# Kontextové vyhledávání pro knowledge management

**dizertační práce**

**Helena Palovská**

**VŠE Praha, 2002**

# Obsah

# Úvod

Tato práce předkládá novou metodu pro počítačovou podporu knowledge managementu řešící přístup k tzv. paměti organizace. Ačkoli by bylo možno navrhovanou metodu také řadit k obecným metodám organizace rozsáhlých souborů dokumentů, nespadá zcela do této kategorie, neboť na jedné straně nabízí i více, na druhou stranu je vhodnější spíše pro užití v určité jedné konkrétní hospodářské organizaci nebo instituci. Tyto se vyznačují specifičností své činnosti a konkrétní oblastí zájmu, a navrhovaná metoda je založena na předpokladu omezenosti „světa zájmu".

Proto tuto metodu navrhuji pro podporu knowledge managementu, a to jak pro tzv. učící se organizace (learning organization), i pro tzv. společnosti tvořící znalosti (knowledge creating company), i pro tzv. intelektuální kapitál (intellectual capital). (Přehled těchto přístupů k knowledge managementu lze nalézt ve [Vod], více např. ve [Dav], [Mal], [Wil].) Tato má práce se nezabývá knowledge managementem v rámci disciplíny managementu, pouze nabízí jednu možnost, jak podpořit procesy v hospodářské či jiné organizaci zpřístupněním potenciálního informačního či znalostního bohatství, které má organizace k dispozici.

Vyslechnuto z úst lidí, kteří pracují na projektech počítačové podpory knowledge managementu v praxi: „není problém vytvořit velkou hromadu dokumentů". Dá se dokonce zlepšovat účinnost přístupu (technologická dosažitelnost) k dokumentům. Problémem je využití, tj. jak z hromady vybrat to, co v aktuálním případě může být využito.

Tento problém je starý, k jeho řešení existovalo a existuje množství přístupů. Nejvíce je jich založeno na využití tzv. *klíčových slov* pro výběr dokumentu. Pro zvyšování relevance výběru dokumentů požadavkům uživatele, což za předpokladu získávání informace o požadavcích na výběr od anonymního uživatele[1] vlastně vyžaduje zvyšování sémantické bohatosti dotazovacího jazyka (postupu kladení požadavku na výběr dokumentů), byly vyvinuty další metody. Některé jsou založeny na možnosti formulovat podmínky pro výběr dokumentů na základě určité vzájemné *pozice klíčových slov* či fragmentů slov v dokumentu, či pozice slov nebo fragmentů slov v rámci struktury celého dokumentu. Další přístupy zakládají svou nabídku na strukturování *doprovodné informace k dokumentu*, kde se nabízejí klíčová slova týkající se určitých atributů dokumentu (např. o autorovi, vydavateli …) (přehled standardů pro strukturování doprovodných atributů – metadat – pro dokumenty lze nalézt v [Bar], specielně standard Dublin Core je popsán v [Žab], viz také [DC]). Ještě další přístupy jsou založeny na využití *tezaurů*, které zohledňují vztahy mezi významovými pojmy, jež mají klíčová slova představovat.

[1] Jiné přístupy jsou založené na získávání informace o uživateli samotném.

Mezi novější přístupy patří využití tzv. *ontologií*[2], které generalizují myšlenku vztahu mezi pojmy. Rozbor těchto přístupů přináším v následující kapitole. Moje navrhovaná metoda je inspirovaná také těmito myšlenkami, ale je založena na jiné koncepci. Přináší prvek vyšší dynamiky do dotazování. Proto myslím, že dále zvýší možnost těžit z dostupných informačních či znalostních zdrojů.

[2] Toto slovo je informatiky převzato z filozofie, ale má zde jiný význam.

# Motivace

ontologie v informatice

konceptuální grafy

ORM

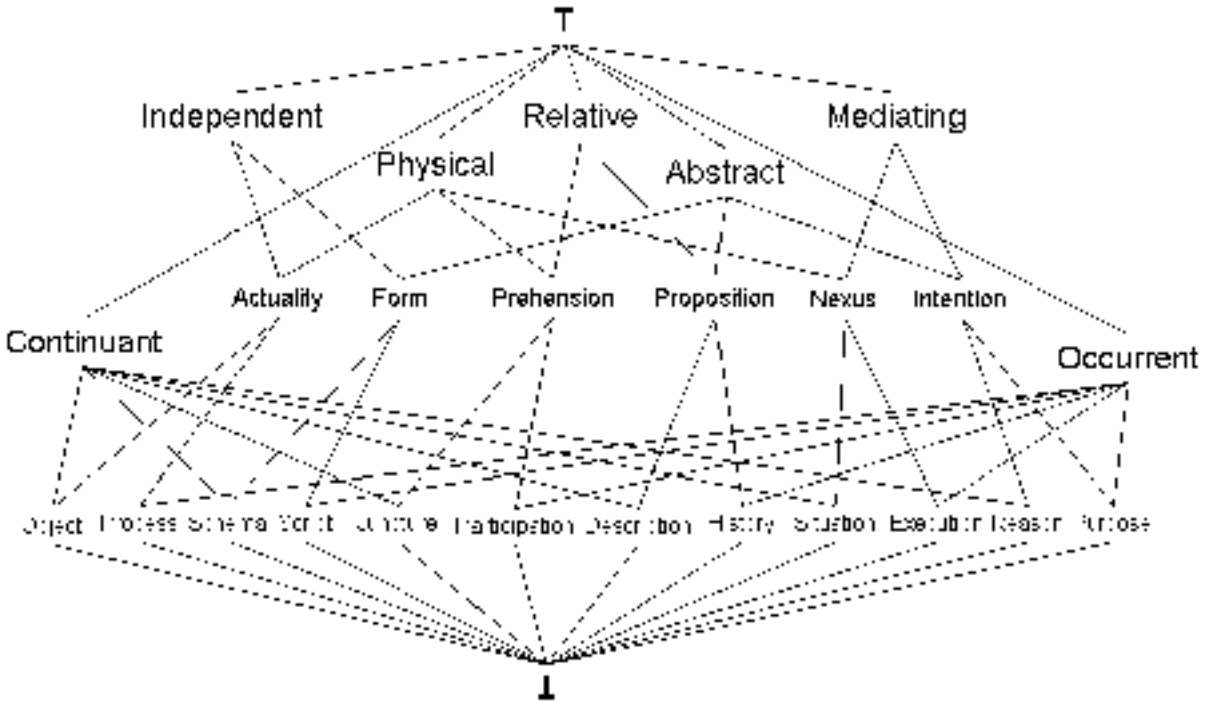
# Výsledky, z kterých práce vychází

Zde přináším pouze ty z cizích teoretických i prakticky vyzkoušených výsledků, které byly pro mou práci hlavní inspirací.

## *Ontologie v informatice*

Slovo ontologie bylo do informatiky převzato z filozofie, ale označuje se jím v informatice něco poněkud jiného. Ačkoli informatický význam slova ontologie s filozofickým významem souvisí a z něj vychází, v informatice se jedná o striktně inženýrský konstrukt. Jde o strukturu znázorňující obecné vztahy uvnitř vymezené množiny pojmů, záměrem je zmapování univerza, kterým se v daném případě někdo zabývá. Mezi informatickým a filozofickým pojetím není ostrá hranice.

V [Sow], apendix B, předkládá autor ontologii, jejíž nejvyšší úrovně vypadají takto:




**obrázek 1: vzorová ontologie J. F. Sowy**

Je vidět, že se jedná o síť[3] pojmů, zobrazující vztahy nadřízený—podřízený[4]. Je to pokus o vytvoření skutečně univerzální ontologie. Zde ho uvádím jednak jako příklad takovéhoto filozoficko-informatického konstruktu, ale také ho uvádím jako názornou ukázku velké šíře takové konstrukce, když „univerzem“ se myslí skutečné univerzum zahrnující vše. V tomto směru je

[3] svaz v matematickém smyslu

[4] Srozumitelněji lze tyto úrovně znázornit do třírozměrného modelu, v němž jeden rozměr je rozdělen na Independent, Relative a Mediating, druhý rozměr na Physical a Abstract, a třetí na Continuant a Occurent.

významná iniciativa Cyc (viz [CYC]), pokoušející se vytvořit „encyklopedickou“ ontologii, zahrnující veškeré pojmy z lidských znalostí[5].

V praktických užitích i v teoretických úvahách pro praktické využití se uvažují „univerza“ omezená, nazývaná např. *Universe of discourse* , oblast zájmu, sféra života, doména, či prostě ontologie. Výrazná je snaha o vytvoření jakýchsi standardů či veřejných knihoven ontologií (viz [FARQ]), aby tyto mohly sloužit k podpoře komunikace (vzájemnému porozumění lidí z různých organizací zabývajících se tímtéž oborem) i k interoperabilitě autonomních automatických systémů využívajících ontologií k formální registraci abstraktních modelů.

Ve formálních ontologiích se uvažují nejen vztahy nadřízenosti a podřízenosti (is a), ale také vztahy částí a celku, vztahy typu a instancí, vztahy entit a jejich atributů, i některé další vztahy, např. vztah „používá“, nebo obecný „vztahuje se“[6].

Dále mohou být ve formálních ontologických modelech zahrnuty i konstrukty tzv. faset: pro pojmy/kategorie[7] v modelu a konkrétní typy atributů[8] entit z těchto kategorií se vyjadřuje jejich kardinalita (tj. kolik různých možných výskytů v dané kategorii přichází do úvahy) nebo obor hodnot (explicitní formální vyjádření množiny možných výskytů, či ji omezující (nad)množiny) apod. (Jednu z možných definic faset viz [fas].)

### *Využití ontologií pro knowledge management*

Ontologie mohou být v projektech počítačové podpory pro knowledge management využity ke tvorbě anotací dokumentů i jiných informačních zdrojů. Tyto anotace mají být využívané při vyhledávání informačních zdrojů v praxi organizace.

Jeden z projektů v tomto směru je částečně popsán v [Abe]. Autoři na tomto projektu dále pracují.

Jejich koncepce, dle mého názoru, jasně ukazuje přednosti i omezení využití ontologií pro podporu knowledge managementu. Ontologie mají být základem, na kterém se všichni lidé zabývající se danou doménou[9] shodnou, jde o jakýsi základní soubor trvale platných konstatování o dané doméně. Zároveň se předpokládá jistá míra *úplnosti* tohoto souboru, tj. že vše podstatné je v něm obsaženo.

To vyžaduje zabývat se všemi pohledy, jakými se na danou problematiku kdo dívá. Už toto prakticky vyžaduje redukovat obecnost a omezit se jen na některé určité pohledy. Ty lze považovat za separátní ontologie. Mezi nimi ovšem můžeme uvažovat souvislostní vztahy.

---

[5] Zde se myslí explicitní vyjádření znalostí.

[6] Metamodel jazyka SHOE, určeného ke znalostní anotaci dokumentů v prostředí WWW, zahrnuje i možnost definice speciálních vztahů mezi pojmy/kategoriemi, např. „psi *mohou honit* kočky“ ([Šim]).

[7] tzv. rámce

[8] tzv. sloty

[9] a kteří samozřejmě se těší v komunitě zabývající se daným oborem dostatečné autoritě, tj. tzv. experti

V [Abe] nazývaná „informační ontologie“ popisuje *strukturní vztahy informačních zdrojů* (kniha-článek-sekce…), *jejich spolehlivost či důvěryhodnost zdrojů*, *vazby k osobám či organizačním jednotkám* kompetentním vzhledem ke zdroji (původci, recenzenti…). Je zde zahrnut například i koncept „množiny pravidel“ jako specielního typu informačního zdroje, a uvažuje se jeho vztah k nějaké podnikové aktivitě, či koncept obecné poznámky k něčemu, co je obsaženo v tzv. „podnikové ontologii“, o ní viz dále.

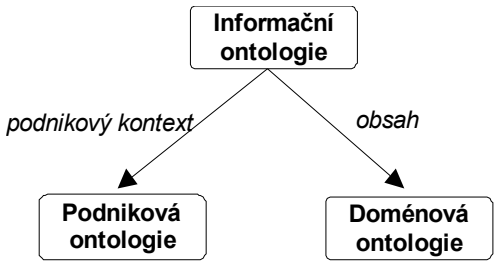
Zdrojem pro vytváření takovéto „informační ontologie“ mohou být již hotové či rozpracované veřejně dostupné ontologie v rámci internetových projektů, či standardy bibliografických metadat (viz např. [Šim]). Využitelných hotových produktů je v této oblasti velké množství.

„Informační ontologie“ se zabývá formálními vlastnostmi dokumentu či informačního zdroje.

Druhou uvažovanou ontologií v [Abe] je tzv. „podniková ontologie“, která popisuje *organizační strukturu* podniku, jak z hlediska pracovníků, tak z hlediska podnikových procesů. I v této oblasti lze využít množství hotových modelů, využívajících jednak obecné manažerské zkušenosti, ale hlavně interních podnikových dokumentů, ve kterých jsou dobře popsány. „Podniková ontologie“ se zabývá vztahy dokumentů či informačních zdrojů k organizační struktuře podniku.

Třetí částí koncepce [Abe] je tzv. „doménová ontologie“. Má se týkat samotného *předmětu činnosti* podniku. Autoři zde navrhují možnost využít některé z dostupných ontologií pro daný obor činnosti, i když přiznávají pravděpodobnou nutnost vytvářet tuto ontologii „na zelené louce“. „Doménová ontologie“ se zabývá samotným obsahem dokumentů či informačních zdrojů.

Vztah všech tří ontologií z [Abe] naznačuje obrázek 2 s vysvětlením: konkrétní informační zdroj je popsán pomocí konceptů z informační ontologie, jejichž různé atributy nabývají hodnot v konceptech z podnikové a doménové ontologie.




**obrázek 2: tři dimenze popisu informačního zdroje dle [Abe]**

### *Kritika použití ontologií pro podporu knowledge managementu*

Využití „informační“ a „podnikové“ ontologie (ve významu popsaném výše) nepředstavuje na jednu stranu velký problém, protože jsou v tomto směru dostatečně dostupné analýzy i návrhy

využitelných objektů. Na druhou stranu jejich zakomponování do projektu počítačové podpory knowledge managementu, kromě centrální evidence, přinese „pouze“ zefektivnění přístupu k informačním zdrojům.

Budování „doménové“ ontologie je podle mého kruciálním bodem celého projektu, neboť je na jedné straně nejobtížnější z hlediska množství vynaložené práce, ale hlavně je kritickým bodem přínosu celého projektu pro cíle podniku. To proto, že cílem počítačové podpory knowledge managementu typu organizational memory či knowledge repository je podpora znalostních pracovníků, přičemž největší důraz se klade na sdílení zkušeností a znalostí z *předmětu jejich činnosti*. Doménová ontologie má sloužit k systemizaci anotací *obsahů* dokumentů a jiných informačních zdrojů. Využití projektu podpory knowledge managementu má probíhat prostřednictvím zkonstruovaných ontologií, takže relevance dokumentů či jiných informačních zdrojů vůči uživatelově informační potřebě bude hledána pomocí dotazů založených na pojmech zahrnutých v těchto ontologiích. Největší přínos celého projektu je potenciálně ukryt v zpřístupnění *obsahové informace* dokumentů a jiných informačních zdrojů.

Vybudování doménové ontologie pro tento účel podle mého mínění narazí na problém *konsensu uceleného obrazu oboru činnosti*, a pokud by přesto byl takový model vytvořen, velká část z něho by mohla následně ležet ladem. Proto se mnou navrhovaná metoda anotace informačních zdrojů zakládá na jiné koncepci, i když podobné představě doménové ontologie.

I myšlenka „podnikové“ ontologie se podle mého může setkat s problémy, protože v moderní (tzv. turbulentní) době mohou být reorganizace v podniku velmi časté. Řešením by snad bylo uchovávání verzí ontologií i s jejich případnými vzájemnými vazbami.

Další inspirací pro mou práci byla konstrukce konceptuálních grafů.

## *Konceptuální grafy*

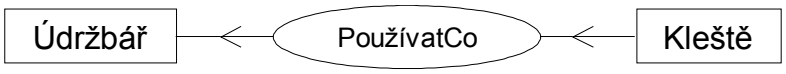
Konceptuální grafy jsou formální nástroj pro zaznamenání explicitní znalosti. Zachycují abstraktní, konceptuální strukturu jazykové podoby vyjádřené znalosti. Mohou být znázorněny graficky či zapsány v podobě jim odpovídajícího lineárního textového kódu.

Metamodel konceptuáních grafů, tj. představa o tom, z jakých typů částí a jakým způsobem se mohou konceptuální grafy vytvářet, vychází z lingvistické znalosti gramatické stavby vět přirozeného jazyka. Vychází z pojetí, že věty v přirozeném jazyce vyjadřují *vztahy mezi koncepty*. Následující obrázek 3 představuje konceptuální graf pro větu „Údržbář používá kleště.“:


**obrázek 3: příklad konceptuálního grafu**

Zde „Agnt“ znamená „agens“, tj. „ten, kdo je původce činnosti“, a „Thme“ znamená „theme“, tj. „to, co je objektem činnosti“. „Údržbář“, „Používat“ a „Kleště“ jsou koncepty, „Agnt“ a „Thme“ jsou konceptuální vztahy. Částečně formální výklad konstrukce konceptuálních grafů obsahuje Dodatek 1.

Teorie konceptuálních grafů zahrnuje jisté množství primitivních typů konceptuálních vztahů, jako jsou „Agnt“ a „Thme“ nahoře, ale je možno definovat nové typy vztahů (viz Dodatek 1). Pro příklad uvedený výše by bylo možno definovat vztah „PoužívatCo“ a vytvořit konceptuální graf, viz obrázek 4:




**obrázek 4: jiná verze konceptuálního grafu**

Tato verze je zřejmě srozumitelnější, lépe čitelná pro eventuální uživatele systému, který by měl pomocí konceptuálních grafů komunikovat s uživateli.

Konceptuální grafy jsou paralelní vývojovou linií[10] k vývoji tzv. konceptuálních modelů v databázové teorii, v němž jsou významnými milníky např. modely ER(A), NIAM, UML[11]. Podle mého názoru je zde patrný společný původ v univerzální abstraktní gramatické struktuře přirozeného jazyka, který je univerzálním jazykem k vyjadřování jak znalostí, tak fakt (i jiného). Obě cesty, „znalostní“ i „databázová“ se setkávají ve vzájemné přeložitelnosti konceptuálních grafů a modelů NIAM (viz str. 59).

Ve své práci předpokládám využití konceptuálních grafů se speciálními druhy konceptů: existenční koncepty bez určení i s určením (s designátorem), koncepty kvantifikované množstvím a množinové koncepty (kolekce) – příklady viz dále. Také předpokládám využití konceptů s vnořenými grafy, tzv. „kontextů“ v terminologii konceptuálních grafů.

Prve uvedený příklad (obrázek 3, obrázek 4) obsahuje existenční koncepty bez určení. Příklad konceptuálního grafu s existenčním konceptem s určením je následující :


**obrázek 5: konceptuální graf s konceptem typu „Údržbář“ s určením instance tohoto typu „Jan Novák“**

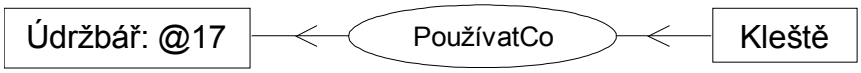
[10] v umělé inteligenci, konstrukce k budování znalostních bází, k zaznamenání explicitní znalosti

[11] Záměrně vynechávám jistou skupinu modelů, jejímiž představiteli jsou např. SDM či HIT. Jejich původ není v analýze přirozeného jazyka, ale v analýze sémantických konstrukcí.

V následujícím příkladu jsou oba koncepty určené (s určením):


**obrázek 6: CG z předchozího obrázku s určením konceptu typu „Kleště“ instancí s ident. kódem „K457“**

Příklad konceptu kvantifikovaného množstvím je následující:

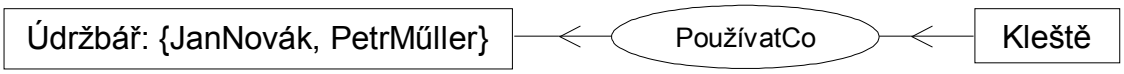



**obrázek 7: „Existuje 17 údržbářů, kteří používají kleště.“; či prostě: „17 údržbářů používá kleště.“**

a jiná verze s určením konceptu „Kleště“:


**obrázek 8: „17 údržbářů používá kleště K457.“**

Dále uvádím příklad konceptuálního grafu s množinovým konceptem:




**obrázek 9: „Údržbáři Jan Novák a Petr Műller používají kleště.“**

Zde je potřeba vysvětlení a úmluva pro další výklad: Konceptuální graf viz obrázek 9 znamená, že *oba* údržbáři, Jan Novák i Petr Műller, používají kleště. Důležitost této úmluvy je vidět v následujícím příkladu:


**obrázek 10: „Údržbáři Jan Novák i Petr Műller oba používají jak kleště K457 tak i kleště K129.“**

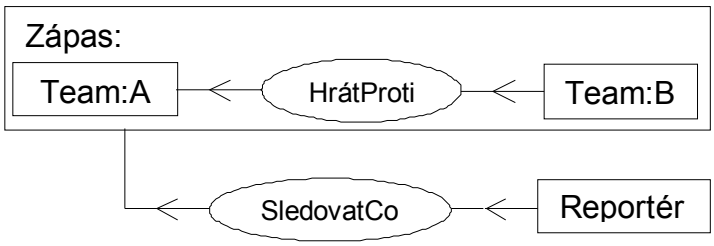
Tedy, je-li v konceptuálním grafu množinový koncept, interpretuji ho tak, že všechny instance uvedené v určení tohoto konceptu vstupují do všech vztahů s instancemi z určení konceptů, připojených k odpovídajícímu konceptuálnímu vztahu.

Výslovně zde uvádím, že pro svou práci nepředpokládám využití univerzálně kvantifikovaných konceptů ve významu podle [Sow] (viz Dodatek 1). Určitý typ univerzálního referentu však míním využívat.

Koncepty s vnořenými konceptuálními grafy předpokládám využít ve složitých kontextech[12], ve kterých některé jevy závisí na určitých sub-kontextech. Příklad takového konceptu viz obrázek 11.

[12] Slovo „kontext“ v této práci používám ve významu: „situace, která provází nějaký jev“, na rozdíl od významu [Sow].

Důvodem k použití takto složité struktury je, že reportér může sledovat více různých zápasů, různí reportéři sledují různé zápasy… Globální kontext bude zachycovat, kteří reportéři sledují které zápasy.




**obrázek 11: Reportér sleduje zápas, ve kterém tým A hraje proti týmu B.**

Narozdíl od [Sow] budu používat i koncepty, jež budou hybridy množinových konceptů a konceptů s vnořeným konceptuálním grafem, tzn. designátorem konceptu může být dle mne i množina konceptuálních grafů.

### *Použitelnost konceptuálních grafů pro podporu knowledge managementu*

Původním účelem vytvoření teorie konceptuálních grafů bylo zaznamenat explicitní znalosti pro automatické zpracování ve znalostní bázi. Tento účel samozřejmě respektuji, ale pro svou práci zamýšlím jiné využití. Toto má být využití konceptuálních grafů pro *anotaci informačních zdrojů*, kterážto anotace bude *záznamem kontextu* vzniku či použitelnosti informačního zdroje.

Takový kontext nechci zaznamenávat pouze nominálně, tj. vytvářením jakési hierarchie či ontologie možných kontextů, a následným zařazováním informačních zdrojů k nim[13]. Chci umožnit vyjádření dynamické stránky kontextu, vyjádření, že v kontextu došlo nebo dochází k určitým událostem, že kontextem je určitá obchodní (business) situace. Vycházím z názoru, že podnik je hlavně zařízením pro provádění činnosti, a tato činnost je pro knowledge management ohniskem zájmu.

Zaznamenané kontexty ve formě konceptuálních grafů vždy mohou být zpracovávány odvozovacím strojem znalostní báze, pokud to bude účelné a možné. – Já však předpokládám využití kontextových anotací k *vyhledávání informačních zdrojů*. Moje představa modelu vyhledávání spočívá v tom, že informační požadavek, tzv. *dotaz, bude položen také jako kontext*, a vyhledávány budou zdroje, které tomuto požadovanému kontextu (nejlépe) odpovídají.

Takovýto mnou navržený způsob podpory vyhledávání může být kombinován s jinými nástroji k vyhledávání.

Poslední inspirací, které zde uvádím, je metoda i nástroje ORM.

[13] běžná kategorizace v knihovnických a jiných systémech

## *ORM*

ORM (Object Role Modeling) je verzí modelovací metody NIAM (Natural-language Information Analysis Method), s kterou jsem se blíže seznámila, a která jako jediná z verzí NIAM má podporu v programových CASE nástrojích.

ORM je primárně metodou pro modelování a dotazování informačního systému[14] na konceptuální úrovni, zahrnuje ale také procedury pro transformaci konceptuálního modelu do logického databázového modelu. Jako verze NIAM, ORM se snaží stavět modely na konstrukcích přirozeného jazyka a umožňuje vyjádřit model i dotazy do informační báze v „přirozeném" jazyce[15] jako jednu z alternativ.

Z bohatých možností syntaktických konstrukcí přirozeného jazyka se ORM omezuje pouze na konstrukty *objektů* (což v konceptuálních grafech odpovídá existenčním konceptům bez designátorů) a *vztahů*, v nichž tyto objekty hrají *role*. V tomto se tedy podobá principům tvorby konceptuálních grafů. Obojí je založeno na poznatku, že ostatní konstrukce vět přirozeného jazyka lze z těchto dvou odvodit, nebo lépe řečeno je lze jimi nahradit, opsat (viz závislostní syntaktické struktury v [Stro], str.149-157).[16] Specielně je tedy třeba zdůraznit, že ORM *nepoužívá* při tvorbě modelů *konstrukty atributů*, narozdíl od jiných známých metod a jazyků datového modelování. Tento na první pohled nedostatek se stává výhodou v případě nutnosti přizpůsobit konceptuální model světa zájmu nově vzniklému pohledu, novým tzv. uživatelským požadavkům. V nich často dochází k nutnosti změnit konstrukci atributu na konstrukci vztahu k nějakému objektu/entitě (např. atribut „země_narození" je nutno nahradit vztahem k entitě „země"). Jiným přínosem přístupu vyhýbajícímu se konstruktům atributů je unifikovaný přístup k vyjadřování integritních omezení a dotazů[17].

Nevýhodou ORM modelů plynoucí z absence konstruktu atributů je poměrná rozlehlost modelů. Tu lze zmírnit mechanizmy, které abstrakcí tvoří kompaktnější modely. To je také zastánci ORM doporučovaná metoda odvození ER(A) nebo UML modelů z ORM modelu. Tedy ER či UML podobu nabízejí jako *pohled* na ORM model.

(ORM také nabízí bohatou škálu grafických značení pro rozličné typy integritních omezení modelu. Do jisté míry tím umožňuje zahrnovat do modelu i sémantické konstrukce.)

---

[14] Ačkoli některá rozšíření ORM byla navržena pro modelování procesů a událostí, střediskem zájmu ORM je datové modelování.

[15] ve strukturovaném jazyce, v němž vyjádřené věty mají podobu vět přirozeného jazyka

[16] Odpovídající podoba v přirozeném jazyce faktů vyhovujících modelu ORM se skládá z jednoduchých vět, eventuelně pospojovaných slovy jako „který", „jenž".

[17] Poslední výhodou je snadná možnost ověřování modelu na základě *populace* příkladů.

Velkou pozornost věnuje ORM otázce *atomů modelu*, kterým říká „elementární typy faktů“. Jsou to takové pod-modely (fragmenty) modelu ORM, které nelze rozložit na menší části bez ztráty informace v zaznamenávaných faktech. (Bezesporu lze model vždy rozložit na množinu fragmentů, z nichž každý obsahuje pouze jediný vztah a objekty, které v tomto vztahu hrají role[18]. To znamená, že otázka elementarity je otázkou atomičnosti takovýchto fragmentů.) Tento důraz na elementaritu typů faktů (typů vztahů) je veden snahou o odstranění redundance faktů v informační bázi, o adaptabilitu k informačním požadavkům a o maximální nezávislost modelu na implementaci. Následující příklady nejsou elementární fakty:

Údržbář Jan Novák byl přijat Josefem Suchým dne 13.5.1992.
Údržbář Jan Novák dnes třikrát použil kleště.

První z nich může být rozdělen na následující elementární fakty:

Jan Novák je údržbář. Jan Novák byl přijat Josefem Suchým. Jan Novák byl přijat dne 13.5.1992.

Druhý z nich takto (zde je použito hnízdění 2.faktu ve 3.faktu):

Jan Novák je údržbář. Jan Novák dnes použil kleště. Toto použití bylo trojnásobné.

Naopak následující je elementární fakt (za předpokladu, že jeden zaměstnanec má v jednom dni jedinou helmu, a s nikým se o ni nedělí):

Jan Novák má dnes helmu č. H4387.

Uvážíme-li informační funkce (fragmenty dotazů):

Zaměstnanec, který měl 14.3.2002 helmu č. H1278,…
Helma, kterou měl Petr Műller předevčírem,…

pak vidíme, že nelze rozumně použít „hnízdění“[19] jako v případě trojnásobného použití kleští nahoře. Pro první typ dotazů by se hodilo hnízdění:

Helma(kód) byla ve Dni(datum) použita. Toto použití realizoval Zaměstnanec(jméno).

Pro druhý typ dotazů by se hodilo:

Zaměstnanec(jméno) Dne(datum) použil helmu. Použitá helma byla Helma(kód).

### *Objektivizace vztahů, hnízdění v ORM*

Objektivizace vztahů v datovém modelování se podobá konstrukci tzv. kontextů v konceptuálních grafech. Datové modelování však nepřipouští zahnízdění – objektivizaci – ničeho jiného než

[18] pokud některým z těchto objektů je objektivizovaný, tj. zahnízděný, vztah, pak do příslušného pod-modelu patří i objekty hrající role v tomto zahnízděném vztahu

[19] vybrat jedno z několika možných hnízdění, a tím ostatní potlačit

jediného vztahu, což v konceptuálních grafech odpovídá připustit zahnízdění pouze kontextů designovaných hvězdovými grafy (pro vysvětlení pojmů viz Dodatek 1).

Formálně jde o to, že nějaký vztah v modelu považujeme také za objekt, který může vstupovat do dalších vztahů. Z hlediska lingvistické syntaxe se jedná o nominalizaci predikátu[20].

ORM značení, konstrukce a metody jsou zevrubně popsány v [Hal], další doplňující texty je možno nalézt v [orm], [Hal] obsahuje reference na další odbornou literaturu o ORM. Dodatek 2 v mé práci ORM stručně popisuje.

### *Použitelnost ORM pro knowledge management*

ORM je svébytná metoda pro datové modelování pro strukturované databáze. Jako taková jistě může sloužit k podpoře knowledge managementu, a to hlavně

- přispěním k porozumění datovému modelu uživateli
- možností dotazovat databázi prostřednictvím konceptuálního modelu, a v jazyce, který je snadný k užití
- přispěním ke stabilitě datového modelu, a tím k přetrvání informačních funkcí (dotazů).

Tedy může výrazně přispět k informační či znalostní zužitkovatelnosti transakčních procesních systémů (TPS). Tento její příspěvek však není hlavním středem zájmu pro mnou navrhovanou metodu podpory knowledge managementu. Mým záměrem je použít ORM jako mezistupeň pro implementaci anotací informačních zdrojů. Mojí metodou bude návrh „business“ konceptuálního grafu, jeho překlad do modelu ORM, interpretace konkrétních anotací v modelu ORM, eventuelně dotazování prostřednictvím ConQuer – dotazovacího jazyka ORM.

Zamýšlený „business“ konceptuální graf může obsahovat i koncepty obsažené v modelech pro TPS, a pak je možno navrhnout interpretaci dotazu nad „business“ CG i jako dotaz do databází TPS.

[20] Přirozený jazyk používá různé konstrukce pro vyjádření objektivizovaného vztahu (predikátu), například vedlejší věty, ale datové modelování se omezuje na malý počet konstruktů.

# Popis metody

Základní plán metody

Návrh business konceptuálního grafu

Transformace business CG do ORM

Tvorba anotací nad business CG

Dotazy

XML záznam anotací

# Základní plán metody

Cílem mé práce je navrhnout metodu anotace informačních zdrojů, založenou na strukturaci anotací do podoby konceptuálních grafů. Tyto konceptuální grafy mají vyjadřovat *kontexty vzniku a nebo kontexty užití informačních zdrojů*.

Otázku, zda je možno předjímat užití informačních zdrojů, a pokud ano, do jaké míry, ponechávám stranou své pozornosti, tím se zabývají projekty komplexní podpory knowledge managementu (viz např. [Mit]). Rozhodně je formulování předpokládaného užití obtížnější než prosté zaznamenávání kontextů vzniku informačních zdrojů. Ale u zdrojů, které jsou nositeli expertní znalosti[21], je formulování toho, jak má vypadat kontext užití, podstatné. Já zde nadále nebudu rozlišovat mezi kontextem vzniku a kontextem užití, a budu používat výraz „kontext informačního zdroje“.

Smysluplnost celé metody je podmíněna možností modelovat a strukturovat „business kontext“, generalizovaný souhrn abstraktních kontextů – business situací, procesů a objektů – k nimž lze dostupné informační zdroje připojovat jako nosiče potenciální užitečné znalosti. (Pro příklad viz Příklad (univerzita) na str.28.)

Návrh takového *modelu kontextů* jistě vyžaduje vhodné postupy analýzy, z nichž některé mohou být znovupoužity z již známých zkušeností a metodik informační analýzy pro návrh strukturovaného modelu informačního systému, ale jistě bude třeba vytvořit i postupy nové. Jako možné východisko pro analýzu mohou například sloužit otázky: „V jakých situacích či kontextech byste ocenili pomoc nabídkou dostupných informačních zdrojů? Charakterizujte tyto situace! Charakterizujte business situace, ve kterých byste ocenili informaci o odpovídajících dostupných informačních zdrojích!“

Protože odpovědi na předchozí navržené otázky se mohou s časem měnit, musí celá metodika *umožňovat vývoj* business kontextového modelu. Použité realizační technologie musí být schopné adaptace na změnu základního konceptuálního návrhu. Předpokládám hlavně změny *přidáváním konceptů a konceptuálních vztahů*, odstraňování částí modelu považuji spíše za patologické, připouštím je ale také.

Prostředkem k zaznamenání business kontextového modelu má být *konceptuální graf*. Tento konceptuální graf má obsahovat koncepty pouze existenční, bez určení referentů. Tento graf bude dále mnou navrženým postupem přeložen do ORM modelu. Vzniklý ORM model může být metodami vlastními ORM mapován do logického návrhu databáze, a takto může být realizována implementace celého systému. Nebo je také možno vynechat použití prostředků ORM a vytvořit návrh implementačních struktur přímo.

[21] explicitní znalosti

*Anotační záznamy do databáze* budou přidávány uživateli komunikujícími prostřednictvím business konceptuálního grafu. V něm ve zvolených konceptech uživatelé zvolí odpovídající kvantifikátory a referenty. K tomuto účelu lze vytvořit vhodné uživatelsky přívětivé prostředí využívající nabídky již exitujících instancí konceptů[22]. Takovýmto „instancováním" business konceptuálního grafu se definuje anotace informačního zdroje, které pak bude odpovídat nový záznam[23] v databázi anotací.

*Vyhledávání zdrojů* má také probíhat prostřednictvím business konceptuálního grafu. Uživatel v něm bude „instancovat" některé koncepty v grafu a takto charakterizovat svůj kontext. To bude interpretováno jako dotaz na informační zdroje, jejichž anotace odpovídá uživatelem zadanému kontextu. Otázkou, co znamená ono „odpovídá", se ve své práci také zabývám. V první fázi nepředpokládám tvorbu nějaké náhražky uživatelova uvažování, neboť mnou zamýšlenými uživateli mají být tzv. znalostní pracovníci. Přepokládám tedy, že uživatel interaktivně zadá takové dotazy, jež budou dostatečně obecné, aby poskytly odpověď, a zároveň dostatečně přesné, aby odpověď byla relevantní.

V pozdější fázi je možno vytvořit nějaké náhražky uživatelovy tvůrčí invence, a pro dotazy, které neposkytnou odpověď (množina odpovídajících informačních zdrojů bude prázdná), je možné navrhnut typové uvolnění určitých podmínek dotazu, tj. uvolnění instancí v určitých konceptech. – Takováto konstrukce umělé inteligence jistě vyžaduje další analýzu business kontextu.

Prostředníkem k definici dotazu do databáze anotací může být i ConQuer, dotazovací jazyk ORM. O něm viz Dodatek 2.

[22] Tj. existujících referentů a event. dalších ozřejmujících atributů (jména, obrázky…) objektů, na něž tyto referenty odkazují.

[23] z hlediska logické databázové struktury může jít o více záznamů

# Návrh business konceptuálního grafu

Business konceptuální graf má zachycovat popis typových kontextů, které budou sloužit k anotaci informačních zdrojů. Doporučovaný postup analýzy od příkladů k typům lze v i tomto případě přijmout, přičemž začátkem by, dle mého názoru, měl být sběr příkladů vět v přirozeném jazyce popisujících určitý konkrétní kontext.

Poté je třeba věty seskupit do typových skupin, jež sice nemusí mít stejnou gramatickou strukturu, ale mají stejnou sémantickou strukturu a dají se do jednotné gramatické struktury přeformulovat beze změny smyslu. Touto jednotnou gramatickou strukturou myslím mimo jiné také to, že typy pojmů (konceptů) a predikátů (konceptuálních vztahů) budou na stejných místech ve struktuře typové skupiny stejné (např. „Údržbář … používá kleště ….“). Poté je třeba vybrat jednu ze všech do úvahy připadajících typových struktur („Údržbář … používá kleště ….“ nebo „Opravář … zachází s kleštěmi ….“) jako konsensus pro komunikaci uživatelů s budoucím systémem[24].

Protože konkrétní kontext může být vyjádřen skupinou vět, v nichž se některé pojmy z jedné věty odkazují na jiné pojmy z jiných vět (koreferenční propojení, v češtině vyjádřené např. vazbou „který“), je třeba vyřešit volbu typových struktur vět ve vzájemné závislosti na možné takové odkazy.

Poté může následovat rozklad typových vět do struktur konceptuálních grafů, jeden konceptuální graf s existenčními kvantifikátory bez určení referentů pro jednu typovou skupinu vět. Zde je třeba domluvy s uživateli o *pojmenování konceptů a konceptuálních vztahů*[25]. V případě, že jeden koncept má stejný pojmový obsah jako jiný koncept v modelu, je vhodné pro ně zvolit stejná pojmenování. Pokud je přesto pro lepší porozumění vhodné použít různá pojmenování, stávají se tato pojmenování synonymy, a tato synonymie je evidována.

Dále je navíc možno přidat *pojmenování rolí* v konceptuálních vztazích, například pojmenováním hran grafu.

Velkou nevýhodou pro užití navrhované metody v češtině je to, že pro přirozené porozumění strukturám grafu jsou důležité vhodné tvary slov. Toto je možno řešit typovými překlady konkrétního konceptuálního grafu do češtiny či jejich přizpůsobením situaci uživatelovy interakce, v závislosti na referentech konceptů[26].

---

[24] Pravděpodobně je možné navrhnout několik alternativních variant, které budou vyhovovat různým skupinám uživatelů.

[25] odpovídajícím neutrálním tvarům slov abstrahovaných z vět

[26] Například využití informace o mluvnickém rodu, nebo množném či jednotném čísle.

Některá fakta jsou zjistitelná z jiných částí informačního systému, například z TPS. Pokud je při analýze zjištěno, že součástí některého popisu typového kontextu jsou takové typy fakt, je toto evidováno.

## *Porovnání s datovým modelováním pro strukturované databáze*

Hlavní rozdíl spočívá v tom, že model *ne*představuje přehled typů jednotlivých zaznamenávaných faktů, ale typ zaznamenávaných *komplexů faktů*.

Specifickou zvláštností je, že v konkrétním business kontextu se mohou určité typové situace vyskytovat vícenásobně, což znamená, že v business konceptuálním grafu některé pod-grafy budou moci být pro konkrétní anotaci použity vícenásobně. Je možné, že se v aplikační oblasti takové pod-grafy dají předem vytipovat.

Někdy může být vyžadována koreference mezi koncepty z různých faktů. Pokud tomu tak není, různé koncepty téhož typu v různých faktech nejsou obecně totožné (mohou být instancovány jinak).

*Uživatelská srozumitelnost modelu, jeho pochopení a přijetí*, je důležitější než minimalita vzhledem k účelu tvorby datového modelu. Například není třeba vylučovat takové typy konceptů, jež nemají vhodnou extenzi, tzn. že v konkrétních business kontextech připadá do úvahy pouze jeden jediný výskyt, jediná instance v tomto konceptu. „Náprava“ se pak provede při následujícím překladu modelu.

Ze stejného důvodu je možno do modelu zahrnovat i vztahy (konceptuální relace), které jsou odvoditelné z jiných vztahů v modelu. Při překladu do datového modelu se použije zaznamenaná definice odvození.

Podobně s odvozenými koncepty.

Péče o *elementaritu typů faktů* (elementaritu konceptuální relací) je důležitá pro zachování ortogonality těch částí modelu, které představují nezávislé části modelovaného business kontextu. Znamená to *žádoucí volnost při tvorbě kontextových anotací a dotazů.* Některé otázky elementarity konceptuálních relací jsou rozebrány v Dodatku 3, v části Elementarita konceptuálních relací.

## *Formální struktura business konceptuálního grafu*

### *Koncepty*

Koncepty se připouštějí pouze existenční bez určení referentů (s jedinou výjimkou popsanou dále v odstavci Hnízděné koncepty), to znamená, že se definují *pouze typy konceptů*. V business konceptuálním grafu se může vyskytovat více konceptů téhož typu, tj. jakoby jeden a tentýž koncept byl v modelu na více místech.

Pro definici typu je předně třeba zvolit vhodné jméno typu (či alternativní jména pro různé uživatele[27]) a pak je dokumentačně popsat, výstižně a srozumitelně, což lze následně nabídnout při uživatelské interakci s modelem jako vysvětlení na požádání. Jeden a tentýž typ konceptu může mít synonymní jména.

U konceptů evidujeme, zda se u nich při užití v anotacích připouští kvantifikování množstvím nebo kolekcemi. Takové koncepty nazvu *množinovými koncepty*.[28]

Také speciálně označíme ty typy, jež mají jedinou možnou instanci (viz výše poznámka k modelování).

Typy konceptů, které jsou v business konceptuálním grafu odvoditelné z jiných typů, je třeba definovat, např. za pomocí λ-výrazů (viz str.58 zde). Koncepty těchto typů nazvu *odvozenými koncepty*[29].

### *Konceptuální relace*

Konceptuální relace doporučuji používat spíše „odvozené“ dle terminologie CG v [Sow] uvedené na str.13 zde, tj. takové relace, kterým uživatelé *přirozeně rozumí* a používají je.

Konceptuální relace, které lze v business konceptuálním grafu odvodit z jiných relací *přítomných v grafu*, je třeba definovat, např. za pomocí λ-výrazů (viz str.58 zde). Nazvu je *odvozenými konceptuálními relacemi29*.

Pro uživatelské návěští konceptuální relace (její pojmenování) doporučuji oproti zvyklostem naznačeným v [Sow] použít slovesné tvary, nikoli jmenné, protože slovesné tvary jsou lépe čitelné v souvislostech grafu. Doporučuji nabídnout různá slovesná pojmenování pro různé směry čtení (u binárních relací to jsou dva směry čtení: „Údržbář používá Kleště“ – „Kleště jsou používány Údržbářem“). Problémy s ohebností češtiny lze řešit přizpůsobením návěští relací a jmen typů konceptů situaci uživatelovy interakce.

V modelu typového konceptuálního grafu lze použít návěští neutrální. – Péči o tvarosloví lze chápat jako nadstavbu modelu.

Podobně jako koncepty, i konceptuální relace dokumentačně popíšeme, a i tento popis může sloužit jako vysvětlení na požádání.

---

[27] Jména se nemají v modelu opakovat, pokud nejde o stejný typ. Tedy jméno má typ identifikovat.

[28] Koncepty, jež by měly být vždy kvantifikovány pouze množstvím, jsou chybně pojaté. Koncepty, jež lze kvantifikovat kolekcemi, v mém pojetí má smysl připouštět i kvantifikovatelnými množstvím, protože zamýšlím umožnit i designování konceptů neznámými instancemi.

[29] Vývojem business konceptuálního grafu se mohou některé původně neodvozené objekty stát odvozenými.

### *Role*

Oproti konstrukcím konceptuálních grafů popsaným v [Sow] navrhuji hrany v grafu eventuelně označit pojmenováním rolí, jež v příslušných vztazích hrají instance konceptů, připojených těmito hranami ke konceptuálním relacím. To přispěje k uživatelské srozumitelnosti, a při interakci uživatele s modelem je možno zobrazení jmen rolí na požádání zapínat a vypínat.

### *Hnízděné koncepty*

Zvláštní pozornost je třeba věnovat „kontextům“ dle terminologie [Sow] popsané zde na str.56. Jde vlastně o zahnízděné grafy v celkovém konceptuálním grafu. Protože celý model nemá připouštět tvorbu libovolných anotací, ale jen typových, musí být takovéto konceptuální grafy, zahnízděné v nějakém konceptu celkového konceptuálního grafu, také předem typově určeny. Toto tedy tvoří jedinou výjimku zmíněnou dříve v odstavci Koncepty: designátory zahnízďujících konceptů musí být v modelu *typově* definovány.

Pokud by nebylo možno při anotacích některé části grafu použít vícenásobně, bylo by hnízdění zbytečné, neboť kontext relací, v nichž zahnízděný koncept hraje roli, by byl jasný. Kvůli možnému opakování je však nutno zahnízdění využít pro evidenci souvislostí mezi sub-kontexty.

V datovém modelování je zvykem, že hnízděné objekty se pojmenovávají jmenným tvarem objektivizovaného predikátu, pojmenování může však být i jiné (viz obrázek 11 na str.15); odpovídající pojmenování se projeví jako *pojmenování typu* hnízděného konceptu.

Pro zvýšení přehlednosti je možno připustit, že některé koncepty, jež jsou koreferenčně propojeny (viz následující odstavec Koreferenční propojení) s nějakým hnízděným konceptem, mohou mít prázdný referent (při zobrazení pro uživatele), ale znamená to, že jejich referentem je tentýž konceptuální graf.

Ačkoli [Sow] to výslovně nepotvrzuje, je nutno umožnit „množinové kvantifikování“ konceptů se zahnízděnými grafy, tj. umožnit instancování takovýchto konceptů vícenásobně. Takto vytvořené grafy zahnízděné v jediném konceptu všechny najednou vstupují do relace, k níž je hnízděný koncept připojen. (K takovémuto množinovému instancování zahnízděného CG je třeba vytvořit vhodný interface.)

### *Koreferenční propojení*

V modelu není třeba přímo zobrazovat vztahy *podtyp/nadtyp* (ISA) mezi typy konceptů, ale tuto informaci zahrneme do modelu jako „neviděnou“. Navíc je možno uvažovat a zahrnout další vztah mezi typy konceptů, a to *typovou příbuznost*, jež znamená, že typy mohou mít společné instance. Smysl takového vztahu je srovnatelnost instancí obou typů.

Oba zmíněné vztahy (podtyp/nadtyp, příbuznost) se projeví jako integritní omezení pro konkrétní tvorbu anotací. Tato omezení budou působit v situacích, kdy se uživatel pokusí spojit dva koncepty v modelu koreferenčním propojením. (Zároveň může působit jako nápověda, kdy uživatel začne koreferenční propojení vytvářet. V takovém případě mohou být ukázány, např. zvýrazněním, ostatní koncepty s použitým konceptem typově příbuzné.)

Narozdíl od ORM, jeden a tentýž typ konceptu se v modelu může opakovat, aby uživatel při tvorbě anotace mohl zvolit různé referenty pro různé konceptuální relace, v nichž koncepty tohoto typu hrají role. Jeden a tentýž koncept se může v modelu opakovat také proto, aby by model přehledný, potom jsou opakování tohoto konceptu propojena koreferenčním propojením.

Opakování konceptu v jediné anotaci může také nastat vícenásobným použitím nějaké části konceptuálního grafu.

Koreferenční propojení, vytvořené při anotaci, mezi různými koncepty téhož typu konceptu je samozřejmě přípustné. (Použitou terminologii můžeme zjednodušit tak, že koncepty téhož typu v modelu budeme považovat také za typově příbuzné.)

Množinové koncepty vyžadují zvláštní objasnění: ISA vztahy jsou možné i mezi dvěma koncepty, z nichž jeden je a druhý není množinový, ISA vztah se týká *typu* konceptu. Koreferenční propojení je ovšem možné pouze tehdy, když jsou množinové oba propojované koncepty, nebo žádný z nich. Koreferenční propojení znamená identitu kolekcí instancí obou množinových konceptů.

Je možno si představit pojem typové příbuznosti či pojem ISA vztahu i mezi hnízděnými koncepty založený na množinových operacích se zahnízděnými konceptuálními grafy, ale pro praktickou aplikaci se mi to zdá být příliš složité.[30]

V business konceptuálním grafu mohou být některé koncepty propojeny koreferenčním propojením „napevno“. To pak znamená, že pokud uživatel použije při anotaci oba propojené koncepty, musí být jejich instancování stejné.

Vztahy ISA, eventuelně i typová příbuznost, se uplatní při tvorbě datového modelu pro business konceptuální graf.

### *Použití odvozených konceptů a konceptuálních relací*

Obecně, *odvozené konceptuální relace jako takové není možno použít při tvorbě anotací* – lze je použít pouze při formulaci dotazů.

*Odvozený koncept* je možno při anotaci použít jedině tehdy, když je předchozím instancováním v anotaci již *definován* – toto si lze přestavit jako integritní omezení pro tvorbu anotací. (Je možno

[30] Ideu, na které by bylo možno takové pojmy postavit, poskytují poznámky, na str. 80.

si představit takovou funkci uživatelského interface, která při pokusu instancovat odvozený koncept přiměje uživatele instancovat nejprve ty koncepty, jež jsou potřebné pro odvození tohoto konceptu.)

Úmluva: *Budu-li nadále hovořit o* business konceptuálním grafu*, budu tím myslet jeho pod-graf tvořený neodvozenými relacemi. Jako o* konceptuálních relacích *budu hovořit pouze o neodvozených konceptuálních relacích, jinak budu výslovně hovořit o* odvozených konceptuálních relacích.

### *Atomy grafu, integritní omezení*

Zcela obecně, k jedné anotaci je možno použít kterýkoli hvězdový pod-graf business konceptuálního grafu samostatně a vícenásobně. Ale některé konceptuální relace mohou být spolu v aplikační doméně svázány natolik, že v jakémkoli konkrétním reálném kontextu se musí vyskytovat vždy společně. Pak pod-graf jimi tvořený představuje *atom* pro použití k anotacím.

Atomy dělí business kontext na celistvé, nedělitelné části.

Všechny atomy budou definovány. – Atom je dán výčtem konceptuálních relací, jež zahrnuje, tedy je podmnožinou množiny všech konceptuálních relací v business konceptuálním grafu; atomy tvoří rozklad této množiny[31]. Pod-grafy odpovídající různým atomům však nemusí být disjunktní, mohou mít společné koncepty. Konstrukty atomů tak mimo jiné vyjadřují logické vztahy mezi členstvím konceptů v konceptuálních relacích.

Při instancování konceptu, který není hnízděný[32], musí uživatel vybrat některý atom, v jemuž odpovídajícím pod-grafu koncept leží, a v tomto atomu instancovat všechny připojené koncepty[33].

Pokud v konkrétním atomu musí být instancování některých konceptů totožné, namísto koreferenčního propojování je vhodné tyto koncepty sloučit do jediného konceptu, a vytvořit souvislý pod-graf. Pro účely srozumitelnosti však toto sloučení nemusí být vhodné, např. jestliže tyto koncepty jsou různých, typově příbuzných typů[34].

Pokud lze pro jednotlivé atomy nalézt smysluplná jména, uděláme to, a jména atomů zaznamenáme. Jinak zvolíme pro atomy nějaké umělé identifikátory.

[31] Otázku atomů a zahnízděných konceptuálních grafů řeší část Atomy na str. 79.

[32] V terminologii ORM to lze vyjádřit tak, že pouze hnízděné koncepty mohou být „líné“, tj. nezávislé. U ostatních konceptů je třeba specifikovat, jakou roli hrají v kontextu anotace. Hnízděný koncept sám o sobě není ničím jiným , než zahnízděným kontextem. Pravidlo ne-lenosti se pak induktivně aplikuje na zahnízděný kontext…

[33] V kapitole Tvorba anotací nad business CG popíšu, jak uživatel instancuje koncept, ve kterém má na mysli libovolnost. Neuvažuji ovšem, že by anotace vyjadřovaly pravidla. Anotace může případně vyjadřovat antecedent pravidla, jehož sukcedent je obsažen v anotovaném informačním zdroji.

[34] Ve skutečnosti by v takovém případě bylo možno zvolit pro typ uvažovaných konceptů průnik dotyčných typů. Takovýto typ však může být umělý, uživatelům nesrozumitelný, nebo zbytečný. Speciální případ je, když propojené koncepty jsou v ISA vztahu. Pak by bylo možno zvolit typově užší z nich.

Je možné si představit další, složitější integritní omezení pro anotace, například vzájemně se vylučující použití nějakých částí modelu za určitých podmínek, či nutnost použít alespoň jednu část z nějaké množiny částí za jiných podmínek. Všechna taková integritní omezení je možno do modelu zahrnut, a při tvorbě anotací je aplikovat. Pro obě výše zmíněná omezení lze zvolit případné vhodné vizuální zvýrazňování v situacích uživatelovy interakce, ve kterých instancováním některého konceptu grafu uživatel omezení aktivuje. Všechna integritní omezení mají být uživateli neviditelná do té doby, než nastanou podmínky jejich uplatnění.

#### *Fakta z TPS*

Některé konceptuální relace zahrnuté do business konceptuálního grafu představují typy faktů, zjistitelných z databází TPS[35]. Takovéto relace budou zvlášť označeny, včetně toho, jak a zda vůbec, budou při konkrétních anotacích naplňovány daty z TPS[36].

### *Textová podoba business konceptuálního grafu*

Jde vlastně o textovou podobu uživatelského interface. Model sám bude zřejmě kódován textově (viz str.54, více [Sow]).

Podobně jako je tomu v uživatelském nástroji InfoAssistant pro tvorbu dotazů v jazyce ConQuer (viz [orm]), je možné vytvořit „textové“ uživatelské prostředí pro tvorbu anotací a dotazů. Uživatelská interakce začne nabídkou typů konceptů, jež lze instancovat. Uživatel si vybere, vybraný typ instancuje textovou podobou zápisu designátorů. Poté jsou mu nabídnuty všechny konceptuální relace, v nichž tento typ konceptu hraje roli. Uživatel si vybere, při anotaci instancuje ve vybrané relaci všechny koncepty, jež v ní hrají role – integritní omezení se projeví nabídkou dalších relací a nutností je instancovat. Uživatel pak může „koreferenčně“ (podobně propojovacímu znaku ⌊ v jazyce ConQuer) pokračovat u vybraných konceptů, které instancoval, nebo může znovu začít nabídkou typů konceptů. Nějaký druh označení spojuje instancované konceptuální relace do atomů, může to být např. svorka. Při dotazování instancuje uživatel pouze ty koncepty, jež chce specifikovat – o dotazování v textové podobě viz Textové „outline“ dotazování nad business CG na str.46.

### *Příklad (univerzita)*

Příklad zde uvedený neilustruje zcela ty rysy mnou navrhované metody, které ji činí vhodnou pro podporu knowledge managementu. Moje volba tohoto příkladu je dána tím, že prostředí univerzity

---

[35] Fakta v TPS mohou být historická či aktuální, některá poskytují identifikační vztahy, jiná vztahy m:n…

[36] Je na zvážení celkové koncepce IS v podniku, zda data budou udržována redundantně v TPS systémech i v anotačních záznamech mnou navrhovaného systému. Možných řešení architektur je mnoho, rozbor tohoto je nad rámec této práce.

důvěrně znám, a naproti tomu analýzu v prostředích, ve kterých by aplikace mé metody byla vhodnější, nemám možnost provést.

Nejprve uvedu prvotní „textovou“ podobu modelu, to je sběr vět v přirozeném jazyce, které mají být východiskem pro návrh business konceptuálního grafu. V zápisu těchto vět se odchýlím od běžné gramatiky českého jazyka (ve shodě s zápisem ORM modelů podle [Hal]) v tom směru, že jména předpokládaných typů konceptů budu zapisovat s velkým počátečním písmenem, a víceslovná jména propojím znakem „_“. Předpokládané množinové typy konceptů podtrhávám (jinou variantou by bylo použití tvaru množného čísla):

(1) Učitel vyučuje Předmět.
(2) Učitel garantuje Předmět.
(3) Student navštěvuje Předmět.
(4) Předmět se zkouší na počítači [37].
(5) Učitel zkouší Studenta z Předmětu.
(6) Učitel v Roce navštívil konferenci s Názvem_konference.
(7) Zaměstnanec žádá o Vybavení.
(8) Zaměstnanec dostal Vybavení
(9) Zaměstnanec je Učitel.

(Jedná se o fragment modelu. Vztah k realitě není pro mne zde důležitý, jde pouze o ilustraci.)

V tomto příkladu nenalézám smysluplný prostor pro uplatnění konstrukce hnízděného konceptu, jako příklad z jiného business kontextu je možno uvést textovou podobu pro obrázek 11 ze str.15:

Tým hraje proti Týmu. Tento Zápas sleduje Reportér.

Poznamenávám, že narozdíl od případu tvorby datového modelu pro *databázi faktů* týkající se provozu univerzity, v tomto případu *není třeba* zabývat se a *do business CG zahrnovat* jinak běžná integritní omezení, např. „Jestliže Učitel garantuje Předmět, pak Učitel vyučuje Předmět“. Modelovaná databáze kontextových anotací k informačním zdrojům nemá primárně sloužit k *evidenci fakt* o provozu univerzity, ale k *poznamenávání* těch z fakt, které tvoří kontext jednotlivých informačních zdrojů.

Dále je třeba vyřešit otázku atomů. Vztahy (4) a (5) by mohly tvořit atom, s tím, že fakt (4) by mohl být automaticky propagován z TPS systémů, ale v této podobě takovýto atom není možné smysluplně vytvořit. Přemodeluji tedy vztah (4) následovně:

(4*) Předmět se zkouší v Typu_místnosti.

[37] Syntaxe využívající velkých písmen k vyznačení konceptových typů pomáhá vyřešit případy potenciálních typů konceptů s jedno-výskytovou extenzí.

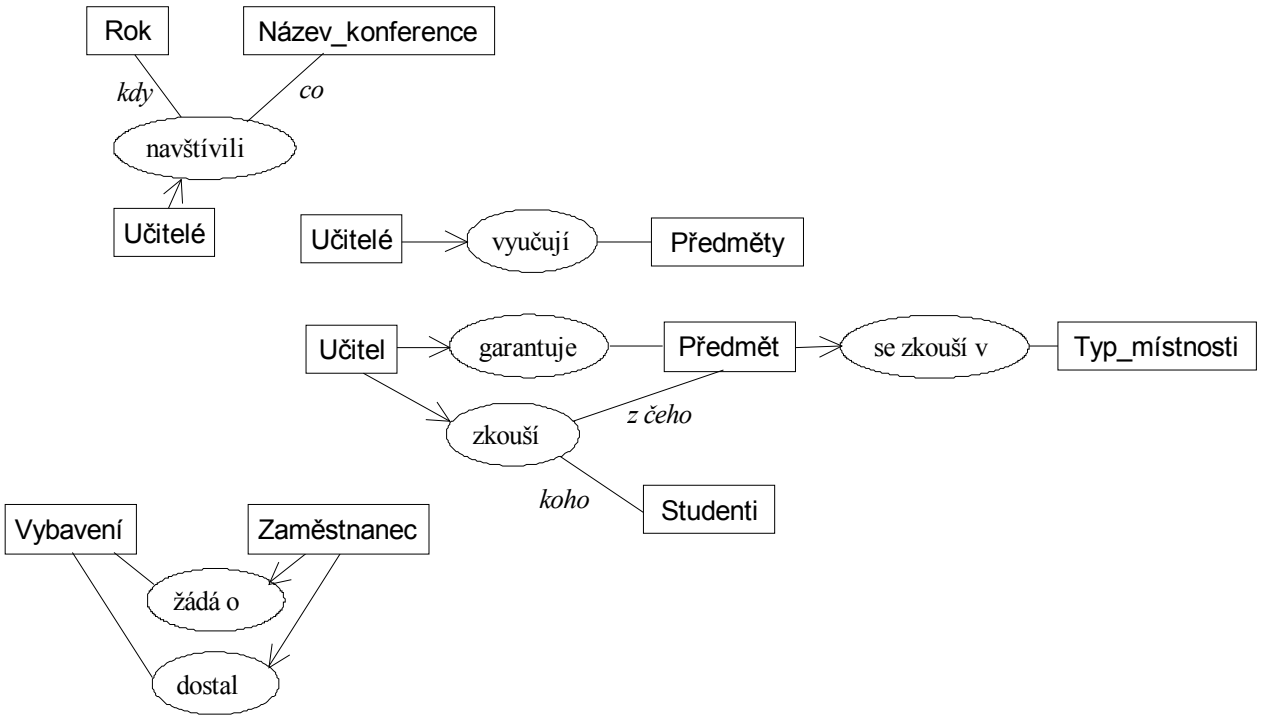
Dvojice vztahů (7),(9) a (8),(9) obě vypadají jako adepti na atom, ale vztah (9) se bude do business CG modelovat jako ISA vztah, tedy „neviděný“, nebude to konceptuální vztah. – V obecném případě, kdy by některý vztah byl modelován jako konceptuální vztah a byl zahrnutelný do více skupin konceptuálních vztahů potenciálně tvořících atom, žádnou skupinu nezvolíme za atom a všechny budou realizovány pouze integritním omezením. Řešení takových případů „denormalizací“, tj. zahrnutím příslušného konceptuálního vztahu do business CG vícekrát, aby mohly potenciální atomy být definovány, není vhodné, neboť se pak naruší funkce business CG jako nástroje k dotazování.

Textová podoba business CG je následující:

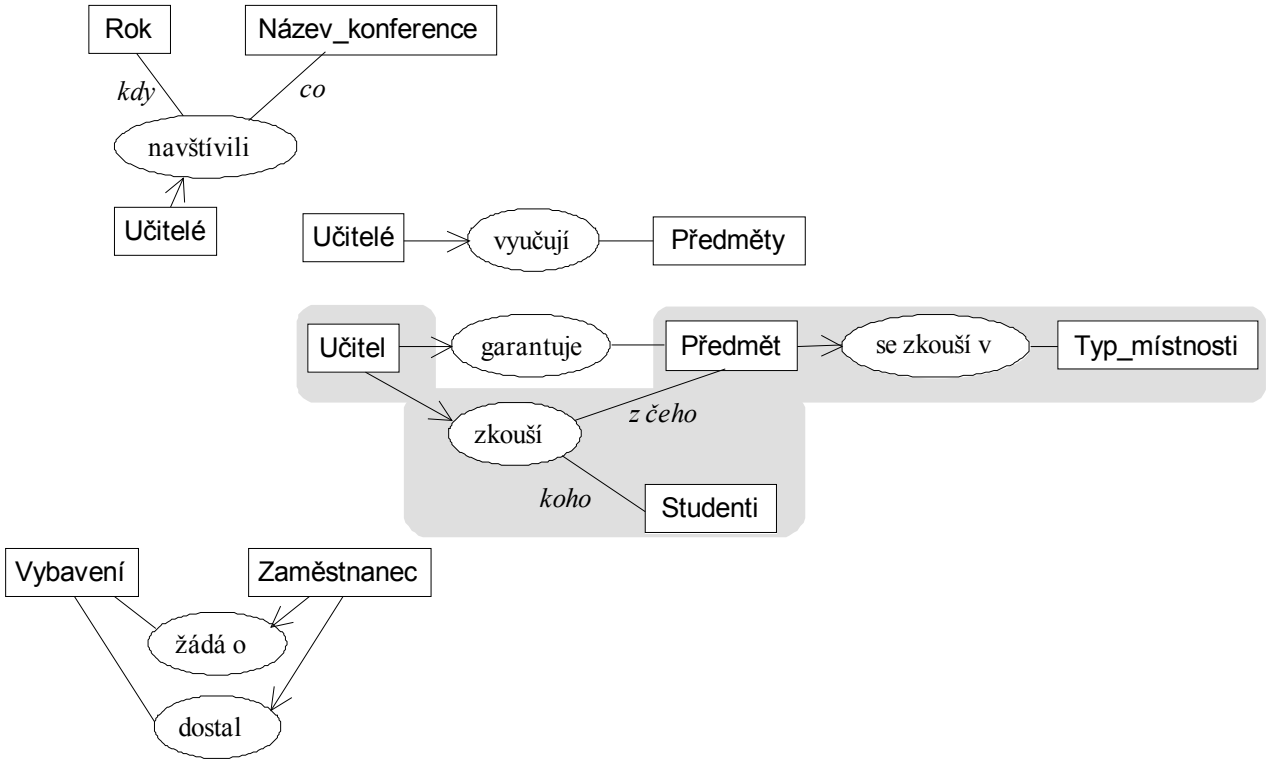
```
[Učitel] → (Vyučuje) → [Předmět]
[Učitel] → (Garantuje) → [Předmět]
[Student] → (Navštěvuje) → [Předmět]
[Učitel] → (ZkoušíKohoZČeho)-1→[Student]
                             -2→[Předmět]→(SeZkoušíV)→ [Typ_místnosti]
[Učitel] → (NavštívilKdyCo)-1→ [Rok]
                           -2→ [Název_konference]
[Zaměstnanec] → (ŽádáO) → [Vybavení]
[Zaměstnanec] → (Dostal) → [Vybavení]
```

Grafická podoba modelu může vypadat například jako viz obrázek 12. Pro větší přehlednost jsou množinové koncepty označeny návěštím ve tvaru množného čísla (u konceptu Vybavení to ale není zřetelné), návěští relací není v infinitivu, šipky vedou od podmětů příslušných vět, místo očíslování hran připojujících konceptů k relacím je použito návěští rozlišující role ve vztahu. Atom není vyznačen, atomy mají být zvýrazněny pouze v situaci, kdy to je potřeba (viz obrázek 13).

Množinový koncept Učitelé je v grafu dvakrát, zatímco Učitel a Vybavení pouze jedenkrát. To je výsledkem úvahy, zda bude pravděpodobné, že pro jednu anotaci bude v různých vztazích použito různé instancování příslušného konceptu.




**obrázek 12: grafická podoba business CG pro universitu (fragment)**




**obrázek 13: zvýraznění atomu**

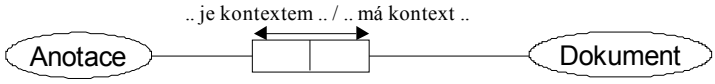
# Transformace business CG do ORM

Tato kapitola pojednává o tom, jak může vypadat datový model pro situaci, kdy anotace vytvořené pomocí business CG popisují kontext informačních zdrojů. Tento datový model zahrnuje tedy i sémantický datový typ pro anotované informační zdroje.

Informačními zdroji, které připadají do úvahy, mohou být dokumenty, části dokumentů, pohledy do databází v TPS, externí informační zdroje, osoby z podniku i mimo podnik,…. Pro zjednodušení budu nadále používat termín *dokument*, a budu jím myslet informační zdroj kteréhokoli z možných druhů.

Dále používám další, umělý, sémantický datový typ: anotace. Ten potřebuji k tomu, že skutečná anotace přestavuje *komplex* faktů, charakterizujících kontext dokumentu. Tento komplex je určen výčtem svých částí, faktů. Ovšem takovýto sémantický typ, jehož instancemi jsou kolekce, je v databázích obtížně implementovatelný, v souladu s [Hal4] vnáším tedy do modelu umělý prvek, typ „anotace“, jenž bude mít umělý identifikátor[38].

Vztah dokumentů a anotací se modeluje takto:



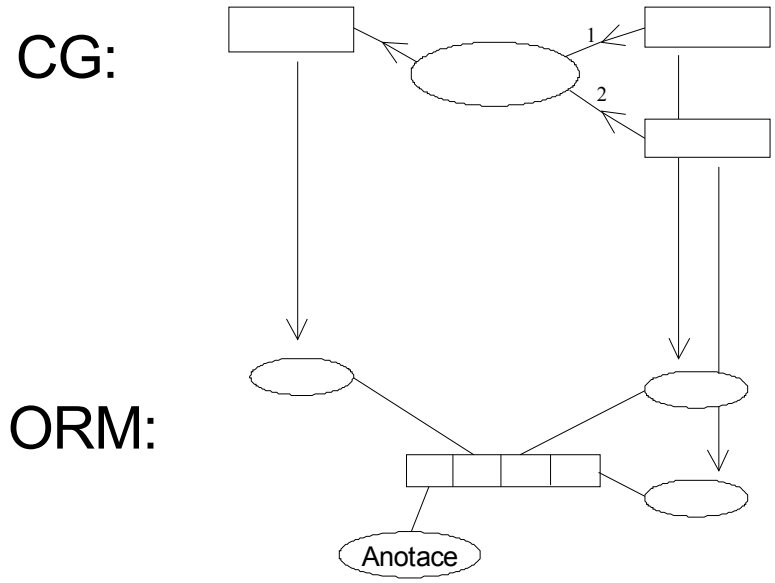

Zdůrazněme, že tento vztah obecně má kardinalitu m:n (srv. též poznámku č.38).[39]

Koncepty z business CG se v zásadě modelují jako objektové typy v ORM modelu, případy množinových konceptů a případy hnízděných konceptů odložme na později. Koncepty téhož typu se modelují jako jediný objektový typ.

Podstatou transformace business CG do modelu ORM je transformace každé n-ární konceptuální relace v CG do (n+1)-árního vztahu v modelu ORM, v němž přidanou roli hraje anotace, jak to ilustruje obrázek 14 (narozdíl od CG podle [Sow] podle mého návrhu mohou být hrany patřící ke konceptuálním relacím označeny nikoli čísly a šipkami, ale jmény rolí). Přesnější formulace transformace je obsahem Dodatku 3, části Popis transformace, kterou doporučuji číst až po kapitole, kterou právě čtete. Vysvětlení oprávněnosti základního transformačního kroku je obsahem Dodatku 3, části Základní krok transformace business CG do ORM.

[38] Identifikátorem „anotace“ může být i KIF textový kód odpovídajícího CG. Potom nepřipadá v úvahu námitka z poznámky na str. 75, omezení „extensional uniqueness“ zajistí algoritmus přidávající záznamy do databáze. Je ovšem otázkou, jestli pravděpodobnost výskytu dvou zcela shodných anotací je natolik velká, aby stálo za to udržovat KIF kód či jiný textový překlad CG jako primární klíč anotací.

[39] Jiná alternativa modelu zohlední potřebu uchování informace o tom, kdo danou anotaci k danému dokumentu připojil, takže bychom modelovali ternární vztah. Změny v tomto případě jsou drobné a snadné, pro všechny další úvahy.




**obrázek 14: idea transformace business CG do modelu ORM**

## *Atomy*

Transformace atomů business CG do modelu ORM je založena na názoru, že konceptuální relace v atomu tvoří sémantický celek. Koncepty v pod-grafu odpovídajícím atomu jsou v myšlené, leč nevyslovené, a tedy nepojmenované[40], relaci. Tuto myšlenou relaci můžeme nazvat *super-relací konceptuálních relací tvořících atom*. Do modelu ORM tedy nemodelujeme jednotlivé konceptuální relace z atomu (podle základního kroku transformace), nýbrž tuto super-relaci.

Přesnou definici super-relací obsahuje Dodatek 3, část Super-relace atomů.

## *Koncepty s jedinou instancí*

Koncepty v business CG, jež mají jedinou možnou instanci, se do ORM modelu nemodelují. Každá n-ární konceptuální relace, ke které jsou takové koncepty připojeny, se modeluje jako (n-k+1)-ární vztah v ORM modelu, kde k je počet konceptů, jež jsou připojeny k této relaci a mají jedinou možnou instanci. Původní konceptuální relace se tedy nejprve transformuje do (n-k)-ární konceptuální relace[41], a tato se modeluje v ORM podle základního kroku transformace.

Například relace

[Zaměstnanec] → (PodatHlášeníKomu) → [Vedení]

se transformuje do relace

[Zaměstnanec] → (PodatHlášeníVedení).

[40] Jak je uvedeno v odstavci Atomy grafu, integritní omezení na str.27, je možné také atomy v business CG smysluplně pojmenovat.

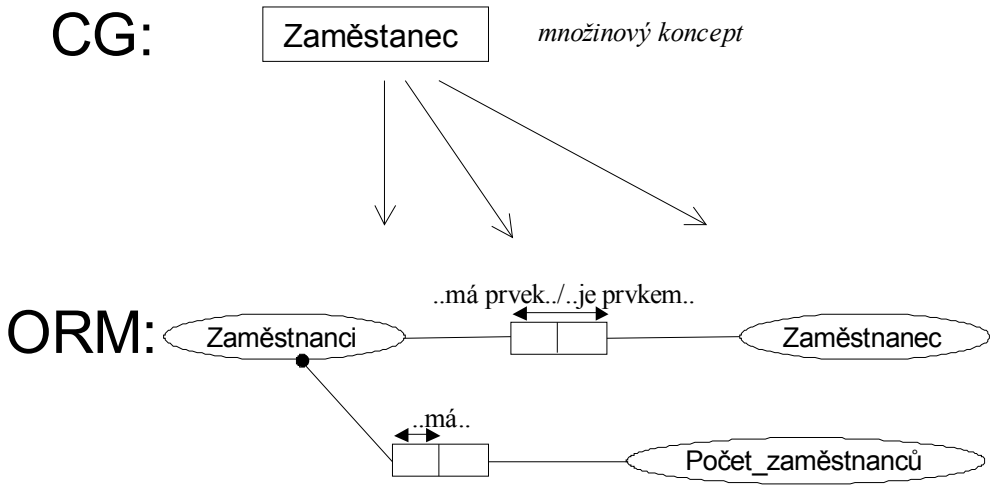
[41] jména typů konceptů, jež se nebudou modelovat, je možno propagovat do jmen transformovaných relací

## *Množinové koncepty*

Koncepty, jež mohou být kvantifikovány množstvím nebo kolekcemi, nazývám množinovými koncepty. Takový koncept může být „instancován“ pouze kvantifikací množství, nebo může být uveden výčet prvků kolekce, a nebo, narozdíl od pojetí [Sow], kombinovaně „instancován“ kvantifikací množství a *výčtem některých* prvků kolekce. V úplné obecnosti je třeba předpokládat, že takové koncepty jsou v business CG připojeny k více konceptuálním relacím, a že mohou být v konkrétní anotaci použity opakovaně. Proto je třeba takové koncepty modelovat jako samostatně odlišitelné entity, pro něž známe počet prvků a výčet některých prvků. Výčet prvků je třeba modelovat jako sémantický typ, jehož instancemi jsou kolekce. Podobně jako [Hal4] modeluji tyto koncepty jako objekty s umělým identifikátorem, přidávám vztah těchto objektů s jejich „prvky“ a funkcionální vztah k počtu prvků.

Množinový koncept typu A modeluji jako objekt typu A’ (jméno je možno volit např. jako tvar množného čísla jména typu A), funkcionální predikát „..má..(počet)“ a predikát „..má prvek..“, v němž první roli hraje objekt typu A’ a druhou roli hrají objekty typu A. Tento predikát má kardinalitu m:n, tj. v terminologii ORM má klíč zahrnující obě jeho role. Příklad uvádí obrázek 15.

Pro podporu porovnávání kolekcí je možno pro objekty typu A’ vytvořit další funkcionální vztah – atribut, který vznikne konkatenací abecedně uspořádaných identifikátorů vyjmenovaných instancí A, jež jsou prvky daného objektu typu A’.



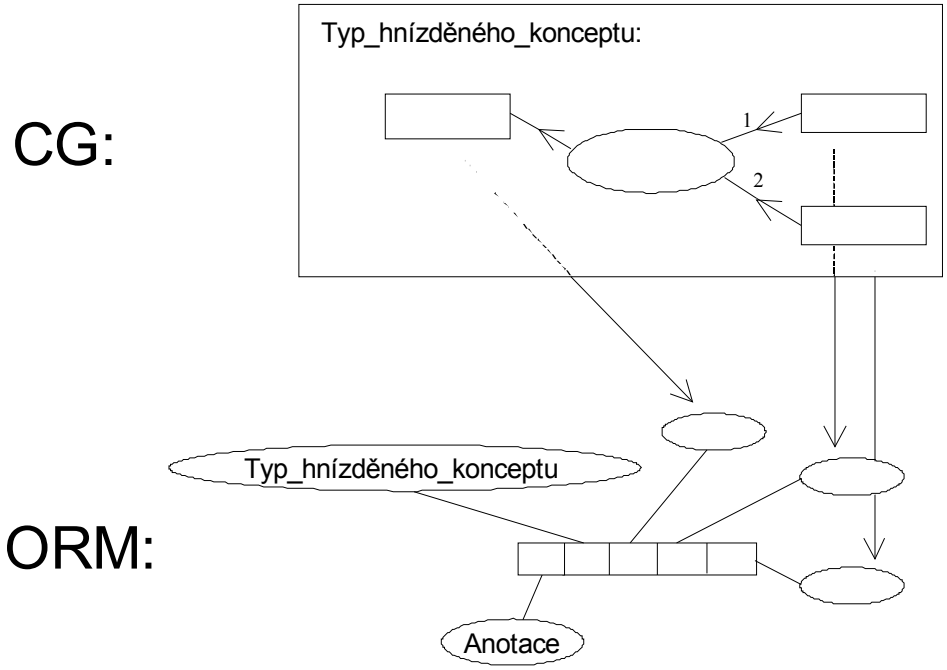

**obrázek 15: transformace množinového konceptu**

Tato volba modelování množinových konceptů usnadňuje jak porovnávání kolekcí instancí tak porovnávání jednotlivých instancí.

## *Hnízděné koncepty*

Hnízděné koncepty, narozdíl od ostatních konceptů, jsou „líné“[42] tj. nemusí být při konkrétní anotaci připojeny k žádné konceptuální relaci, v této anotaci použité. Proto je třeba uchovávat informaci o tom, že tento koncept je do anotace zahrnut.

CG zahnízděný v hnízděném konceptu lze považovat za jiný případ jiného business CG, a aplikovat na tento transformaci do ORM. Rozdílem je pouze to, že namísto nového objektového typu „anotace“ pro tento vnořený CG by sloužil objektový typ, který v ORM odpovídá typu hnízděného konceptu, jenž vnořený CG zahnízďuje. Navíc by bylo třeba zachovávat informaci o tom, který eventuelní koncept zahnízďuje tento hnízděný koncept, který koncept zahnízďuje tento další koncept, atd., a která anotace toto vše zahrnuje. Z důvodů, které vysvětluje Dodatek 3, část Transformace hnízděných konceptů, se hnízděné koncepty v principu modelují tak, jak to ukazuje obrázek 16 na str. 35.




**obrázek 16: transformace hnízděného konceptu**

## *ISA vztahy, typová příbuznost*

ISA vztahy se přímo transformují do modelu ORM. Typovou příbuznost je možno v modelu ORM zohlednit způsobem, který doporučuje [Hal], a to vytvořením nadtypu typově příbuzných typů[43].

[42] tuto terminologii používám shodně s terminologií ORM, v níž je termín „líný objektový typ“ alternativou pro „nezávislý objektový typ“ – co to znamená viz Přehled ORM zde.

[43] metody ORM návrhu týkající se typové hierarchie považuji však za dogmatické

### *Odvozené koncepty a konceptuální relace*

V business CG jsou odvozené koncepty a odvozené konceptuální relace definovány např. pomocí λ-výrazů. K těmto λ-výrazům je třeba dodat informaci, zda konceptuální relace v nich, jež jsou v business CG v jediném atomu, mají v definičním λ-výrazu být také takto chápány, či nikoli. Jinou variantu definice odvozených objektů nabízí použití „dotazovacího jazyka“ popsaného v kapitole Dotazy, části Textové „outline“ dotazování nad business CG.

Definice odvození se přetransformují do definic v ORM modelu, Dodatek 3, část Popis transformace, Transformace odvozovacích pravidel, popisuje transformaci λ-výrazů do jazyka logiky. Definice využívající dotazovacího jazyka zmíněného výše lze snadno transformovat do jazyka ConQuer nad modelem ORM.

### *Integritní omezení*

Integritní omezení v business CG se týkají vždy aktuálně vytvářené anotace, a tak jejich zajištění může provádět klientská část systému. Propagovat je do ORM modelu není důvod, kromě rozhodování o transformaci ORM modelu do logického datového modelu (mohou ovlivnit sdružování do tabulek). Kromě těch, která jsem u konkrétních částí ORM modelu vyjádřila buď slovně nebo v grafické podobě modelu, mohou do úvahy ještě přicházet omezení množinovým porovnáním mezi predikáty modelujícími super-relace atomů.

### *Implementační poznámky*

U odvozených objektů a vztahů je třeba rozhodnutí, zda se budou implementovat.

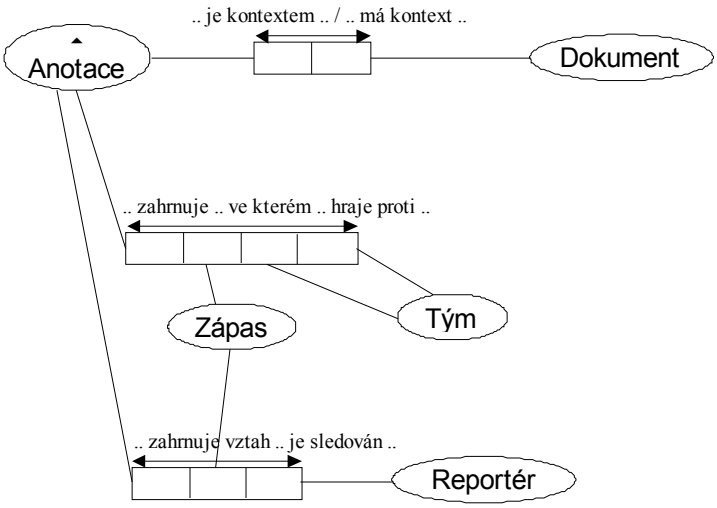
Některé fakty se mohou z transakčních procesních systémů nebo jiných částí informačního systému načítat aktuálně na požádání při dotazování, takové vztahy jsou opatřeny implementačními poznámkami.

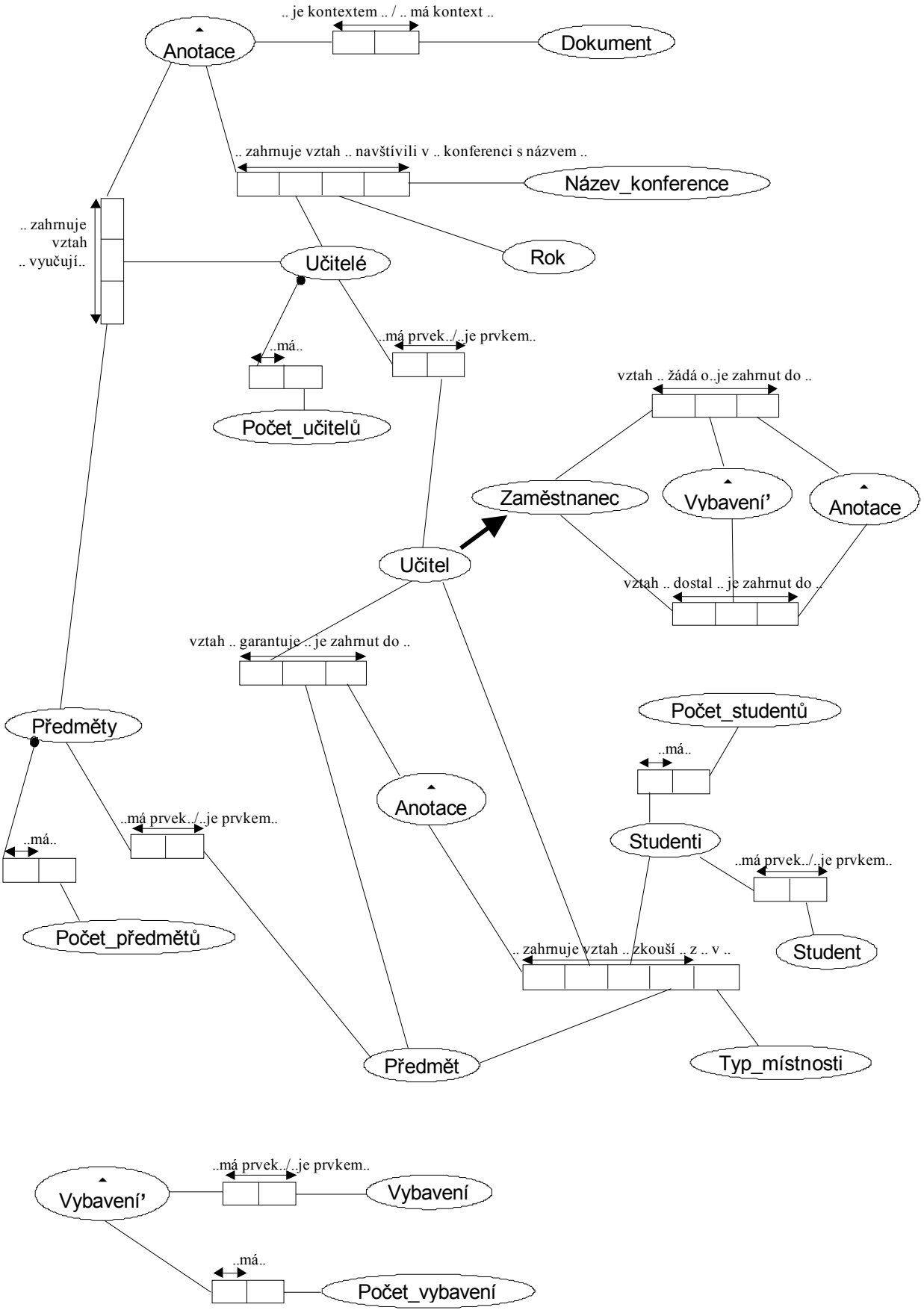
### *Překlad příkladu (universita)*

Příklad „universita“ jak ho zobrazuje obrázek 12 na str.31 bude do modelu ORM přeložen jako obrázek 17 na str.38.

Příklad, který naznačuje obrázek 11 ze str. 15, by byl přeložen následovně[44]:

[44] Je zde nevyřešený problém symetrie rolí obou týmů.


.. je kontextem .. / .. má kontext ..
Anotace
Dokument
.. zahrnuje .. ve kterém .. hraje proti ..
Zápas
Tým
.. zahrnuje vztah .. je sledován ..
Reportér




**obrázek 17: překlad příkladu „universita“ do modelu ORM**

# Tvorba anotací nad business CG

Dvě koncepční otázky týkající se tvorby anotací uvádím na začátek. První: uživatel je sice definovanými atomy nucen specifikovat některé rysy toho určitého kontextu, jejž svou anotací popisuje, ale vždy může z různých důvodů některé fakty neuvést. Nemůžeme tedy předpokládat, že co není v anotaci, není pravda, volím tedy *odmítnutí předpokladu uzavřeného světa*.

Druhá otázka se týká toho, že uživatel při veškeré své snaze některé fakty specifikovat neumí nebo nechce, například protože je nezná. Také může chtít nespecifikovat, protože dokument, pro nějž vytváří anotaci, má obecnou platnost, a anotační záznam má vymezovat celou množinu různých možných kontextových situací.

K podpoře těchto možností volím toto řešení: „instancování“ může být konkrétní, tj.určité, nebo neurčité nebo všeobecné (přesněji to vysvětlím dále). Protože jak při tvorbě konkrétní anotace tak při dotazování může docházet k porovnávání referencí instancí, je třeba vytvořit nástroj k vyjádření identity navzájem mezi neurčitými designátory a navzájem mezi všeobecnými designátory v jedné anotaci, a zároveň definovat operace porovnávání mezi určitými, neurčitými a všeobecnými hodnotami v datových záznamech.

Každý objektový typ modelu ORM, který modeluje[45] nějaký typ ne-hnízděných konceptů v business CG[46], a všechny typy „Počet_A-prvků“, rozdělím do tří podtypů: *určité, neurčité a všeobecné*[47]. Pro neurčité instance zvolím vhodný efektivně rozpoznatelný způsob identifikace, může například začínat symbolem „?“, za kterým následuje identifikátor jedinečný pro anotaci, a po něm následuje číslice odlišující jednotlivé neurčité ale různé hodnoty v jedné anotaci. Podobně pro všeobecné instance, jejich identifikace může například začínat symbolem „-“. Porovnávání hodnot z těchto tří podtypů definuje Dodatek 3, část Definice porovnávacích operací.

(K podpoře dotazování je možno do modelu přidat funkcionální atribut anotací vyjadřující, zda je anotace „všeobecná“, tj. obsahuje nějaký všeobecně instancovaný koncept, nebo konkrétní, nebo neurčitá: „Všeobecná“ anotace by neměla obsahovat žádné neurčité instance, proto je toto rozdělení korektní.)

## *Procedura tvorby anotace*

Uživatel vytvoří anotaci a určí, ke kterému dokumentu či ke kterým dokumentům anotace patří.

[45] Jak jsem vyjádřila na str. 77, nerozlišuji mezi typy konceptů v business CG a odpovídajícími objektovými typy v modelu ORM.

[46] *typy* množinových konceptů jsou shodné s objektovými typy pro *prvky* množin v ORM modelu

[47] Mohly by do úvahy přicházet také přibližné hodnoty, ale toto v této práci nerozvíjím. Bezesporu by se tím zvýšila užitná hodnota systému.

Při tvorbě anotace uživatel volí instance pro některé jednoduché, tj. ne-množinové a ne-hnízděné, koncepty, a některé množinové koncepty kvantifikuje množstvím a/nebo pro ně vybere množinu instancí typu příslušného konceptu. Tomu říkám, že tyto koncepty „instancuje“. Navíc při tvorbě anotace může uživatel vytvořit koreferenční propojení mezi koncepty, z nichž alespoň jeden instancoval – jde o zkratku, aby nemusel instancování opakovat. Zároveň to znamená, pokud s jiným konceptem koreferenčně propojil koncept s neurčitou nebo všeobecnou hodnotou nebo množinový koncept[48], že tyto koncepty v rámci jeho anotace mají být totožné. Pokud nemá být uživatel obtěžován očíslováváním různých neurčitých nebo všeobecných hodnot, musí být přijata úmluva, že co nepropojil koreferenčním propojením, to je obecně navzájem různé[49]. Toto bude platit i pro množinové koncepty – není-li z koreferenčního propojení nebo z úplného výčtu prvků kolekce zřejmá identita mezi koncepty, pokládají se obecně za různé.

Uživatel instancuje jeden koncept, a pak je přinucen vybrat jeden z atomů, k nimž je tento koncept připojen, a instancovat všechny koncepty připojené k tomuto vybranému atomu – k tomu mu mohou posloužit neurčité hodnoty, pokud konkrétní fakta nezná[50]. Integritní omezení business CG mohou uživatele donutit, aby pokračoval některým dalším atomem, atd. Pokud má uživatel volnost, může pokračovat libovolným konceptem dále. Může požádat o další „stránku“ typového business CG, aby mohl použít znovu některý typový koncept: Koreferenční propojení mezi jednotlivými stránkami může uživatel dělat některým příhodným způsobem (například „kopírováním“ a „vložením“, což se již v mnoha jiných systémech osvědčilo).

Při tvorbě anotací nelze použít odvozené konceptuální relace v business CG. Pokud chce uživatel použít odvozený koncept, ke kterému je třeba specifikovat některé fakty, je uživatel nucen toto nejprve udělat.

Některé fakty potřebné k doinstancování použitého atomu mohou být načteny z TPS či jiných informačních systémů, uživatel tím nemusí být zatěžován.

TPS nebo jiné systémy mohou uživateli poskytovat zobrazení takových atributů (jména, obrázky, další ověřující vlastnosti…) entit, které volí jako konkrétní instance typů v konceptech, aby uživatelova interakce byla pohodlná, a aby si mohl ověřit správnost výběru instance.

---

[48] Množinové koncepty nemusí být zcela „určitě“ designovány, i když budou použity určité hodnoty pro počet prvků i vvyjmenované prvky množiny…

[49] Reálné hodnoty, jež tyto neurčité symboly zastupují, mohou být popřípadě stejné, ale systém o tom nebude mít informaci. To se může změnit, když uživatel v budoucnu v záznamu změní neurčité hodnoty za určité a správné.

[50] nebo může být jejich výčet pro něj nudný…

## *Přidávání anotačních záznamů do databáze*

Protože ORM nemá kromě dotazovacího jazyka žádný další jazyk manipulace objektů v úrovni konceptuálního schématu, nelze popsat přidávání anotačních záznamů na konceptuální úrovni nijak standardně. Bylo by možno zvolit některé mapování modelu ORM do logického databázového modelu, a popsat přidávání záznamů na databázové úrovni, ale tomu se chci vyhnout. Zaprvé proto, že téměř vše je jednoduché a zřejmé, zadruhé proto, že nechci předčasně dávat přednost některému z databázových modelů.

Vyjasním tedy všechny otázky, o kterých si myslím, že by mohly být potřeba vyjasnit.

Pro vytvořenou anotaci se určí její referenční identifikátor, určí se referenční identifikátory případných instancí objektů odpovídající hnízděným a množinovým konceptům: Jeden možný návrh identifikace objektu odpovídajícího hnízděnému konceptu je zmíněn na str.81 v části Modelování konceptů v ORM. „Množinové“ objekty odpovídající množinovým konceptům můžeme rozdělit do dvou podtypů – úplně určené (určitý počet prvků je roven počtu vyjmenovaných určitých prvků) a neurčené. Neurčené mají vždy nové různé identifikátory, kromě případů, kdy jde o koreferenčně propojené koncepty v jedné anotaci. Pro úplně určené množinové koncepty lze hledat k nim identické v datových záznamech z jiných anotací[51], a eventuelně použít shodný identifikátor[52] – jinak mají také nový identifikátor, kromě případů, kdy jde o koreferenčně propojené koncepty v jedné anotaci.

Jedna nevyjasněná otázka zůstává u množinových konceptů, u kterých uživatel nezvolil kvantifikaci množství, ale vyjmenoval některé prvky množiny. Je možné přijmout dohodu, že v takovém případě se počet prvků spočítá z vyjmenovaných, tedy se to bude chápat tak, že vyjmenoval všechny (je i možno se na to uživatele dotázat).

Každý instancovaný atom vyústí v záznamy o nových instancích příslušných vztahů (viz Modelování konceptuálních relací na str.81).

Jestliže v případě určeného *určitého* kvantifikátoru množství je mezi vyjmenovanými prvky nějaká neurčitá instance, která není v rámci uživatelovy anotace koreferována, následně ji ignorujeme.

Sémantika případů, kdy uživatel uvede nějakou kvantifikaci množství, a počet vyjmenovaných prvků je *menší než* tato kvantifikace, je chápána tak, že ostatní hodnoty jsou neurčité a různé[53] od ostatních v anotaci užitých neurčitých hodnot.

---

[51] k tomu může dopomoci atribut množinových objektů tvořený konkatenací abecedně uspořádaných identifikátorů prvků

[52] Jak vysvětluji v kapitole Dotazy, části Porovnání množin, porovnání množinových konceptů na rovnost považuji za nevhodné. Proto je možno takovéto složité integritní omezení vypustit.

[53] nejde o porovnání na rovnost podle Definice porovnávacích operací, ale o shodu identifikátorů

Poslední nevyjasněná otázka ohledně množinových konceptů zůstává v případě, že uživatel zvolil *neurčitou* nebo *všeobecnou* kvantifikaci množství, a mezi vyjmenovanými prvky množiny jsou nějaké neurčité. Jedna možnost je tyto ignorovat, když nejsou koreferovány, ale tím se ztrácí informace o tom, že počet prvků množiny je alespoň takový, jaký je počet vyjmenovaných. Potom by bylo řešení v tom, že mezi instancemi typu Počet budou i hodnoty „>n“ , kde n je číslo – pak bylo nutno dodefinovat porovnávání a eventuelní další početní operace s takovýmito hodnotami. Jiná možnost je ponechat uživatelem vyjmenované neurčité instance, a při dotazování v případě potřeby vypočítávat i počet objektů ve vztahu „patří do“ k tomu kterému „množinovému“ objektu. Volba řešení závisí na možnostech použitých technologií a požadavcích na efektivitu, v této obecné koncepční úrovni je možno pouze si uvědomit, že běžný uživatel nebude „ručně“ vyjmenovávat příliš velká množství hodnot. Pokládám za pravděpodobné, že takovéto případy vůbec budou vzácné. Proto pokládám za vhodnější druhou možnost řešení.

## *Vizuální evidence*

Uživatel by měl mít přehled o tom, jakou anotaci vytvořil/vytváří. Interface konceptuálního grafu může být nepřehledné z důvodu rozsáhlosti grafu i z důvodu „vícestránkovosti“ anotace. Proto je dobré uživateli nabídnout nějakou kontrolu, například textovou podobu anotace. Takové textové podoby jsou ale různě čitelné, podoba vět přirozeného jazyka je pravděpodobně nejvhodnější. Pro tuto možnost je ovšem potřeba vyřešit dva problémy: první se týká ohebnosti češtiny (možné řešení by bylo ve vytvoření slovníku tvarosloví potřebných slov), druhá je v algoritmizaci vhodného spojování vět do souvětí pomocí zájmen „který/á/é“ a pod. vyjadřující koreferenci.

## *Formální textová podoba anotace*

Formální textovou podobu anotace je vhodné a efektivní uchovávat pro její možné další zpracování, a pro případnou obnovu záznamů po haváriích v anotační databázi.

## *Příklad*

Zkouškový protokol může mít anotaci

```
[Učitel:Jan_Ostrý] → (ZkoušíKohoZČeho)-
-1→[Student: @13,{Petr_Chytrý, Jana_Pilná, Matěj_Pohodlný}]
-2→[Předmět: Práce_na_PC]→(SeZkoušíV)→ [Typ_místnosti: Počítačová_místnost]
```

Sborník z konference může mít anotaci

```
[Učitel: {Oleg_Nádherný, Kateřina_Rudá}] → (NavštívilKdyCo)-
-1→ [Rok:2001] -2→ [Název_konference: Datakon]
```

Reportáž ze zápasu může mít anotaci

```
[Reportér: Tutti_Ponti] → (Sleduje)→ [Zápas:[Tým: Sparta] → (HrajeProti) →
[Tým: Juventus_Turín]]
```

# Dotazy

Dotazy do mnou navrhovaného systému lze charakterizovat jako požadavky na vyhledání dokumentů, jejichž anotace má charakteristické rysy v dotaze zadané. V principu lze tedy formulovat dotazy tak, že zadáváme charakteristické rysy anotace – což je téměř shodné s procesem tvorby anotace pro vložení záznamů. Jsou zde ale rozdíly: požadovanou anotaci nebo požadované anotace nemusíme charakterizovat zcela, naopak, vymezujeme pouze potřebné požadavky, ostatní rysy ponecháváme volné. V termínech business CG to znamená, že při dotazování *atomy nenutí uživatele instancovat koncepty, jež instancovat nechce*, ovšem musí vzhledem k atomům dát najevo případnou souvislost požadavků. Ostatní integritní omezení business CG též nemají vliv na zadávání dotazů. Ani omezení týkající se odvozených objektů: uživatel může zadat požadavek na odvozený koncept, aniž předem specifikoval fakty potřebné k jeho odvození, uživatel při dotazování *může použít odvozenou konceptuální relaci*.

Anotační záznamy primárně neslouží, dle mé koncepce, jako datová či znalostní báze, tedy dotazování jiné, než výše popsané, nemá podle mne velký smysl. Okolnosti praktického užití však mohou ukázat opak. Pak lze použít jakýkoli dotazovací prostředek dostupný nad anotační databází, anotační záznamy budou dostatečně strukturované. Jako uživatelsky přívětivý vhodný prostředek se mi jeví dotazovací jazyk ConQuer nad ORM modelem, k podpoře jeho užití je potřeba volit taková jména predikátů v ORM modelu, která budou uživatelům srozumitelná (za účelem kompaktnosti asi budou přímo viditelná jména stručnější, pak je možno interaktivně poskytovat obšírnější vysvětlení na požádání).

K takovémuto složitějšímu dotazování poznamenávám hlavě to, že jak jsem uvedla v kapitole Tvorba anotací nad business CG, nelze přijmout předpoklad uzavřeného světa, a tomu je třeba přizpůsobit cíle i strategii dotazování. Navíc anotace, které obsahují nějaký všeobecně instancovaný koncept, nemusí obsahovat skutečná fakta, pouze hypotetické skutečnosti.

Dotazování nad business CG probíhá tak, že jsou vybírány konceptuální relace, zadávány podmínky na některé koncepty, jež jsou k těmto konceptuálním relacím připojeny, a jsou uvedeny případné souvislosti mezi zvolenými konceptuálními relacemi vzhledem k atomům business CG. Ve skutečnosti uživatel instancuje atomy podobně jako je tomu při tvorbě anotací, pouze není nucen instancovat atom celý.

Dotaz se pak chápe jako *logická konjunkce jednotlivých zadaných podmínek* na koncepty a konceptuální relace v atomech. K zadání alternativ (logického OR mezi skupinami podmínek), se mi grafická podoba business CG nejeví jako dostatečně vhodná. Proto by bylo možno vytvořit

alternativu, podobnou jazyku ConQuer, textového „outline“ dotazování nad business CG (vyhledávaný objekt „anotace“ by v dotazech nebylo třeba zmiňovat, neboť by byl jejich imanentní součástí).

Interfaceová funkce podporující vyznačení souvislostí mezi konceptuálními relacemi vzhledem k atomům může být například následující: umístěním kurzoru na konceptuální relaci se vizuálně zvýrazní celý atom, do kterého daná konceptuální relace patří, a v něm se dále zvýrazní ty konceptuální relace a k nim připojené koncepty použité v dotazu, jež uživatel určil, že mají do tohoto atomu patřit. Přidání dalších částí do atomu lze dělat některým standardním uživatelským způsobem pro vytváření skupin objektů.

## *Podmínky kladené na koncepty v rámci dotazu*

Nejjednodušší podmínka na koncept je konkrétní instancování tohoto konceptu určitou hodnotou či hodnotami. Interpretace instancování jednoduchého, tj. ne-množinového[54], konceptu je nade vší pochybnost: je to podmínka žádající rovnost instance zadané hodnotě. Instancování množinového konceptu lze interpretovat různě, mě se zdá nejpřirozenější takovéto chápání: uživatel tímto žádá, aby hledaná množina instancí jím zadanou množinu hodnot *obsahovala*.

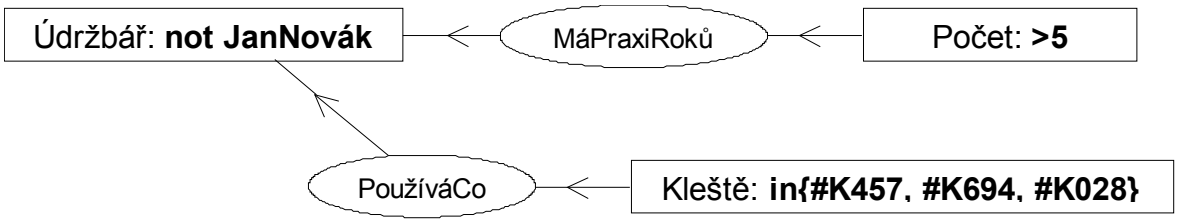
Ostatní podmínky kladené na koncept jsou dány vždy volbou operace porovnání a výrazu, s nímž má být hledaná instance či množina instancí porovnána s výsledkem TRUE (srv. Definice porovnávacích operací). Pro porovnání instancí dvou konceptů použitých v dotazu lze nabídnout různé alternativy interface: koreferenční propojení nebo použití proměnné (podobně jako je to v jazyce QBE i ConQuer); porovnání jiné než na rovnost je možno podpořit „koreferenčním“ propojením modifikovaným symbolem operace porovnání a s vyznačením pořadí porovnávaných konceptů – jde vlastně o „obecnou“ konceptuální relaci, jež není součástí business CG, proto je vhodné použít pro ni jinou grafickou podobu. Použití proměnných se stejně nevyhneme, pokud chceme k porovnání používat výrazy tvořené z instancí jiných konceptů.

### *Porovnání jednoduchých instancí*

Instance jednoduchých konceptů lze porovnat operací =, < , > , ≤ , ≥ (pokud to má smysl), `not`[55] a `in` (event. `not in`), všechno lze zapsat do místa určeného pro referent konceptu:

[54] hnízděné koncepty se zásadně instancují tak, že se instancuje CG v nich vnořený

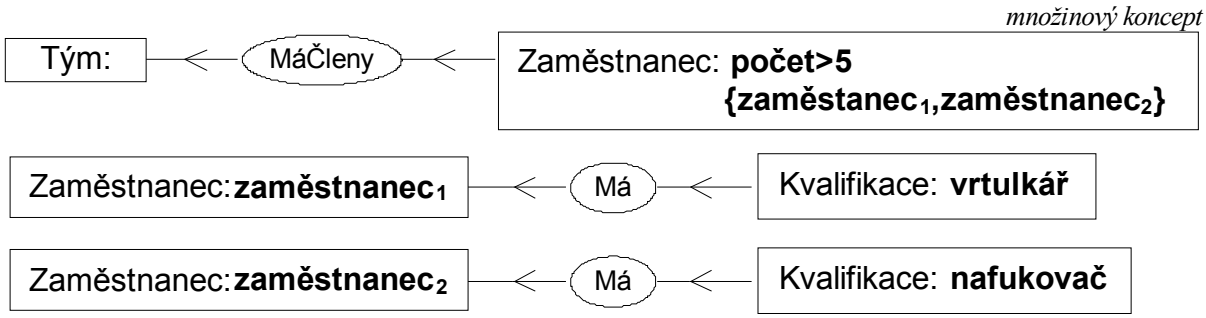
[55] Sémantika porovnání not může vyvolávat pochybnosti, zda se myslí, že pro hledanou anotaci nemá existovat instance konceptu s danou hodnotou, či zda má pouze existovat instance, jež tuto hodnotu nemá (s dalšími podmínkami). Konceptuální grafy pro vyjádření prvního významu používají jinou konstrukci. Dotazovací jazyky QBE a ConQuer, které jsou mi vzorem, užívají interpretaci druhou. Konečně odmítnutí předpokladu uzavřeného světa je dalším argumentem pro přijetí druhé interpretace.




Lze porovnat jednoduchý koncept s množinovým konceptem operací (`not`) `in` .

### *Porovnání množin*

Pro množinový koncept lze zadat podmínku na kvantifikaci množství a/nebo ho porovnávat s množinou:




Transformace porovnání kvantifikace množství do jazyka ConQuer je následující:

A’

⌊ má prvků Počet (porovnání_s_hodnotou)

kde A’ je objektový typ modelu ORM modelující množinový koncept typu A. Porovnání množin se budu věnovat dále.

Běžná porovnání množin jsou inkluze $\subset$ a $\supset$ , a porovnání neprázdnosti průniku, jež označuji „$\exists$`in`“. Toto poslední porovnání znamená existenci prvku z jedné množiny, který splňuje `in` s druhou množinou, takže jde o rozšíření porovnání `in` na množinové koncepty[56]. Poslední operací je rovnost = . Kvůli odmítnutí předpokladu uzavřeného světa se mi jeví jako nejužitečnější operace $\supset$ a „$\exists$`in`“.

Transformaci celého dotazu do jazyka ConQuer uvedu později, zde uvádím pouze transformaci porovnání množin: Je-li A typ množinového konceptu, A’ modeluje příslušný množinový typ v modelu ORM, tedy tomuto konceptu odpovídá v ORM vztah „A patří do A’“, a je-li $B=\{B_1,..,B_n\}$ množina, s kterou je tento koncept porovnáván, pak

A: $\exists$**in B** se transformuje do A’

⌊ má prvek A in B

[56] Tato operace poskytne více dokumentů do odpovědi na dotaz než operace $\subset$ , což v případě chybného pochopení je výhodnější, než obráceně.

| A: ⊃ **B** | se transformuje do | A’<br>└ má prvek $A_1$ $B_1$<br>└ …<br>└ má prvek $A_n$ $B_n$ |
|---|---|---|

(Množinová porovnání budou mnohem snazší, když se pro identifikaci množinových objektů nebo jako jejich funkcionální atribut bude udržovat kód tvořený konkatenací abecedně seřazených identifikačních kódů prvků množiny.)

## *Transformace dotazu*

Dotaz nad business CG mohu transformovat do jazyka ConQuer nad modelem ORM. Příslušný dotaz v ConQuer vždy vypadá takto:

```
✓Dokument
    └ má kontext Anotace
                  └ zahrnuje super-relaci'1 …
                                          └ …
                  └ zahrnuje super-relaci'2 …
                                          └ …
                  └ …
```

Je třeba vysvětlit, jak budou vypadat řádky počínaje třetím: Zhruba řečeno zde budou všechny predikáty ORM modelu odpovídající super-relacím atomů použitým v dotazu, a některé použité množinové koncepty budou mít pomocí └ připojenu jednu či více dalších podmínek, jak je to ukázáno na str. 45. Pro koreferenční propojení a pro porovnání mezi instancemi konceptů se využije očíslování typových proměnných.

Přesnější výklad obsahuje Dodatek 3, část Transformace dotazů nad business CG do jazyka ConQuer.

## *Textové „outline“ dotazování nad business CG*

Dotazy na anotace/dokumenty lze zadávat i v textové podobě, podobně jak je to zmíněno v části Textová podoba business konceptuálního grafu na str. 28. Uživatel vybere konceptuální relaci, eventuelně instancuje koncepty připojené k této relaci (ke konceptům se může později vrátit a instancování provést nebo upravit). Chce-li specifikovat další části téhož atomu, pokračuje na dalším řádku buď koreferenčně pomocí znaku └ připojenému ke konceptu v předcházejícím řádku, nebo novým konceptem a následně relací z téhož atomu, atd. Poté dá najevo, že atom dokončil, a může začít znovu – atomy budou po straně spojeny svorkou. Později může řádky v dotazu přesouvat, a tím přeskupovat do atomů, systém kontroluje to, aby přemísťovaná relace mohla do cílového atomu patřit.

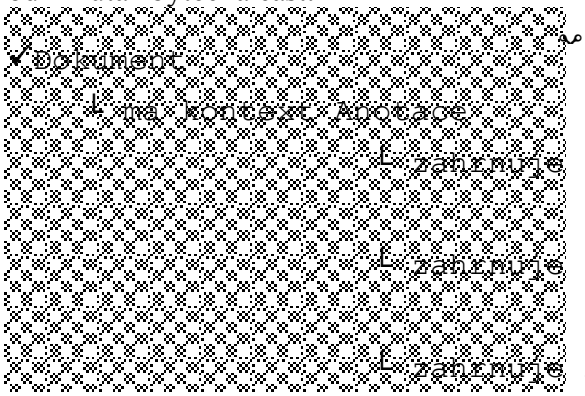
Specifikaci hnízděného konceptu, který je obsažen v dotazu, je možno pomocí znaku ┗ připojenému pod ním vytvořit na dalším řádku textem „ve které/ém“ následovaném analogií dotazu v outline podobě (viz příklad níže). Obrácené směry čtení je možno navrhnout jako připojení textu „v“ následovaného jménem typu hízděného konceptu.

Porovnání konceptů provádí umístěním znaku pro operaci porovnání (jeho vynechání značí default porovnání zmíněné v části Podmínky kladené na koncepty v rámci dotazu) za symbol pro koncept, po znaku porovnání vybere následně instance či *proměnné*, s nimiž má být koncept porovnáván. Proměnné se nabízejí jako jména typů konceptů v business CG, postupně oindexovávaná – index 1 se objeví až v případě přítomnosti indexu 2 (podobně jako v InfoAssistant). Jméno typu konceptu připojeného k nějaké konceptuální relaci je vlastně vždy proměnnou, v dotazu ji lze oindexovat nějakým uživatelsky přívětivým způsobem, při němž každý z případně nabízených indexů ukáže výskyty stejné proměnné se stejným indexem v dosud zadaném dotazu. Koreferování, tj. zastoupení stejnou proměnnou, lze také definovat např. tažením proměnné na proměnnou – cílová proměnná bude mít stejný index, jaký byl u startovní proměnné. Kontrola koreferenčních propojení v dotazu může být umožněna např. tak, že při kurzoru umístěném na nějaké proměnné se zvýrazní ostatní výskyty této proměnné.

Systém při ověřování syntaktické správnosti dotazu kontroluje, že proměnné použité při porovnání s nějakým konceptem se někde jinde vyskytují jako koncepty připojené ke konceptuálním relacím použitým v dotazu.

Logické operace `or`, `not` se zadávají buď za znak ┗ propojující koncepty následujících řádků, nebo před další řádek začínající nějakým konceptem; týkají se tohoto řádku a celé větve dotazu z něj vycházející.

Vše co se týká textové „outline“ podoby dotazů nad business CG jsem převzala z jazyka ConQuer, za použití základní jednoduché představy: z dotazu v jazyce ConQuer nad ORM modelem je odříznuta zbytečná část:



Pak jsou super-relace atomů dekomponovány do jednotlivých konceptuálních relací z business CG a uzavřeny do svorek. Přidány jsou další vyjadřovací možnosti jazyka ConQuer…

## *Příklad*

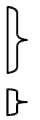
Dotaz v outline podobě z příkladu „universita“ může vypadat například takto:

```
Učitel zkouší Studenti z Předmět
                          └ se zkouší v Typ_místnosti „Počítačová místnost“
or
Učitel garantuje Předmět
```

V novinářském příkladu může dotaz vypadat takto:

```
Reportér sleduje Zápas
                  └ ve kterém Tým in {„Sparta“,„Bohemians“} hraje proti Tým
                  └ not ve kterém Tým hraje proti Tým „Slavie“
```

(Svorky spojující atomy je nutno specifikovat jen tam, kde to je třeba.)



# XML záznam anotací

Anotační záznamy mohou být též mapovány do XML struktury, což může být použito pro distribuci databáze anotací na místo vzdálené od anotovaných informačních zdrojů. Vyhledávání na vzdáleném místě může pak probíhat jako výběr vhodných anotací, a následovat může požadavek na zaslání dokumentů, jež mají kontext mezi vybranými anotacemi.

Následuje XML podoba anotací příkladů ze str. 42.

## *XML překlad příkladů*

```
<zkousi>
  <ucitel>
    Jan_Ostrý
  </učitel>
  <studenti>
    <pocet>
      13
    </pocet>
    <student>
      Petr_Chytrý
    </student>
    <student>
      Jana_Pilná
    </student>
    <student>
      Matěj_Pohodlný
    </student>
  </studenti>
  <predmet>
    Práce_na_PC
  </predmet>
  <typ_mistnosti>
    Počítačová_místnost
  </typ_mistnosti>
</zkousi>
```

```
<sleduje>
  <reporter>
    Tutti_Ponti
  </reportér>
  <zapas>
    <hraje_proti>
      <tym>
        Sparta
      </tym>
      <tym>
        Juventus_Turín
      </tym>
    </hraje_proti>
  </zapas>
</sleduje>
```

Je vidět, že je odstraněna těžkopádnost relačního modelu pro množinové a hnízděné koncepty, atomy lze snadno realizovat jako vhodně pojmenované tagy.

Nevhodnost překladu do ORM modelu pro symetrický vztah týmů v zápase je také odstraněna.

Výhodou XML podoby anotačních záznamů by mohlo být kromě jiného i využití nově vznikajících dotazovacích jazyků na XML dokumenty.

# Závěr

Porovnání ontologické a CG anotace

Přínosy a výsledky práce

# Porovnání ontologické a CG anotace

Tato práce předkládá novou metodu podpory pro vyhledávání informačních zdrojů založenou na strukturaci anotací informačních zdrojů do podoby konceptuálních grafů. Pro danou doménu, *svět zájmu*, navrhuje vytvořit tak zde nazývaný business konceptuální graf, představující typový model kontextů, jež mají být vlastním obsahem anotací. Tuto navrhovanou metodu je možno srovnat z metodou autorů [Abe], kteří k analogickému účelu používají konstrukci informatických ontologií.

Jaké jsou přednosti metody v této práci navrhované, a je zde nějaká podobnost obou přístupů? Nejprve podobnost: Business konceptuální graf je možno považovat za analogii k podnikové i doménové ontologii, s tím že osobně nevidím ostrou hranici mezi těmito dvěma (ve shodě s paradigmatem procesního řízení v managementu). Rozdíly konstrukce ontologií oproti business konceptuálnímu grafu vidím hlavně v použitých typech vztahů – namísto obecných „patří do“, „je částí“, „má vlastnost/atribut“, „vztahuje se k“ atd. navrhuji používat hlavně doménově specifické vztahy, zejména predikáty popisující specifické podnikové činnosti.

Dalším rozdílem je myšlený požadavek na úplnost ontologií, jež považuji za prakticky špatně dosažitelný.

Podstatnou předností je *srozumitelnost* konstrukcí konceptuálních grafů, tedy speciálně business konceptuální grafy považuji za podstatně uživatelsky přátelštější komunikační prostředek, než [Abe] navrhované ontologie.

Proto podle mého může být mnou navrhované použití business konceptuálních grafů vhodnou alternativou k jiným metodám podpory pro knowledge management.

# Přínosy a výsledky práce

Práce zvažuje možnosti využití ontologií v informatice a konceptuálních modelů ORM.

V této práci je podrobně vypracována konstrukce business konceptuálních grafů, je vytvořen překlad business konceptuálních grafů do modelů ORM, je vytvořena a popsána metoda tvorby anotací k informačním zdrojům pomocí business konceptuálních grafů, je vytvořen a popsán postup kladení dotazů nad business konceptuálním grafem a je vytvořen a dostatečně popsán eventuelní nový dotazovací jazyk nad business konceptuálním grafem.

Je naznačeno, jak lze anotace tvořené v business konceptuálním grafu přeložit do struktury vyhovující standardu XML.

Výhledem do budoucna je vytvoření podpory vyhledávání za pomocí speciálních indexních struktur.

# Dodatky

# Dodatek 1

Formální definice konceptuálních grafů je vyložena v [Sow]. Proto zde nepodávám úplný rigorózní výklad, ale poloformální výklad těch částí, které dále ve své práci využívám.

## *Konceptuální graf*

Konceptuální graf je bipartitní graf se dvěma typy uzlů, zvanými *koncepty* a *konceptuální vztahy*.

- Každá hrana grafu spojuje vždy jeden koncept s jedním konceptuálním vztahem. Říkáme, že hrana *patří* tomu konceptuálnímu vztahu a *je připojena* k tomu konceptu.
- Některé koncepty nemusí být spojeny s žádným konceptuálním vztahem, pak jim říkáme *osamělé*.
- Konceptuální graf sestávající pouze z jediného konceptuálního vztahu a konceptů připojených k hranám patřícím tomuto vztahu se nazývá *hvězda*.

Konceptuální graf s *n* konceptuálními vztahy může být zkonstruován z *n* hvězdových grafů překrytím odpovídajících si konceptů.

Konceptuální grafy budu znázorňovat jako grafy (viz obrázek 3) na str.13, koncepty v nich jako obdélníky a konceptuální vztahy v nich jako ovály, nebo je budu zapisovat v lineárním textovém zápisu (pro obrázek 3):

`[Údržbář] ← (Agnt) ← [Používat] → (Thme) → [Kleště]`

koncepty v hranatých a konceptuální vztahy v kulatých závorkách (obojí v souladu s [Sow]).

Nadále budu místo „konceptuální graf“ používat označení CG.

## *Koncept*

Každý koncept má *typ* a *referenta*. V zobrazení CG je typ obvykle umístěn vlevo v konceptu a referent vpravo, a jsou odděleny dvojtečkou. Jestliže referent chybí (je prázdný), je jím implicitně existenční kvantifikátor. Následující obrázek 18 ukazuje příklady:


**obrázek 18: příklady konceptů**

Zde je v prvním případě typ konceptu „Autobus“ a referent je prázdný, čili se mluví o nějakém existujícím autobusu bez bližšího určení. V druhém konceptu je typ „Osoba“ a referent je „Jan“, tj. mluví se o konkrétní osobě, jež se jmenuje Jan.

## *Typ konceptu*

*Typová hierarchie* je částečně uspořádaná množina, v níž částečné uspořádání je dáno vztahem podtyp-nadtyp. Každý typ je buď primitivní nebo definovaný. Definované typy jsou určeny λ-výrazy[57] v primitivních nebo předtím definovaných typech. Následující příklad je definice typu „ŘidičReferent“:

`ŘidičReferent = [Zaměstnanec : λ] ← (AbsolvovatŠkoleníŘidičů)`

## *Referent konceptu*

Referent konceptu ukazuje na výskyt či výskyty (instance) z typu konceptu, které jsou pojmovým obsahem konceptu. Skládá se z *kvantifikátoru* a *designátoru*. Kvantifikátor je buď *existenční*, jejž je možno i vynechat, nebo definovaný. Konkrétní příklady definovaných kvantifikátorů uvedené v [Sow] jsou: *universální* ∀, *množství* (např. @1, @2 atd.), a *kolekce* (např. {Tomáš, Petr, Pavel}, kde Tomáš, Petr a Pavel jsou designátory)[58]. Designátor je buď *literál*, *lokátor* nebo *deskriptor*. Literály jsou jakýmisi zobrazeními referentů, jakýmisi jejich zakódovanými záznamy, je k nim přidána i informace o použitém způsobu zakódování. Lokátor je buď jméno či jména, jako Tomáš, Petr a Pavel v příkladu nahoře, nebo ukazovatel, #ISBN-201-14472-7, tj. identifikační kód v nějakém dohodnutém způsobu identifikace (je uvozen znakem #). Deskriptory jsou CG, které referent popisují. Příklady konceptů s jednotlivými druhy referentů viz obrázek 19.

(V mnou navrhované metodě mohou být koncepty s literály využity v uživatelském rozhraní ke zvýšení tzv. přívětivosti systému. Ve své práci předpokládám využití hlavně lokátorů, a referentů s kvantifikátory množství a kolekcemi. Počítám však i s využitím konceptů s deskriptory, jež nazývám hnízděnými konceptuálními grafy.)

---

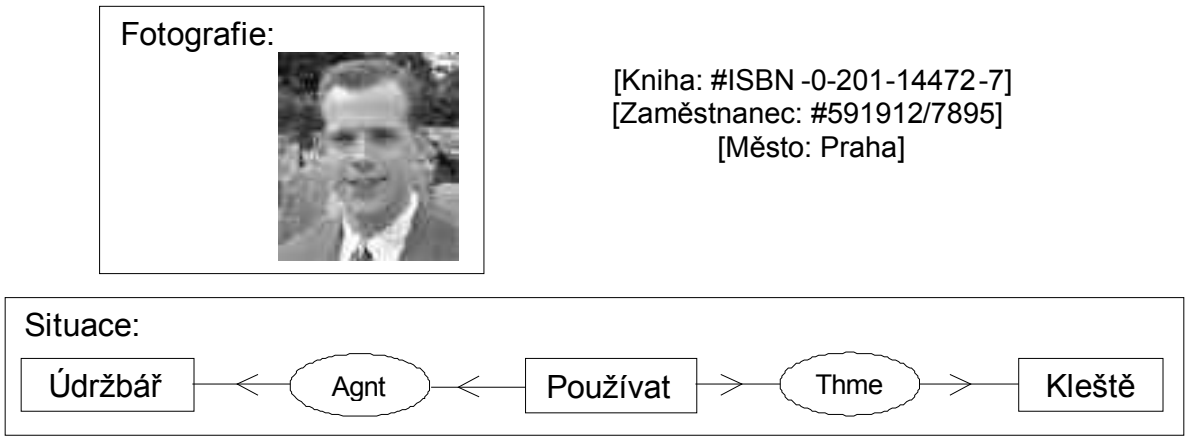
[57] Co jsou λ-výrazy viz Lambda výraz na str.58

[58] V [Sow] není zmíněn jiný typ kolekcí než množiny, tedy explicitně se nehovoří ani o multimnožinách ani o posloupnostech. V případě potřeby takových konceptů lze využít umělých konstrukcí – např.:

`[Multimnožina: {[Typ_prvku:…]→(MáVýskytů)→[Počet:…],…}]`

nebo

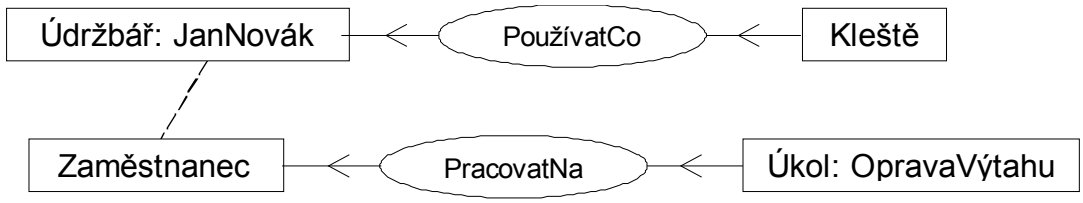
`[Posloupnost: {[Pozice:…]→(JeObsazena)→[Typ_prvku:…],…}]`




**obrázek 19: koncepty s existenčním referentem: literál, lokátory a deskriptor**

## *Koreferenční propojení*

Některé koncepty v CG mohou být propojeny tzv. koreferenčním propojením, jehož význam je ten, že spojují totéž, tj. že pojmové obsahy propojených konceptů jsou totožné. Příklad užití viz obrázek 20:




**obrázek 20: koreferenční propojení**

Koreferenční propojení v lineární textovém zápisu toho CG vypadá takto:

```
[Kleště] → (PoužívatCo) → [Údržbář: JanNovák] - - -
 - - -[Zaměstnanec] ← (PracovatNa) ← [Úkol : OpravaVýtahu]
```

(pokud jsou propojené koncepty hned za sebou) nebo za použití koreferenčních návěští:

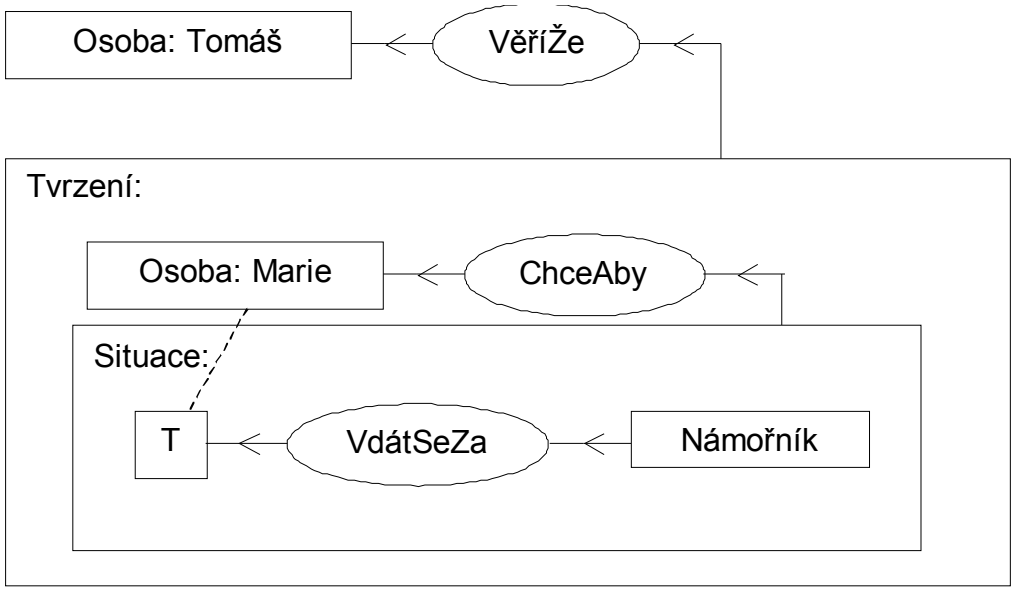
```
[Údržbář: JanNovák *x] ← (PoužívatCo) ← [Kleště]
[Zaměstnanec: ?x] ← (PracovatNa) ← [Úkol : OpravaVýtahu]
```

zde `*x` je definiční a `?x` je vázané návěští.

Koreferenční propojení se používá hlavně v CG s tzv. kontexty. Ve své práci předpokládám možnost využití i těchto tzv. kontextů, kterým říkám zahnízděné CG, a v části týkající se praktické realizace případného projetu se zabývám využitím těchto konstruktů.

## *Kontexty v CG*

Kontextem se CG nazývá každý koncept, jehož deskriptor je neprázdný konceptuální graf.
Následující obrázek 21 je příklad CG se dvěma kontexty, jedním vnořeným ve druhém:

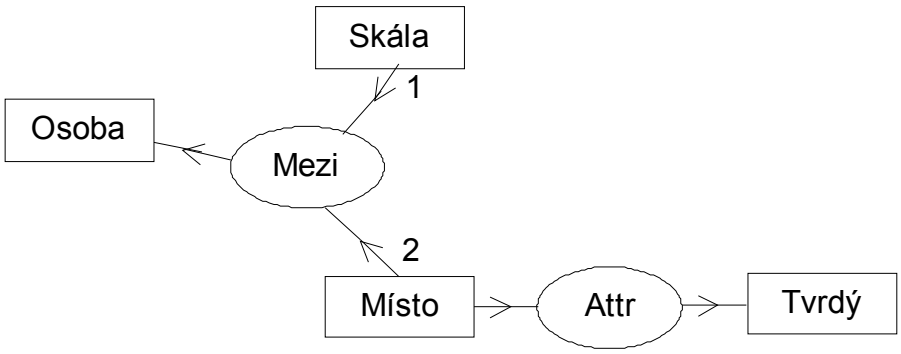



**obrázek 21: CG vyjadřující „Tomáš věří, že se Marie chce vdát za námořníka."**

Znak **T** v CG označuje „cokoli", přesněji univerzální všeobjímající typ. Pro mnou zamýšlené užití konceptuálních grafů k anotaci dokumentů, i v ostatních v této práci zamýšlených využitích pro podporu knowledge managementu, zdají se mi být konstrukce podobné příkladu viz obrázek 21 příliš složité. Nacházím však jiné důvody k zahnízďování CG do konceptů připojeným k dalším konceptuálním relacím.

## *Konceptuální relace*

Konceptuální relace má *relační typ*[59], *vaznost*, a *signaturu*. Vaznost je počet hran, které k ní patří. Hrany, které ke konceptuální relaci patří, jsou uspořádány, a ke každé je přiřazen typ konceptů, které k ní mohou být připojeny. Signatura je uspořádaná n-tice těchto typů, když n je vaznost. Koncepty, které mohou být připojeny k i-té hraně konceptuální relace, musí mít typ, jenž je shodný s i-tým členem signatury relace, nebo je jeho podtypem. Následující obrázek ukazuje relaci typu „Mezi" s vazností 3 a signaturou <**T**,**T**,**T**>:




**obrázek 22: „Osoba je mezi skálou a tvrdým místem."**

[59] relační typ je identifikován jménem relace

Konceptuální relace Agnt, kterou obsahuje obrázek 19, má vaznost 2 a signaturu <Čin, Živoucí>[60]. V grafickém zobrazení grafu se k prvním n-1 hranám připisuje jejich pořadí, a jejich šipky směřují k relaci, poslední hrana má šipku směřující ven. U relací vaznosti 1 (*unárních*) a 2 (*binárních*) není číslování hran třeba. Relace vaznosti 3 nazývám *ternární* [61]. Lineání textový zápis CG předchozího příkladu je

```
[Osoba] ← (Mezi) -
     ← 1 - [Skála]
     ← 2 - [Místo] → (Attr) → [Tvrdý]
```

## *Lambda výraz*

Pro každé přirozené číslo n, n-ární lambda výraz je CG, ve kterém je n konceptů s designátory ve formě volných parametrů, obyčejně označených $\lambda_1$, $\lambda_2$, …, $\lambda_n$. Lambda výrazy jsou užívány k definici nových typů konceptů a nových typů konceptuálních relací.

## *Typ relace*

Relační hierarchie je částečně uspořádaná množina typů relací, částečné uspořádání je dáno vztahem podtyp-nadtyp. Každý typ relace je buď primitivní nebo definovaný. Definované typy jsou určeny λ-výrazy s primitivními nebo předtím definovanými typy. Následující textový zápis je příklad definice nového typu relace:

```
PoužívatCo =
 [Živoucí : λ1] ← (Agnt) ← [Používat] → (Thme) → [T: λ2]
```

Relace nově definovaného typu „PoužívatCo“ mají signaturu <Živoucí, **T**>.

## *Překlad CG do přirozeného jazyka*

[Sow] na str.140 píše, že CG mohou být přeloženy do stylizované angličtiny. Pro překlad do češtiny není, pokud je mi známo, vytvořen žádný algoritmus. Dalo by se použít přeložení anglického překladu do „češtiny“ nezohledňující ohebnost češtiny. Pro mnou zamýšlené aplikace je možno vytvořit překladový stroj pro určitý daný model „světa zájmu“. Opačný překlad, z přirozeného jazyka do CG, je problematický. Algoritmizace takového překladu je v dnešní době nemyslitelná. Pro expertní překlad probíhá stále vývoj v teorii CG, pro podporu základních lingvistických struktur přirozeného jazyka. Na str.448 v [Sow] píše autor, že CG jsou navrhovány tak, aby měly přímý převod do a z přirozeného jazyka.

---

[60] <Act, Animate>

[61] [Sow] používá termíny *monadické*, *dyadické* a *triadické* relace.

## *Překlad CG do KIF a ORM*

Dle [Sow] na str.438, autoři [Han] se svými studenty vytvořili programy na překlad z a do CG těchto notací: ER diagramy, NIAM[62], dataflow diagramy (a další). ORM je verze NIAM, tedy se domnívám, že existuje přinejmenším překlad z ORM do CG. Překlad z CG do ORM je možný jen v omezené množině případů, jeden takový překlad ve své práci předkládám.

Protože konceptuální grafy lze prostřednictvím predikátového kalkulu přeložit do KIF, je mnou navrhovaná aplikace dále použitelná pro zpracování odvozovacími stroji, které podporují KIF.

[62] o eventuálních omezeních na možnost překladu nemám žádné informace, i když je předpokládám

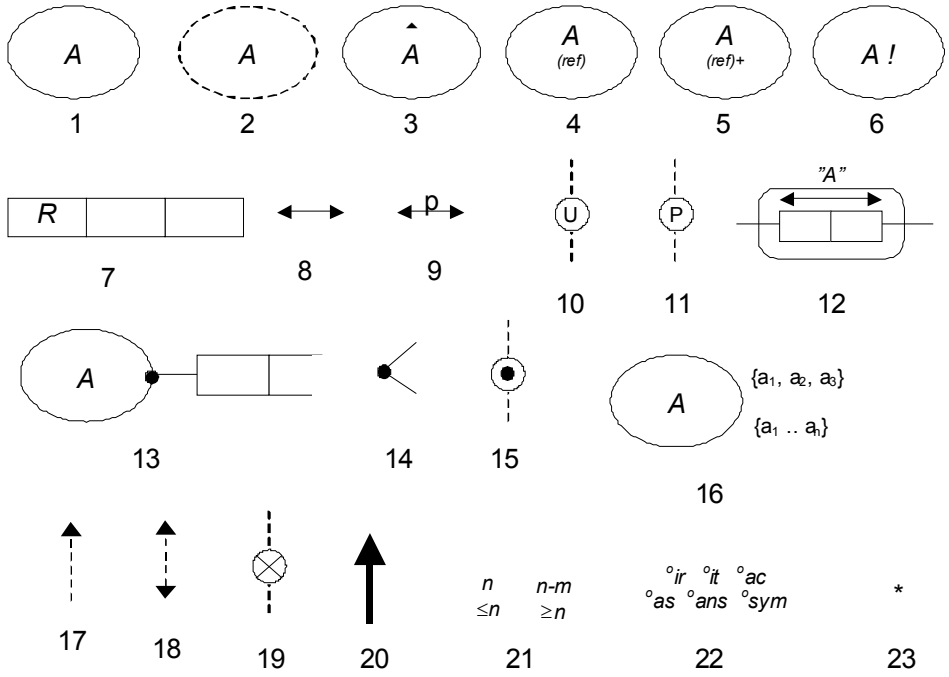
# Dodatek 2

## *Přehled ORM*

ORM zobrazuje svět zájmu v termínech objektů (entit a hodnot), jež hrají role ve vztazích ([Hal1]), odtud jméno ORM – Object Role Modeling. ORM je modelovací metoda, a jako taková zahrnuje jak značení, tak i procedury použití tohoto značení.

Značení ORM zahrnuje formální, textový specifikační jazyk, jak pro modely tak i dotazy, stejně jako formální grafický modelovací jazyk. Textový jazyk má větší expresivitu než grafický, ale zde budu zmiňovat pouze grafický jazyk. Grafická podoba modelu je doplňována textovými dodatky.

Hlavní symboly používané v grafickém jazyce ORM zobrazuje obrázek 23. Symboly jsou očíslovány kvůli odkazům.




**obrázek 23: hlavní ORM symboly**

Objektové typy se dělí na entitní a hodnotové. Hodnotový typ je lexikální objektový typ, např. textový řetězec nebo číslo.

1 *Entitní typ* se zobrazuje jako pojmenovaná elipsa.

2 *Hodnotový typ* se obvykle označuje jako pojmenovaná tečkovaná elipsa. Jiný způsob značení, vhodný pro textovou podobu modelu, je uzavření jména typu do závorek.

3 Objektové typy, které se ve schématu objevují na více místech (kvůli přehlednosti), jsou označeny špičkou šipky, jež má „ukazovat“ na existenci dalších výskytů.

Každý entitní typ musí mít alespoň jedno *referenční schéma*, jež říká, jak mohou být instance tohoto typu jednoznačně určeny pomocí jedné či více hodnot.

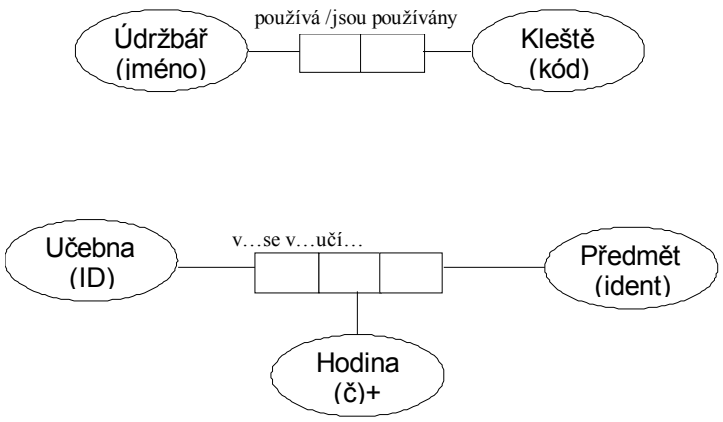
4 Jednoduché referenční schéma (určení entity pomocí jediné hodnoty) může být zkráceně zapsáno jako *referenční mód* do závorek pod jméno typu do elipsy nebo bezprostředně za jméno typu v textové podobě zápisu (např. Země(kód) )

5 Jsou-li hodnoty, jež slouží k referenci entit, číselné, může být přidán znak + za referenční mód (např. Váha(kg)+ )

Hodnoty jsou obecně srozumitelně zapisovány, takže nepotřebují referenci. Ačkoli se nejedná o striktně konceptuální záležitost, je obvyklé vyžadovat, aby pro každý entitní typ bylo některé z jeho referenčních schémat *primární*. Typy vztahů, které jsou použity pro primární referenční schéma, se pak nazývají *referenční typy*. Ostatní typy vztahů jsou *typy faktů*.

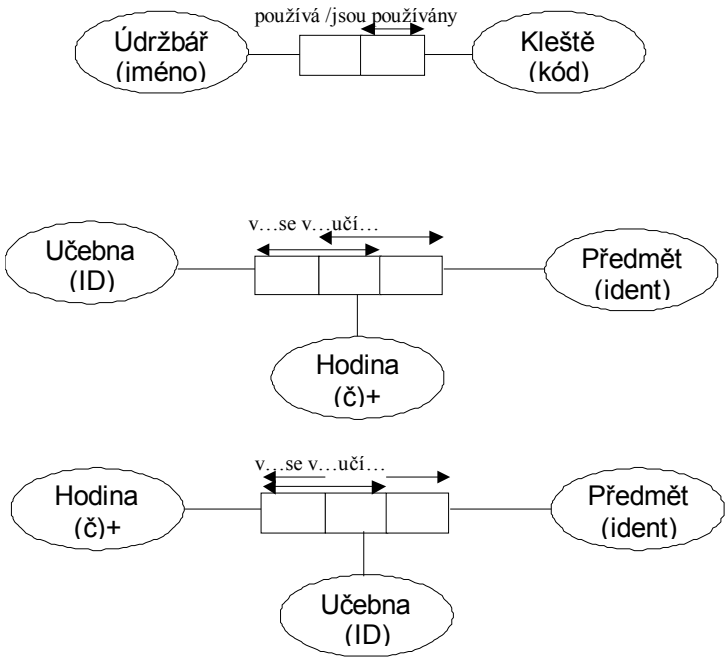
6 *Nezávislý* entitní typ se označuje vykřičníkem. Nezávislost typu znamená, že instance tohoto typu mohou existovat, aniž by hráli roli v některém faktu. Toto se nepovažuje za běžné, a není-li typ explicitně označen jako nezávislý, zahrnují se do něj instance pouze tehdy, pokud se účastní nějakého faktu.

7 Symbol 7 označuje ternární predikát, tj. predikát složený ze tří *rolí*. Každá role se zobrazuje jako obdélník, a musí být připojena k právě jedinému objektovému typu (viz symbol 13). Predikát je v podstatě věta s mezerami pro objekty, které hrají ve větě role, pro každou roli jedna mezera (např. pro větu „Údržbář používá kleště.“, jež odpovídá typu faktu „Údržbář používá Kleště“ je odpovídající predikát „…používá…“).
Počet rolí se nazývá *aritou* predikátu. ORM připouští predikáty libovolné arity (1 – unární, 2 – binární, 3 – ternární, atd.). Role v predikátu jsou obvykle uspořádány.
V takovém případě se jméno predikátu zapisuje do nebo blízko prvního obdélníku pro roli (eventuelně se symbolem „…“ pro objektové mezery).
V modelech mohou být zahrnuty různé způsoby „čtení“ predikátu (např. také „Kleště jsou používány údržbářem“ – „…jsou používány…“). ORM doporučuje pro n-ární typ faktu n směrů čtení, pro každou z rolí takový směr čtení, který by od ní začínal. Příklad ternárního predikátu pro typ faktu „v Učebně se v Hodinu učí Předmět“ je „v…se v… učí…“, jiný směr čtení je „… se učí v…v…“ a třetí: „v…se v …učí…“, přičemž v posledním slouží mezery po řadě pro objekty Hodina, Učebna a Předmět. Grafické značení pro oba zmíněné typy faktů ukazuje obrázek 24. Pro binární predikát je možno alternativní směr čtení zapsat za znak / , jak to ukazuje obrázek.




**obrázek 24:** binární a ternární typy faktů

8 *Omezení vnitřní jedinečnosti* se zobrazují jako pruhy se šipkami, a umísťují se nad jednu či více rolí v predikátu. Znamenají, že instance pro tyto role (resp. jejich kombinace) jsou v populaci tohoto typu faktu jedinečné. Predikát může mít jedno či více omezení vnitřní jedinečnosti. Následující obrázek 25 přidává tato omezení. Ternární typ faktu je zde zobrazen i v jiném pořadí rolí v predikátu, a obrázek ukazuje, že je možno pruh označující omezení jedinečnosti přerušit nad rolí či rolemi, které nezahrnuje.

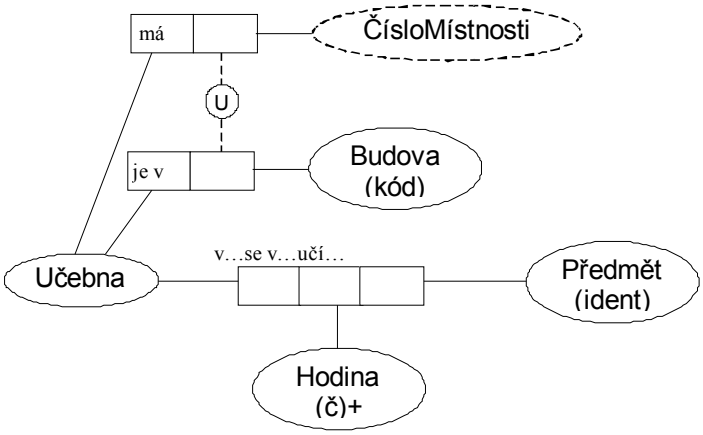



**obrázek 25**: omezení vnitřní jedinečnosti

9 Jedno z omezení vnitřní jedinečnosti pro predikát může být označeno jako *primární* přidáním „P“.

10 *Omezení vnější jedinečnosti* se zobrazuje jako „u“ v kroužku. Může být použito pro dvě nebo více rolí z různých predikátů tak, že je s tímto symbolem spojíme tečkovanými čarami. Toto znamená, že kombinace instancí těchto rolí ve spojení (join) těchto

predikátů jsou jedinečné. Například Učebna v předchozím příkladu je identifikována kombinací identu budovy a číslem místnosti. Pokud v modelu chceme také používat, pro nějaký účel, typ Budova (např. se chceme ptát, které učebny jsou v dané budově, či kteří pracovníci mají kancelář ve stejné budově, jako daná učebna), bude odpovídající část modelu vypadat viz obrázek 26. (Jak je zde vidět, modely ORM mohou být relativně rozsáhlé, oproti kompaktnějším modelům UML.) Situace může být ještě komplikovanější, neboť můžeme chtít rozlišovat patra v budovách, a pak omezení vnější jedinečnosti bude spojeno se třemi rolemi v různých predikátech. Referenčními typy vztahu pak budou: „Učebna je v Budově“, „Učebna je na Patře“, „Učebna má Číslo_místnosti“. Model může zahrnovat i typ faktů: „Patro je v Budově“ (viz obrázek 28).




**obrázek 26**: omezení vnější jedinečnosti

11 Chceme-li deklarovat omezení vnější jedinečnosti jako primární, použijeme „P“ místo „u“.

12 Typ vztahu může být objektivizován, tak aby mohl hrát role v jiných vztazích. Objektivizovaný typ vztahu se označuje obkreslením obdélníkem s kulatými rohy, a je pojmenován. Objektivizovaným typům vztahu se také říká *zahnízděné* objektové typy. Typicky musí mít objektivizovaný predikát omezení vnitřní jedinečnosti přes všechny role, ale případy 1:1 jsou výjimkou (přesněji viz 0.7 v odstavci Transformace do logického databázového schématu).

13 *Omezení povinnosti role* říká, že každá instance objektového typu, který je k této roli připojen, musí hrát tuto roli. Obvykle se značí černým puntíkem. Povinné role se také nazývají *totální* role (viz obrázek 27).

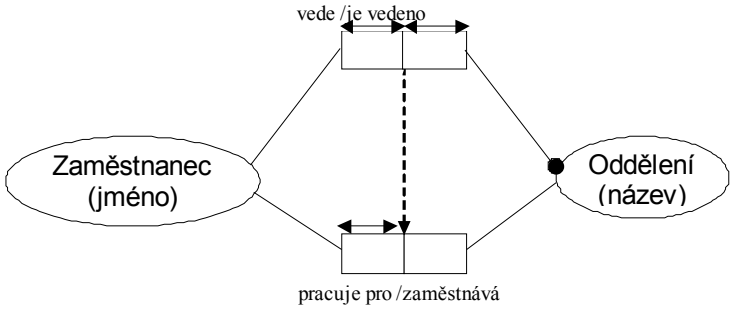
14 *Disjunktivní* omezení povinnosti může být použito pro dvě či více rolí v případě, že instance objektového typu musí hrát alespoň jednu z označených rolí. Je možno ho označit černým puntíkem u společného připojení těchto rolí k objektovému typu.

15 Jiné označení disjunktivní povinnosti se dělá spojením černého puntíku v kroužku tečkovanou čarou s příslušnými rolemi.

16 *Omezení hodnot* říká, že populace objektového typu má být omezena na daný seznam. Může být označeno připsáním seznamu do složených závorek vedle elipsy (symbol 16, nahoře), nebo, jsou-li hodnoty uspořádány, může být rozsah deklarován první a poslední hodnotou oddělenými „..“ (symbol 16, dole).

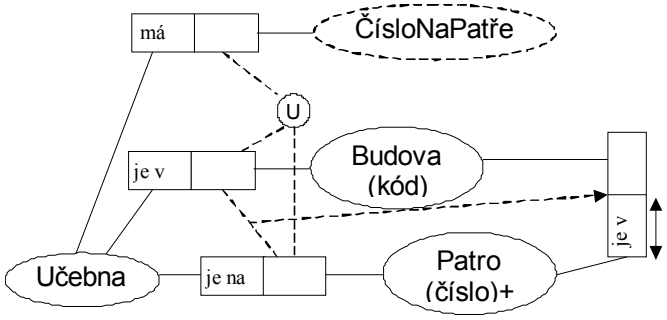
Symboly 17 až 19 označují *omezení množinovým porovnáním*. Mohou být použity jen v případě srovnatelných posloupností typů, tj. posloupností jedné či více rolí, jejichž „hostitelský“ objektový typ je tentýž nebo srovnatelný typ na stejných pozicích srovnávaných posloupností.

17 *Omezení podmnožinové* se označuje tečkovanou šipkou od jedné posloupnosti rolí ke druhé. Následující obrázek 27 znázorňuje takové omezení mezi dvojicí rolí typu faktu „Zaměstnanec vede Oddělení“, a typu faktu „Zaměstnanec pracuje pro Oddělení“.




**obrázek 27**: omezení povinnosti role a omezení podmnožinové

Jiný příklad užití podmnožinového omezení ukazuje následující obrázek 28:




**obrázek 28**: jiné omezení podmnožinové, omezení vnější jedinečnosti

18 *Omezení rovností* se označuje dvojitou šipkou. Znamená, že obě populace si musí být rovné.

19 *Omezení vylučující* se označuje křížkem v kroužku a znamená, že populace jsou vzájemně disjunktní. Například dvojice rolí v predikátu typu faktu „Osoba je autorem Článku“ a dvojice rolí v predikátu typu faktu „Osoba recenzuje Článek“ jsou vzájemně výlučné.

20 (Vlastní) *podtyp* nějakého typu je s ním spojen tučnou šipkou. (Podle ORM musí podtypy být definované na základě specifikace vztahů, v nichž hraje roli nadtyp.)[63]

21 Tři typy *omezení frekvence* ukazuje symbol 21. Je možno je aplikovat k posloupnosti jedné či více rolí.

22 Symbol 22 ukazuje šest druhů *kruhových omezení*. Mohou být použity pro dvojici rolí, jež obě jsou hrány stejným objektovým typem. Po řadě znamenají: antireflexivitu (ir), antitransitivitu (it), acykličnost (ac), asymetrii (as), antisymetrii (ans) a symetrii (sym).

23 Hvězdička může být připojena blízko k typu faktu, pokud tento typ faktu je odvozen z jiných typů faktů v modelu zahrnutých.

Ne všechny verze ORM (NIAM) podporují všechny tyto symboly, ale (jediná) verze podporovaná CASE nástrojem je podporuje všechny. Některá další značení používaná v teoretických pracích dosud nejsou v CASE nástroji podporována.

Navíc je možno přidávat další omezení v textové podobě, ve formalizovaném jazyce FORML nebo v jazyce ConQuer.

## *Procedura tvorby konceptuálního schématu*

[Hal] podrobně popisuje proceduru tvorby schématu, zde jen vyjmenuji jednotlivé kroky:

1. Převeďte známé příklady informací do podoby elementárních faktů, a proveďte kontrolu kvality.
2. Načrtněte diagram typů faktů a proveďte kontrolu populací.
3. Zkontrolujte entitní typy, jež by měly být zkombinovány (tvorba typové hierarchie), a poznamenejte všechna odvození.
4. Přidejte omezení jedinečnosti, a zkontrolujte aritu typů faktů.
5. Přidejte omezení povinných rolí, a zkontrolujte logická odvození.
6. Přidejte všechna hodnotová, množinová omezení a vymezení podtypů.
7. Přidejte ostatní omezení a proveďte závěrečnou kontrolu.

[63] To se může zdát pro tvorbu modelů zbytečně omezující, protože to vnáší nutnost zahrnovat umělé typy vztahů.

## *Transformace do logického databázového schématu*

Programový CASE nástroj podporující tvorbu ORM schémat nabízí transformaci do logického databázového schématu řady cílových SŘBD. Obecná procedura tvorby schématu v relačním databázovém modelu, nazvaná **Rmap**, je popsána v [Hal][64]. Je téměř deterministická, jedině v několika bodech nechává možnost volby pro uvážení analytika.

Procedura **Rmap** je navržena s ohledem na to, aby výsledné schéma bylo *korektní*, tj. ekvivalentní se schématem ORM, *efektivní*, tj. aby umožňovalo dobrou dobu odezvy pro aktualizace dat a dotazy, a *zřejmé*, tj. aby bylo relativně snadné ho pochopit a pracovat s ním (péče o zřejmost je založena na vhodných volbách pro jména tabulek a atributů).

Jestliže výchozí ORM schéma obsahuje pouze elementární fakty (viz str.17 zde), pak výsledné relační schéma je normalizované.

Jednotlivé kroky stručně popisuje následující přehled:

0. Absorbujte všechny podtypy do jejich vrchního nadtypu[65]. Odmyslete si všechny explicitní primární identifikační schémata, s objektovými typy nadále zacházejte jako s černými skříňkami, obsahující objekty bez zřetele na způsoby jejich identifikace.
1. Mapujte každý typ faktu s omezením vnitřní jedinečnosti zahrnujícím více než jednu roli (vztahy n:m:…) do samostatné tabulky.
2. Typy faktů s jednoduchým omezením vnitřní jedinečnosti (vztahy 1:n) seskupte podle typu objektů, jež hrají funkcionální role zahrnuté v těchto omezeních (podle konce 1-).
   Seskupení utvoří vždy samostatnou tabulku, s primárním klíčem založeným na identifikačním schématu daného objektového typu.
   Případy 1:1 mapujte do samostatné tabulky (obecně se dává přednost vyhnout se prázdným hodnotám[66]).
3. Každý nezávislý objektový typ mapujte do samostatné tabulky.
4. Rozbalte každou černou skříňku (nyní vždy v nějakém sloupci nějaké tabulky) do atributů založených na identifikačním schématu příslušného objektového typu.
5. Transformujte všechna ostatní omezení a pravidla odvození.
   Omezení podtypů (v ORM mají být podtypy definovatelnými typy na základě faktů, v nichž

[64] Jejím výsledkem je popis schématu v jakémsi abstraktním relačním jazyce definice dat, z něhož je pak prováděn překlad do konkrétního jazyka zvoleného SŘBD. Ne vždy jsou všechny součásti ORM modelu v cílovém schématu obsaženy, záleží to na tom, jestli zvolený SŘBD podporuje potřebné rysy.

[65] Toto může být analytikem změněno, defaultní je absorpce.

[66] I pro toto má analytik jinou volbu.

hraje roli nadtyp) mapujte do podmíněně volitelných sloupců tabulky nadtypu, pro role ve vztazích n:m:… definujte podmíněná omezení podmnožin.

Všechny kroky, v nichž má analytik možnost volby, a všechny ostatní obtížnější části transformační procedury, jsou zahrnuty do rozvinuté podoby kroku 0:

0.1 V duchu binarizujte všechny unární typy faktů, zvažte vhodný do úvahy připadající „uzávěr světa“.

0.2 V duchu si odmyslete primární referenční schémata. Se složeně identifikovanými typy objektů zacházejte jako s černými skříňkami.

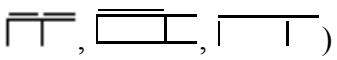
0.3 Označte všechna nedefaultní rozhodnutí tam, kde podtypy nebudou absorbovány do svých nadtypů[67].

0.4 Označte všechny odvozené typy faktů, jež mají být ukládány.

0.5 Indikujte rozhodnutí o volbě mapování symetrických 1:1 predikátů.

0.6 Zvažte nahrazení libovolného disjunktivního identifikačního schématu (schématu, v němž objekty, jež mají být identifikovány, hrají nepovinné role[68]) použitím umělého či zřetězeného identifikátoru, či volbou default hodnot (a změnou nepovinných rolí na povinné).

0.7 Označte volbu mapování pro eventuelní případy objektivizovaných predikátů, v nichž omezení vnitřní jedinečnosti nezahrnuje všechny role. (V korektním ORM schématu mají všechna omezení vnitřní jedinečnosti objektivizovaného predikátu délku alespoň n-1, kde n je arita predikátu. Pokud některá role není zahrnuta do žádného z omezení vnitřní jedinečnosti, má být predikát rozložen na dva. Přípustné jsou tedy typově tyto možnosti:



*Ve všech případech, kdy je ponechána volba při mapování ORM schématu do relačního modelu, je pro metodu anotací dokumentů popisovanou v této práci vhodné zvolit tu z možností, která poskytuje nejlepší efektivitu dotazů. Aktualizace anotací předpokládám že budou pouze výjimečné, z hlediska provozu zanedbatelné.*

[67] [Hal] popisuje tři rozumné způsoby mapování ISA vztahů do relačního modelu: absorpce (tabulka je jediná pro všechny objekty jedné ISA hierarchie, atributy podtypů tvoří volitelné sloupce), separace (fakty specifické pro podtyp jsou mapovány do samostatné tabulky), rozklad (každý podtyp tvoří samostatnou tabulku, nadtyp je tak rozdělen do několika tabulek – to je vhodné při úplném disjunktním členění nadtypu do podtypů).

[68] Například v biologické klasifikaci se může použít název druhu, a pokud je druh použit, ještě je pak volitelný název poddruhu.

*Ze stejného důvodu je možné zvážit porušení elementarity faktů v ORM modelu, a tím následnou transformaci do nenormalizovaného relačního schématu, čímž může být podstatně zvýšena efektivita dotazů.*

*Další z možností je zvolit pro mapování jiný než relační model.*

## *Jazyk ConQuer*

Jazyk ConQuer je dotazovací jazyk nad konceptuálními schématy v modelu ORM. Konceptuální schéma v modelu ORM může být postupem reverzního inženýrství vytvořeno z relačního databázového schématu, a tak může být jazyk ConQuer použit nad jakoukoli relační databází. Ve zmíněném procesu reverzního inženýrství je designérovi umožněno vybírat vhodná jména typů objektů a formulace predikátů.

Zvláštností je, že použití jazyka ConQuer v nástroji InfoAssistant nevyžaduje, aby uživatel znal konceptuální schéma, nad kterým klade dotazy, ani aby znal značení modelu ORM. Nástroj uživateli střídavě nabízí objekty a predikáty, jejichž výběrem se formulují věty ve formalizovaném jazyce ConQuer, jež jsou současně srozumitelnými větami v angličtině. Odpovídající podpora pro češtinu by vyžadovala péči o české tvarosloví.

### *Vyjadřovací prostředky jazyka ConQuer*

*Vyjadřovací síla* jazyka se zakládá na:

Základní koncepci ORM modelu, v níž typy objektů jsou sémantické domény, jež se při tvorbě modelu používají tak, aby vzájemně srovnatelné entity i hodnoty byly výskyty buď jednoho a téhož typu, či nadtypu a podtypu. Tak objektové typy tvoří „sémantické lepidlo“ modelu, a informační pohledy na data je možno vytvářet jako cesty v grafu, procházející od nějakých objektových typů k jiným objektům vybranými predikáty, přičemž jsou označovány (zaškrtnuty znakem ✓) požadované informační položky.[69]

Omezující podmínky výběru se připisují k objektovým typům, jimiž dotaz „prochází“. Nejjednodušší podmínka omezující možný výskyt typu, jímž může dotazová cesta procházet, je

[69] Jazyk ConQuer se velmi podobá relačnímu dotazovacímu jazyku QBE. Jsou zde však dva podstatné rozdíly: návrh dotazu se dělá nad schématem, jež je souvislým grafem a propojení již zahrnuje v sobě; multimnožina predikátů dotazu je částečně uspořádaná „průchodem s větvením“.

konkrétní hodnota referenčního módu objektu připsaná bezprostředně za jméno typu (referenční mód je možno explicitně pojmenovat nebo ne). Pro porovnání dvou hodnot ve stejném typu (či v nadtypu a podtypu) je možno použít proměnné. Další možné omezující podmínky mohou být založeny na porovnání >, <, >=, <=, <>, `in`, `like`[70]... Logické kombinace jsou podpořeny použitím `and`, `or`, `not` , přičemž `and` je implicitní součástí větvení cesty dotazu, `or` je možno k větvení připsat, `not` se připisuje před predikát a znamená negaci celé podvětve dotazu pod tímto predikátem.

Analogicky k levému vnějšímu spojení v relačních dotazovacích jazycích nabízí ConQuer operátor `maybe` či synonymně `possibly`, jež se připisuje před predikát.

Spojení mezi podtypem a nadtypem se zapisuje predikátovým jménem `is`.

Je možno vytvářet agregace `count`, `sum`, `avg`,...[71] .

V dotazech je možno používat odvozené predikáty.

Následují příklady dotazů v jazyce ConQuer, v tzv. „outline“ podobě:

```
✓Zaměstnanec
    └ bydlí ve Město
            └ je sídlo Pobočka 52
```

(Vypiš všechny zaměstnance co bydlí ve městě, ve kterém je pobočka 52.)

```
✓Zaměstnanec
    └ má ✓Plat > 15000
    └ either umí count(Jazyk)> 1
    └ or – řídí Auto
            └ má Barva 'červená'
```

(Vypiš všechny zaměstnance, kteří mají plat více než 15 000 a hovoří více než jedním jazykem nebo řídí červené auto.)

---

[70] není velký problém nahradit anglická klíčová slova jazyka českými ekvivalenty

[71] Buď jako v QBE aplikováním agregační funkce na hodnotový typ v cestě dotazu, přičemž „group by“ je specifikováno zaškrtnutími ✓ v dotazu. Nebo je možno pro „having“ podmínku použít explicitní specifikaci „group by“ objektů:

```
✓Oddělení
   └ zaměstnává ✓Zaměstnanec
                  └ dosáhl Hodnocení
                          └ max(Hodnocení) for Zaměstnanec > avg(Hodnocení) for Oddělení
```

✓Manažer
    └ **is** Zaměstnanec
        └ **possibly** řídí ✓Auto
        └ **not** pracuje v Oddělení 52

(Vypiš všechny manažery a jejich auta (když nějaká mají), kteří nepracují v Oddělení 52.)

✓$Zaměstnanec_1$
    └ bydlí ve $Město_1$
    └ pochází ze $Země_1$
    └ dohlíží na $Zaměstnanec_2$
        └ bydlí ve $Město_1$
        └ pochází ze $Země_2$ <> $Země_1$

(Kdo dohlíží na zaměstnance, který bydlí ve stejném městě ale pochází z jiné země?)

Pro podpoření *sémantické srozumitelnosti* nabízí InfoAssistant verze dotazů přímo v angličtině. Pro češtinu by byla nutná tzv. lokalizace. Uvádím anglickou verzi prvního z ConQuer dotazů uvedených výše, s anglickými jmény objektových typů a predikátů:

**List each** Employee **that** lives in City **that** is location of Branch 52.

a s českými:

**List each** Zaměstnanec **that** bydlí v Město **that** je sídlo Pobočka 52.
**Vypiš všechny** Zaměstnanec **, kteří** bydlí v Město **, které** je sídlo Pobočka 52.[72]

Příklady uvedené výše mají ukázat, že ConQuer je *snadný*. Dále mají ukázat, že ConQeur je *sémanticky relevantní*, neboť formulace dotazů nevyžaduje nic, co souvisí s implementací datových struktur, ale nesouvisí se smyslem dotazu (v relačním modelu užití tabulek implementujících vztahy, vícehodnotové atributy, znalost o tom, které atributy jsou klíčem, které implementují spojení…)

Posledním zdůrazňovaným rysem jazyka ConQuer je *sémantická stabilita*, která se projeví, když se časem schéma změní například tak, že se do měj přidají nějaké typy faktů, nebo se změní kardinalita nějakého vztahu (což v odpovídajícím logickém databázovém modelu může způsobit změnu atributů na referenční propojení s nějakými tabulkami). ConQuer dotaz není třeba v takových případech přeformulovávat.

[72] Překlad klíčových slov není obtížný, ale srozumitelnost pro češtinu vyžaduje navíc péči o tvarosloví.

### *Překlad ConQuer do SQL*

Jazyk ConQuer má rigorózně definovanou formální sémantiku, a na základě ní a na základě informace o mapování ORM schématu do relačního databázového schématu má InfoAssistant definovaný překladový stroj do SQL. Tento překladový stroj využívá informace z integritních omezení konceptuálního schématu k *sémantické optimalizaci* překladu (viz [Hal2]).

# Dodatek 3

## *Elementarita konceptuálních relací*

Elementarita konceptuální relace se dá posuzovat tak, že zvážíme rozložení této relace na více relací a zjišťujeme, zda tím dojde ke ztrátě informace, tzn. zda následným opětným složením vzniklých relací vzniknou falešná tvrzení. Například fakt

Údržbář Jan Novák použil dne 31.5.2002 kleště číslo K578.

bychom se mohli pokusit rozložit na

Údržbář Jan Novák použil dne 31.5.2002 kleště.
Údržbář Jan Novák použil kleště číslo K578.

Zvážíme-li příslušné typy faktů

Údržbář(jméno) použil Dne(datum) Kleště(číslo).
Údržbář(jméno) použil Dne(datum) kleště.
Údržbář(jméno) použil Kleště(číslo).

pak zjistíme, že složením 2. a 3. typu faktu a použitím instancí z nich vytvoříme falešná tvrzení.

Takovéto zjišťování rozložitelnosti se zakládá na tom, že dokážeme najít instance faktů, jež vyvrátí hypotézu rozložitelnosti. Ne vždy však má analytik k dispozici „signifikantní populaci“ typů.
V takovém případě lze použít postup, jenž je analogický postupu doporučovanému ORM metodou. Tento postup je založen na nalezení *klíče* (klíčů) *relace*.

Klíčem konceptuální relace bude každá minimální množina hran patřících k této relaci taková, že objekty v reálném světě, jež jsou označeny referenty konceptů připojených k této relaci těmito hranami, již jednoznačně určují objekty, které jsou označeny referenty zbylých konceptů a jsou v odpovídajícím vztahu k nim. Specielně v případě konceptů, jež mohou být v business CG kvantifikovány množstvím nebo kolekcemi, myslím příslušnými *objekty* v reálném světě *množiny objektů* typu, jenž je typem tohoto konceptu.

Pak platí: Jestliže je klíčem konceptuální relace celá množina všech hran patřících k této relaci, pak relaci nelze rozložit.

Toto pravidlo nedává postačující podmínku rozložitelnosti relace (jak ukazuje příklad s helmami na str.17, v něm jsou dva klíče délky 2). Také neříká, jak lze eventuelně nejlépe rozložit rozložitelnou relaci:

Údržbář Jan Novák použil dne 31.5.2002 kleště číslo K578, které byly červené.

lze vhodně rozložit:

Údržbář Jan Novák použil dne 31.5.2002 kleště číslo K578.
Kleště K578 jsou červené.

(když barva kleští je stálá).

Formálně lze sice definovat rozložitelnost i samotné rozložení nalézt na základě funkčních závislostí v relaci, ale to je pro praktickou analýzu málo použitelné v případě, že informace o závislostech je třeba získávat od uživatelů – doménových expertů. I samotné získávání informace o klíčích relace u více-árních relací je pracné (ORM metoda doporučuje ověřování protipříkladů na konkrétních instancích typů).

Příznakem toho, že příslušný typ faktu je rozložitelný, je například použití spojky „a“. Není to však pravidlem, jak ukazuje následující:

Jan Novák šel 31.5.2002 do skladu a půjčil si kleště č. K578.

může být pouze reformulací dříve uvedeného faktu[73].

Všechny poznámky i úvahy, které uvádím v tomto odstavci, jsou převzaty z metod ORM, až na jednu výjimku: Pokud koncept v konceptuální relaci může být kvantifikován množstvím nebo množinou, chápu to tak, že to, co tento koncept zastupuje v reálném světě, jsou množiny objektů reálného světa – tyto množiny jsou nepojmenované, jsou identifikovány výčtem svých prvků. Klíč příslušné relace se pak týká jedinečnosti n-tic *množin* instancí typů konceptů, připojených k relaci uvažovanými hranami. Ověřování na příkladech s uživateli, doménovými experty, pak je v tomto ohledu složitější.

Ještě jednu změnu oproti metodě ORM zde přijímám: ORM metoda říká, že pokud je klíč relace (typu faktu) kratší než n-1, kde n je arita relace (typu faktu), pak je třeba relaci (typ faktu) rozložit a použít hnízdění (objektivizaci predikátu). Já pro účely své práce považuji hnízděné koncepty za prvek narušující srozumitelnost business CG, a tak doporučuji se jim pokud možno vyhýbat.

Hnízděný koncept je podle mne vhodné použít jen tehdy, když mu uživatelé zcela rozumí, a to je nejlépe tehdy, když sami takový koncept používají, tj. *pojmenovávají*. – Vytvoření umělého konceptu skrývá nebezpečí, že nebude pochopen.

Pojetí elementarity relace pak lze lépe vyjádřit takto:

*Relace není elementární, pokud lze nalézt* uživatelům srozumitelný *rozklad této relace na více jiných relací, jejichž opětným složením nemohou vzniknout falešné fakty. Jinak je relace elementární.*

[73] ve skutečnosti fakt, že si někdo půjčil kleště, a fakt, že je použil, nejsou tytéž, a je možné, že je v analýze rozlišíme…

Tato formulace nechává nevyjasněn ještě jeden problém, a to, zda připustíme, aby rozkladem relace vznikly redundance:

Zaměstnanec Alois Vodička získal kvalifikaci „vrtulkář“ dne 25.7.1997 udělením komise č.13.

Toto můžeme rozložit, vyhýbajíce se hnízdění:

Zaměstnanec Alois Vodička získal kvalifikaci „vrtulkář“ dne 25.7.1997.
Zaměstnanec Alois Vodička získal kvalifikaci „vrtulkář“ udělením komise č.13.

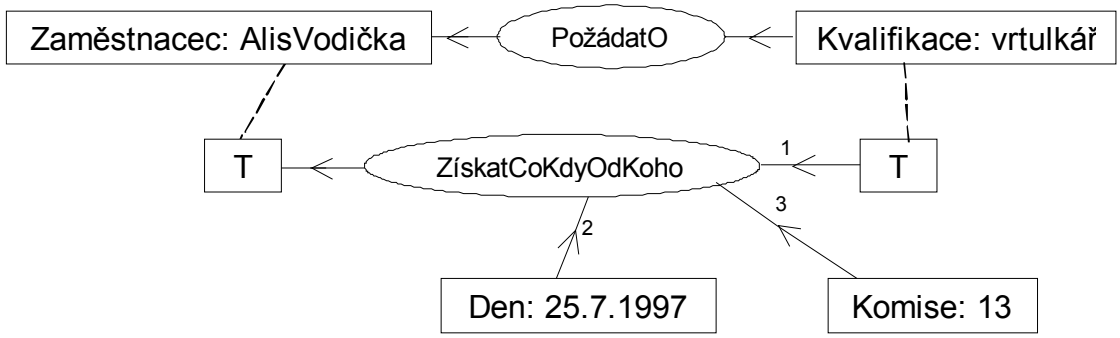
ale fakt, že Alois Vodička je vrtulkář, je zde obsažen dvakrát, redundantně. Proto je lépe ponechat relaci v původním tvaru.

Někdy ovšem může jít o jiný případ:

Zaměstnanec Qido Horáček požádal o kvalifikaci „vrtulkář“. (Tuto kvalifikaci ještě nezískal.)
Zaměstnanec Alois Vodička požádal o kvalifikaci „vrtulkář“. Tuto kvalifikaci získal dne 25.7.1997 udělením komise č.13.

Fakta o Aloisi Vodičkovi lze vyjádřit konceptuálním grafem viz obrázek 29.[74] Zde je jasně vidět, že „požádat o“ a „získat co od koho“ jsou různé predikáty, a pouze shodou okolností (pravidly, jež se v organizaci dodržují), je zde vztah inkluze mezi predikátem „získat co“, jenž je projekcí predikátu „získat co od koho“, a predikátem „požádat o“.

Metoda ORM navrhuje zahnízdit predikát „požádat o“ a *substituovat jím* predikát „získat co“, pokud k tomu získání došlo[75]. *Sémantický obsah* obou predikátů je ovšem *jiný*.




**obrázek 29: Zaměstnanec Alois Vodička požádal o kvalifikaci „vrtulkář“. Tuto kvalifikaci získal dne 25.7.1997 udělením komise č.13.**

Z předchozích příkladů zobecňuji kritérium vhodnosti rozkladu relace:

[74] Zda by v business CG koncepty typu Zaměstnanec a T splynuly, je věc přehledného uspořádání, nebo tvorby atomů – o atomech viz dále v kapitole Návrh business konceptuálního grafu. Podobně Kvalifikace a T.

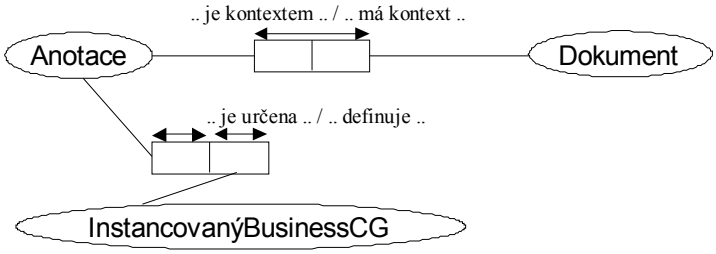
[75] Tento zahnízděný predikát do konceptu „Žádost“ pak má mít nepovinné členství ve vztahu „ …(Žádost) je naplněna …(Dne) …(Komisí)“.

*Relaci je vhodné rozložit na více jiných, pouze když nemohou vzniknout redundantní fakty. Také je třeba respektovat sémantický obsah navrhovaných relací.*

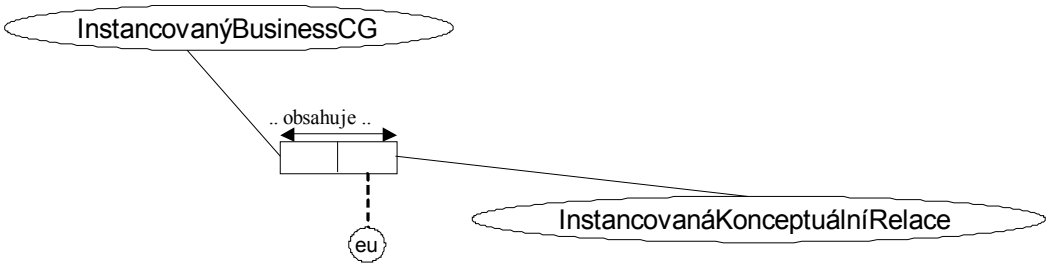
Poznámka: Pokud uživatelé budou souhlasit se zahnízděním vztahu/ů, je možno CG tímto způsobem upravit. Výhoda toho je v menší rozsáhlosti modelu, model je pak kompaktnější.

## *Základní krok transformace business CG do ORM*

Instancovaný business CG je kontextem dokumentu, anotace je určena instancovaným business CG, můžeme tedy zjednodušeně říci, že anotace je kontextem dokumentu:




Instancovaný business CG je množinou, tj. výčtem, jednotlivých instancovaných konceptuálních relací[76]:

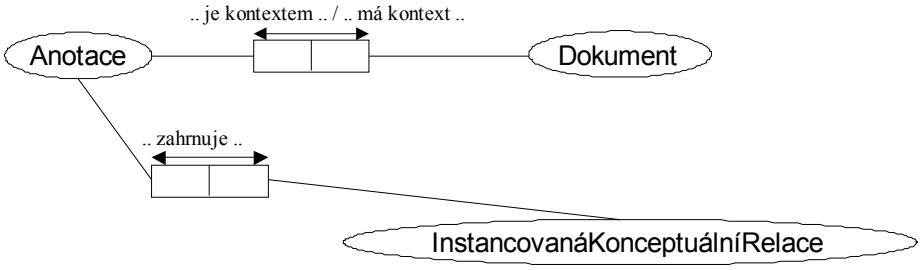



(Zde „eu“ v kroužku je další z omezeních použitelných v ORM modelech, které však CASE nástroj InfoModeller nepodporuje. Znamená „extensional uniqueness“. Nadále toto omezení vynechám, neboť se nebude mapovat do logického modelu databáze[77].) Vzhledem k 1:1 vztahu anotace a instancovaného business CG, instancovaný business CG z modelu vynechám[78]:

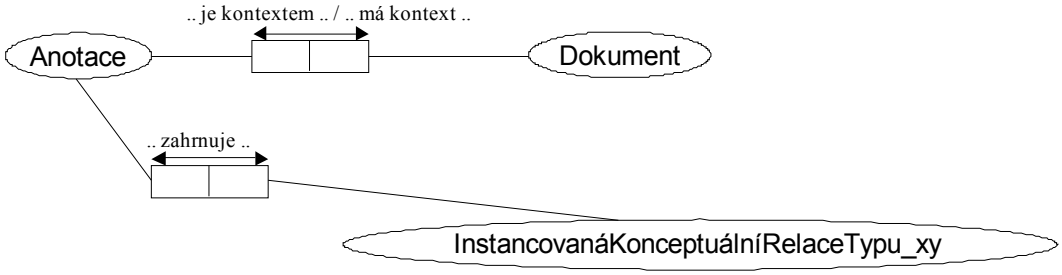
[76] Ve skutečnosti je instancovaný business CG výčtem instancovaných knceptuálních relací a „líných“ konceptů, tj. těch, které nejsou připojeny k žádné relaci. Podle zásad návrhu busines CG to však mohou být pouze hnízděné koncepty. O nich pojednává další část, Transformace hnízděných konceptů.

[77] takové omezení je velmi drahé provozovat jako databázové integritní omezení

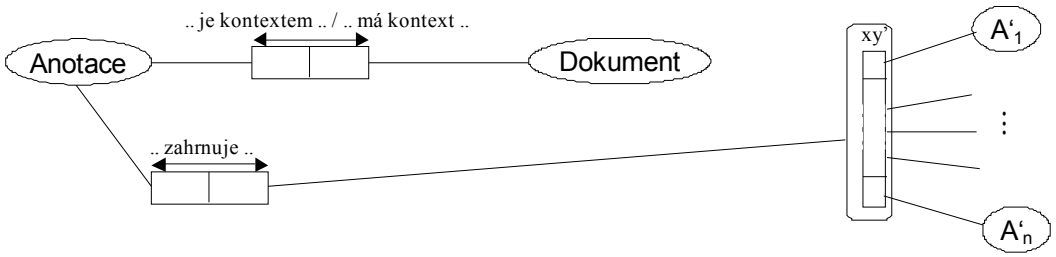
[78] Vynechání omezení „eu“ způsobí, že v datovém modelu budou považovány za různé dvě anotace, které se skládají ze stejného komplexu instancovaných konceptuálních relací, ale jsou vytvořeny nezávisle na sobě. (srv. též poznámku 38 na str.32)



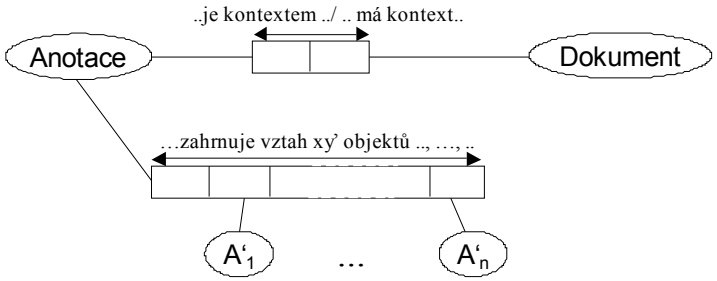

Nyní je třeba instancované konceptuální relace roztřídit podle jejich typů, a v duchu udělat krok, kdy předchozí obrázek velmi zkomplikujeme tím, že pro každý typ konceptuální relace xy modelujeme typ vztahu „anotace zahrnuje instancovanou konceptuální relaci typu xy“. Omezíme se dále jen na jediný typ konceptuální relace:




kde xy je nějaký typ konceptuální relace z business CG, jehož vaznost je n a signatura je $<A_1,..,A_n>$. Instancované konceptuální relace můžeme v ORM modelu modelovat jako objektivizované vztahy objektů odpovídajících konceptům připojeným k této konceptuální relaci:




kde xy’ volíme pro jméno predikátu, odpovídajícího typu relace xy, $A'_1,..,A'_n$ označují sémantické datové typy odpovídající typům konceptů $A_1,..,A_n$. V dalším kroku provedeme „odhnízdění“ objektivizovaného vztahu:

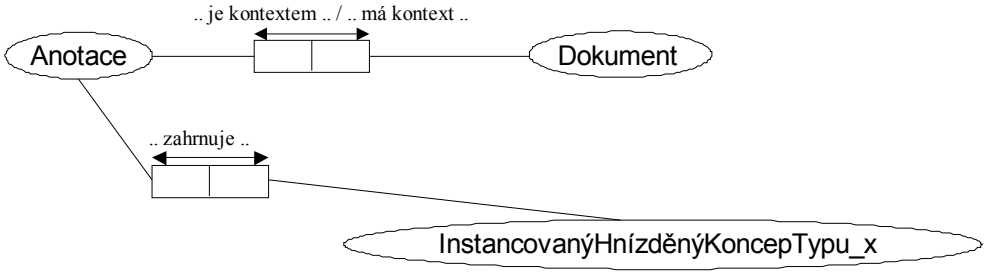


(Zda omezení vnitřní jedinečnosti, tj. klíč, posledního predikátu zahrnuje všechny jeho role či nikoli, není podstatné. Záleží to na tom, zda predikát xy' má klíč zahrnující všechny jeho role, či nikoli. Jak již bylo poznamenáno na str.67, není snahou vytvořit v maximální míře normalizovaný logický databázový model, ale kritériem efektivity implementace bude rychlost vyhodnocování dotazů.)

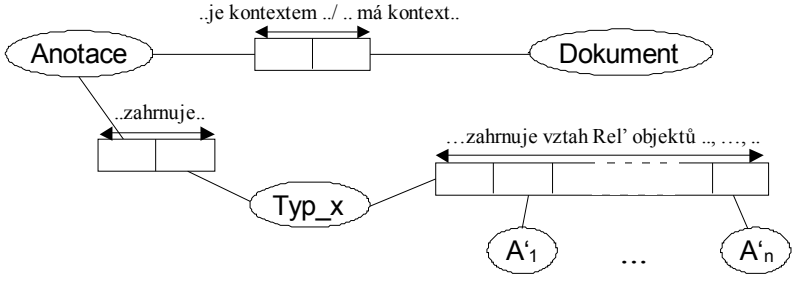
Protože nemůže dojít k nedorozumění, budu typy konceptů v business CG a jim odpovídající typy objektů v ORM modelu pojmenovávat stejně.

## *Transformace hnízděných konceptů*

Jak zmiňovala poznámka 76 na str.75, instancovaný business CG je výčtem instancovaných konceptuálních relací *a* instancovaných hnízděných konceptů (ty jediné mohou být „líné"). Analogicky k úvahám v části Základní krok transformace business CG do ORM, pro každý typ x hnízděného konceptu ORM model obsahuje:

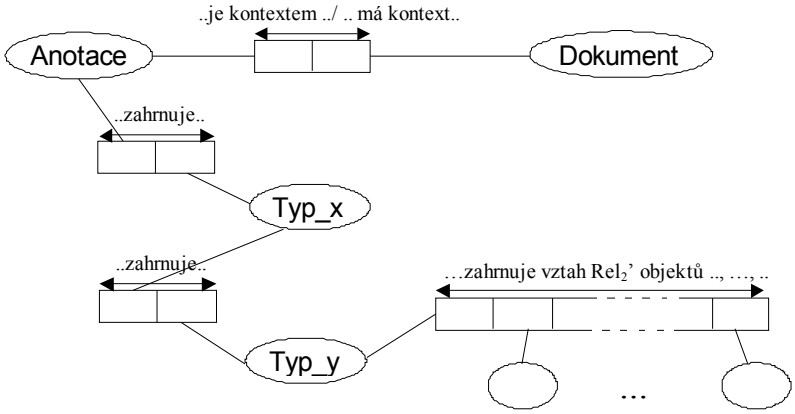



Instancovaný hnízděný koncept lze považovat za „anotaci" nad jiným „business CG", jejž tvoří graf vnořený v hnízděném konceptu typu x. Můžeme znovu provést úvahy předchozí části o základním kroku transformace, a modelovat:




(*)

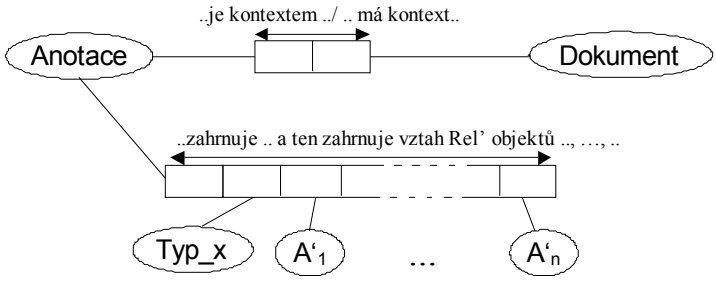
pro každou konceptuální relaci Rel se signaturou $<A_1,\ldots,A_n>$ obsaženou v CG vnořeném v uvažovaném hnízděném konceptu. Jestliže ve vnořeném CG je další „líný" hnízděný koncept typu y, dostáváme:



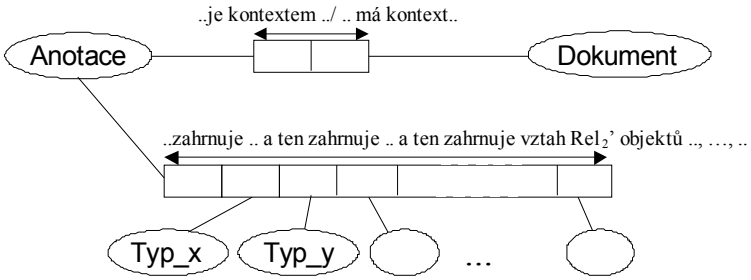

(**)

pro každou konceptuální relaci $Rel_2$ obsaženou tomto dalším hnízděném konceptu. Stejně lze postupovat dál a dál. Otázkou je, zda v nejhlubší úrovni zahnízdění najdeme nějaké konceptuální relaci – musíme, protože tam již nejsou žádné „líné“ koncepty.

Typy faktů „Anotace zahrnuje Typ_x“ a „Typ_x zahrnuje Typ_y“ jsou sice elementární, ale jejich elementarita není pro účely celého sytému přínosná, spíše naopak. Převažující manipulace nad databází anotací budou vyhledávání dokumentů podle jejich kontextových anotací, a přidávání anotačních záznamů. Vyhledávání napříč anotacemi považuji za netypické a vzácné (lze ho například provádět za pomoci jazyka ConQuer nad ORM modelem). Takže typické manipulace by vyžadovaly neustálé propojování faktů „Anotace zahrnuje Typ_x“ a „Typ_x zahrnuje Typ_y“, což by bylo málo efektivní. Proto navrhuji upravit ORM model takto, pro případ (*):




pro případ (**):




mapování dále zahnízděných konceptů je zřejmé.

## *Super-relace atomů*

Mějme libovolný atom v CG, buďte $K_1,\ldots,K_n$ všechny koncepty pod-grafu odpovídajícího tomuto atomu. Řekneme, že $K_i$ je ekvivalentní[79] s $K_j$, jestliže jsou $K_i$ a $K_j$ stejného typu a jsou propojeny koreferenčním propojením. Buď $\{ K'_1,\ldots,K'_m \}$ nějaká maximální podmnožina $\{ K_1,\ldots,K_n \}$, ve které nejsou žádné dva prvky vzájemně ekvivalentní. Vytvořme konceptuální graf z pod-grafu odpovídajícího uvažovanému atomu, v němž v konceptech $K'_1,\ldots,K'_m$ jako designátory budou po řadě volné proměnné $\lambda_1,\ldots, \lambda_m$ a v každém ze zbylých konceptů $K_i$ bude jako designátor volná proměnná $\lambda_j$ , pokud $K_i$ je ekvivalentní s $K'_j$.

Takto vytvořený konceptuální graf je λ-výrazem, který definuje super-relaci daného atomu.

Definujme ještě pro pozdější účely pod-graf super-relace jako CG, v němž k super-relaci připojíme i-tou hranou koncept $K'_i$.

Jméno (typ) super-relace může odpovídat nějakému zvolenému identifikátoru atomu, toto zde není třeba specifikovat. Pokud jsou atomy pojmenované, může super-relace nést stejné jméno[80].

## *Popis transformace*

Nejprve objasním otázky konceptů a konceptuálních relací z CG zahnízděných v jiných konceptech.

### *Terminologie hnízdění*

Všechny koncepty a konceptuální relace business CG jsou v úrovni zahnízdění 0. Business CG je v úrovni zahnízdění 0 v sobě.

Nechť n je celé číslo, n≥0. Je-li koncept $K_1$ či konceptuální relace $Rel_1$ součástí nějakého konceptuálního grafu $G_1$ zahnízděného v nějakém konceptu $K_2$ konceptuálního grafu $G_2$, a je-li $K_2$ v úrovni zahnízdění n, řeknu, že

*$K_1$ a $Rel_1$ a $G_1$ jsou v úrovni zahnízdění n+1 v daném business CG*

a

*$K_1$ a $Rel_1$ jsou přímo zahnízděny v $K_2$.*

### *Atomy*

Každý CG v nějaké úrovni zahnízdění má definovány atomy, jež tvoří rozklad množiny všech konceptuálních relací tohoto CG.

---

[79] Koreferenční propojení je implicitně tranzitivní (tzn. že koreferenčně jsou propojeny i koncepty, které nejsou viditelně graficky propojeny, ale existuje cesta z viditelných koreferenčních propojení mezi nimi). Jeden a tentýž koncept je sám se sebou také implicitně koreferenčně propojen.

[80] Jako dokumentační vysvětlení můžeme super-relaci popsat obšírněji.

### *Množina všech konceptů*

Množina všech konceptů pro business CG je množinou všech konceptů z nějakých CG, pro něž existuje n tak, že jsou v úrovni zahnízdění n v tomto business CG.

### *Typy konceptů*

Jak bylo nepřesně uvedeno v části Koncepty na str. 23, v množině všech konceptů pro business CG se mohou některé koncepty „opakovat“, což přesně znamená, že se v této množině mohou vyskytovat různé koncepty se stejným typem. Navíc v různých konceptech stejného typu mohou být použita různá synonymní jména typu – synonymie je na pozadí business CG evidována.

Explicitně ujasňuji, že dva koncepty, z nichž jeden je množinový a druhý ne, mohou být stejného typu.

Pokud není v business CG dodržena zásada, že synonymní jména typů pro hnízděné koncepty se používají pouze pro shodné[81] zahnízděné CG, je možno provést malou modifikaci typů zahnízděných konceptů, při níž se tyto ještě specifikují podle typu zahnízděných CG[82].

### *Množina všech konceptuálních relací*

Množina všech konceptuálních relací pro business CG je množinou všech konceptuálních relací z nějakých CG, pro něž existuje n tak, že jsou v úrovni zahnízdění n v tomto business CG.

### *Před-transformace*

Nejprve provedeme transformaci popsanou v části Koncepty s jedinou instancí na str.33 se všemi koncepty z množiny všech konceptů pro business CG a konceptuálními relacemi z množiny všech konceptuálních relací pro business CG.

Dále ztotožníme všechny koncepty z množiny všech konceptů pro business CG, jež jsou stejného typu, jsou propojeny koreferenčním propojením, a jsou společně přímo zahnízděny v nějakém konceptu. Připojení hran ke konceptuálním relacím odpovídajícím způsobem přesuneme.

Bez újmy na obecnosti předpokládejme, že náš business CG je již výsledek takovéto před-transformace.

### *Vztah anotace a dokumentu*

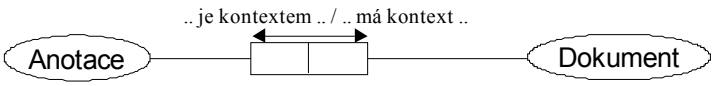
Do ORM modelu vložíme sémantický datový typ „anotace“, s umělým identifikátorem, jímž může např. být textový kód instancovaného business CG.

---

[81] Přesná definice shodnosti vyžaduje vzájemně jednoznačné zobrazení mezi koncepty, mezi konceptuálními relacemi a připojeními konceptů ke konceptuálním relacím, v němž odpovídající si koncepty i konceptuální relace jsou stejného typu. Vztah koncept-připojení/hrana-konceptuální relace se musí tímto zobrazením zachovávat.

[82] Typ CG lze definovat na základě shodnosti CG. Oba pojmy – shodnost i typ CG – je třeba nejprve definovat pro CG bez hnízděných konceptů, a pak indukcí pro ostatní CG.

Dále do modelu vložíme sémantický datový typ „dokument“. Vložíme n:m vztah „anotace“ a „dokumentu“:




### *Modelování konceptů v ORM*

Každý typ konceptu z množiny všech konceptů pro business CG transformujeme do ORM modelu jako objektový typ. Pro jeho jméno zvolíme některé ze synonym, jež pro tento typ připadají v business CG do úvahy.

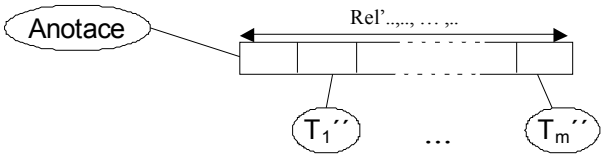
Pro každý typ konceptu A, který je typem nějakého množinového konceptu v množině všech konceptů pro business CG, přidáme do modelu ORM hodnotový typ Počet_A-prvků a ještě jeden objektový typ A’, dále přidáme vztah „..patří do..“ kardinality n:m mezi A a A’ a vztah „..má..“ kardinality n:1 mezi A’ a Počet_A-prvků. Referenční schéma objektového typu A’ je založeno na umělém identifikátoru. Všechny hodnotové typy Počet_A-prvků jsou podtypy hodnotového typu Počet.

Objektové typy, jež odpovídají hnízděným konceptům, budou mít umělý identifikátor, může jím být např. textový kód instancovaného CG vnořeného v tomto konceptu.

### *Modelování konceptuálních relací*

Modelování konceptuálních relací definujeme podle úrovně zahnízdění. Všechny konceptuální relace v úrovni 0 sdružíme do atomů v této úrovni, transformujeme všechny super-relace těchto atomů:

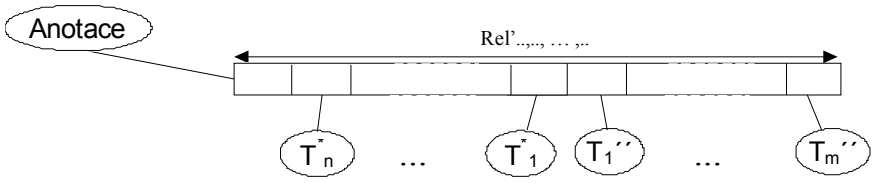
Je-li Rel super-relace nějakého atomu, m její arita, $<T_1,\ldots,T_m>$ její signatura, $K_1,\ldots,K_m$ koncepty připojené k pod-grafu této super-relace, pak tuto relaci transformujeme do (m+1)-árního vztahu Rel’ v ORM modelu takto:




kde $T_i'' = T_i$ , pokud $K_i$ není množinový, jinak $T_i'' = T_i'$, kde „$T_i$ patří do $T_i'$ “ modeluje množinový koncept $K_i$. (Sémantiku této transformace vysvětlují části Základní krok transformace business CG do ORM na str. 75, Atomy na str.33, a Množinové koncepty na str.34.)

Nechť Rel je super-relace nějakého atomu A v úrovni zahnízdění n, n≥1, nechť m je arita Rel, $<T_1,\ldots,T_m>$ signatura Rel, $K_1,\ldots,K_m$ koncepty připojené k pod-grafu Rel. Buďte $K^*_1,\ldots,K^*_n$ koncepty takové, že konceptuální relace atomu A jsou přímo zahnízděny v konceptu $K^*_1$ , $K^*_i$ je přímo

zahnízděn v $K^*_{i+1}$ pro $1<i<n$ , $K^*_n$ je v úrovni zahnízdění 0. Buďte $T^*_1,\ldots,T^*_n$ po řadě typy konceptů $K^*_1,\ldots,K^*_n$ . Relaci Rel transformujeme do (m+n+1)-árního vztahu Rel' v ORM modelu takto:




kde $T_i´´=T_i$ , pokud $K_i$ není množinový, jinak $T_i´´=T_i´$, kde „$T_i$ patří do $T_i´$ " modeluje množinový koncept $K_i$. (Sémantika této transformace vysvětlena v části Transformace hnízděných konceptů Dodatku 3.)

### *ISA vztahy*

ISA vztahy v business CG se přímo transformují do odpovídajících ISA vztahů v modelu ORM. Tedy explicitně vyjádřeno, pokud v ISA vztahu v business CG je některý z konceptů v tomto vztahu množinový, ISA vztah v ORM modelu se týká objektového typu, který odpovídá modelovanému typu konceptu, a nikoli umělého objektového typu představujícího příslušnou kolekci: je-li A typ konceptu, jenž je množinový, a modeluje-li tento koncept typ vztahu „A patří do A' ", pak případný ISA vztah se týká objektového typu A, nikoli typu A'.

### *Transformace odvozovacích pravidel*

Transformaci odvozovacích pravidel zakládám na transformaci λ-výrazů, jež odvození definují, doplněných o informaci souvislostí konceptuálních relací s atomy. Pokud by byla odvozovací pravidla dána jinak než λ-výrazy, následující popis poskytne alespoň ideu, jak transformaci provést.

Mějme λ-výraz definující odvozený koncept nebo odvozenou konceptuální relaci v business CG, k tomuto λ-výrazu je přidána informace, zda konceptuální relace v něm, jež patří do množiny všech konceptuálních relací pro business CG a jsou z jednoho atomu v nějaké úrovni zahnízdění, jsou takto při odvození také myšleny, tj. mají být ve společné super-relaci. Ve skutečnosti musí být množina konceptuálních relací z definičního λ-výrazu, jež patří do množiny všech konceptuálních relací pro business CG, rozložena do disjunktních podmnožin, z nichž každá je částí nějakého atomu v příslušné úrovni zahnízdění v business CG, a z nichž každá značí, že všechny konceptuální relace v ní mají být v jediné super-relaci. Nazvěme tyto podmnožiny sub-atomy definičního λ-výrazu.

V definičním λ-výrazu mohou být navíc obecné porovnávací relace ($\neq$, $<$, $>$, $\leq$, $\geq$, in, …) a koreferenční propojení, a dále logické relace (and, or, not) mezi hnízděnými koncepty. Protože jde o odvození v business CG, všechny typy konceptů, kromě hnízděných konceptů připojených k logickým relacím, v definičním λ-výrazu pocházejí z množiny všech konceptů pro business CG, a

také typy hvězdových pod-grafů definičního λ-výrazu, jež odpovídají konceptuálním relacím patřícím do množiny všech konceptuálních relací pro business CG, jsou součástí CG v některé úrovni zahnízdění v business CG.

Označím definiční λ-výraz jako $Výraz_1$. V následujícím popisuji rozšíření výrazu $Výraz_1$ o relace z „dotknutých“ atomů, a dále záměnu atomů za super-relace:

Rozšiřme $Výraz_1$ na $Výraz_2$ :

- připojme k sub-atomům z výrazu $Výraz_1$ všechny konceptuální relace, jež jsou součástí je zahrnujících atomů v příslušné úrovni zahnízdění v business CG
- přidejme k těmto připojeným konceptuálním relacím k nim připojené koncepty z CG v příslušné úrovni zahnízdění v business CG, pokud ve výrazu $Výraz_1$ již nejsou; evidujme rozdělení relací nově vzniklého výrazu do atomů
- přidejme všechna koreferenční propojení z business CG mezi koncepty nového výrazu
- všem konceptům nového výrazu, které jsou koreferenčně propojeny s nějakým konceptem s neprázdným referentem tohoto výrazu, připišme stejný referent (je to nějaké $\lambda_i$ , nebo konkrétní instance konceptů či jejich množina, nebo kvantifikátor množství)

Platí: $Výraz_2$ je λ-výraz, jenž je ekvivalentní s λ- výrazem $Výraz_1$ – důkaz:

1. Přidali jsme podmínky existence nějakých instancí nějakých typů konceptů, jež jsou s instancemi konceptů ve $Výraz_1$ v přidaných relacích z odpovídajících atomů. Sémantika definice atomů však zaručuje, že takové instance existují, takže nejde o zúžení odvozovaného konceptu či konceptuální relace.
2. Všechna koreferenční propojení z $Výraz_1$ i z business CG implicitně znamenají, že referenty příslušných konceptů musí být stejné.

Nahraďme $Výraz_2$ výrazem $Výraz_3$ :

- Nahraďme každý pod-graf odpovídající evidovanému atomu ve $Výraz_2$ pod-grafem super-relace tohoto atomu.
- Jestliže ve $Výraz_2$ byly nějaké koncepty propojeny koreferenčním propojením, propojme ve $Výraz_3$ jim odpovídající koncepty, pokud nesplynuly, také koreferenčním propojením.
- Jestliže jednomu konceptu ve $Výraz_2$ odpovídají při nahrazení dva či více různých konceptů ve $Výraz_3$ , propojme tyto ve $Výraz_3$ koreferenčním propojením.

Platí: Každému konceptu z $Výraz_2$ při tomto nahrazení odpovídají nějaké koncepty stejného typu z $Výraz_3$ , dvěma různým konceptům z $Výraz_2$ odpovídá jeden a tentýž koncept z $Výraz_3$ , pouze když tyto dva koncepty ve $Výraz_2$ byly stejného typu a byly koreferenčně propojeny – důkaz: Koncept z $Výraz_2$ je nahrazen nějakým konceptem, pouze když je stejného typu a je s ním koreferenčně propojen (viz Super-relace atomů na str. 79).

Důsledek: Jestliže dva koncepty z $Výraz_2$ jsou nahrazeny stejným konceptem ve $Výraz_3$ , pak tyto koncepty ve $Výraz_2$ mají stejné referenty.

Vytvořme $Výraz_4$ : Ve $Výraz_3$ vždy připišme všem konceptům, jež odpovídají nějakému konceptu z $Výraz_2$ , stejný referent.

*Výraz $Výraz_4$ je λ-výraz, jenž je ekvivalentní s λ-výrazem $Výraz_1$.*

Každá relace Rel z $Výraz_4$, jež není porovnávací relací (≠, <, >, ≤, ≥, in, …) nebo logickou relací (and, or, not), se modeluje jako predikát Rel' v modelu ORM, v němž odpovídající role hrají objekty modelující typy konceptů z $Výraz_4$ nebo objektové typy modelující jejich kolekce. Navíc v tomto predikátu hraje roli objektový typ „anotace". Ten je „lepidlem" dotazu[83], jenž je transformací původního λ-výrazu definujícího odvození. V dalším poloformálně popíšu odpovídající formuli typovaného kalkulu 1.řádu v predikátech a typech modelu ORM:

Každé množině vzájemně koreferenčně propojených konceptů v λ-výrazu $Výraz_4$ , které mají prázdný referent, a každé množině vzájemně koreferenčně propojených hnízděných konceptů, kromě hnízděných konceptů připojených k logickým relacím, a každé množině vzájemně koreferenčně propojených množinových konceptů přiřadím jednu nezávislou proměnnou, nechť jsou to např. proměnné $x_1,\ldots,x_n$, každá z nich bude odpovídajícího typu. Pro objekt anotace zvolím proměnnou α .

Každá porovnávací relace (≠, <, >, ≤, ≥, ) z $Výraz_4$ se transformuje do porovnávacího predikátu mezi příslušnými proměnnými (porovnání „in" transformuji za chvíli).

Pro každý množinový koncept s neprázdnou množinou designátorů se vytvoří konjunkce literálů tvořených predikátovým symbolem pro predikát „patří do", na místě první proměnné je $x_i$ odpovídající tomuto množinovému konceptu, na druhém místě symbol příslušného designátoru (konstanta nebo některé $\lambda_j$). Pro každý množinový koncept s kvantifikátorem množství se vytvoří literál tvořený predikátovým symbolem

[83] Odvození uvažuji jen v rámci jedné libovolné ale pevné anotace, ačkoli odvození napříč anotacemi by bylo také možné.Pro taková odvozování i dotazování považuji za vhodnější přímo prostředí ORM modelu, v němž je množné použít např. jazyk ConQuer. K tomu je ovšem důležité, aby ORM model byl „uživatelsky čitelný".

pro predikát „má“, v němž na místě první proměnné je $x_i$ odpovídající tomuto množinovému konceptu, na druhém místě je symbol pro příslušný počet. Množinový koncept s neprázdný referentem se transformuje do konjunkce výrazů, které jsou pro něj takto vytvořeny.

Porovnávací relace in se transformuje do disjunkce rovností.

Pro každý hnízděný koncept připojený k logické relaci ve $Výraz_4$ vytvořím výraz, jenž je konjunkcí:

- literálů tvořených predikátovými symboly odpovídajícími relacím zahnízděným v tomto hnízděném konceptu a na odpovídajících místech je vždy buď některá proměnná $x_i$ nebo $\lambda_j$ nebo $\alpha$
- výrazům transformujícím porovnávacím relacím „in“ zahnízděným v tomto hnízděném konceptu
- všech výrazů transformujících množinové koncepty s neprázdnými referenty zahnízděné v tomto konceptu

Nakonec z nich vytvořme výraz $V$ pomocí logických spojek and, or, not, jak to odpovídá logickým relacím and, or, not ve $Výraz_4$.

Definice odvozeného objektového typu T nebo m-árního predikátu P pak je

$T(\lambda_j) \equiv (\exists x_1 | T_1 \ \ldots \ \exists x_n | T_n \ (V))$

nebo

$P(\lambda_1, \ldots, \lambda_m) \equiv (\exists x_1 | T_1 \ \ldots \ \exists x_n | T_n \ (V))$

Formulace odpovídajícího dotazu v jazyce ConQuer je v takovéto naprosté obecnosti obtížná, mohla by být založena na algoritmu „průchodu“ definičním λ-výrazem. – V kapitole Dotazy popsaná tvorba dotazů na dokumenty nad business CG a jejich transformace do jazyka ConQuer nad modelem ORM může být vodítkem, jak v business CG definovat odvozovací pravidla pro koncepty nebo konceptuální relace, která lze snadno transformovat do jazyka ConQuer.

## *Definice porovnávacích operací*

Toto řešení otázek manipulace s určitými, neurčitými a všeobecnými hodnotami je jedno z možných, je založeno na té volbě, že neurčité a všeobecné hodnoty nesou v sobě informaci o svém původu – anotaci, ve které byly vytvořeny. Tím se zvětšuje objem potřebné paměti, ale logika i složitost manipulací nad anotačními záznamy se tím zjednodušuje.

Porovnávat lze pouze typově porovnatelné hodnoty.

Určité hodnoty se mezi sebou porovnávají obvyklým způsobem.

Neurčitá hodnota má význam jedné skutečné hodnoty, jen nevíme které: Porovnání na rovnost je TRUE pouze tehdy, když porovnávané neurčité hodnoty jsou z jedné anotace a je explicitně vyjádřeno, že mají být stejné – mají stejnou identifikaci. Porovnání na nerovnost je negací porovnání na rovnost. Všechna ostatní porovnání neurčitých hodnot mezi sebou, a porovnání určitých a neurčitých hodnot, je UNKNOWN.

Porovnání všeobecných hodnot mezi sebou na rovnost je TRUE, když jsou z jedné anotace a mají stejnou identifikaci, jinak je UNKNOWN. Porovnání určité nebo neurčité hodnoty na rovnost s všeobecnou hodnotou je UNKNOWN. Porovnání na nerovnost je negací porovnání na rovnost.

Porovnání na < , > u typů, u nichž to má smysl, je dáno:

| | | | |
|---|---|---|---|
| $\alpha < \beta$ | $\equiv$ | $\alpha < \beta$ | , když $\alpha$ i $\beta$ jsou určité |
| | | FALSE | , když $(\alpha = \beta)$ je TRUE |
| | | UNKNOWN | jinak[84] |

Porovnání ≤ , ≥ je disjunkce < či > a rovnosti. Porovnání `in(seznam hodnot)` je disjunkcí rovností.

Logické operace s pravdivostními hodnotami TRUE, FALSE, UNKNOWN jsou obvyklé:

| **AND** | TRUE | FALSE | UNKNOWN |
|---|---|---|---|
| TRUE | TRUE | FALSE | UNKNOWN |
| FALSE | FALSE | FALSE | FALSE |
| UNKNOWN | UNKNOWN | FALSE | UNKNOWN |

| **NOT** | TRUE | FALSE | UNKNOWN |
|---|---|---|---|
| | FALSE | TRUE | UNKNOWN |

$\mathcal{T}_1$ **OR** $\mathcal{T}_2$ ≡ NOT( NOT($\mathcal{T}_1$) AND NOT($\mathcal{T}_2$) )

Obvyklé dotazování má sémantiku takovou, že se mají vybrat ty objekty, pro něž podmínka výběru má hodnotu TRUE. Je možné si představit takové dotazování, ve kterém můžeme chtít výběr i těch, pro které podmínka má hodnotu UNKNOWN (popřípadě s nižší preferencí výběru). V kombinaci s použitím funkcionálního atributu anotací, podle kterého se dají vybrat „všeobecné“ anotace (viz str.39), je toto volba, která umožní „přirozenou“ interpretaci dotazů.

[84] Je na pováženou, zda porovnání nemá být TRUE v případě, že jedna porovnávaná hodnota je všeobecná a druhá určitá nebo neurčitá. Zde jsou argumenty proti tomu: V takovém případě by bylo $\alpha = \beta \equiv$ TRUE i $\alpha < \beta \equiv$ TRUE i $\alpha > \beta \equiv$ TRUE , a tak by nebylo možno v úvahách používat $\alpha < \beta \equiv$ not $(\alpha \geq \beta)$. Druhý argument je sémantický: Pokud hledám $\beta$ k danému určitému $\alpha$ tak, že $\alpha = \beta$ (nebo $\alpha < \beta$) , pak nalezení takového $\beta$ implicitně znamená, že mezi hodnotami, jež $\beta$ zastupuje, existuje nějaká rovna $\alpha$ (nebo větší než $\alpha$). Je-li však $\beta$ všeobecné, nic o hodnotách, jež $\beta$ sémanticky zastupuje, nevíme.

## *Transformace dotazů nad business CG do jazyka ConQuer*

Dotaz nad business CG v grafické podobě se skládá z multimnožiny hvězdových pod-grafů konceptuálních relací v různých úrovních zahnízdění, rozdělené do atomů (korektně vůči atomům v příslušných úrovních zahnízdění v business CG), podmínek kladených na některé koncepty v nich, a koreferenčních propojení mezi některými koncepty. Každá skupina hvězdových pod-grafů tvořících atom v dotazu se transformuje do řádku

(*) `└ zahrnuje Rel X(podmínka`$_1$`),Y(podmínka`$_2$`),…`

kde `Rel` je jméno super-relace odpovídajícího atomu v business CG, sekvence `X,Y…` vyjmenovává objektové typy hrající postupně všechny role v predikátu „`..zahrnuje Rel ..,…`“, `podmínka`$_1$, `podmínka`$_2$,… jsou podmínky kladené v dotazu na odpovídající koncepty v případě, pokud nejsou některé z:

- `in(množinový_koncept)`
- `⊃(sekvence_hodnot|proměnných)`
- `∃in(množina|množinový_koncept)`
- podmínka kvantifikace množinového konceptu.

(**)

(Množinová porovnání = a ⊃ , jež považuji za nevhodná, se mohou realizovat jako definovaná porovnání množinových typů. Takovou definici popisuje následující část Porovnání množin.)

Je-li porovnání u některého z konceptů: `in B` , kde `B` je množinový koncept, pak se pod řádek (*) připojí řádek (se znakem `└` pod porovnávaným konceptem):

`└ je prvkem B′`

kde `B′` je objektový typ modelující množiny prvků typu `B` a je oindexován stejně jako příslušný výskyt typu `B′` v řádku, ve kterém koncept B z tohoto porovnání hraje roli v odpovídajícím predikátu.

Je-li porovnání u některého z množinových konceptů: `⊃{B`$_1$`,..,B`$_n$`}` , kde `B`$_1$`,..,B`$_n$ jsou hodnoty nebo proměnné, je-li `A` typ tohoto konceptu, je-li `A′` objektový typ modelující množiny prvků typu `A` , pak se pod řádek (*) připojí řádky (se znaky `└` pod objektem `A′`):

`└ má prvek A B`$_1$
`└ …`
`└ má prvek A B`$_n$

Je-li porovnání u některého z množinových konceptů: `∃in{B`$_1$`,..,B`$_n$`}` , kde `B`$_1$`,..,B`$_n$ jsou hodnoty nebo proměnné, je-li `A` typ tohoto porovnávaného konceptu, `A′` objektový typ modelující množiny prvků typu `A` , pak se pod řádek (*) připojí řádek (se znakem `└` pod objektem `A′`):

└ má prvek A in{$B_1$,..,$B_n$}

Je-li porovnání u některého z konceptů: ∃in B , kde B je množinový koncept, je-li A typ tohoto porovnávaného konceptu, je-li A′ objektový typ modelující množiny prvků typu A , pak se pod řádek (*) připojí řádek (s prvním znakem └ pod objektem A′):

└ má prvek A
└ je prvkem B′

kde B′ je objektový typ modelující množiny prvků typu B a je oindexován stejně jako příslušný výskyt typu B′ řádku odpovídajícím atomu dotazu, ve kterém je porovnávající koncept B.

Je-li porovnání u některého z konceptů počet(porovnání)(hodnota|proměnná), je-li A typ tohoto konceptu, A′ objektový typ modelující množiny prvků typu A , pak se pod řádek (*) připojí řádek (se znakem └ pod objektem A′):

└ má prvků Počet(porovnání)(hodnota|proměnná)

Všechny připojené řádky nebo skupiny řádků popsané výše se pod řádek (*) připojí v pořadí podle klesající vzdálenosti prvního znaku připojení └ od začátku řádku, tím se odstraní možné nedorozumění o tom, co je k čemu připojeno.

Všechny proměnné použité v dotazu nad business CG se transformují do proměnných odpovídajícího typu na příslušném místě porovnání v dotaze v jazyce ConQuer, shoda oindexování odpovídá použití stejných proměnných v dotaze nad business CG.

Je-li v některé podmínce v dotazu nad business CG použita proměnná, která je u nějakého konceptu v dotazu použita jako referent, pak transformace tohoto konceptu v dotazu jazyka ConQuer má stejné oindexování jako tato proměnná.

Každá množina koreferenčně propojených konceptů stejného typu z dotazu nad business CG se transformuje v jednotné oindexování příslušných jmen typů nebo proměnných v dotazu jazyka ConQuer. Každé koreferenční propojení mezi typově příbuznými koncepty rozdílného typu se transformuje do podmínky rovnosti jedné z proměnných, stojící na místě jména typu hrajícího roli v nějakém predikátu, s druhou proměnnou.

Všechny skupiny řádků vzniklé jako (*) s příslušnými připojenými řádky odpovídajícími použitým podmínkám (**) se v libovolném pořadí vloží pod úvodní řádky:

✓Dokument
└ má kontext Anotace

řádky vzniklé jako (*) mají první znak └ pod objektem „Anotace“.

### *Porovnání množin*

Zde jsou definovány porovnání = a ⊃ pro množinové objekty.

Jsou-li oba porovnávané množinové objekty úplně určené, je jejich porovnání obvyklé. Je-li alespoň jeden z porovnávaných objektů neurčený, má porovnání = výsledek UNKNOWN. Porovnání A⊃B, kde B je úplně určený, je TRUE, když množina (vyjmenovaných) prvků objektu A obsahuje množinu prvků B, jinak je UNKNOWN. Porovnání A⊃B, kde B je neurčený, má výsledek UNKNOWN.

# Použitá literatura


[Abe] ABECKER A., BERNARDI A., HINKELMANN K., LIAO M.,SINTEK M.: *Ontologies for Knowledge Retrieval in Organizational Memories*, SEKE'99 - Workshop on Learning Software Organizations, 1999

[Bar] BARTOŠEK M.: *Digitální knihovny*, sborník konference Datakon 2001, Vydavatelstvo STU, Bratislava 2001

[Duž] DUŽÍ M.: *Logical Foundations of Conceptual Modelling*, docentská habilitační práce, Technická Universita Ostrava, 2002

[Hal] HALPIN T.: *Information Modeling and Relational Databases, From Conceptual Analysis to Logical Design*, Morgan Kaufman Publishers, 2001

[Hal1] HALPIN T.: *UML Data Models from an ORM Perspective*, Part 1 – 10, Journal of Conceptual Modeling, www.jcm.net

[Hal2] HALPIN T., BLOESH A.C.: *ConQuer: A Conceptual Query Language*, [orm]

[Hal3] HALPIN T., BLOESH A.C.: *ConQuer: A Conceptual Query Language-II*, [orm]

[Hal4] Halpin T: *Modelling Collections in UML and ORM*, [orm]

[Mit] MITTELMANN A., HANTSCHEL I., ERHART W., HAHN T., WIENERROITHER S.: *Holistic Knowledge Management*, IDIMT 2001

[Pok] POKORNÝ J.: *Databázové systémy a jejich použití v informačních systémech*, Academia Praha, 1992

[Pok1] POKORNÝ J., SNÁŠEL V., HÚSEK D.: *Dokumentografické informační systémy*, Karolinum Praha 1998

[Smi] SMITH B.: *Ontology and Information Systems*, draft k přednášce konané na VŠE Praha v roce 2001

[Sow] SOWA J.F.: *Knowledge Representation, logical, philosophical, and computational foundations*, Brooks/Cole 2001

[Stro] STROSSA P.: *Vybrané kapitoly z počítačového zpracování přirozeného jazyka*, Slezská univerzita v Opavě, 1999

[Šim] ŠIMEK P.: *Ontologie a WWW*, diplomová práce, VŠE Praha 1999

[Vod] VODÁČEK L., EHLEMAN J.: *Strategic goals for the use of knowledge management*, sborník konference IDIMT, Johannes Kepler Universität, Linz 2001

[Žab] ŽABIČKA P.: *Dublin Core – metadata pro popis elektronických dokumentů*, sborník konference Datasem 2000, Masarykova universita v Brně, 2000

# Reference


[Dav] DAVENPORT T.H., PRUSAK L.: *Working Knowledge — How Organizations Manage What They Know*, Harvard Business School Press, Boston 1998

[Wil] WILSON D.A.: *Managing Knowledge*, Butherworth Heineman, Oxford 2000

[DC] *Dublin Core Metadata Initiative homepage*, http://purl.oclc.org/dc/index.htm

[CYC] *CYCORP homepage*, http://www.cyc.com

[FARQ] FARQUAR A., FIKES R., PRATT W., RICE J.: *Collaborative Ontology Construction for Information Integration*, Knowledge Systems Laboratory Departmant of Computer Science, KSL-95-63, Stanford University, 1995, http://www-ksl.stanford.edu/KSL_Abstracts/KSL-95-63.html

[fas] www.ai.sri.com/~okbc/okbc-faq/Knowledge_models/what_is_facet.htm

[Fil] FILLMORE CH. J.: *The case for case*, In: E. Bach & T.R. Harms eds., Universals in Linguistic Theory, Hlt,Rinehart and Winston, New York, 1973, s.1-88

[Han] HANSEN H. R., MŰHLBACHER R. & NEUMANN G.: *Begriffsbasierte Integration von Systemanalysemethoden*, Physica-Verlag, Heidelberg, 1992

[Mal] MALHOTRA Y.: *Knowledge Management & New Organization Forms: A Framework for Business Model Innovation*, In: Knowledge Management and Virtual Organizations, Idea Group Publishing, Hershey, PA, April 2000, p.2-19

[orm] www.orm.net

[Pal] PALOVSKÁ H.: *Knowledge separated from Heads*, sborník IDIMT, 2001

[Pok2] POKORNÝ J.: *XML functionally*, In: Proc. of IDEAS2000 Conf., IEEE, Yokohama, Japan Sep. 18-20, 2000