Vysoké učení technické v Brně
Fakulta informačních technologií

Seminární práce do předmětu: **VPD**

Téma: Dolování v objektově orientovaných databázích založené na generalizaci

Zpracoval: Lukáš Stryka **2. 9. 2006**

Obsah

1 Úvod 3
2 Operátory pro generalizaci atributů, metod a komplexních komponent objektů 4
2.1 Generalizace identifikátoru objektů a hierarchie tříd/podtříd 4
2.2 Generalizace hodnot jednoduchým atributů 5
2.3 Generalizace strukturovaných dat 6
2.4 Generalizace na multimediálních datech 7
2.5 Generalizace na zděděných nebo odvozených vlastnostech 8
2.6 Generalizace nad hierarchií skládání tříd 8
2.7 Generalizace tříd 9
2.8 Objektový krychlový model 10
2.9 Algoritmus generalizace tříd pro konstrukci objektové krychle 11
2.10 Generalizace založená na dimenzích – příklad 13
3 Odvozování generalizovaných pravidel 17
3.1 Získání charakteristických pravidel 17
3.2 Získávání diskriminačních pravidel 18
3.3 Získávání asociačních pravidel vysoké úrovně 18
4 Shrnutí a závěr 22
5 Literatura 23

1 Úvod

S rostoucími objemy dat v databázích je vyvíjet efektivní a výkonné nástroje pro získávání užitečných znalostí z těchto dat (knowledge Discovery in Databases - KDD, nebo též data mining). V posledních letech bylo vyvinuto mnoho technik pro získávání znalostí z transakčních či relačních databází. V dnešní době však však dochází k rozvoji a rozšiřování objektově-orientovaných databází (OO databáze), které slouží zejména pro uchovávání složitě strukturovaných dat. Je tedy třeba vyvinout nové metody a techniky pro získávání znalostí z těchto OO databází.

OO modely dat v sobě sdružují datové struktury a sémantiku pro tvorbu složitých databází, např. složité datové objekty, hierarchii tříd/podtíd, dědičnost, metody, aktivní data, atd. Tím se stává mocným nástrojem pro vývoj databázových systémů, ale přináší to s sebou i složitost implementace nástrojů kro získávání znalostí.

V této práci jsou popsány základní principy při získávání užitečných znalostí z OO databází. Dolovací proces začíná shromážděním dat, která jsou relevantní vůči požadované dolovací úloze, pomocí některého OO dotazovacího nástroje dané OO databáze. Dále je prováděna generalizace komplexních datových objektů, pro která existují, nebo jsou vyvíjeny generalizační operátory. Dále je aplikována generalizace založená na krychli, která vede ke generalizaci a zhuštění relevanntích dat do kompaktní objektové krychle. Nakonec jsou z této krychle získávány požadované typy znalostí.

Tato práce je organizována následujícím způsobem:

- v kapitole 2 jsou popsány základní generalizační operátory pro generalizaci komplexních datových objektů,
- kapitola 3 popisuje významné metody a techniky pro generalizaci založenou na krychli a
- v kapitole 4 jsou nastíněny techniky pro získávání různých druhů znalostí.

2 Operátory pro generalizaci atributů, metod a komplexních komponent objektů

OO databáze organizují velké množiny složitých datových objektů, které jsou organizovány do hierarchií tříd/podtříd. Každý objekt v dané třídě je asociován s:

- identifikátorem objektu,
- množinou atributů, které mohou obsahovat sofistikované datové struktury, hierarchii skládání tříd, multimediální data, atd.,
- množinu metod, které specifikují výpočetní rutiny nebo pravidla pracující nad danými objekty.

Pro vývoj nástrojů pro získávání znalostí je třeba vyvinout relativně malý počet generalizačních operátorů, nad kterými mohou být postaveny generalizační operace. Abstraktně lze vyjádřit generalizaci komponenty p objektu O_i jako $Gen(O_i.p)$, kde Gen je abstraktní generalizační objektový operátor, který může být transformován na konkrétní operaci pro danou komponentu objektu.

Objekt v OO databázi je je popsán množinou atributů a množinou metod. Hodnota atributu může být znak, řetězcem číslo, struktura, množina/seznam dat nebo ve formě nestrukturovaných dat jako je text, mapa, obrázek, hlas, atd.

Přepokládejme, že vlastnost objektu O_i je p_k. Lze tuto vlastnost generalizovat na minimálně generalizovaný koncept p_{k-1}, který může být dále generalizován na p_{k-2}, atd. Sekvence generalizací vlastnosti objektu může být vyjádřena jako:

$$\forall j(1 \leq j \leq k): Gen(p_j) = p_{j-1}.$$

V následujících podkapitolách bude nastíněno, jak se provádí generalizace různých komponent složitého objektu v OO databázích.

2.1 Generalizace identifikátoru objektů a hierarchie tříd/podtříd

Jedna ze základních komponent objektově orientované databáze je identifikátor objektu, který jednoznačně identifikuje objekt. Zůstává nezměněn při strukturální reorganizaci dat. Na první pohled se zdá, že je nemožné generalizovat tento identifikátor, nicméně objekty v objektově orientované databázi jsou organizovány do tříd, které jsou začleněny do hierarchie tříd/podtříd, tedy generalizace objektů může odpovídat této hierarchii. Tedy identifikátor objektu může být generalizován jménu odpovídající nejnižší podtřídy. Toto jméno může být dále generalizováno na jméno třídy/podtřídy vyšší úrovně dané hierarchie. Stejným způsobem mohou být generalizovány třídy/podtřídy do nadřazených tříd/podtříd.

Předpokládejme, že identifikátor objektu O_i je O_i.oid jméno nejnižší podtřídy, které přísluší O_i je C_{ij} a jméno supertřídy třídy C_{ik} je C_{ik-1} pro všechna k ($1 \leq k \leq j$). Generalizace identifikátoru objektu O_i.Oid objektu O_i a odpovídající třídy může být reprezentována jako:

$$Gen(O_i.oid) = C_{i,j}.$$
$$\forall k(1 \leq k \leq j)\ Gen(C_i,k) = C_{i,k-1}.$$

2.2 Generalizace hodnot jednoduchým atributů

Jednoduché atributy mohou být numerická nebo nenumerická data, která jsou nejčastěji zastoupeny *položkami* v databázi.

Generalizace nenumerických hodnot může spočívat v postupné konceptuální hierarchii specifikované doménovými experty nebo uživateli. Konceptuální hierarchie reprezentují nezbytný znalostní základ, který řídí celý generalizační proces. Různé úrovně konceptu jsou organizovány do taxonomie nebo hierarchie konceptů, které mohou být uspořádány od obecných ke specifickým. Nejobecnější koncept (odpovídající úrovni 0), je *null* popis (popsané rezervovaným slovem any) a nespecifičtější koncept odpovídá specifické hodnotě atributu v databázi.

Při použití konceptuální hierarchie nebo taxonomie mohou být nalezená pravidla reprezentována v termínech zobecněných konceptů a mohou být vyjádřena jednoduchou a explicitní formou, která je srozumitelná pro většinu uživatelů.

Mnoho konceptuálních hierarchií je uloženo implicitně v databázích, např. geo_location(street, city, province, country), a pomocí specifikace se explicitně mohou stát určitým zobecňujícím datovým pravidlem, např. generalizace pomocí odstranění atributů *street* a *city* v geo-location() apod. Některé hierarchie mohou být vytvořeny znalostními inženýry nebo doménovými experty, kteří vytvoří pro rozsáhlé databáze registr hierarchií konceptů, avšak pouze omezený počet diskrétních atributů nebo rozsahu numerických hodnot pro dané atributy, které obecně nejsou příliš rozsáhlé. Některé hierarchie mohou být odvozeny ze znalostí vložených mimo databázi nebo z vypočtených aplikací určitých pravidel nebo algoritmů, jako je např. British Columbia $\Rightarrow$ Western Canada z geografické mapy uložené v prostorové databázi. Kromě toho může být hierarchie definována i pro jeden atribut nebo množinu příbuzných atributů ve tvaru vyváženého stromu, nevyváženého stromu, mřížky nebo přímého acyklického grafu.

Určitá rozšíření konceptuální hierarchie mohou být generovány automaticky, a to pomocí uživatelova učení se a požadavků založených na specifikaci ze strany uživatelů, sémantice dat nebo distribučních charakteristikách dat. Někdy se daná konceptuální hierarchie ukáže jako ne zcela vhodná pro konkrétní učící úlohu, a je tedy třeba ji dynamicky upravit na základě distribuční charakteristiky pro požadované výsledky učení. Např. pokud

jsou požadavky na učení analyzovat místo narození studenta pod úroveň jedna (top level) konceptu, může být {B.C., other_provinces_in_Canada, foreign}. Oproti tomu, pokud se jedná o místo narození studenta univerzity, příslušná nejvyšší úroveň konceptu může být tvaru {North_America, Europe, Asia, other_countries}. Obě konceptuální hierarchie mohou být získány dynamickým přizpůsobením dané konceptuální hierarchie založené na analýze statistické distribuce relevantní datové množiny.

Různé konceptuální hierarchie mohou být konstruovány pod stejným atributem podle rozdílných úhlů pohledů nebo preferencí. Např. místo narození může být organizováno podle administrativních oblastí, geografické polohy, velikosti měst atd. Všeobecně doporučovaná hierarchie je asociována s atributem, který je standardní. Ostatní hierarchie mohou být explicitně voleny uživatelem během procesu učení. Někdy je pro zlepšení výsledků použito paralelně více hierarchií a nakonec je zvolena ta, která dává nejlepší výsledky.

Generalizace na numerických atributech může být provedena podobně, ale pro automatizaci procesu je voleno rozdělení dle distribučních charakteristik. Předdefinovaná konceptuální hierarchie nemusí být v mnoha případech vyžadována. Např. věk studentů na univerzitě může být rozčleněn do několika skupin, jako např. {pod 23, 23-26, 26-30 a 30+}, což odpovídá uniformnímu rozdělení dat nebo použití některé ze statistických analytických nástrojů. Vhodná jména mohou být uživateli nebo experty přiřazena zobecnělým numerickým rozsahům, jako např. {velmi mladý, mladý,…} pro zvýšení sémantického významu.

2.3 Generalizace strukturovaných dat

Komplexně strukturovaná data, jako např. množina či seznam jsou v OO databázích obecná. Mohou být generalizována několika způsoby pro získání zajímavých vztahů z takovýchto datových množin. Množinový atribut může být homogenního nebo heterogenního typu. Typicky množinová data mohou být generalizována dvěma způsoby:

- generalizace každé hodnoty v množině do sobě odpovídající vyšší úrovně konceptu nebo derivací obecného chování množiny, jako je např. počet elementů v množině typu nebo rozsahu hodnot v množině, váženého průměru pro numerická data atd.,
- generalizace může být vylepšena aplikací různých generalizačních operátorů pro získání alternativních generalizačních cest. V tomto případě výsledek generalizace je heterogenní množina.

Např. koníček je množinový atribut, který se skládá z množiny hodnot, jako je např. {tenis, hokej, šachy, housle atd.}, které mohou být generalizovány do množiny konceptů vyšší úrovně, jako např. {sporty, hudba, videohry} nebo do 5 (počet koníčků v množině) nebo obojí. Počet může být asociován s generalizovanou hodnotou pro určení, kolik elementů je zobecněno v příslušné generalizované hodnotě, jako např. {sporty (3), hudba (1), videohry (1)}, kde

sporty (3) označují 3 druhy sportu. Množinový atribut může být generalizován do množinového nebo jednoduchého atributu, stejně tak jako jednoduchý atribut může být generalizován do množinového atributu, pokud hierarchie mřížka nebo generalizace prochází různými cestami. Další generalizace na generalizovaných množinových atributech může sledovat cestu každé hodnoty v množině. Obdobně lze generalizovat atributy tvořené seznamem nebo sekvencí hodnot.

Obecně, atributy obsahující strukturované hodnoty mohou obsahovat množiny, n-tice, seznamy, stromy, záznamy a jejich kombinace. Dále může být jedna struktura obsažena v jiné struktuře na jiné úrovni.

Obecně strukturovaný atribut může být generalizován několika způsoby. Hlavními jsou např.:

- generalizace každého atributu ve struktuře,
- zplošťováním struktury, generalizování na zploštělé struktuře,
- odstranění struktur nízké úrovně nebo shrnutí struktur nízké úrovně pomocí konceptů vyšší úrovně nebo agregace a
- navrácení typu struktury.

2.4 Generalizace na multimediálních datech

Multimediální databáze mohou obsahovat souhrn textů, grafiky, obrázků, map, hlasu, hudby a jiné formy audio či video informací. Taková multimediální data jsou typicky uložena v sekvencích bytů s proměnnou délkou. Na segmenty dat jsou provázány dohromady pro snadnější odkazování. Generalizace na multimediálních datech může být provedena pomocí rozpoznání a extrakce esenciálních rysů nebo obecných vztahů takovýchto dat.

Existuje mnoho způsobů pro extrakci esenciálních rysů nebo obecných vztahů ze segmentů multimediálních dat. Pro obrázky se jedná o velikost a barvu obsažených objektů nebo hlavní oblasti v obrázcích získané agregací nebo aproximací. Pro segmenty hudby to může být melodie založená na aproximaci vztahů, které se opakovaně vyskytují v segmentu nebo styl, který může být založen na tonech, tempu nebo rychlosti, hlavních hudebních nástrojů atd. Pro články to může být abstrakt nebo obecná organizace jako např. obsah, předmět, index výrazů často se opakující v článcích atd. Souhrnně řečeno je generalizace multimediálních dat pro získání zajímavých znalostí implicitně uložených v datech velkou výzvou budoucnosti při vývoji nástrojů pro získávání znalostí.

2.5 Generalizace na zděděných nebo odvozených vlastnostech

Objektově orientované databáze jsou organizovány do hierarchií tříd/podtříd. Některé atributy nebo metody tříd objektů nejsou explicitně specifikovány v třídě jako takové, ale jsou zděděny ze třídy vyšší úrovně. Některé OO databázové systémy umožňují zdědění vlastností z více než jedné supertřídy (vícenásobná dědičnost). Zděděné vlastnosti objektů mohou být odvozeny v OO databázích zpracováním dotazů. Z pohledu získávání znalostí není nezbytností rozlišovat, která data jsou uložena uvnitř třídy anebo zděděna ze supertřídy. Pro proces získávání znalostí jsou relevantní data získávána zpracováním dotazů a je tedy jedno, zda jsou data uložena ve třídě objektů nebo jsou zděděna.

Další důležitou OO databází jsou metody. Mnoho dat může být získáno z objektů pomocí aplikace těchto metod. Metoda je obvykle definována výpočetní procedurou nebo funkcí nebo množinou deduktivních pravidel. Je velmi obtížné generalizovat metody jako takové, proto je třeba , aby aplikační programátoři vytvořili nové metody, které jsou požadovanou generalizací daných metod. Obecně, generalizace na datech získaná aplikací metod by měla být prováděna ve dvou krocích:

- získání relevantní množiny dat aplikací metod a
- provedení generalizace ošetřením/úpravou získaných dat.

2.6 Generalizace nad hierarchií skládání tříd

Atribut objektu může být složen z nebo popsán jiným objektem a některé z těchto atributů mohou být dále složeny z dalších objektů, tedy jde o hierarchii skládání tříd. Generalizace nad hierarchií skládání tříd může být provedena jako generalizace na množině shluku strukturovaných dat (pokud je shluk rekurzivní, může být množina nekonečnou).

V principu je reference na složený objekt možné vyjádřit jako dlouhou sekvenci referencí odpovídajících hierarchií skládání tříd. Ve většině případů jsou dlouhé sekvence preferencí převedeny do slabších sémantických spojení mezi původním objektem a odkazujícím složeným objektem. Např. jeden atribut *vehicles owned* objektu třídy *student* může odkazovat na jiný objekt třídy *car*, který může obsahovat atribut *auto_dealer*, který může odkazovat na manager s atributem *children*. Obvykle je nepravděpodobné, že se najde nějaká zajímavá obecná vazba mezi studentem a dítětem ředitele dealera jeho auta. Proto je generalizace na třídě objektů prováděna především na nejbližších hodnotách popisných atributů nebo metodách s omezením referencí nepřímo odkazované části hierarchie skládání tříd.

Pro objevování zajímavých znalostí by měla být generalizace utvářena pouze na objektech hierarchií složených tříd, které jsou úzce sémanticky provázány s právě testovanou třídou, ale ne s těmi, které jsou vzdáleny nebo mají slabé sémantické provázání.

3 Generalizace tříd

S dostupností operátorů pro generalizaci objektů může být provedena generalizace relevantních objektů založena na třídách. Vzhledem k tomu, že množina objektů ve třídě může sdílet mnoho atributů a metod a generalizace každého atributu a metody může být provedena pomocí aplikace sekvence generalizačních operátorů, které jsou popsány v předchozí kapitole, hlavní důraz je kladen na to, jak skloubit generalizační proces nad různými atributy metodami při získávání zajímavých výsledků.

Generalizace třídy objektů může být brána jako sekvence množinově orientovaných generalizačních procesů, které transformují třídu do relativně více generalizované třídy. Třída předkládaná do generalizačního procesu se nazývá pracovní třída W (working class). Počáteční třída se nazývá počáteční pracovní třída W_0 (initial working class). Třída získaná aplikací generalizačního operátoru se nazývá výsledná třída R (result class).

Atributově orientovaná indukční metoda, vyvinutá při získávání znalostí v relačních databázích, může být upravena pro objektově orientované databáze. Např. věk může být získán aplikací metody compute_age založené na aktuálním datu a hodnotě atributu birth_date objektu třídy person. Pro generalizaci založenou na třídě je vybrána a aplikována množina generalizačních operátorů na atribut v pracovní třídě, s ohledem na vztahy mezi různými atributy předtím, než je každý atribut generalizován na vyšší úroveň.

V prvních fázích generalizace se netestují vztahy mezi různými atributy, neboť by to mohlo vést k velkému množství nežádoucích generalizací na nízké úrovni. To by nevedlo ke generování elegantních pravidel vyjádřených ve stručné formě. Proto je indukce orientovaná na atributy, která zahrnuje i vztahy mezi atributy, použita až tehdy, kdy jsou tyto atributy generalizovány na relativně vysoké úrovni konceptu. To vede ke snížení výpočetní náročnosti generalizačního procesu databáze.

Formálně, generalizační operátor Gen může být aplikován na atribut A_i každého objektu pracovní třídy W_k , což vede na novou generalizovanou třídu R_k. Tudíž třídní generalizátor ClassGen je de facto aplikace objektového generalizačního operátoru Gen na atribut A_i každého objektu v pracovní třídě. Tedy:

$$\mathbf{R_k = ClassGen(W_k,a_i) = \{o' : (o \in W_k) \wedge (o'.a_i = Gen(o.a_i)) \wedge (\forall (j \neq i) o'.a_i = o.a_j)\}}$$

Např. pro pracovní třídu „W_0 = Person“ v univerzitní databázi studentů ClassGen (W_0,address) získává výslednou třídu R_0 s jednostupňovou generalizací na atributu „address“ (např. z čísla ulice k bloku ulice), přičemž ostatní atributy zůstávají neměnné.

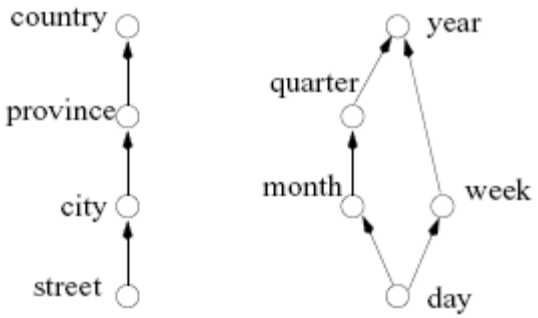
Na proces objevování znalostí může být nahlíženo jako na aplikaci sekvence operátorů generalizace databáze ClassGen na odlišné atributy, dokud výsledná třída neobsahuje pouze malý počet generalizovaných objektů, které mohou být následně uskupeny do stručného generalizovaného pravidla v termínech vyšší úrovně.

3.1 Objektový krychlový model

Populární konceptuální model, který ovlivňuje návrh datových skladů a nástroje pro OLAP (OnLine Analytical Processing) je multidimenzionální datový model. Multidimenzionální databáze, často nazývané datová krychle, je databáze skládající se z velkého množství faktů nebo multidimenzionálních bodů a relativně malého množství dimenzí, s ohledem na to, jaká data jsou analyzována. Např. datový sklad pro tržby může ukládat *objem tržeb* jako fakt v tabulce faktů, může použít *prodejna*, *produkt* a *datum* jako dimenze a ukládat jejich popis v tabulce asociovaných dimenzí.

Tabulky dimenzí ukládají textový popis dimenzí obchodů, které jsou uloženy. Každá dimenze má množinu atributů. Např. dimenze poloha prodejny může mít atributy č.p., ulice, město, země atd. Atributy dimenze mohou být uspořádány částečně dle potenciální vzájemné hierarchie. Např. ulice-město-země. Stejná situace může nastat u data, jako např. datum-měsíc-kvartál-rok nebo den-týden-rok.

Hierarchie a struktura dimenzí atributů je ukázána na následujícím obrázku.



a) hierarchie „region“ b) mřížka „čas“

Obr. 1: Hierarchická a mřížková struktura vlastností v dimenzích skladu

Fakta jsou číselná měřítka uspořádaná v odpovídajících dimenzích a jsou poskytnuta v datových skladech pro analýzy dat, jako např. objem tržeb, rozpočet atd. Tabulky faktů ukládají klíče nebo vícenásobné dimenze a číselná měřítka obchodu.

Objektový krychlový model je dále vyvíjen z krychlového modelu pro datové sklady do objektově orientovaných databází.

Definice: Objektová krychle je multidimenzionální databáze konstruovaná na vrcholu generalizovaných tříd v OO databázích. *Generalizovaná třída* se skládá z množiny *generalizovaných atributů*, které jsou použity jako dimenze objektové krychle a z jednoho nebo více atributů, obsahujících agregované hodnoty nebo jiná rozšíření, která slouží jako měřítka krychle.

Tato definice ukazuje, že multidimenzionální databáze může být konstruována, pokud výslednou třídu objektů z procesu generalizace OO databáze lze převést do množiny měřítek a dimenzí v podobném duchu, jako je tomu u relačních databází. Měřítka multidimenzionálních databází mohou obsahovat číselné zobecněné hodnoty, stejně jako jiná rozšíření, která mohou být kolekcí identifikátorů objektů nebo kolekcí jiných rysů generalizovaných objektů.

Např. třída prostorových objektů může obsahovat odlišné kolekce útvarů prostorových oblastí. Rys *útvar* může být příliš odlišný od tvaru sdílené dimenze, která obvykle obsahuje malý počet různých hodnot. Útvar může být dále generalizován na menší počet různých, ale generalizovaných útvarů, jako je např. polygon, elipsa, kruh, kolekce polygonů atd. nebo může být z konstrukce objektové krychle vyřazena. Nebo může být začleněna jako kolekce ukazatelů na objekty nebo na různé rysy, které jsou např. kolekcí podobných generalizovaných objektů, jež sdílejí stejné nebo jiné generalizované rysy.

3.2 Algoritmus generalizace tříd pro konstrukci objektové krychle

Konstrukce objektové krychle a derivace hlavní třídy (cuboid) pro objektovou krychli může být implementována aplikací sekvence databázových generalizačních operátorů, každý operátor na odpovídající pracovní třídu s výstupem do zobecnělé výsledné třídy, která se stává pracovní třídou v dalším kole databázové generalizace. Takováto iterativní generalizace sekvencí výsledných tříd může vyžadovat mnoho průchodů rozsáhlými databázemi.

Efektivní generalizační algoritmus, který je zde popsán, prochází množinu relevantních objektů nejvíce dvakrát:

- První průchod získává statistické informace o rozložení pro hodnoty atributů a určuje množinu generalizovaných hodnot, na které mohou být atributy generalizovány. Informace bude použita pro přípravu generalizace. Pokud je množina dat rozsáhlá, může být pro zlepšení generalizace použito shlukování.
- Po určení úrovně konceptu hlavní třídy je proveden druhý průchod, který generalizuje všechny vlastnosti každého objektu v počáteční pracovní třídě do konceptů na úrovni hlavní třídy.

Z pohledu objektové krychle je hlavní třída odvozena ze základního cuboidu nebo z některého obecného cuboidu určeného přednastaveným prahem, který určuje maximální počet různých hodnot atributů v generalizaci třídy.

Základní cuboid lze brát jako zvláštní případ hlavní třídy, kde práh každého atributu dimenze je maximum pro to, aby bylo formování dimenze proveditelné. Pokud je práh atributu určen uživatelem, obvykle generalizace nevede na základní cuboid. Pokud není práh definován pro žádný atribut, vede použití algoritmu k získání základního cuboidu.

Algoritmus (Dimensionálně založená generalizace v objektově orientovaných databázích) Generalizace v OO databázích dimenzionálně založenou indukcí založené na generalizaci uživatelova požadavku.

Vstup. 1. DBQuery (psáno v OO jazyka dolování znalostí OML) související s OO databází a úkolu generalizace,

2. Gen(a_i), množina hierarchií konceptu nebo generačních operátorů na dimenzi a_i ´s,

3. T_i , množina prahů dimenzí pro dimenzi a_i ´s.

Výstup. Hlavní (generalizovaná) třída, např. hlavní cuboid objektové krychle založený na generalizačních požadavcích.

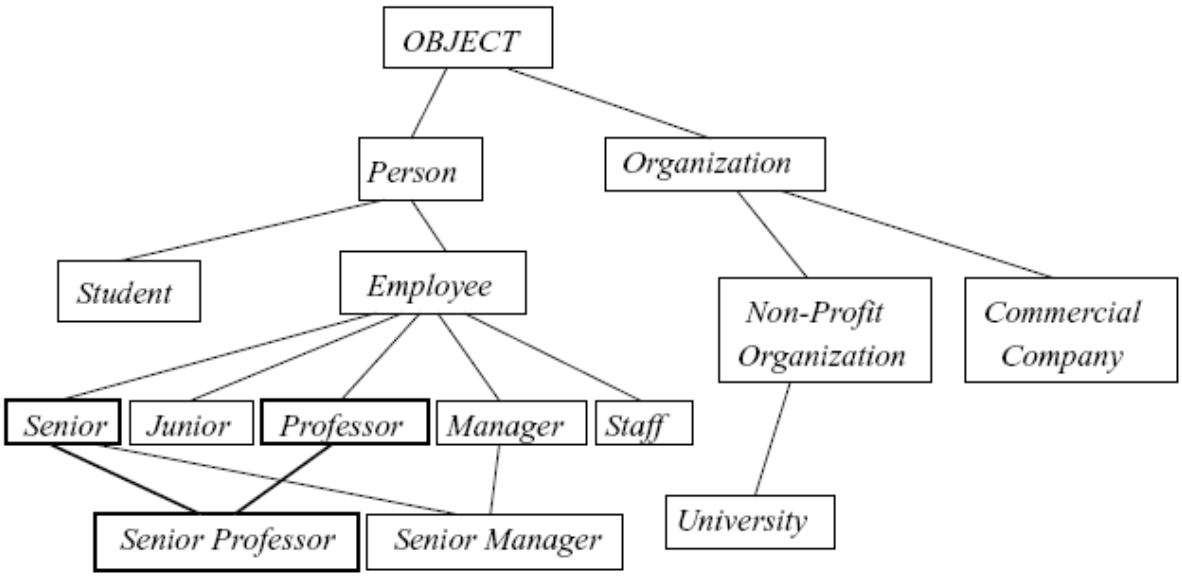
Postup. Indukce založená na dimenzích tvořená následujícími třemi kroky:

1. *initClass*: Spouští databázový dotaz pro určení počáteční pracovní třídy W_o.
2. *BaseGen*: generalizuje požáteční pracovní třídu W_o, minimálně do formy minimálního generalizovaného základního cuboidu $cube_o$. Spočtení počtu různých hodnot pro každý atribut a_i v počáteční pracovní třídě W_o. Pro numerické hodnoty jsou generovány minimální intervaly. Na základě počtu různých hodnot/intervalů v každé dimenzi je určena minimální generalizovaná úroveň pro každou dimenzi základního cuboidu.
3. *PreGen*: Příprava pro generování hlavní krychle. Výpočet požadované úrovně L_i pro každou dimenzi a_i podle daného prahu dimenze t_i. Dimenze ai by měla být odstraněna, pokud je v ní velká množina různých hodnot, ale neexistuje generalizační operátor. Určení mapovacích dvojic (v, v´), kde *v* je hodnota dimenze a_i a v´ je odpovídající generalizovaná hodnota na úrovni L_i.
4. *PrimeGen*: Určuje hlavní třídu $cube_p$. Generalizace každého objektu o počátečním základním cuboidu $Cube_0$ do o´ nahrazením každé *v* v *o* na *v´* pomocí mapujících dvojic (v,v´) získaných pomocí PreGen. Vkládání

takových o´do hlavní třídy, ve které jsou objekty se stejnými hodnotami atributů uloženy ve stejném slotu s jejich počtem a potencionálně s dalšími měřítky.

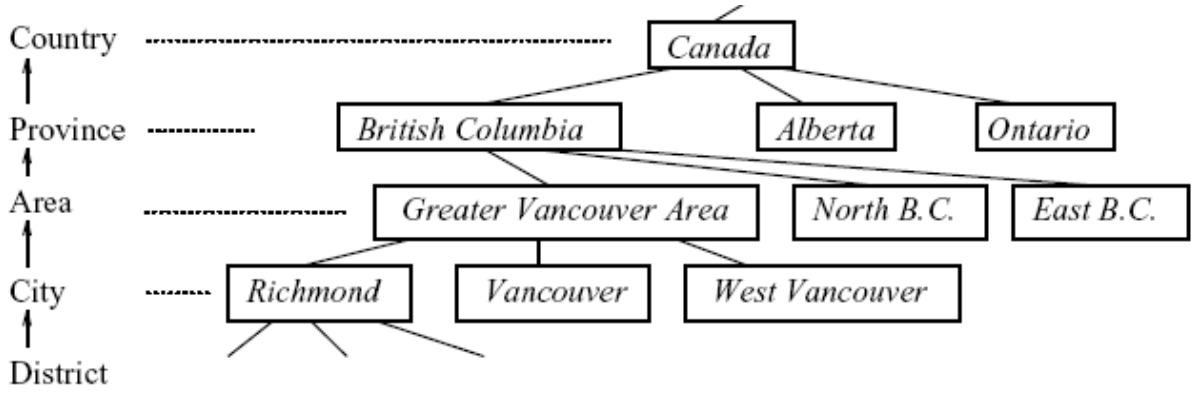
3.3 Generalizace založená na dimenzích – příklad

Máme OO databázi skládající se z množiny tříd: person, organization atd. a jejich asociovaných podtříd. Ukázka hierarchie tříd a podtříd je uvedena na následujícím obrázku, kde podtřída *senior_professor* je podtřídou tříd *senior_employee* a *professor* (vícenásobná dědičnost).



Obr. 2: Hierarchie tříd a podtříd

Ukázka konceptuální hierarchie pro atribut *address* je ukázána na následujícím obrázku:



Obr. 3: Konceptuální hierarchie pro atribut address

Generalizační úloha generalizuje množinu dat ve třídě *person* ve vztahu k house_size, residential_area a salary pro lidi s Phd. titulem., kterým je více než čtyřicet let a řídí japonské auto. Úloha může být reprezentována jako následující dotaz v syntaxi OML.

```
generalize Person P into Generalized_P
where P.Age >= 40 and P.Car.Maker = 'Japan'
           and 'Ph.D' in P.Education and P.Workplace.Name = 'U.B.C.'
in relevance to P.Name, P.Home.House_size, P.Salary, P.Home.Address
```

Krok 1: Vezmi množinu objektů, které jsou vztaženy ke generalizační úloze v pracovní třídě W_0. Jeden z daných objektů může vypadat např. takto:

ObjectID:	02EREV	; String value is given by system
Name:	Alex Flemming	; String value
Home:	⟨House_ObjectID⟩	; House_Object has "House_size", "Address", "Price" and so on.
Education:	(···, (Ph.D. in Computer Science, UCLA, 1987))	; Set-data values
Workplace:	⟨Workplace_ObjectID⟩	; Workplace_Object has "Name", "Address" and so on.
Salary:	$73,854	; Real value
Birthdate:	March 23, 1950	; String value
Age:	Method(Birthdate, Today)	; Method
Face:	⟨Bitmap data⟩	; Multimedia data
⋮		

Age je metoda, která vypočítá věk osoby z atributu *birthday* a aktuálního data.

Krok 2: V algoritmu BaseGen je generován minimální generalizovaný cuboid $cube_0$, jehož dimenze jsou generovány na základě relevantní množiny atributů a numerická data jsou začleněna do minimálních intervalů.

Krok 3: V algoritmu PreGen je každá dimenze základního cuboidu $cube_0$ prohledána právě jednou. Distribuce hodnot dimenzí jsou určeny generalizováním mapujících párů. Atribut *P.Name* je odstraněn, pokud již nejsou nadřazené koncepty, kromě *any*. Generalizace odvozuje následující koncepty vyšší úrovně spojené s jejich odpovídajícími koncepty nižší úrovně relevantních objektů pro danou úlohu: {Richmond, Burnaby, Vancouver} pro *Address* (na konceptuální úrovni city), {senior professor, senior manager} pro objektový identifikátor (podle hierarchie tříd/podtříd), {big, medium, small} pro *House_size* a {L(low),M(medium), H(high)} pro *Salary*. Navíc celková hodnota salaries na jakékoli generalizačních úrovních je také uložena v buňkách krychle. Můžeme tak snadněji spočítat hodnotu průměrného platu z *counts* a *total_salary* uložených v objektové krychli.

Krok 4: V algoritmu PrimeGen jsou objekty generalizovány pomocí mapování párů určených v kroku 3 a každý generalizovaný objekt je vložen do hlavní třídy. Jeden z generalizovaných objektů je zobrazen níže:

Generalized_Object[1]

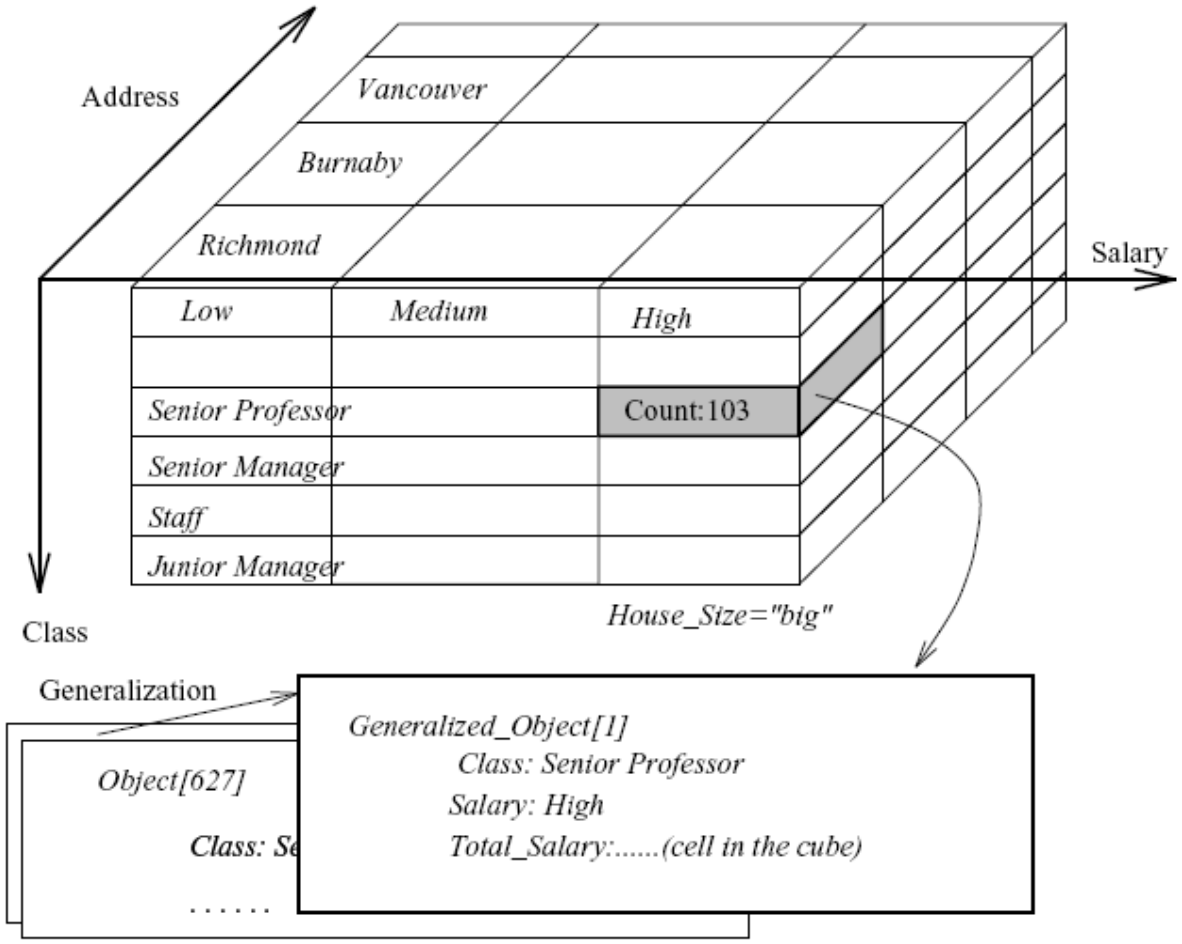
Class:	senior professor
ObjectID:	60@WUH
House_size:	big
Address:	Richmond
Salary:	high
Count:	103

ObjectID je nový identifikátor objektů přidělený systémem.

Pokud generalizovaný objekt je reprezentován jako n-tice, hlavní třída je reprezentována jako relace zobrazená v následující tabulce, kde atribut ObjectID je vynechán, jelikož nehraje žádnou roli v sémantické reprezentaci generalizované třídy.

Class	*House_size*	*Address*	*Salary*	*count*
senior_professor	*big*	*Richmond*	*medium*	64
senior_professor	*big*	*Richmond*	*high*	103
.....	...	.....	...	...
senior_manager	*small*	*Vancouver*	*low*	49

Tab.1: Hlavní třída reprezentovaná jako relace (4-dimenzionální krychle)



Obr. 4: Multidimenzionální objektová krychle s generalizovanými objekty

Hlavní třída může být též reprezentována ve formě vícedimenzionálního pole, které se nazývá crosstab (generalized Gross-tabulation), kde první sloupce ukazují různé podtřídy: senior professor, senior manager, druhé sloupce ukazují úroveň platů: low, medium a high, řádky první úrovně ukazují velikost domů: big, medium a small a řádky druhé úrovně ukazují adresu: Richmond, Burnaby a Vancouver.

		senior_professor				*senior_manager*				*Total*
		L	*M*	*H*	*Total*	*L*	*M*	*H*	*Total*	
big	*Richmond*	0	64	103	167	0	16	20	36	203
	Burnaby	2	27	46	75	1	8	9	18	93
	Vancouver	0	11	39	50	0	4	8	12	62
	Total	2	102	188	292	0	28	37	66	358
medium	*Richmond*	1	42	20	63	0	12	1	13	76
	Burnaby	0	21	62	83	0	3	18	21	104
	Vancouver	0	12	32	44	1	2	5	8	52
	Total	1	75	114	190	1	17	24	42	232
small	*Richmond*	4	10	0	14	8	0	0	8	22
	Burnaby	11	4	2	17	2	9	0	11	28
	Vancouver	35	24	5	64	49	41	6	96	160
	Total	50	38	7	95	59	50	6	115	210
Total		53	215	309	577	61	95	67	223	800

Tab. 2: Hlavní třída (cuboid) - všeobecná kontingenční tabulka

4 Odvozování generalizovaných pravidel

Při generalizaci založené na dimenzích je z relevantní množiny dat generalizována jedna nebo více hlavní (generalizované) třídy (cuboidy). Poté mohou být na takového hlavní generalizované třídy (cuboidy) aplikovány různé techniky pro získání různých typů znalostí, klasifikace konceptů, redukce atributů nebo další generalizace. Generalizovaná znalost může následně zahrnovat charakteristická pravidla, diskriminační pravidla, klasifikační pravidla, zákonitosti v evoluci dat, odchylková pravidla, pravidla popisu shluků a víceúrovňová asociační pravidla.

4.1 Získání charakteristických pravidel

Charakteristické pravidlo je funkce, která charakterizuje koncepty, které pokrývají všechny nebo aspoň většinu objektů v množině dat odpovídající dolovací úloze.

K získání zajímavých charakteristických pravidel z velké množiny dat se předpokládá, že koncepty v pravidlech budou reprezentovány na relativně vysoké úrovni, takže pravidla mohou být reprezentována stručně, a že budou shrnovat obecné chování množiny objektů.

Pokud je hlavní třída v generalizované podobě množiny relevantních dat, mohou být z této hlavní třídy získána charakteristická pravidla. Pro získání takovýchto znalostí mohou být použity techniky jako např. přímá prezentace

hlavní třídy jako hlavní tabulka, prezentace hlavní třídy ve formě generalizované crosstab, další generalizace hlavní třídy a redukce počtu atributů v hlavní třídě.

4.2 Získávání diskriminačních pravidel

Diskriminační pravidlo je funkce, která určuje koncepty třídy, jež je prozkoumávána (cílová třída) z jiných tříd (nazývaných kontrastní třídy). Např. pro odlišení jedné choroby od jiných, by mělo diskriminační pravidlo shrnout symptomy, které odlišují tuto chorobu od jiných.

Diskriminační pravidlo může být získáno generalizací dat současně v cílové třídě i v kontrastních třídách, s výjimkou vlastností, které se překrývají v obou třídách v generalizovaném pravidle. Obvykle vlastnost, která dobře diskriminuje třídu, se může stále v menší míře překrývat s jinými třídami v rozsáhlé množině dat. Proto se často používá n-tice vlastností, které určují, jak přesně může jít použít daná vlastnost pro odlišení cílové třídy od ostatních. Takovéto kvantitativní měřítko se nazývá *diskriminační váha* a je definována jako procento výskytů vlastnosti v cílové třídě vůči výskytům v obou třídách.

4.3 Získávání asociačních pravidel vysoké úrovně

Asociační pravidlo je pravidlo typu:

$$\mathbf{A_i \wedge \ldots \wedge A_j \rightarrow B_k \wedge .. \wedge B_l}$$

kde $A_i,\ldots,A_j$, $B_k,\ldots$, B_l jsou množiny vlastností asociovaných v databázi.

Asociační pravidlo může být vyjádřeno na základní úrovni konceptu, např.každá konstanta vyskytující se v pravidle se vyskytuje i v databázi, nebo na vysoké úrovni konceptu, kdy některé nebo všechny konstanty v daném pravidle jsou na vyšší úrovni konceptu, než je uloženo v databázi. Např. pravidlo:

$$status(x,manager) \rightarrow house\_size(x,big) \qquad [0.08, 70\%]$$

je asociační pravidlo vysoké úrovně, protože konstanty v obou případech status (manager) a house_size(big) jsou konstanty vysoké úrovně. V hranatých závorkách jsou uvedena dvě důležitá čísla: podpora a spolehlivost (support, confidence), které jsou hlavními parametry výsledného pravidla. První parametr *podpora vztahu* v množině vyjadřuje pravděpodobnost, se kterou se vztah p vyskytuje v dané množině. Druhý parametr *spolehlivost pravidla* $p \rightarrow q$ v množině S je poměr pravděpodobnosti výskytu p i q v S vůči výskytu p v S, jedná se tedy o podmíněnou pravděpodobnost.

Při vývoji technik získávání asociačních pravidel z transakčních a relačních databází byly vyvinuty metody pro získávání jedno i vícedimenzionálních asociačních pravidel. Tyto metody mohou být dále upraveny a vylepšeny pro získávání asociačních pravidel z OO databází. Např. níže popsaný algoritmus pro dolování víceúrovňových pravidel může nejprve získávat velké množiny položek (množina položek s podporou větší nebo rovnou předdefinovanému prahu podpory) na vysoké úrovni konceptu, a dále pak zpracovávány jejich odpovídající následníci.

Dolování víceúrovňových nebo jednoúrovňových asociačních pravidel je náročným procesem, protože zde existuje velmi mnoho možností, jak seskupit velké množiny dat do různě velikých skupin. Nicméně uživatelé se někdy zajímají o asociační pravidla složená z dat na vysoké úrovni konceptu, a proto je tedy možné zefektivnit dolovací techniky tak, že jsou asociační pravidla dolována na úrovni ne nižší, než je vyjádřeno v generalizované hlavní třídě.

Hlavní třída uchovává vztahy mezi daty na vysoké úrovni konceptu a s počtem uloženým v každé generalizované n-tici. Tedy z hlavní třídy mohou být velké množiny položek s podporou větší nebo rovnou předdefinované minimální podpoře. Podpora 1-množin může být určena s počtem výskytů každé hodnoty atributu v hlavní třídě. Velké 1-množiny mohou být spárovány do kandidátních velkých 2-množin, jejichž podpora může být spočtena jako součet výskytů pro každou z takovýchto 2-množin v hlavní třídě. Velké 2-množiny jsou takové kandidátní množiny, které splňují hodnotu minimální podpory. Tento proces pokračuje až do získání velkých k-množin, kde k je maximální počet atributů v hlavní třídě nebo dokud neexistují větší množiny než k-množiny. Po získání velkých množin jsou odvozována asociační pravidla, jejichž rozsáhlé množiny splňují hodnotu předdefinované minimální spolehlivosti.

Algoritmus získávání asociačních pravidel vysoké úrovně z OO databází

Vstup: 1. Úloha pro získávání asociačních pravidel ze specifikované množiny dat na vysoké úrovni konceptu v OO databázi.

2. Gen(a_i) – množina hierarchií konceptů či generalizační operátory nad atributem a_i.

3. Ti – množina prahů atributů pro atribut a_i.

4. Prahová hodnota podpory a spolehlivosti.

Výstup: Množina asociačních pravidel pro danou množinu dat na vysoké úrovni konceptu.

Postup:

1. Shromáždění relevantní množiny dat pomocí objektově-orientovaného dotazování.
2. Získání hlavní třídy počáteční pracovní třídy podobným způsobem jako je indukce orientovaná na atributy.

3. Výpočet velkých k-množin podle dat v hlavní třídě:

a) Výpočet velkých 1-množin L_1 tak, že se spočte výskyt všech 1-množin obsažených v základní třídě a následné vyřazení těch 1-množin, které nesplňují minimální podporu,

b) Iterativní výpočet velkých množin L_k (pro k>1) podle algoritmu Apriori. Nejprve jsou vytvořeny k-množiny ze dvou sousedních (k-1)-množin v Lk-1 (liší se pouze v jednom prvku). Poté jsou vyřazeny ty k-množiny, které nesplňují minimální podporu,

c) Tento proces je opakován, dokud nejsou získány všechny k-množiny, kde *k* je maximální počet atributů v hlavní třídě nebo dokud nejsou získány větší množiny než jsou k-množiny.

4. Testování všech asociačních pravidel, zda splňují hodnotu minimální spolehlivosti. Výsledná asociační pravidla jsou tvaru:

$$\mathbf{A_i \wedge \ldots A_j \rightarrow B_m \wedge \ldots \wedge B_n,}$$

kde $A_j,\ldots A_j$, B_m, …, B_n jsou dvojice hodnot atributů velkých k-množin (pro $k = j - i + n - m + 2$).

Příklad. Nechť dolování asociačních pravidel probíhá nad daty získanými následujícím dotazem a hodnota minimální podpory je 0,15 a minimální spolehlivosti 0,7.

```
mine association rules
from Person P
where P.Age >= 40 and P.Car.Maker = 'Japan'
           and 'Ph.D' in P.Education and P.Workplace.Name = 'U.B.C.'
with respect to P.Name, P.Home.House_size, P.Salary, P.Home.Address
```

Odvození asociačních pravidel probíhá následovně: nejprve je získána množina relevantních dat, dále je získána hlavní tabulka Table_1. Z tabulky Table_1 jsou získávány velké 1-množiny L_1, které jsou v následující tabulce:

large 1-itemset	support
$class = \text{“}senior\_professor\text{”}$	72.12%
$class = \text{“}senior\_manager\text{”}$	27.88%
$house\_size = \text{“}big\text{”}$	44.75%
$house\_size = \text{“}medium\text{”}$	29.00%
$house\_size = \text{“}small\text{”}$	26.25%
$address = \text{“}richmond\text{”}$	37.62%
$address = \text{“}burnaby\text{”}$	28.12%
$address = \text{“}vancouver\text{”}$	34.25%
$salary = \text{“}medium\text{”}$	38.75%
$salary = \text{“}high\text{”}$	47.00%

Tab. 3: Velké 1-množiny v hlavní třídě

Spárováním velkých 1-množin do kandidátních velkých 2-množin, jejichž hodnoty podpory mohou být vypočteny z hlavní třídy.

Velké 2-množiny, splňující minimální podporu, jsou uvedeny v následující tabulce:

large 2-itemset	support
$\{class = \text{“}senior\_professor\text{”}, house\_size = \text{“}big\text{”}\}$	36.50%
$\{class = \text{“}senior\_professor\text{”}, address = \text{“}richmond\text{”}\}$	30.50%
$\{class = \text{“}senior\_professor\text{”}, salary = \text{“}high\text{”}\}$	38.62%
$\{class = \text{“}senior\_professor\text{”}, house\_size = \text{“}medium\text{”}\}$	23.75%
$\{class = \text{“}senior\_professor\text{”}, address = \text{“}burnaby\text{”}\}$	21.88%
$\{class = \text{“}senior\_professor\text{”}, salary = \text{“}medium\text{”}\}$	26.88%
$\{class = \text{“}senior\_professor\text{”}, address = \text{“}vancouver\text{”}\}$	19.75%
$\{house\_size = \text{“}big\text{”}, address = \text{“}richmond\text{”}\}$	25.37%
$\{house\_size = \text{“}big\text{”}, salary = \text{“}high\text{”}\}$	28.12%
$\{house\_size = \text{“}big\text{”}, salary = \text{“}medium\text{”}\}$	16.25%
$\{house\_size = \text{“}medium\text{”}, salary = \text{“}high\text{”}\}$	17.25%
$\{house\_size = \text{“}small\text{”}, address = \text{“}vancouver\text{”}\}$	20.00%
$\{address = \text{“}richmond\text{”}, salary = \text{“}high\text{”}\}$	18.00%
$\{address = \text{“}richmond\text{”}, salary = \text{“}medium\text{”}\}$	18.00%
$\{address = \text{“}burnaby\text{”}, salary = \text{“}high\text{”}\}$	17.12%

Tab. 4: Velké 2-množiny v hlavní třídě

Použitím algoritmu apriori získáme z dvojic velkých 2-množin velké kandidátní třídy. Vypočteme jejich podpory a porovnáme s hodnotou minimální podpory. Získáme tak velké 3-množiny, které jsou uvedeny v následující tabulce:

large 3-itemset	support
$\{class = \text{“}senior\_professor\text{”}, house\_size = \text{“}big\text{”}, address = \text{“}richmond\text{”}\}$	20.88%
$\{class = \text{“}senior\_professor\text{”}, house\_size = \text{“}big\text{”}, salary = \text{“}high\text{”}\}$	23.50%
$\{class = \text{“}senior\_professor\text{”}, address = \text{“}richmond\text{”}, salary = \text{“}high\text{”}\}$	15.38%
$\{house\_size = \text{“}big\text{”}, address = \text{“}richmond\text{”}, salary = \text{“}high\text{”}\}$	15.38%

Tab. 5: Velké 3-množiny v hlavní třídě

Obdobně postupujeme při získávání velkých 4-množin. Nakonec testujeme výslednou množinu asociačních pravidel na splnění hodnoty minimální spolehlivosti. Výsledná pravidla dolovací úlohy jsou uvedena v následující tabulce:

$house\_size = \text{“}big\text{”} \rightarrow class = \text{“}senior\_professor\text{”}$	[36.50%, 81.56%]
$house\_size = \text{“}medium\text{”} \rightarrow class = \text{“}senior\_professor\text{”}$	[23.75%, 81.90%]
$house\_size = \text{“}small\text{”} \rightarrow address = \text{“}vancouver\text{”}$	[20.00%, 76.19%]
$address = \text{“}richmond\text{”} \rightarrow class = \text{“}senior\_professor\text{”}$	[30.50%, 81.06%]
$address = \text{“}burnaby\text{”} \rightarrow class = \text{“}senior\_professor\text{”}$	[21.88%, 77.78%]
$salary = \text{“}high\text{”} \rightarrow class = \text{“}senior\_professor\text{”}$	[38.62%, 82.18%]
$house\_size = \text{“}big\text{”} \wedge address = \text{“}richmond\text{”} \rightarrow class = \text{“}senior\_professor\text{”}$	[20.88%, 82.27%]
$house\_size = \text{“}big\text{”} \wedge salary = \text{“}high\text{”} \rightarrow class = \text{“}senior\_professor\text{”}$	[23.50%, 83.56%]
$address = \text{“}richmond\text{”} \wedge salary = \text{“}high\text{”} \rightarrow class = \text{“}senior\_professor\text{”}$	[15.38%, 85.42%]
$address = \text{“}richmond\text{”} \wedge salary = \text{“}high\text{”} \rightarrow house\_size = \text{“}big\text{”}$	[15.38%, 85.42%]

Tab. 6: Vygenerovaná asociační pravidla vysoké úrovně

5 Shrnutí a závěr

V této práci byly přiblíženy techniky pro získávání znalostí z OO databází. Tyto techniky jsou postaveny na metodách dolování na základě generalizace a indukci založené na dimenzích. Tyto techniky jsou schopny pracovat nad různými vlastnostmi objektů, jako jsou atributy, metody, atd.

Generalizační techniky jsou postaveny na výchozích znalostech o daných objektech, jako je např. konceptuální hierarchie, bez kterých nelze dolovací proces provádět.

V práci byly popsány základní operátory pro generalizaci vlastností objektů jako jsou např. metody, atributy, identifikátor objektu, atd. Dále byly popsány metody pro tvorbu objektové krychle, které využívají tyto generalizační operátory. Na závěr bylo nastíněno získávání znalostí z vytvořené objektové krychle, zejména pak dolování asociačních pravidel vysoké úrovně.

Získávání znalostí v OO databázích má mnoho společného se získáváním znalostí z relačních databází, zejména při generalizování objektů na hlavní třídu vyjádřenou často ve formě cuboidů nebo generalizovaných tabulek. Nicméně metody pro generalizovaní složitě strukturovaných dat, hierarchií tříd/podtříd, děděných nebo odvozených vlastností, hierarchií skládání tříd atd. jsou dosti specifickou oblastí pro získávání znalostí v OO databázích.

Obecně, dolování dat poskytuje mocný nástroj pro automatické generování a verifikování znalostí při konstrukci rozsáhlých bází znalostí. Představuje tak důležitý směr při vývoji datových systémů a systémů bází znalostí.

V současné době existuje na univerzitě v Osace prototyp dolovacího systému, který je založený na technikách, jenž zde byly presentovány.

6 Literatura

[1] Braga D., Campi A., Ceri S., K1emettinen M., Lanzi P.L.: Discovering interesting information in XML data with association rules. Proceedings of the 2003 ACM symposium on Applied computing, p. 450-454, Melbourne, Florida, 2003, ISBN:1-58113-624-2.

[2] Glymour C., Madigan D., Pregibon D., Smyth P.: Statistical Inference and Data Mining. Communications of the ACM archive, Volume 39, Issue 11, (November 1996), p. 35 - 41, New York, NY, USA, 1996, ISSN:0001-0782.

[3] Yaik O. B., Yong C. H., Haron F.: Time Series Prediction using Adaptive Association Rules.First International Conference on Distributed Frameworks for Multimedia Applications (DFMA'05),p. 310-314, Besançon, France, 2005.

[4] Han, J., Nishio S., Kawano H.:Generalisation-Based Mining in Object/Oriented Databases. [online] http://citeseer.ist.psu.edu.

[5] Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers. 2001.

[6] Bartík, V.: Získávání znalostí z dat – víceúrovňová asociační pravidla (Diplomová práce), Vysoké učení technické v Brně, 2001.

[7] Stryka, L.: Modul dolování asociačních pravidel (Diplomová práce), Vysoké učení technické v Brně, 2003.