OBJEKTOVĚ ORIENTOVANÉ DATABÁZE

Vojtěch Merunka

ÚVOD

Objektově orientované programování se dnes víceméně stalo již standardním pracovním nástrojem a postupně ztrácí svůj dřívější nádech exotičnosti. Důkazem tohoto tvrzení jsou stovky osvědčených malých i velmi rozsáhlých programů především v jazycích Smalltalk a C++ a v poslední době i Object COBOLu. Hitem dneška jsou metody objektově orientované analýzy a designu a postupně se do popředí zájmu dostávají i objektově orientované databáze, které však zatím patří mezi nejméně známé (i využívané) aplikační oblasti objektově orientovaného paradigmatu, což bývá u nás i v zahraničí často zneužíváno ve prospěch obchodních zájmů nejrůznějších softwarových firem.

V tomto článku bude nejprve v jeho první části podán obecný pohled na objektový datový model a na jeho srovnání s relačním datovým modelem. Ve druhé části potom budou diskutovány nejrozšířenější mýty z oblasti současné databázové technologie, které mají vztah k objektově orientovaným databázím.

Popisované informace vycházejí z provozních zkušeností autora s objektovou databází ArtBase, z dílčích poznatků o objektových databázích GemStone a OODBMS, z materiálů o databázových systémech ORION, O_2 a IRIS a z provozních zkušeností s relačními databázemi ORACLE 7, Paradox a MS Access. Příklady dotazů psané v jazyce Smalltalk přímo platí pro první tři jmenované objektové databázové systémy.

OBJEKTOVÝ DATOVÝ MODEL

Objektový (ODM) a relační datový model (RDM) se od sebe výrazně liší ve své funkčnosti i v architektuře. Porovnejme si hlavní vlastnosti obou datových modelů:

1) Relační datový model je oproti mnoha možnostem objektového modelu založen pouze na množinách (tabulkách) záznamů, přičemž pokud jednotlivé prvky záznamů mohou být jen atomickými hodnotami, jedná se o tzv. normalizovaný relační model. RDM sice principiálně nebrání za prvky záznamů dosazovat strukturované hodnoty (další záznamy nebo celé tabulky), ale tato možnost se v praxi používá jen velmi ojediněle v tzv. NFNF databázích.

2) Objektový model kromě datové (statické) informace podporuje oproti relačnímu modelu navíc i práci s procedurální (dynamickou) informací v podobě metod jednotlivých objektů.

3) Operace poslání zprávy u objektového modelu má větší funkčnost než manipulace s hodnotami atributů záznamů u relačního modelu. Zprávy mohou obsahovat parametry (v optimálním případě jiné objekty). Metody, kterými příslušné objekty reagují na tyto zprávy, mohou kromě prostého čtení či nastavování vnitřních složek objektů (jako v RDM) implementovat i takové vlastnosti objektů, které nejsou jednoduše dány jejich

vnitřními hodnotami. Z pohledu ODM však jsou všechny metody objektů rovnocenné bez ohledu na jejich nestejný vztah k vnitřním datovým složkám.

Mějme například objekty třídy `Osoba`, každý se složkami `jméno`, `adresa` a `rodnéČíslo`. Do protokolu takovýchto objektů potom kromě „triviálních" zpráv `jméno`, `adresa` a `rodnéČíslo` manipulujících se složkami objektů stejně jako v RDM mohou patřit i zprávy `pohlaví`, `datumNarození`, `věkováKategorie`, protože jim odpovídající metody mohou tyto informace získávat libovolně složitým výpočtem.

) Objektové množinové operace nejsou omezeny na množiny obsahující objekty stejné struktury jako jim odpovídající relační operace. Je-li množina objektů složena z různých objektů (tj. objektů různých tříd), potom stačí aby všechny objekty této množiny měly společnou neprázdnou množinu zpráv. S využitím polymorfismu a dědění mezi objekty se potom může jednat i o poměrně velmi heterogenní soubory objektů, což může být využito již při konceptuálním návrhu modelované aplikace.

Možné odlišné vlastnosti objektů a relačních záznamů i množin objektů a relačních tabulek mají mj. i vliv na možnosti i vlastnosti dotazování. V ODM lze využívat například operace sjednocení a průnik i na množinách s různými objekty. Na další operace objektové algebry se podívejme podrobněji s pomocí jednoduchého příkladu.

Pro demonstraci objektových dotazů si vezměme heterogenní množinu objektů s názvem `Zákazníci`. Díky polymorfismu v ní mohou být objekty třídy `Osoba` (a jejích podtříd podle zásad dědění) i objekty třídy `Firma` (ev. podtříd). Mezi společné vlastnosti objektů, které lze využít, patří např. zprávy `jméno` a `adresa`.

4a) Provedeme-li selekci, jak ukazuje příklad,

```
Zákazníci
  select: [:each | each jméno isLike: '*Novák*'].
```

tak dostaneme podmnožinu z množiny `Zákazníci`. Došlo zde však k zajímavému úkazu. Protože všichni „novácí" jsou jistě nějaké osoby, tak na výsledné množině lze rovnou aplikovat další dotazy i s využitím takových zpráv (např. `pohlaví`, `datumNarození`, `příjmení`), které by nebyly možné použít v původní množině. Ve výsledcích objektových selekcí tedy na rozdíl od relačních může docházet ke zvětšování protokolů, což může v některých případech zajistit takovou funkčnost, kterou v relačních databázích lze docílit jen oklikou přes operace spojení tabulek.

4b) Pokračujme operací projekce:

```
Zákazníci project: #(jméno adresa).
```

výsledek této operace opět vlivem vlastností naší množiny `Zákazníci` je také svojí potenciální funkčností odlišný od výsledku relační projekce. Zde nám vzniká množina nových objektů (v tomto konkrétním případě typu `Array`), které zřejmě obsahují po jednom objektu třídy `Name` a jednom objektu třídy `Address`.

4c) V objektové algebře existuje tzv. operace kolekce, která nemá v relační algebře přímý ekvivalent a která zobecňuje operaci projekce. Kolekce je taková funkce, která pro danou množinu objektů vrací jako výsledek množinu, která obsahuje všechny objekty první množiny transformované libovolným výrazem. Ukázka přináší možné dva dotazy kolekce množiny Zákazníci:

```
Zákazníci
  collect:  [:each | each adresa obec početObyvatel].
```

a nebo

```
Zákazníci
  collect:  [:each | Array
                       with: each jméno
                       with: each adresa obec název
            ].
```

První dotaz vrací množinu s číselnými údaji o počtu obyvatel sídel, ze kterého pocházejí zákazníci. Druhý dotaz vrací množinu dvojic jméno zákazníka - název obce.

5) Pojetí identity u objektů se nekryje s pojetím identity u záznamů. Identita objektu je určena kromě vnitřních hodnot i všemi vnějšími vazbami daného objektu. Identita objektu tedy nemusí být přerušena ani při změně všech jeho vnitřních hodnot (složek). Identita záznamu je určena pouze jeho vnitřními hodnotami (viz. koncept primárního klíče).

Shrňme si nyní uvedené rozdíly a jejich praktický dopad na rozdílnou funkčnost objektových a relačních systémů, a hlavně na proces analýzy a návrhu těchto systémů:

1) Objektový model podporuje teoeticky neomezeně složité a strukturované datové entity - objekty v různých vzájemných hierarchiích (skládání, dědění, závislost, vztah třída-instance), které mohou snáze korespondují s "reálnými objekty" v zadání, resp. konceptuálním modelu aplikace. Kromě jednoduchých typů objektů (texty, čísla, grafika, zvuky), je možné vytvářet i nové typy objektů jako kombinace existujích. Každý objekt se navenek projevuje jako jedna celistvá entita. Tento fakt se příznivě projevuje na dotazovacích a vůbec funkčních možnostech aplikace.

   Relační databáze je omezena na záznamy z jednoduchých přípustných datových typů, jakými jsou např. čísla či texty (u moderních relačních systémů i grafika a zvuky - diskutováno v následující kapitole „Mýty ..."), přičemž každý složitější údaj či struktura musí být v relačním modelu transformován na soustavu záznamů ve více tabulkách a sestavených z přípustných datových typů.

2) U relačního datového modelu jsou v entitách databáze uloženy pouze statické vztahy - data. Dynamická stránka je záležitostí programu pracujícího s takovouto databází. Objekty, se kterými pracuje objektový model, kromě statických údajů - dat obsahují v databázi i kódy metod - tj. dynamickou stránku uložených dat nezávisle na manipulujícím programu. Objektovou databázi lze navrhnout takovým způsobem, že nepotřebuje

externí manipulující program, protože veškerá její funkčnost (i rozhraní) je přímo zajištěna jejími objekty.

3) Údaje v relačních tabulkách (tj. složky jednotlivých záznamů) jsou přímo přístupné přes příslušné domény, jejichž názvy tvoří protokol daných záznamů. Údaje v objektovém modelu jsou ukryté v objektech a namísto techniky přímého přístupu do vnitřní struktury podle atributu se pracuje s takovými protokoly objektů, které vymezují každému objektu množinu přípustných zpráv. Mezi protokolem objektu a jeho vnitřní datovou strukturou nemusí být přímá souvislost. Zprávami posílanými objektům lze získávat i takové údaje, které nejsou přímou součástí struktury daných objektů, neboť mohou být výsledkem libovolného výpočtu (viz. ad 2).

4) Vzhledem k 3) je u relačního modelu nutné, aby všechny údaje v rámci jedné domény zachovávaly nějaký stejný datový typ a všechny záznamy jedné tabulky stejnou strukturu, což prakticky znamená, že všechny navrhované množiny v relačním systému musejí být navrhovány již od konceptuální fáze analýzy pouze z prvků stejného typu. Množiny z prvků různého typu (podobně jako vztahy **m:n**) je nutno rozkládat. Vztahy **1:n** je nutné u normalizovaných relačních modelů implementovat pomocí propojení dvou tabulek, kde vazba (propojení pomocí klíčů) vždy směřuje od entit kardinality "**n**" směrem k entitám s kardinalitou "**1**", což má přímý vliv i na sémantiku dotazů v relačním dotazovacím jazyce, kde je při jejich formulaci třeba se podřizovat skutečnému směru příslušné vazby.

   Objektový model umožňuje implementaci vazeb mezi objekty vždy v souladu s konceptuálním modelem ve směru od celku k části bez ohledu na případnou kardinalitu a datové typy.

5) U relačního modelu během činnosti DBMS, tj. během zpracovávání některých dotazů, dochází k dočasnému spojování jednotlivých relačních tabulek, neboť každá složitější struktura byla při své tranformaci do relačního datového modelu rozdělena do více záznamů uložených v různých relačních tabulkách. Při dotazu proto musejí být komplexní údaje jdoucí mimo rámec údajů uložených v jedné tabulce spojovány (pomocí klíčů) z údajů více tabulek, což představuje časově i uživatelsky náročnou operaci.

   U objektového modelu k spojování množin dochází jen v záměrných netriviálních případech, neboť možnosti skládání objektů nejsou teoreticky nijak omezené.

6) U relačního modelu je pro potřeby zajištění identity navrhovat kromě běžných atributů tzv. klíče záznamů, které se nemusí vždy krýt s potřebami zadání. Na klíčích relačních záznamů, které jsou de facto pouze implementačním prostředkem (abstrahované ukazatele do paměti), závisí celý aparát výrazových prostředků relačního dotazovacího jazyka.

   U objektového modelu stačí v návrhu modelovat pouze ty složky, které vždy odrážejí nějaký informační aspekt ze zadání. Klíče jako implementační prostředek jsou výhradně vnitřní záležitostí systému.

Z uvedených souvislostí mezi popisovanými datovými modely lze vytušit i formálně dokázat, že **relační datový model může za určitých podmínek představovat zvláštní případ objektového datového modelu** (tj. že objektový model pokrývá všechny vlastnosti relačního modelu), přičemž rozbor vzájemných souvislostí mezi modely se jeví jako velmi prospěšný například pro pochopení činnosti či návrhu relačních databázových systémů s

„objekty", jakými je například Paradox či MS Access a nebo „relačně-objektová" klient-server řešení.

Závěrem této kapitoly můžeme dosavadní poznatky o porovnávaných datových modelech a jejich vzájemně si odpovídajících pojmech shrnout do následující tabulky:

relační datový model	objektový datový model
relační tabulka (množina záznamů)	množina objektů
entita, jedna relace z tabulky (záznam)	objekt
atribut, operace s atributem záznamu	zpráva

MÝTY O OBJEKTOVÝCH DATABÁZÍCH

Na základě předložených úvah je nyní možné i vhodné se zaměřit na některé všeobecně rozšířené mýty o objektově orientovaných databázích.

1. BLOBs a možnost ukládání procedur = objekty.

Na současných relačních databázích je patrný trend směrem k postupnému rozšiřování relačního datového modelu. První z nich jsou tzv. BLOBs (binary-large objects), v různých systémech označované i např. jako „RAW" nebo i „MEMO". Druhou z nich je možnost přidávání procedur k záznamům v relačních tabulkám (tzv. triggers). Toto vede mnohé prodejce databází k reklamním tvrzením, že jejich produkty jsou objektově orientované, přičemž tato tvrzení jsou dovedně dokládána možností ukládání např. obrázků a využíváním výše zmíněných triggerů v příkladech předváděných na výstavách, konferencích či v literatuře.

Odhlédneme-li od faktu, že uvedené databáze jsou stále založeny na relačním (tabulkovém) datovém modelu - což by už mělo jako argument stačit, tak stejně BLOBs samy o sobě nejsou rovnocenné s objekty. Je tomu tak proto, že na rozdíl od objektů nemají dostatečnou funkčnost k tomu, aby operace s nimi byla například součástí selekční podmínky v dotazu - např. nelze přímo položit selekci takových záznamů, které obsahují zvukový záznam (BLOB) delší než 1 minutu. V podmínce selekce totiž mohou figurovat pouze tzv. „klasická" data, protože BLOBs se v záznamech nechovají jako atributy záznamů - jsou to jen bajty obsahující přídavnou informaci, jejíž interpretaci musí uživatel databáze (programátor) znát.

Podobně je tomu i s procedurami - triggery. Triggery sice představují výhodnou možnost implementace chování dat, což jistě přináší veliké úspory při tvorbě aplikací, ale nevytvářejí se záznamy stejně těsnou vazbu, jako metody objektů, a neexistuje u nich polymorfismus ani dědičnost. Navíc se triggery vesměs spouští změnou hodnoty v záznamech a jsou to většinou jen přidružené SQL příkazy na rozdíl od objektových metod, které mohou obsahovat a také obsahují libovolný kód.

2. Každá správná databáze musí podporovat standard SQL.

SQL je standardem pouze pro relační databázové systémy. Jazyk SQL totiž přímo vychází z matematické teorie relačního datového modelu. Je také pravdou, že jazyk SQL se liší od systému k systému a od verze k verzi. Správnější by tedy bylo hovořit o „jazycích typu SQL". Samotný jazyk SQL však ani v relačních databázích nestačí k vybudování celé programové aplikace. Proto se kombinuje s programovacími jazyky C, 4GL apod. Tato kombinace přináší známé impedanční problémy mezi programovacím a dotazovacím jazykem.

Realitou současných objektových databází jsou možnosti více různých rozhraní k objektově orientovaným programovacím jazykům, a to nejčastěji Smalltalku, C++ a nebo CLOSu. Otázka jazyka SQL není v objektových databázích tak jednoduchá, jak by se mohlo zdát, protože vlastnostem (možnosti dotazování, funkčnost aplikace, ...) objektového datového modelu mnohem lépe vyhovují dnes již standardizované obj. orientované programovací jazyky (např. Smalltalk), které navíc umožňují napsat na rozdíl od SQL celou aplikaci včetně např. uživatelského rozhraní. Jazyk SQL je tedy bez nadstandardních rozšíření (např. SQL3, O_2SQL, Object SQL) pro objektové databáze v podstatě nepoužitelný.

3. Databáze, které nejsou založeny na relačních tabulkách, neumožňují transakce a možnost obnovení obsahu databáze v případě havárie.

Tento mýtus má kořeny v historii, kdy skutečně existovaly první experimentální systémy s těmito nedostatky. Dnes toto tvrzení samozřejmě neplatí. Mnohé víceuživatelské objektové databáze dokonce navíc dovolují každému uživateli jeho aplikaci v rozpracované transakci přerušit, (i vypnout počítač) a v transakci pokračovat např. druhý den (tzv. long transactions). Možnost obnovení stavu databáze do bodu před začátkem transakce („rollback") je v některých databázích doplněna i o možnost práce (např. při obnovování) s několika časovými verzemi jednoho a téhož objektu.

S transakcemi souvisí i problematika uzamykání dat při současných přístupech více uživatelů na jeden údaj. Moderní objektové databáze nabízejí dvě možnosti. První z nich - tzv. pesimistické řízení - je shodná s mechanismem známým z relačních databází, kdy se na počátku transakce objekt uzamkne a další uživatel již předem nemůže provést operaci, která by vedla k možné konfliktní situaci. Protože však možnosti uzamykání a přístupu na data nejsou v objektových databázích omezeny na celé záznamy (objekty), ale uzamykání lze provádět mnohem jemnějším způsobem (po částech objektů, na volání zpráv, ...), je v objektových databázích k dispozici (tj. volitelně) i mechanizmus tzv. optimistického řízení. Optimistické řízení spočívá v tom, že různým uživatelům není předem zakázáno provádět operace nad stejnými objekty a případné konflikty jsou řešeny až při uzavírání transakcí, kdy systém nedovolí uzavřít takovou transakci, jejíž výsledek by byl v konfliktu s nějakou dříve uzavřenou transakcí.

Je však pravdou, že vlastnosti transakcí a schopnosti obnovování a zálohování se velmi liší v závislosti na konkrétním systému. Objektový datový model ale principiálně neobsahuje žádné technické bariéry pro implementaci všech vlastností známých z relačních systémů.

4. Objektové databáze neumožňují dotazování.

Tento mýtus částečně souvisí i s mýtem o SQL. Vznikl proto, že v prvních verzích objektových databází se kladl velký důraz na podobnost se síťovým datovým modelem a z ní vyplývající možnost navigačního přístupu k objektům, který je v objektové technologii označován jako „browsing". Browsing je silným nástrojem především v takových databázích, kde složky jednotlivých objektů jsou opět nějakými strukturovanými objekty a pohyb v databázi lze založit na přechodech mezi jednotlivými vzájemně se skládajícími objekty.

Zdůrazňování browsingu a formální podobnost se síťovým datovým modelem vedla mnohé odpůrce objektové technologie k přesvědčení, že nepodporují dotazování. Podívejme se tedy ještě jednou na možnosti dotazování v objektovém datovém modelu.

- Objektové dotazy mohou kromě výběru hodnot obsahovat i provádění operací s nimi. (např v operaci kolekce).
- Predikáty v dotazech mohou kromě logických operací využívat i navigační přístup pro zajištění vazby mezi objekty, jak i ukazují příklady dotazů např. ad 4c) v předchozí kapitole.
- Objektové dotazy pracují s různými typy množin objektů (`Set`, `Bag`, `Array`, `OrderedCollection`, `SortedCollection`, `Dictionary`, ...) a využívají i jejich vzájemné konverze (`asSet`, `asArray`, ...), relační dotazy pracují jen s jedním typem množiny - s tabulkou.

5. Mezi objektovými databázemi nejsou standardy.

Od roku 1993 probíhá standardizační proces v objektové technologii a to jak na aplikační úrovni (jazyky), tak i na fyzické úrovni (komunikační protokoly, formáty v paměti). Jsou již první výsledky ve formě zpráv ODMG-93 a specifikací CORBA 1.0 (Common Object Request Broker Architecture) a CORBA 2.0. V rámci norem ANSI existují a již první výsledky podávají výbory ANS X3J20 pro Smalltalk, ANS X3J16 pro C++ a pro objektová rozšíření jazyka SQL výbor ANS X3H2.

Mezinárodní sdružení firem zabývajících se objektovou technologií OMG (Object Management Group) vydává dokument „Request for Technology for Object Query Services" obsahující specifikaci abstraktního jazyka OQL. Každým rokem ve světě probíhá několik velkých konferencí, kde je diskutována jak praxe, tak i matematická teorie objektově orientovaného datového modelu.

6. Objektově orientovaná databázová technologie = klient-server.

Tvrzení některých obratných dodavatelů technologií klient-server mohou vést u jejich zákazníků k závěrům, že dobře navržená klient-server aplikace je objektově orientovaná - tj. klient-server řešení kombinované s moderním hardwarem činí každou databázi objektově orientovanou. Je zajímavé, že často je slyšet od stejných lidí, kteří v jiných situacích zastávají názory ad 3) a ad 4).

Bez ohledu na použitý databázový systém lze rozlišit několik různých architektur klient-server a to podle způsobu rozložení funkcí celého systému mezi klienty a server. Je-li

však použita jako centrální databáze databáze relační, tak ani jedna z možných architektur řešení nemůže způsobit změnu chování datového modelu směrem k objektům.

Samostatnou kapitolou je použití objektově orientovaných prostředků pro návrh a implementaci klientovských aplikací, ale i zde platí, že použití objektově orientovaného programovacího jazyka (a ani analýzy) automaticky neznamená, že v nich vytvořený produkt musí být z databázového pohledu také objektově orientovaný.

7. velké databázové systémy musejí být založené na relačním datovém modelu

Tento mýtus má opět kořeny v historii OODBMS a je založen na domněnce, že jedinou možností, jak efektivně ukládat a pracovat s rozsáhlými objemy dat je relační datový model.

Přímý vztah mezi datovým modelem a fyzickým způsobem ukládání a správy dat na diskových pamětech mají pouze relační databáze X-base (Dbase, Fox, Paradox, ...), kdy jednotlivé tabulky přímo odpovídají příslušným souborům v operačním systému. U velkých databázových systémů typu ORACLE jsou relační data na diskových pamětech uložena podstatně sofistikovanějším způsobem, který však již na relačním datovém modelu není přímo závislý a je velmi podobný způsobu ukládání dat ve velkých objektových databázích.

Objektové a relační databáze se však velmi liší ve filosofii práce s daty vzhledem k uživatelům databáze. Relační databáze svým uživatelům poskytuje prostředky, jak lze pomocí programů běžících v operační paměti pracovat s daty na discích, takže na logické úrovni se všechna data databáze presentují jako disková (tablespaces, files, ...). Objektové databáze na rozdíl od relačních vytvářejí svým uživatelům na svých rozhraních zdání toho, že všechna data jsou uložena v operační paměti podobným způsobem jako běžné proměnné, jak je známe z vyšších programovacích jazyků a o práci s diskem se interně stará systém, který neustále odlehčuje operační paměti odkládáním dat na disk a naopak načítá z disku do paměti (tzv. „swizzling"), čímž se zajišťuje i aktualizace a synchronizace stavu báze dat.

Současné velké objektové databáze, jejímž představitelem je Gemstone, jsou téměř vždy přirovnávány a testovány s relačními databázemi ORACLE, Ingres, Informix, Progress a nebo Sybase, protože patří do stejné velikostní třídy (Gemstone podporuje teoreticky až 3200 uživatelů a až 4 mld. objektů v systému) a v porovnávacích testech s databází ORACLE rozhodně nezůstává pozadu (čím je báze dat strukturovanější, tím je rychlejší - například v dotazování až 2x - 6x).

ZÁVĚR

V článku byl podán pohled na současnou situaci v databázových technologiích se zaměřením na objektové databáze smalltalkového typu. Cílem tohoto článku nebylo jen prosté vychvalování objektově orientované technologie v oblasti zpracování dat, ale spíše snaha autora vyjádřit se k této aktuální problematice a podělit se o některé praktické zkušenosti s některými objektovými databázemi.

Pokud obsah tohoto článku pomůže jeho čtenářům v zorientování se v této dnes poměrně nepřehledné oblasti či vyvolá nová zamyšlení a diskuse, tak svůj účel splnil.

LITERATURA

- E. Bertino, L. Martino; Object-Oriented Database Systems - Concepts and Architectures, Addison-Wesley 1994.
- J. Pokorný; Dotazovací Jazyky, Nakl. Science 1994.
- M. Loomis; série článků o OODBMS v časopise Journal on Object-Oriented Programming ročník 1994.
- P. Coad, E. Yourdon; Object-Oriented Analysis, Yourdon Press 1991.
- J. Martin, J. Odell; Object-Oriented Methods, Prentice Hall 1995.
- I. Jacobson; Object-Oriented Software Enginnering, Addison-Wesley 1992.
- D. Maier, J. Stein; Development and Implementation of an OODBMS, interní materiál firmy Servio.
- Artbase - Distributed Smalltalk and Object-Oriented DBMS, ArtInApples 1994.
- Informační materiály o databázi Gemstone od firmy Servio.
- Informační materiály o databázi O_2 od firmy O_2 Technology.
- V. Merunka, J. Polák; Objektově orientovaná analýza a design, SWN leden 1995.
- R. Bébr; Mlamoj: sociologicko-psychologická studie, SWN leden 1995.

Vojtěch Merunka

Katedra informatiky
PEF ČZU v Praze
merunka@pef.vsz.cz

Katedra počítačů
FEL ČVUT v Praze
merunka@cs.felk.cvut.cz