytech (na různých místech a různých úrovních řízení). Jsou mu přiznány různé druhy odměn, část určité odměny může být deponována a vyplacena až po uplynutí stanovené doby.



V relační formě není zavedena tabulka *Obchodník*. Je to proto, že entita *Obchodník* reprezentuje skutečnou osobu, které budou vyplaceny odměny za práci všech jeho výskytů, tyto odměny však primárně náleží výskytům (o výskytech je mj. na jiném místě systému účtováno, entita obchodník slouží pouze pro tisk sestav).



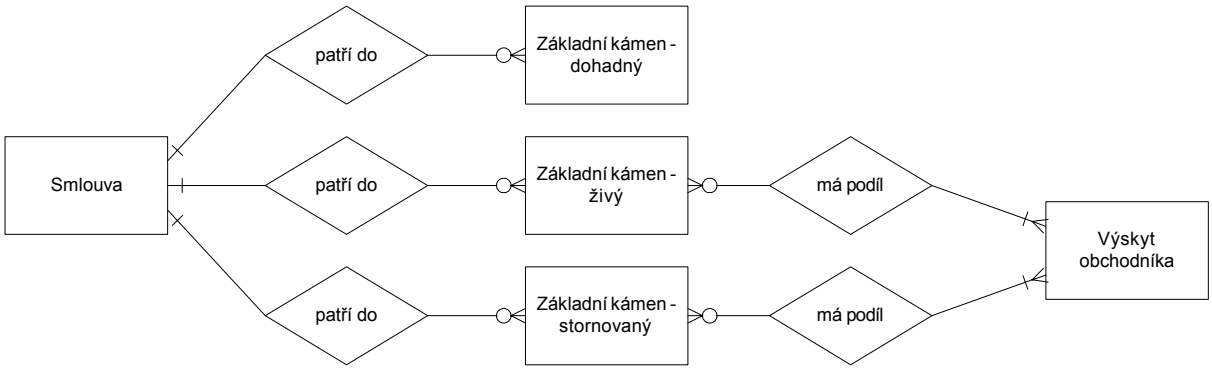
Pokud je systém postaven na objektové databázi, navrhují se algoritmy výpočtu odměn přímo v metodách objektů. V systému pak mohou být třídy, jejichž existence není vyžadována pro správu dat, ale právě pro provádění metod. Proto je v příkladu namodelována třída *Obchodník*, která zajistí výpočet odměn za všechny výskyty daného obchodníka.
Podobný účel má i speciálně vytvořená třída *Depozitní období*. Přestože entita není součástí ER diagramu, její vytvoření si vyžádala funkční analýza, konkrétně potřeba metod *odložVýplaty* a *odložVýplatyO*.
Metod objektů lze využít také při získávání odvozených údajů. Např. metoda *celkem* třídy *Odměna* vrací součet všech jejích složek.

Sémantická nedostatečnost relačního modelu

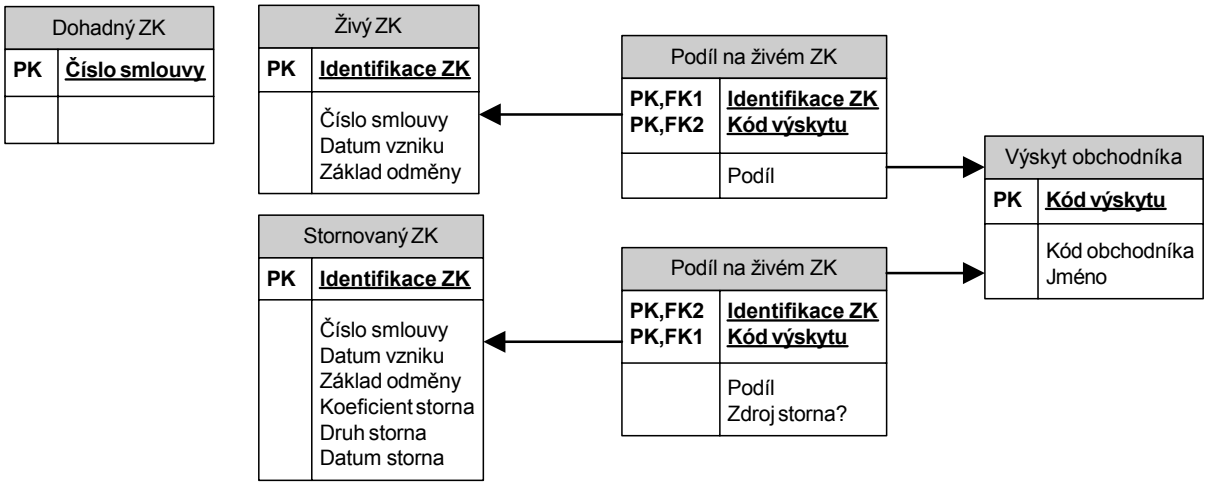
Nedostatek modelovacích konstruktů vede k neúplnosti sémantických informací obsažených v relačním schématu. Objektový model poskytuje další konstrukty (skládání, agregace, generalizace...), které v případě použití relační databáze musí být implementovány výhradně pomocí tabulek a cizích klíčů. Některé informace tak bývají obsaženy spíše v datech, než ve schématu.
Na příkladech v ostatních oddílech je patrné, jaké informace byly při tvorbě relačního modelu zachovány, a jaké ztraceny. Informace, které se takto ztratí, musejí být na počátku transformace modelů doplněny.

Identita objektu a jeho příslušnost třídě

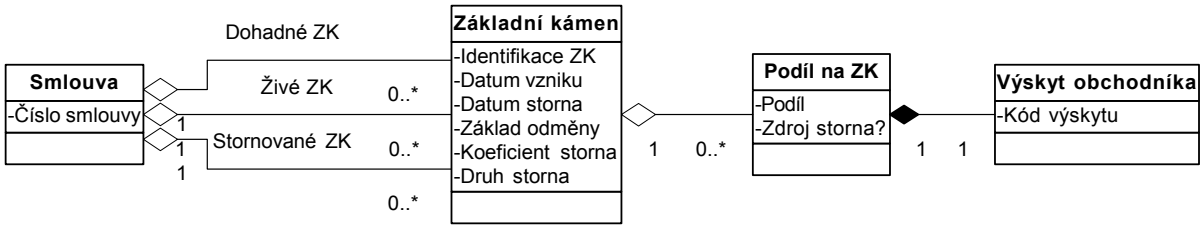
Jednoznačná identifikace záznamu v tabulce je dána primárním klíčem, jeho hodnotu je možné změnit. Jednoznačnost platí v rámci jediné tabulky, unikátní číslo pro celou databázi neexistuje. Platí také, že dva záznamy jsou totožné, pokud jsou všechny jejich složky shodné.
Naproti tomu mají objekty v databázi jedinečné OID – číslo určující identitu objektu. Dva objekty jedné třídy se shodnými atributy totožné nejsou.
Z uvedeného mj. vyplývá, že v relační databázi lze záznam libovolně přenášet mezi tabulkami (se stejnou strukturou), aniž by se změnila jeho identita. Podobný přenos objektů není možný – vytvořením stejně naplněného objektu v jiné třídě vzniká odlišný objekt. Proto vznikají při obou návrzích rozdílné struktury.



Jako příklad je použita část stejného systému. Obchodník sjednává smlouvy, jejichž části (*Základní kameny - ZK*) generují odměny. Každý *ZK* patří jednomu nebo více obchodníkům, a může nabývat těchto stavů: dohadný (fiktivní *ZK* nepatřící žádnému obchodníkovi), živý (obchodníkovi náleží odměna v plné výši) a stornovaný (obchodníkovi náleží odměna podle *Koeficientu storna*).



V relační databázi mohou být pro každý z možných stavů vyhrazeny zvláštní tabulky. Každá z těchto tabulek má jen potřebné sloupce a ZK je během svého životního cyklu mezi tabulkami podle potřeby přemisťován. Uvedené řešení je výhodné v situaci, kdy se často provádějí operace nad izolovanými tabulkami stavů ZK.



Stejný objekt (*ZK*) nesmí v průběhu svého životního cyklu opustit svou třídu. Při návrhu tříd je s tímto nutné počítat – ekvivalentní objektový model má jen jednu třídu *ZK* a jednu třídu *Podíl*.

Různé způsoby přístupu k datům

Struktura statických modelů odráží i různé způsoby přístupu k datům. Zatímco relační databáze využívají tzv. množinový přístup, mezi objekty se postupuje navigací. Díky zapouzdření mohou objekty pracovat jen se svými daty, k přístupu k „cizím“ datům je zapotřebí využívat metody dalších objektů, ke kterým má původní objekt přímý přístup.
Na diagramech k prvnímu příkladu je vidět změněná struktura objektového statického modelu, třída *Provizní období* byla přesunuta přímo ke třídě *Výskyt obchodníka*.

ZÁVĚR

Objektový datový model je správný (přínosný), pokud je výsledkem čistě objektového návrhu. Prosté převedení relačních prvků na objektové není korektní a nepřináší uspokojivé výsledky. Proto je při transformaci nutné doplnění sémantických informací a zpětné sestavení konceptuálního modelu.

Literatura:

Michael Blaha, William Premerlani. *Object-Oriented Modeling and Design for Database Applications*. Prentice Hall, 1998. ISBN: 0131238299

J. Rumbaugh et al. *Object-Oriented Modeling and Design*. Prentice Hall, 1991. ISBN: 0136298419

V. Merunka. *Objektový přístup v databázových systémech*. ČZU Praha, 2002. ISBN: 8021308826

Kontakt:

Ing. Vít Holub
Unicorn a.s.
+420267991723
vit.holub@unicorn.cz