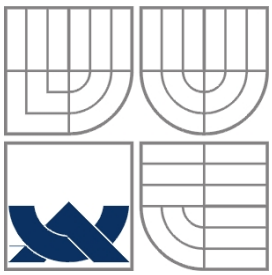
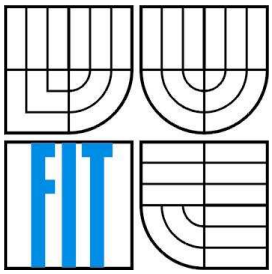


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

IMPLEMENTACE ČÁSTI STANDARDU SQL/MM DM PRO ASOCIAČNÍ PRAVIDLA

IMPLEMENTATION OF SQL/MM DM FOR ASSOCIATION RULES

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

ZDENĚK ŠKODÍK

VEDOUCÍ PRÁCE
SUPERVISOR

doc. Ing. JAROSLAV ZENDULKA, CSc.

BRNO 2007

Abstrakt

Diplomová práce se zabývá problematikou získávání znalostí z databází, konkrétně se zaměřuje na asociační pravidla, která jsou součástí systému dolování z dat. Dolováním se snažíme získat znalosti (data), která nejsou v databázi přímo definována, ale mohou být užitečná. V dokumentu je popsána problematika standardu SQL/MM DM, speciálně pak všech uživatelem definovaných typů předepsaných pro asociační pravidla, stejně jako všech obecných typů tvořících rámec pro dolování. Před samotnou implementací těchto typů jsou ještě uvedeny prostředky, kterými je to umožněno, tedy jazyk PL/SQL a podpora Oracle Data Mining. Správnost implementace je ověřena ukázkovou aplikací. Na závěr jsou zhodnoceny dosažené výsledky a zmíněno možné pokračování projektu.

Klíčová slova

Databáze, získávání znalostí z databází, dolování dat, asociační pravidla, podpora, spolehlivost, algoritmus apriori, SQL/MM DM, uživatelem definovaný typ, PL/SQL, Oracle Data Mining, DBMS_DATA_MINING

Abstract

This project is concerned with problems of knowledge discovery in databases, in the concrete then is concerned with an association rules, which are part of the system of data mining. By that way we try to get knowledge which we can´t find directly in the database and which can be useful. There is the description of SQL/MM DM, especially then all user-defined types given by standard for association rules as well as common types which create framework for data mining. Before the description of implementation these types, there is mentioned the instruments which are used for that – programming language PL/SQL and Oracle Data Mining support. The accuracy of implementation is verified by a sample application. In the conclusion, achieved results are evaluated and possible continuation of this work is mentioned.

Keywords

Database, knowledge discovery in databases, data mining, association rules, support, confidence, apriori algorithm, SQL/MM DM, user-defined type, PL/SQL, Oracle Data Mining, DBMS_DATA_MINING

Implementace části standardu SQL/MM DM pro asociační pravidla

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením doc. Ing. Jaroslava Zendulky, CSc. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Zdeněk Škodík
28. května 2007

Poděkování

Děkuji doc. Ing. Jaroslavu Zendulkovi, CSc., vedoucímu diplomové práce, za jeho shovívavost a trpělivost při vypracovávání tohoto projektu. Dále děkuji i Ing. Jaroslavu Rábovi za jeho rady při ladění ukázkové aplikace ve Form builderu.

Obsah

1 Úvod 3
2 Získávání znalostí z databází 4
2.1 Zdroje pro získávání znalostí z databází…….5
2.2 Proces získávání znalostí……9
2.2.1 Příprava dat……9
2.2.2 Modelování……9
2.2.3 Vyhodnocování výsledků……...……...……..10
2.3 Typické úlohy dolování dat……..11
2.3.1 Asociační pravidla……...……...……...……...11
2.3.2 Shlukování……...……...……...……...……12
2.3.3 Klasifikace……...……...……...……...……12
2.3.4 Prediktivní modelování..……...…12
2.4 Předpoklady úspěchu dolování dat……...12
2.5 Možné problémy při získávání znalostí z db……..……...13
2.6 Získávání znalostí v praxi..……..14
2.7 Nové směry…...……..……..15
3 Asociační pravidla……16
3.1 Základní charakteristiky pravidel…….16
3.2 Generování a počet kombinací…….18
3.3 Jednoúrovňová a zobecněná as. pravidla…….18
3.4 Jiné typy asociačních pravidel…….19
3.4.1 Pravidla s výjimkami……...19
3.4.2 Časové sekvence……..20
3.4.3 Více tabulek…...……..20
3.5 Získávání asociačních pravidel...……...……...21
3.6 Algoritmus apriori……...……...……...21
4 Standard SQL/MM DM 22
4.1 Standard SQL/MM...……...……...……...22
4.2 Standard SQL/MM DM...……...……...……...…22
4.3 Uživatelem definované typy…...……...……23
4.3.1 Typy pro přípravu dat…….23
4.3.2 Typy pro dolování asociačních pravidel…….…….25
5 Aplikace v prostředí Oracle……..29
5.1 Jazyk PL/SQL……..……...……...……...29

6 Oracle Data Mining……..31
6.1 ODM a asociační pravidla…….……...32
6.1.1 ODM a algoritmus apriori……...…….32
6.1.2 Moduly DBMS_DATA_MINING_TRANSFORM A DBMS_DATA_MINING……..33
7 Popis implementace……..40
7.1 Implementace typů standardu……...40
7.2 Typ DM_LogicalDataSpec...……...42
7.3 Typ DM_MiningData……...44
7.4 Typ DM_RuleSettings...……...45
7.5 Typ DM_RuleBldTask..……...47
7.6 Typ DM_RuleModel.……...48
7.7 Implementace ukázkové aplikace..……...49
7 Závěr……...……..55
Literatura..……...……..56
Seznam příloh...……...……..57

1 Úvod

Dnešní společnost produkuje obrovské množství dat. V každodenním životě je tak člověk nepřetržitě obklopován různými informacemi. S databázemi se pak setkává v případě, kdy se s těmito daty snaží manipulovat. Databázové technologie totiž představují osvědčený prostředek, jak uchovávat rozsáhlé množství dat a vyhledávat v nich informace.

Dolování z velkého množství dat (data mining) je novou, rychle se rozvíjející disciplínou, která dává prostor výzkumným projektům, a zároveň je již prakticky využívána v řadě oblastí. V obchodním světě je nejúspěšnější aplikací dolování z dat tzv. databázový marketing. Jde o způsob analýzy zákaznických databází, který umožňuje vyhledávat budoucí preference zákazníků. Uvádí se, že s jejím použitím lze zvýšit prodej až o 20%. Řada investičních společností využívá metody dolování z dat k analýze finančních a akciových trhů. Dolování z dat se také uplatňuje při detekci a prevenci pojišťovacích a daňových podvodů. Metody dolování se používají i v řadě vědeckých aplikací (astronomie, geologie, biologie, meteorologie). Využití ve všech těchto oblastech by ale nebylo možné bez odpovídajícího softwarového vybavení. Zařadit data mining do informačních technologií jako celku lze následovně: informační technologie – databázové jazyky – SQL multimediální a aplikační balíčky – data mining.

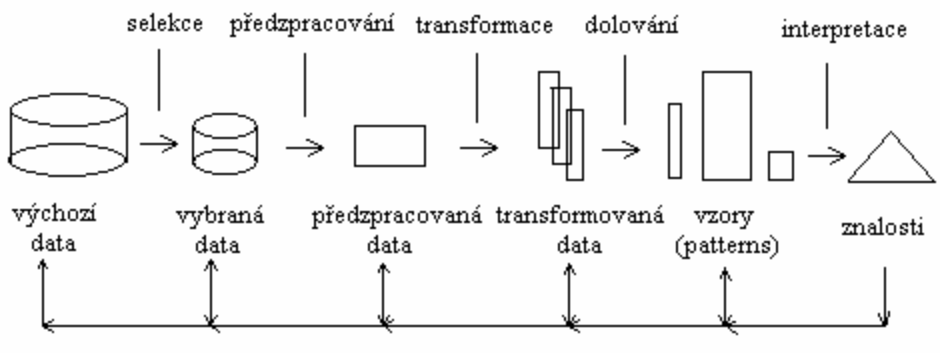
Cílem této práce je nastudování problematiky dolování dat a následné využití těchto znalostí v praxi při implementaci uživatelem definovaných typů standardu SQL/MM DM pro asociační pravidla. Je nutné seznámit se s databázemi a získáváním znalostí z nich, asociačními pravidly, standardem SQL/MM DM a typy definovanými pro získávání asociačních pravidel, podporou Oracle Data Mining. Teprve poté je možné přistoupit k implementační části práce, tedy vytvoření návrhu a jeho realizace. Pro ověření funkčnosti je nakonec vhodné vytvořit ukázkovou aplikaci.

Po úvodu následuje kapitola věnovaná získávání znalostí z databází, popisuje zdroje pro dolování, samotný proces získávání znalostí, typické úlohy i možné problémy při dobývání. Třetí kapitola je zaměřena na definici asociačních pravidel, jejich druhy a získávání z databází. Další kapitola uvádí standard SQL/MM a jeho šestou část DM (Data Mining). V jejím rámci jsou popsány všechny uživatelem definované typy navržené pro získávání asociačních pravidel. Nástroje, pomocí kterých lze přistupovat k databázi a pracovat s ní, jsou popsány v páté kapitole. Šestá kapitola představuje podporu Oracle pro dolování dat nazvanou Oracle Data Mining se zaměřením na asociační pravidla. Následující kapitola pak přibližuje samotnou implementaci problematiky. V závěru jakožto poslední kapitole práce jsou zhodnoceny výsledky a zmíněna možná pokračování.

Diplomová práce navazuje na ročníkový a semestrální projekt, z nichž jsem využil některé teoretické znalosti.

2 Získávání znalostí z databází

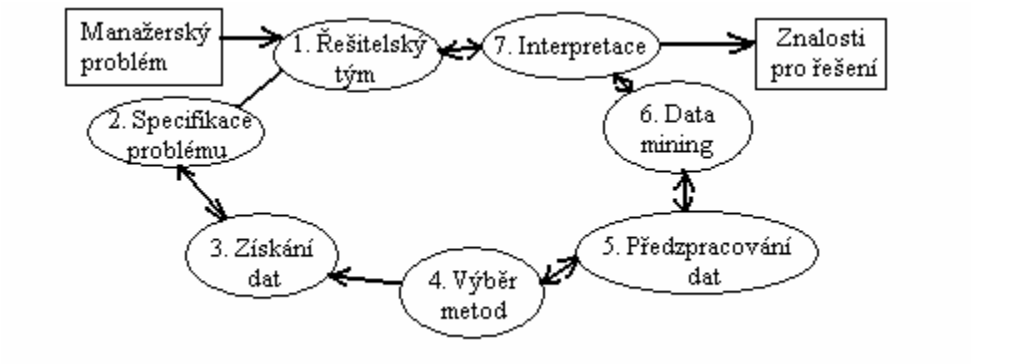
Problém získávání znalostí z databází (Knowledge Discovery in Databases, KDD) se v plné míře vynořil na počátku 90. let minulého století. Jak je charakterizované v knize [2], která slouží jako teoretické východisko pro celou tuto kapitolu, lze získávání znalostí z databází definovat jako netriviální získávání implicitních, dříve neznámých a potenciálně užitečných informací z dat. Po jistém období tápání se ustálilo chápání KDD jako interaktivního a iterativního procesu tvořeného kroky selekce, předzpracování, transformace, vlastního „dolování“ (data mining) a interpretace. Hledají se tedy potenciálně užitečné struktury v datech vedoucích k extrakci zajímavých zákonitostí či informací vyšší úrovně, které mohou být studovány z dalších úhlů pohledu. Jedním krokem v této činnosti je dolování dat (data mining). Dolování dat spočívá v aplikaci specifických algoritmů pro dobývání nebo získávání častých vzorů z dat. Asi by bylo lepší říkat „dolování znalostí z dat“, ale zůstaneme u zavedené terminologie. Pojem dolování dat bohužel nevyjadřuje přesně, co se za ním skrývá, a mnoho lidí si to nedokáže představit, protože představuje to, že je kladen důraz na získávání dat z velkého množství informací, které nejsou v databázi uloženy přímo. Přesto se výraz datového dolování (data mining) stal velmi populární. V praxi se používají ještě další ekvivalenty jako je dolování znalostí z databází (knowledge mining from databases), extrakce znalostí (knowledge extraction), datová nebo vzorová analýza (data/pattern analysis), archeologie dat (data archeology) a bagrování dat (data dredging). V rámci získávání znalostí z databází je používána řada druhů metod analýzy dat, mezi něž patří např. klasifikační metody, či rozhodovací stromy.



Obr.1 Proces získávání znalostí z databází

Zatímco schéma na obr. 1 popisuje „technologický“ pohled na získávání znalostí, lze se na problematiku nahlížet i okem manažerským. Impulsem pro zahájení procesu získávání znalostí je nějaký reálný problém. Cílem procesu je pak získat co nejvíce relevantních informací vhodných k řešení daného problému. Příkladem reálného problému je např. otázka nalezení skupin zákazníků obchodního domu nebo skupin klientů banky, kterým by bylo možné nabídnout speciální služby. U

zákazníků obchodního domu se může jednat o zjištění, že zákazník kupuje potravinářské zboží odpovídající jisté dietě, v případě klientů banky může jít o potenciální zájemce o hypotéční úvěr.



Obr. 2 Manažerský pohled na proces získávání znalostí z databází

S postupem doby začaly vznikat metodiky, které kladou za cíl poskytnout uživatelům jednotný rámec pro řešení různých úloh z oblasti získávání znalostí. Tyto metodiky umožňují sdílet a přenášet zkušenosti z úspěšných projektů. Za některými metodikami stojí producenti programových systémů (metodika 5A firmy SPSS nebo metodika SEMMA firmy SAS), jiné vznikají ve spolupráci výzkumných a komerčních institucí jako „softwarově nezávislé“ (CRISP-DM).

2.1 Zdroje pro získávání znalostí z databází

Hlavním zdrojem dat pro dolování je databázový systém. Pod pojmem databázový systém rozumíme společné označení pro data uchovávaná v centrálně zpracovávané struktuře dat, zvané databáze a pro obslužné speciální programové vybavení nazývané systém řízení bází dat (SŘBD).

Databáze, které dnes představují základní součást moderních informačních systémů, může být specifikována jako souhrn systematicky uspořádaných dat uložených a zpracovávaných odděleně od aplikačních programů. Obecně obsahují čtyři komponenty:

- Datové prvky - vlastní data
- Vztahy mezi datovými prvky - uloženy ve složitějších strukturách
- Schéma - uživatelský popis struktury dat
- Integritní omezení - podmínky kladené na data

Již v 70. letech začaly nahrazovat souborové aplikace postupy, vedoucí k oddělení aplikačních programů od vlastních dat. Programátor tak není nucen znát fyzický způsob uložení dat a metody přístupu k nim. Problémy spojené s organizací dat potom přebírá SŘBD. Vedle rozvoje SŘBD je dnes předmětem velkého vývoje i oblast návrhu databází. V souvislosti se SŘBD se rozlišují dva typy jazyků. Prvním je jazyk pro definici dat (DDL - Data Definition Language), který definuje uživatelská data. Druhým typem je pak jazyk pro manipulaci dat (DML - Data Manipulation

Language), který slouží pro aktualizaci dat a provádění výběrů podle zadaných kritérií. Charakteristickými vlastnostmi současných SŘBD jsou:

- Zpracování v reálném čase
- Transakční zpracování - každé zpracování dat je realizováno jako posloupnost operací (transakce), převádějící databázi z jednoho konzistentního stavu do druhého
- Variabilní architektura dat - centralizované (data jsou fyzicky uložena na jednom počítači) nebo distribuované uložení dat (na více počítačích)
- Ochrana dat - před neoprávněným přístupem, změnou nebo poškozením formou speciálních uživatelských práv zpravidla chráněných heslem
- Zotavení z chyb - pokud se nepodaří transakci úspěšně ukončit, musí být všechny její efekty odstraněny
- Řízení souběžného přístupu více uživatelů - definována jednotlivá práva a priority
- Práce s multimediálními daty - text, zvuk, obraz, apod.

V prehistorii databází byla data ukládána v jednom velkém „plochém“ souboru (tzv. flat file), ke kterému se přistupovalo indexovanými sekvenčními metodami (ISAM). Velkým krokem kupředu pak bylo zavedení relačních databází. Jeden velký datový soubor byl rozdělen do řady relací (tabulek).

Relační databáze je tedy tvořena:

- Množinou relací – relace je reprezentována dvourozměrnou tabulkou (řádky odpovídající záznamům, sloupce atributům, jednotlivé záznamy jsou jednoznačně identifikovány pomocí primárního klíče).
- Operacemi selekce, projekce a spojení pro manipulaci s tabulkami – selekce slouží k výběru záznamů (řádků tabulky), projekce k výběru atributů (sloupců tabulky) a spojení slouží k propojování tabulek (spojují se řádky se stejnou hodnotou nějakého atributu – obvykle klíče).

Pro kladení dotazů nabízejí relační databáze dva způsoby:

- QBE – query by example představuje uživatelům relativně jednoduchý intuitivní způsob kladení dotazů, je tedy vhodnější pro méně zkušené uživatele.
- SQL – structured query language je určen uživatelům zkušeným. Jde o jednoduchý programovací jazyk pro definování dat a pro manipulaci s nimi. Obecně je výrazně mocnější, flexibilnější a je více rozšířen.

Objektově orientované databáze se začaly rozvíjet zejména v poslední době, hlavně díky objektově orientovanému programování. Katalyzátorem byla snaha o odstranění chyb a nedostatků relačních systémů. Objektově orientované databáze mohou všechna data ukládat do objektů, které odpovídají objektům reálného světa. Na rozdíl od relačních systémů má každý objekt svou identitu (v relačních databázích byly jednoznačně identifikovány objekty pomocí primárních klíčů), kterou z

definice dat může dědit, nebo vytvářet úplně nové objekty. S objekty můžeme provádět běžné operace pro objektově orientovaný přístup. V současnosti hlavní nevýhodou je, že objektově orientované databáze ještě nejsou dokonale propracovány a více používány, z čehož plynou i následující nevýhody:

- Na začátku vývoje nebyl žádný standard OODBMS, protože bylo časově náročné tento standard vyvinout
- Nedostatek nástrojů pro manipulaci s objekty
- Obtížný přesun dat z relační do objektově orientované databáze

Dalším logickým krokem vývoje byl vznik objektově relačních databází, u nichž se podařilo eliminovat nevýhody jak relačních, tak objektově orientovaných přístupů uložení dat. OR model je založen na relačním modelu, ale je otevřen pro rozšíření, tudíž je migrace z relačních do objektově orientovaných databází snadná. Jako rozšíření dovoluje uživateli definovat nové, složitější datové typy a metody pro práci s nimi, poskytuje dědičnost, zapouzdření a polymorfismus. Výhodou objektů je vedle lepší reprezentace dat reálného světa zejména fakt, že na rozdíl od tabulek, které ukládají jen data, objekty zahrnují také operace, které potřebujeme pro práci s těmito daty. Pro databázové systémy je charakteristické, že pro každý objekt je generován jedinečný identifikátor OID. Tabulka obsahující tyto objekty se v systému nazývá objektová tabulka. Místo tradičního řešení vztahů v relačním datovém modelu pomocí primárních a cizích klíčů je použito odkazů, které jsou typickým rysem v systému při migraci relační aplikace s objektově relačním či objektovým přístupem. Relačními objekty pak nazýváme takové objekty, na které směřují odkazy z jiných objektů. Jejich vlastností je, že jsou od odkazujících se objektů fyzicky odděleny, odkazy jsou vlastně směrníky na řádky tabulky (= objekty). Objektová tabulka se vytvoří pomocí abstraktního datového typu.

V posledních letech přispěly velkou měrou k rozvoji dolování dat datové sklady, které představují místo, kde jsou analyzovaná data uložena. Datový sklad je subjektově orientovaný, integrovaný, časově proměnný, leč stálý soubor dat, který slouží pro podporu rozhodování. Lze je v procesu dolování dat považovat za jeden z ideálních zdrojů vstupních dat. Procesy transformace a čištění, kterými prošla data při plnění do datového skladu, jsou pro dolování dat současně výhodou i nevýhodou. Předpřipravená data již nemusí vyžadovat další čištění v rámci dolování, na druhou stranu mohou být ztraceny některé informace umožňující odhalení skutečného původu dolováním získaných informací. V databázích se uchovávají především primární data a odvozená data je potřeba vždy znovu (často zdlouhavě) vyhodnocovat, přičemž s odstupem času nemusí být toto odvození dat vlivem změny obsahu databáze vůbec proveditelné. Problematiku řeší právě datový sklad, který uchovává odvozená data, různě agregovaná a vhodně předpřipravená pro hlavní cíl, jímž je operativní podpora rozhodování manažerů. Jednou z hlavních dimenzí datových skladů je dimenze časová, která umožňuje uchováváním a zpracováváním neplatných dat získat často žádaný pohled na vývoj dat v čase. Datový sklad je specifická databáze, která slouží jako neutrální datový prostor pro uchovávání dat z různých dílčích databázových systémů.

Významným aspektem ve využívání databází je bezpečnost. Je přirozené, že chceme mít data nejen přehledně uložena, ale i dobře chráněna. Bezpečnostní úroveň databází je formulována autorizačními pravidly, jejichž dodržování zajišťuje bezpečnostní systém. Tato pravidla jsou většinou vyjádřena v jazyce SQL. Bezpečnost je zajišťována řízením přístupu k datům, které může nabývat následujících podob:

- Volitelné řízení přístupu (Discretionary Access Control) – jednotlivým uživatelům či jejich skupinám přiděluje oprávněný uživatel potřebná přístupová práva k jednotlivým objektům.
- Povinné řízení přístupu (Mandatory Access Control) – objekty jsou členěny do předem daných bezpečnostních úrovní a jednotliví uživatelé jsou zařazeni do skupin s příslušnými přístupovými právy. Jedná se o využití principu víceúrovňové bezpečnosti v databázovém prostředí.

Jako lépe využitelný se z tohoto pohledu jeví relační model. U objektově relačních modelů platí, že koncepce vlastnictví objektů zatím nemá jednoznačnou interpretaci.

Druhý zajímavý zdroj představuje statistika, jež nabízí celou řadu teoreticky dobře prozkoumaných, zdůvodněných a léty praxe ověřených metod pro analýzu dat. Pro oblast získávání znalostí z databází mají význam:

- Kontingenční tabulky – pro zjišťování vztahu mezi dvěma kategoriálními veličinami.
- Regresní analýza – pro hledání funkčních závislostí jedné numerické (spojité) veličiny na jiných numerických veličinách.
- Diskriminační analýza – pro odlišení příkladů (pozorování) patřících do různých tříd.
- Shluková analýza – pro nalezení skupin (shluků) navzájem si podobných příkladů.

Jakožto o posledním podstatném zdroji hovoříme o strojovém učení. Důležitou vlastností živých organismů je schopnost přizpůsobovat se měnícím se podmínkám (adaptovat se), eventuálně se učit na základě vlastních zkušeností. Schopnost učit se bývá někdy dokonce považována za definici inteligence. Je proto přirozené, že vybavit touto vlastností i systémy technické je jedním z cílů umělé inteligence. Navíc v řadě praktických případů, kdy není dostatek apriorních znalostí o řešeném problému, ani jinak postupovat nelze. V zásadě lze rozlišit dva typy učení: učení se znalostem a učení se dovednostem. První typ hledá koncepty, obecné zákonitosti apod. (např. jak rozpoznat defraudanta), u druhého typu jde o zdokonalení schopností, na základě procvičování nějaké činnosti (např. jak nalézt cestu v bludišti).

2.2 Proces získávání znalostí

2.2.1 Příprava dat

Příprava (předzpracování) dat je nejobtížnější a časově nejnáročnější krok celého procesu získávání znalostí z databází. Současně je to ale krok, který má klíčový význam pro úspěch dané aplikace. Je to ta část práce, která (spolu s krokem porozumění problému) vyžaduje největší podíl spolupráce s expertem z dané aplikační oblasti. S příchodem reálných aplikací metod strojového učení se postupně začíná přesouvat pozornost odborníků na získávání znalostí od algoritmů pro modelování k algoritmům pro předzpracování a přípravu dat.

Cíl předzpracování je obvykle dvojí – buďto vybrat (nebo vytvořit) z dostupných dat ty údaje, které jsou relevantní pro zvolenou úlohu získávání znalostí, nebo reprezentovat tyto údaje v podobě, která je vhodná pro zpracování zvoleným algoritmem. Zatímco první cíl úzce souvisí s porozuměním problému i s porozuměním datům (a je tedy závislý na aplikační oblasti), druhý cíl (přizpůsobit data uvažovanému algoritmu) je relativně aplikačně nezávislý.

Z hlediska přípravy dat existují postupy, jak zpracovat:

- strukturovaná data
- více vzájemně propojených tabulek
- odvozené atributy
- data s příliš mnoho objekty
- data s příliš mnoho atributy
- numerické atributy
- kategoriální atributy
- chybějící hodnoty

2.2.2 Modelování

Jádrem celého procesu získávání znalostí je použití analytických metod. Tento krok bývá v anglické literatuře nazýván data mining, modeling, nebo analyzis. Vstupem do analytických procedur jsou předzpracovaná data, výstupem znalosti. Vzhledem k tomu, že vycházíme z představy, že navzájem si podobné příklady (resp. příklady téže třídy) vytvářejí shluky v prostoru atributů, nalezené znalosti reprezentují tyto shluky. Způsob reprezentace znalostí přitom může být rozmanitý. Mohou to být např. funkce přiřazené jednotlivým shlukům (to je případ subsymbolických metod), nebo rozdělení prostoru atributů na snadno popsatelné, pravidelné útvary (to je případ metod symbolických).

Jednotlivé metody se ovšem neliší pouze způsobem reprezentování hledaných znalostí. Další rozdíly mezi metodami spočívají v tom:

- pro jaký způsob úlohy získávání znalostí jsou vhodné,

- jak složité shluky dokáží reprezentovat,
- do jaké míry jsou nalezené znalosti srozumitelné pro uživatele,
- jak jsou nalezené znalosti efektivní při klasifikaci nových případů,
- pro jaký typ dat jsou vhodné.

Ze symbolických metod jsou nejčastěji používané analytické metody zejména rozhodovací stromy, asociační pravidla, rozhodovací pravidla a metody induktivního programování. Tyto metody vesměs chápou učení se z dat jako prohledávání prostoru možných řešení. Z metod subsymbolických si můžeme uvést neuronové sítě, genetické algoritmy a bayesovské metody. V těchto metodách jde většinou o učení chápané jako aproximace nějaké funkce (ztrátové funkce v případě neuronových sítí, „fitness“ funkce u genetických algoritmů nebo pravděpodobnosti). Poznamenejme ještě, že subsymbolické nemusí znamenat protiklad symbolického; nejmarkantnější je to v případě bayesovských metod, kdy pravděpodobnosti můžeme interpretovat i jako vazby mezi atributy. Rozlišujeme i metody založené na analogii (učení založené na instancích, případové usuzování – učení je zde chápáno jako zapamatování si typických příkladů).

2.2.3 Vyhodnocování výsledků

Důležitým krokem v procesu získávání znalostí je interpretace a ocenění nalezených znalostí. V případě deskriptivních úloh je hlavním kritériem novost, zajímavost, užitečnost a srozumitelnost. Tyto charakteristiky úzce souvisejí s danou aplikační oblastí, s tím, co přinášejí expertům a koncovým uživatelům. Z tohoto pohledu můžeme hovořit o:

- zřejmých znalostech, které jsou ve shodě se „zdravým selským rozumem“ – příkladem může být pravidlo, že pokud měl pacient v těhotenství problémy, tak se jednalo o ženu
- zřejmých znalostech, které jsou ve shodě se znalostmi experta z dané oblasti – např. fakt, že pokud se účet klienta banky pohybuje v záporném zůstatku, má tento klient problémy se splácením úvěru
- nových, zajímavých znalostech, které přinášejí nový pohled
- znalostech, které musí expert podrobit bližší analýze, neboť není zcela jasné, co znamenají
- znalostech, které jsou v rozporu se znalostmi experta

Jednou ze součástí vyhodnocování výsledků je testování modelů. Při hledání znalostí pro potřeby klasifikace se obvykle postupuje metodou učení s učitelem. Metody vyhodnocování modelů jsou pak založeny na testování nalezených znalostí na datech, na možnosti porovnat, jak dobře se nalezené znalosti shodují s informací od učitele.

Přestože vizualizace hraje důležitou roli především ve fázi porozumění datům, resp. při interpretaci deskriptivních znalostí, můžeme se s ní setkat i při hodnocení modelů určených pro klasifikaci. Jako na jiných místech procesu získávání znalostí, tak i zde jde o to umožnit expertovi lépe porozumět tomu, co se děje.

Máme-li k dispozici více algoritmů pro získávání znalostí, můžeme je použít na tatáž data. Naskýtá se pak otázka, který z modelů je nejlepší. Odpověď se opět hledá na základě provedeného testování jednotlivých modelů. Výsledky testování se pak navzájem porovnávají.

Vzhledem k tomu, že neexistuje algoritmus, který by předčil ostatní na libovolných datech, dostává se do popředí otázka, jak dopředu poznat, který algoritmus zvolit pro danou úlohu. Odpověď můžeme hledat na základě znalosti silných a slabých stránek jednotlivých algoritmů nebo experimentálně. Mezi známé charakteristiky algoritmů, které jsou brány v úvahu, patří například:

- rozdíl mezi způsobem reprezentace příkladů (hodnoty atributů nebo relace)
- rozdíl mezi vyjadřovací silou jednotlivých algoritmů (rozhodovací stromy a pravidla rozdělují prostor atributů rovnoběžně s osami, neuronové sítě nebo diskriminační funkce naleznou i diagonální hranici mezi třídami)
- schopnost práce se zašuměnými a chybějícími daty (různé způsoby náhrady nepoužitelné hodnoty),
- schopnost práce s maticí cen (ve fázi učení, ve fázi testování, vůbec ne)
- předpoklad nezávislosti mezi atributy
- ostrá nebo neostrá klasifikace (tj. zda odvozujeme jen indikátor třídy nebo i pravděpodobnost či váhu klasifikace)

K empirickým studiím vhodnosti jednotlivých algoritmů na různé typy dat patří dva rozsáhlé výzkumné projekty celoevropského rozsahu, STATLOG a METAL. Možností, jak zlepšit výsledky dosažené jednotlivými modely, je i jejich vzájemná kombinace.

2.3 Typické úlohy dolování dat

Dolování dat není úzce specializované odvětví, spíše naopak se s ním můžeme čím dál častěji setkávat například v databázových systémech, které již samy začínají ovládat některé techniky dolování v datech. K oboustranné spolupráci dochází mezi dolováním v datech a obory, jako statistika nebo strojové učení. Existují obecně čtyři nejtypičtější úlohy, kterými se data mining zabývají. Jsou jimi hledání častých vzorů (tedy asociačních pravidel), shlukování, klasifikace a prediktivní modelování.

2.3.1 Asociační pravidla

V této diplomové práci se budeme blíže věnovat právě hledání častých vzorů, podrobnější popis tohoto dolování je tedy uveden v kapitole 3.

2.3.2 Shlukování

Shlukováním se nazývá proces seskupování množiny fyzických nebo abstraktních objektů do podobných tříd. Shluk je seskupení objektů, které jsou podobné ostatním v rámci jednoho shluku a jsou odlišné od datových objektů ve shlucích ostatních. Se shlukem se pak dá pracovat jako s jednou skupinou. Vstupem pro metody zabývající se shlukováním je obecně n-dimenzionální prostor bodů. Pro představu můžeme uvést jednoduchý příklad – na dokumenty můžeme v určitém smyslu nahlížet jako na body v prostoru vysoké dimenze, kdy prostorové umístění takového dokumentu je dáno výskytem v něm obsažených jistých slov. Shluky bodu pak budou představovat podobné dokumenty, např. zabývající se stejným tématem.

2.3.3 Klasifikace

Klasifikace a shlukování mají určité prvky podobné. Shlukování mělo daný prostor bodů, u každého bodu pak byly známy jisté atributy. K těmto atributům jsme se pomocí technik shlukování snažili přidat nově vytvořený atribut, který by byl příznakem příslušnosti do konkrétního shluku. V klasifikaci již máme množinu entit a u každé známe její hodnoty atributu a navíc každá entita již má přirazený klasifikační atribut. Tento atribut zařazuje entitu do určité kategorie, třídy, shluku. Úkolem klasifikace je pak nově zadané entitě, na základě hodnot jejích ostatních atributů, nastavit klasifikační atribut. To znamená zařadit novou entitu do jisté kategorie entit.

2.3.4 Prediktivní modelování

Prediktivním modelováním rozumíme postup, kdy na základě známé množiny vstupních a výstupních hodnot (odpovídajících vstupům) se hledá nejpravděpodobnější hodnota výstupu pro předem neznámé kombinace vstupních hodnot. Jedním z příkladů prediktivního modelování je např. hodnocení rizika úvěru v bankovnictví. Banky shromažďují záznamy o svých klientech, o nichž je známo, že jsou dlužníky. Pokud vytvoříme prediktivní model popisujícího hodnocení dlužníka (výstup) na základě informací o něm (vstupní data), lze hodnotit rizika nově příchozích zákazníků, o kterých lze zjistit údaje používané jako vstupní data modelu.

2.4 Předpoklady úspěchu dolování dat

Předpoklady pro úspěšné nasazení technologie dolování dat v imaginární firmě:

- Kvalitně připravená vstupní data
- Promyšlená příprava – urychlení implementace procesů dolování dat
- Vyhodnocení nejvhodnější techniky pro dolování dat
- Kvalitní řízení projektů dolování dat
- Spolupráce IT a uživatelů

2.5 Možné problémy při získávání znalostí z db

Při získávání znalostí z databází jsou kritickými atributy úspěchu zejména vybraná dolovací technika a reprezentace reálného světa (zvolení vhodných datových typů). Níže uvedené problémy jsou v současné době předmětem výzkumu v oblasti dolování dat i procesu získávání znalostí z databází.

- Upravování nekompletních dat – data uložená v databázi mohou být neefektivně vyjádřena, nebo dokonce nekompletní. Při dolování mohou taková data zmást dolovací systém a vedou k chybám v získaných znalostech. Metody pro čištění a analýzu dat nejsou většinou součástí dolovacích systémů.
- Práce s relačními a složitými typy dat – relační databáze a datové sklady jsou velmi využívány a je pro ně důležitý vývoj efektivních a výkonných dolovacích systémů. Jiné databáze poskytují složité datové objekty, hypertextová, dočasná a prostorová data. Neexistuje ještě systém, kterým by se dala dolovat všechna data. Dolovací systémy jsou většinou konstruovány pro určitý druh dat.
- Dolování rozdílných druhů znalostí v rozdílných databázích – různí uživatelé se mohou zajímat o různé typy znalostí, a proto by měl dolovací systém využívat široké spektrum datových analýz pro objevování těchto znalostí. Pro tyto úkoly požadujeme užití velkého množství dolovacích technik na stejnou databázi různými způsoby, které musí být vyvinuty.
- Interaktivní dolování znalostí a mnohonásobné úrovně abstrakce – protože je obtížné předvídat, co přesně můžeme najít v databázi, je výhodné, aby byl dolovací systém interaktivní. Interaktivita v dolování dat umožňuje uživateli změnit hledané vzory, poskytuje upravování a čistění dolovaných dat. Uživatel by měl interaktivně vidět a ovlivňovat dolovací systém, který mu zobrazuje data a objevené vzory v různých úrovních a úhlech pohledu.
- Začleňování původních znalostí – původní znalosti nebo informace, které byly znovu přehodnoceny jako zajímavé, mohou poskytovat přístup a vzor pro další dolovací proces. To dovoluje objevovat vzory, které mohou být výstižně popsány na rozdílných úrovních abstrakce. Oblasti znalostí, které se vztahují k databázím, jako jsou integritní omezení a deduktivní pravidla, mohou pomoci k urychlení dolovacího procesu nebo k odhadnutí zajímavosti vzorů.
- Vyhodnocování vzorů – dolovací systémy mohou odhalit tisíce vzorů. Tyto vzory ale mohou být nezajímavé pro uživatele (poskytují mu již známé nebo běžně dosažitelné znalosti). Je to výzva pro metody vyhodnocování vzorů, aby seřadily výsledné vzory podle zajímavosti pro uživatele.
- Prezentace a vizualizace výsledků – získané znalosti by měly být prezentovány tak, aby byly snadno srozumitelné a bylo umožněno přímo s nimi pracovat. Toto je velmi důležité

u interaktivních datových dolovacích systémů. Srozumitelná reprezentace získaných dat požaduje použít techniky jako jsou stromy, tabulky, grafy, křivky atd.

- Efektivita dolovacích systémů – k efektivnímu dobývání znalostí z velkého množství informací potřebují dolovací algoritmy, aby byly efektivní a dostupné. Prováděcí čas by měl být předvídatelný.
- Paralelnost a distributivnost dolovacích algoritmů – velké množství databází, data rozdělená mezi tyto databáze a výpočetní složitost dolovacích metod jsou činitele, které by měly přispívat k vývoji paralelních nebo distribuovaných databázových algoritmů. Takové algoritmy rozdělí data na určité části, které zpracovávají paralelně. Výsledky jsou pak spojeny dohromady. Takové algoritmy zvyšují výkon a sílu dříve objevených algoritmů.
- Dolování informací z různorodých databází a globálních informačních systémů – lokální a celosvětové počítačové sítě (jako je internet) spojují velký počet zdrojů dat, které tvoří velké, distribuované a heterogenní databáze. Objevování znalostí z rozdílných zdrojů buď strukturovaných, nebo nestrukturovaných, s různou sémantikou dat je další směr pro vývoj dolovacích systémů.
- Databázové dotazovací jazyky pro dolování dat – relační dotazovací jazyky, jako je SQL, dovolují uživateli klást dotazy k vyhledávání dat. Stejně tak jazyky vyšší úrovně pro dolování dat a dotazování je třeba vyvinout tak, aby umožnily uživateli popisovat to, co chce dolovat. Také by měly sloužit k usnadnění zadávání důležitých informací pro analýzu dat, podmínek a omezení, které jsou požadovány na objevených vzorech.

2.6 Získávání znalostí v praxi

Lidmi, zabývajícími se problematikou získávání znalostí z databází, řešené úlohy a následné diskuze ukazují, jaké jsou klíčové předpoklady úspěchu používání metod získávání znalostí v praxi:

- Metodika pro standardizaci procesu získávání znalostí – metodika učiní proces KDD srozumitelným a reprodukovatelným. Umožní rovněž přenesení úspěšných postupů a řešení z jedné aplikační oblasti do jiné, a také sdílení zkušeností mezi odborníky zabývajícími se touto problematikou.
- Spolupráce s experty z dané aplikační oblasti – podobně jako v případě expertních systémů, i při získávání znalostí má expert z dané aplikační oblasti (a expert na data) důležitou roli. Jeho spolupráce je klíčová, jak v úvodních krocích (porozumění dané problematice a porozumění datům), tak pro ocenění a využití znalostí.
- Dokonalejší metody předzpracování – algoritmy pro předzpracování a transformaci dat (diskretizace a seskupování hodnot, ošetření chybějících hodnot, vytváření nových atributů) obvykle pracují nezávisle na aplikační oblasti. Zdá se, že využití doménových znalostí může výrazně zvýšit efektivnost těchto metod.

- Algoritmy schopné zpracovávat složitější data – většina algoritmů používaných pro modelování pracuje s jedinou datovou tabulkou tvořenou záznamy s pevnou strukturou. V reálných aplikacích se ale setkáváme s podstatně složitějšími typy dat: vzájemně provázanými relacemi, časovými daty, prostorovými daty, texty, strukturovanými daty. Řada činností v kroku předzpracování jde tedy na vrub „nedokonalým“ nástrojům pro modelování. Přestože tato oblast je předmětem intenzivního výzkumu, do běžně používaných systémů pro získávání znalostí se dosažené výsledky zatím příliš nepromítly.
- Interpretace výsledků srozumitelná expertovi – rozhodujícím kritériem pro úspěch nějaké reálné aplikace KDD je akceptování výsledků experty a potenciálními uživateli. To nejlepší je bezcenné, pokud nebude používáno. Experti nejsou ochotni probírat se stovkami a stovkami pravidel, ani je nezajímají tabulky ukazující zlepšení jednoho klasifikátoru vůči jinému o 1,13%. Co je zajímá, je pochopení nalezených znalostí nebo silná a slabá místa naučeného klasifikátoru. Jako důležité se tedy jeví následné zpracování výsledků a jejich vizualizace.

2.7 Nové směry

V drtivé většině se databázemi, ze kterých se dobývají znalosti, myslí relační databáze (jedna nebo více). U těchto databází se předpokládá vzájemná nezávislost záznamů z hlediska pořadí v databázi. Existují samozřejmě složitější data: časová (např. časové řady), prostorová (např. data z geografických informačních systémů) nebo strukturální (např. data o chemických sloučeninách). Na druhé straně stojí data nestrukturovaná (např. texty). Spolu s tím se objevují nové oblasti aplikací získávání znalostí, které řeší své specifické problémy. Někdy stačí mírně adaptovat existující postupy, někdy je třeba zásadně změnit kroky předzpracování a transformace dat, a někdy přicházejí na řadu zcela nové metody. V dalších dvou odstavcích se můžete letmo seznámit se dvěma dnes tak populárními oblastmi získávání znalostí – získávání znalostí z textu a webu.

Získávání znalostí z textů můžeme chápat jako speciální typ úlohy získávání znalostí z databází. Zatímco u databází pracujeme s údaji uloženými v pevné struktuře, zde máme co do činění s nestrukturovaným textem. Hlavním problémem tedy je, jak vhodně reprezentovat textový dokument, aby bylo možné použít některý z algoritmů.

Získávání znalosti z webu soustřeďuje svoji pozornost na nejdynamičtěji se rozvíjející zdroj informací současnosti, na Internet. V některých případech web slouží jako zdroj dat pro „klasické“ získávání znalostí z databází i pro získávání znalostí z textů, v jiných případech jde o nové typy úloh vyplývající ze zvláštností Internetu. Můžeme pak hovořit o získávání znalostí na základě obsahu webu, struktury webu nebo používání webu.

Zdá se, že jako další krok v řadě text mining – web mining se objeví multimedia mining, tedy získávání znalostí z multimediálních dat kombinujících texty, obrázky, zvuky, videosekvence apod.

3 Asociační pravidla

Pod pojmem asociace si můžeme představit spojení, které se vytváří za určitých podmínek mezi objekty systému tak, že vybavením jednoho se vyvolá i druhý objekt. Jedná se tedy o hledání vztahů mezi analyzovanými daty. Asociací nazýváme vztahy mezi dvěma nebo více atributy, případně mezi určitými množinami atributů [2] (tato kniha je teoretickým základem kapitoly). Lze je rozdělit na klasické (pokud jsou mezi dvěma množinami atributů určité vztahy), transakční (vztahy v rámci jedné množiny atributů) a agregované (vztahy mezi podmnožinou atributů a jejich vlastními charakteristikami).

IF-THEN konstrukce nalezneme ve všech programovacích jazycích, používají se v běžné mluvě (nebude-li pršet, nezmoknem). Není tedy divu, že pravidla s touto syntaxí patří společně s rozhodovacími stromy k nejčastěji používaným prostředkům pro reprezentaci znalostí, ať už získaných od expertů, nebo vytvořených automatizovaně z dat.

Termín asociační pravidla se stal populárním počátkem 90. let v souvislosti s analýzou nákupního košíku. Při této analýze se zjišťuje, jaké druhy zboží si současně kupují zákazníci v supermarketech (např. zelí a knedlík). Jde tedy o hledání vzájemných vazeb (asociací) mezi různými položkami sortimentu prodejny. Přitom není upřednostňován žádný speciální druh zboží jako závěr pravidla. V tomto smyslu budeme chápat pravidla v této kapitole.

3.1 Základní charakteristiky pravidel

U pravidel vytvořených z dat nás obvykle zajímá, kolik příkladů splňuje předpoklad a kolik závěr pravidla, kolik příkladů splňuje předpoklad i závěr současně, kolik příkladů splňuje předpoklad a nesplňuje závěr atd. Tedy zajímá nás, jak pro pravidlo Ant -> Suc (kde Ant je předpoklad, levá strana pravidla, antecedent a Suc závěr, pravá strana pravidla, sukcedent) vypadá příslušná kontingenční tabulka. Pro *n* příkladů je její podoba uvedena v následující tabulce, kde:

- $n(Ant \wedge Suc) = a$ je počet objektů pokrytých současně předpokladem i závěrem
- $n(Ant \wedge \neg Suc) = b$ je počet objektů pokrytých předpokladem a nepokrytých závěrem
- $n(\neg Ant \wedge Suc) = c$ je počet příkladů nepokrytých předpokladem ale pokrytých závěrem
- $n(\neg Ant \wedge \neg Suc) = d$ je počet příkladů nepokrytých ani předpokladem ani závěrem

	Suc	¬Suc	Σ
Ant	a	b	r
¬Ant	c	d	s
Σ	k	l	n

Tabulka 1: Kontingenční (čtyřpolní) tabulka

Z těchto hodnot (místo o počtu objektů pokrytých kombinací se někdy mluví o četnosti, resp. frekvenci kombinace) můžeme počítat různé charakteristiky pravidel a kvantitativně tak hodnotit nalezené znalosti. Základními charakteristikami asociačních pravidel jsou podpora (support) a spolehlivost (confidence). Podpora je počet objektů splňujících předpoklad i závěr, tedy hodnota a, platí

$$P(Suc \wedge Ant) = \frac{a}{a + b + c + d}$$

spolehlivost (též nazývaná platnost - validity, konsistence - consistency, nebo správnost - accurancy) je vlastně podmíněná pravděpodobnost závěru, pokud platí předpoklad, tedy

$$P(Suc \mid Ant) = \frac{a}{a + b}$$

Pomocí těchto dvou parametrů (podpora a spolehlivost) lze určit, jak statisticky významné pravidlo jsme z databáze získali, tj. jak je pro nás zajímavé. Obvykle volíme nějakou spodní mez minimální podpory a minimální spolehlivosti, pomocí nichž se vylučují ta pravidla, která pro nás nejsou tak zajímavá. Další důležité charakteristiky:

- počet objektů splňující předpoklad:

$$P(Ant) = \frac{a + b}{a + b + c + d}$$

- počet objektů splňující závěr:

$$P(Succ) = \frac{a + c}{a + b + c + d}$$

- pokrytí (podmíněná pravděpodobnost předpokladu pokud platí závěr)

$$P(Ant \mid Suc) = \frac{a}{a + c}$$

- kvalita (vážený součet spolehlivosti a pokrytí, w_1 a w_2 se obvykle volí tak, aby $w_1+w_2=1$):

$$Kvalita = w_1 \frac{a}{a + b} + w_2 \frac{a}{a + c}$$

Na základě platnosti a pokrytí lze dělit implikace do několika skupin:

- konzistentní pravidla - pravidla s platností rovnou 1, levá strana implikace je postačující podmínkou pro splnění pravé strany
- úplná pravidla - pravidla s pokrytím rovným 1, levá strana implikace je nutnou podmínkou pro splnění pravé strany
- deterministická pravidla - pravidla s platností i pokrytím rovným 1, levá strana je nutnou a postačující podmínkou pro splnění pravé strany

3.2 Generování a počet kombinací

Základem všech algoritmů pro hledání asociačních pravidel je generování kombinací (konjunkcí) hodnot atributů. Při generování vlastně procházíme prostor všech přístupných konjunkcí. Metod se nabízí několik – generování do šířky, do hloubky a heuristická metoda.

Při generování do šířky se kombinace získávají tak, že se nejprve vygenerují všechny kombinace délky jedna, pak všechny kombinace délky dva atd. Jde tedy o generování kombinací podle délek, kategorie jednoto atributu jsou přitom uspořádány podle abecedy.

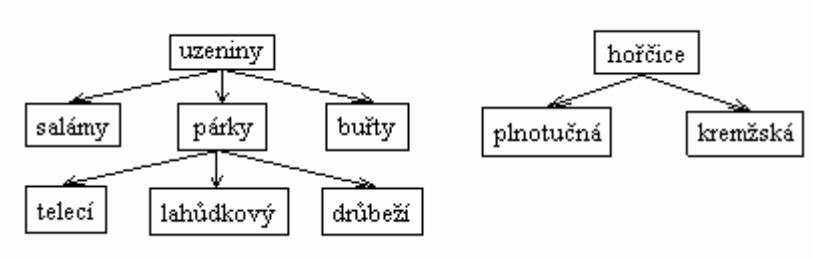
Při generování do hloubky se vyjde od první kombinace délky jedna a ta se pak prodlužuje (vždy o první kategorii dalšího atributu), dokud to jde. Nelze-li kombinaci prodloužit, změní se kategorie „posledního“ atributu. Pokud nemůžeme provést ani to (vyčerpaly se kategorie posledního atributu), kombinace se zkrátí a současně se změní poslední kategorie. Kategorie jednoho atributu jsou opět uspořádány podle abecedy.

Oba zmíněné způsoby jsou „slepé“. Provádějí se pouze na základě seznamu hodnot atributů a neberou do úvahy vstupní data. Proto můžeme vygenerovat kombinace, které se nevyskytují v datech. Poslední zde uváděný způsob generování podle četnosti vytváří kombinace v pořadí podle jejich výskytu v datech. Jde o příklad heuristického prohledávání prostoru kombinací, kde heuristikou je „uvažuj kombinaci s nejvyšší četností“. Při tomto způsobu generování se kombinace s nulovou četností objeví až na konci seznamu.

Generování kombinací je výpočetně náročný proces. S růstem počtu atributů značně roste počet možných konjunkcí. Při počítání počtu kombinací s ohledem na řídící parametry generování a vstupní data lze říci, že počet generovaných (a testovaných) kombinací je exponenciálně závislý na počtu atributů.

3.3 Jednoúrovňová a zobecněná as. pravidla

Jednoúrovňová asociační pravidla představují nejjednodušší typ pravidel, u položek obsažených v transakcích nás zajímá pouze to, zda je v dané transakci daná položka obsažena, či nikoliv. Jsou to tzv. booleovská pravidla. Nejpoužívanější algoritmus pro získávání těchto pravidel je algoritmus Apriori, popř. jeho modifikace (podrobný popis tohoto algoritmu viz kapitola 3.6). V případě, že máme v systému mnoho položek, můžeme získat pouze malé množství asociačních pravidel. Abychom se tohoto vyvarovali, zavádíme pojem zobecněná (víceúrovňová) asociační pravidla. Například zboží, které si zákazník v supermarketu ukládá do košíku je součástí přirozené taxonomie. Obchod nabízí různé druhy nápojů, uzenin, hořčice apod. (obr. 3)



Obr. 3 Taxonomie sortimentu zboží

Takováto taxonomie se využívá při hledání zobecněných (víceúrovňových) asociačních pravidel. Tato pravidla zachycují asociace mezi položkami na různé úrovni obecnosti (granularity). Zajímají nás tedy nejen pravidla na nejnižší úrovni hierarchie, např. lahůdkový párek -> kremžská hořčice, ale i obecnější (a tedy kompaktnější a snad srozumitelnější) pravidla využívající této taxonomie, např. párek -> hořčice. Zobecněné asociační pravidlo je tedy pravidlo tvaru Ant -> Suc, kde $Ant \cap Suc=0$ a žádná položka v Suc není předchůdcem žádné položky v Ant vzhledem k uvažované hierarchii. Jinak se ale v pravidlech objevují položky (kategorie) z různých úrovní hierarchie.

Problém při hledání zobecněných pravidel je ve vhodné volbě minimální požadované podpory (support). Je-li hodnota tohoto parametru příliš vysoká, nenajdeme pravidla na nejnižší úrovni, není ani zaručeno, že nalezneme příslušné obecnější pravidlo (důvodem může být nízká spolehlivost takového pravidla). Je-li hodnota minimální požadované podpory příliš nízká, dojde ke kombinatorické explozi, navíc budou v pravidlech figurovat položky z nižších úrovní hierarchie ve všech možných vzájemných kombinacích. Platí totiž, že pokud má požadovanou podporu kombinace $Ant \wedge Suc$, budou ji mít i kombinace vytvořené z předchůdců Ant a Suc. Jednou z možností, jak se vypořádat s otázkou různé podpory na různých úrovních hierarchie, je dynamicky měnit minimální požadovanou podporu v závislosti na úrovni hierarchie dané kombinace. Využívá se přitom tzv. stupně obecnosti g, který je pro určitou hodnotu v hierarchii definován jako podíl počtu listových hodnot v podstromu uvažované hodnoty a počtu všech listových hodnot v dané hierarchii. Takto můžeme získat i pravidla, která pro nás nejsou zajímavá. Nezajímavá pravidla lze rozdělit na redundantní a nepodstatná. Redundantní asociační pravidlo je možno přímo odvodit z pravidla, které bylo již získáno na vyšší úrovni. To znamená, že všechny položky z pravidla už jsou na vyšší nebo stejné úrovni.

3.4 Jiné typy asociačních pravidel

3.4.1 Pravidla s výjimkami

Výsledkem algoritmů pro hledání asociačních pravidel bývá rozsáhlý seznam pravidel, které je nutné interpretovat. Vodítkem je obvykle nějakým způsobem definovaná zajímavost. V tomto případě považujeme za zajímavá ta pravidla, která se vymykají ustáleným běžným představám (tzv. Common sence).

Formálně je pravidlo s výjimkou definováno na základě trojice pravidel:

$$Comb_A \rightarrow Suc,\ Comb_A \wedge Comb_B \rightarrow \neg Suc,\ Comb_B \rightarrow \neg Suc,$$

kde první pravidlo odpovídá ustáleným představám (toto pravidlo má vysokou podporu i spolehlivost), druhé pravidlo je hledaná výjimka (toto pravidlo má nízkou podporu a vysokou spolehlivost) a třetí pravidlo je takzvané referenční pravidlo (má buď nízkou podporu, nebo nízkou spolehlivost, popř. obojí).

Příkladem pravidla s výjimkou může být následující trojice pravidel:

1. použité bezpečnostní pásy -> přežití automobilové havárie (obecně uznávané pravidlo o účinnosti bezpečnostních pásů)
2. použité bezpečnostní pásy ∧ věk = předškolní -> úmrtí při havárii (překvapivá výjimka, pro malé děti jsou bezpečnostní pásy nevhodné)
3. věk = předškolní -> úmrtí při havárii (referenční pravidlo, při haváriích umírá málo předškolních dětí)

3.4.2 Časové sekvence

Většinou se asociační pravidla hledají v databázích, kde se neuvažuje s faktorem času. Záznamy v databázi prostě zachycují charakteristiky nějakých objektů. Někdy se ale ocitneme v situaci, že potřebujeme nalézt asociace mezi událostmi, které se odehrávají v různých časových okamžicích. (např. databáze poruch v telekomunikační síti). Sekvence událostí, v nichž se hledají opakující se epizody, pak může mít tuto podobu :

(P, 123), (Q, 125), (S, 140), (P, 150), (R, 151), (Q, 155), (S, 201), (P, 220), (S, 222), (Q, 225),

kde v zápise (písmeno, číslo) značí písmeno název události a číslo je údaj o čase, ve kterém událost nastala.

3.4.3 Více tabulek

Většina algoritmů pro získávání znalostí z databází pracuje s jednou datovou tabulkou (maticí). Výjimkou nejsou ani algoritmy pro hledání asociačních pravidel. Přesto můžeme v literatuře nalézt přístupy, které tento problém řeší jinak než „prostým“ spojením všech relevantních tabulek do jedné v kroku předzpracování, tedy před vlastním výpočtem.

Příkladem může být postup, umožňující hledat kombinace s vysokou četností v datovém skladu, organizovaném do hvězdy. Hvězda obsahuje jednu centrální tabulku faktů, ze které se odkazuje pomocí tzv. cizích klíčů do tabulek jednotlivých dimenzí. Navržený způsob hledání četných kombinací pracuje ve dvou krocích: hledání četných kombinací v tabulkách dimenzí a spojení výsledků z jednotlivých tabulek dimenzí s využitím cizích klíčů v tabulce faktů.

3.5 Získávání asociačních pravidel

Obecně se dá říct, že získávání asociačních pravidel je dvoukrokový proces:

- V prvním kroku získáváme všechny tzv. frekventované množiny prvků, tj. množiny mající větší podporu, než je předem stanovená mez. Proces, ve kterém jsou získávány tyto množiny, je iterativní, nejprve se získávají všechny jednoprvkové množiny (vezmou se všechny položky, které se vyskytují v množině transakcí v požadovaném počtu), z nich se pak vytváří frekventované dvouprvkové množiny atd.
- Ve druhém kroku se z frekventovaných množin získávají silná asociační pravidla, tj. ta pravidla, která mají spolehlivost větší, než je stanovená mez. Proces spočívá v tom, že z frekventovaných množin vytvoříme všechna asociační pravidla a posléze z nich vybereme jen ta, která splňují podmínku minimální spolehlivosti a jsou tedy pro nás zajímavá. Tato pravidla se označí jako silná a jsou výsledkem tohoto algoritmu.

3.6 Algoritmus apriori

Nejznámějším algoritmem pro hledání asociačních pravidel je algoritmus apriori. Tento algoritmus navrhl R. Agrawal v souvislosti s analýzou nákupního košíku. Jádrem algoritmu je hledání často se opakujících množin položek (frequent itemsets). Jde o kombinace (konjunkce) kategorií (v případě analýzy nákupního košíku jsou kategorie typu máslo(ano), chléb(ano) apod.), které dosahují předem zadané hodnoty minimální podpory v datech. Algoritmus je založen na generování kombinací do šířky, pro kombinace, které vyhovují svou spolehlivostí (předem určená hodnota minimální spolehlivosti), se vytvářejí asociační pravidla. Následuje stručný popis algoritmu:

1. do L_1 přiřaď všechny kategorie, které dosahují alespoň požadované četnosti
2. polož k = 2
3. dokud L_{k-1} != 0
    3.1. pomocí fce apriori-gen vygeneruj na základě L_{k-1} množinu kandidátů C_k
    3.2. do L_k zařaď ty kombinace z C_k, které dosáhly požadovanou četnost
    3.3. zvětši počítadlo k

kde C_k je kandidátská množina položek velikosti k a L_k frekventovaná množina položek

Funkce apriori-gen(L_{k-1}):

1. pro všechny dvojice kombinací $Comb_p$, $Comb_q$ z L_{k-1}
    1.1. pokud $Comb_p$ a $Comb_q$ se shodují v k-2 kategoriích, přidej $Comb_p \wedge Comb_q$ do C_k
2. pro každou kombinaci Com_b z C_k
    2.1. pokud některá z jejich podkombinací délky k-1 není obsažena v L_{k-1}, odstraň Com_b z C_k

4 Standard SQL/MM DM

4.1 Standard SQL/MM

První zmínky o standardu SQL jakožto programovacím jazyku nad databázemi sahají až do 80. let minulého století. V roce 1986 vznikl standard ISO/SQL – 86, v němž byla zřetelná dominantní úloha dialektu SQL firmy IBM (DB2). Další výrazným posunem ve vývoji byl standard SQL/92 z počátku 90. let, který se již vyznačoval třemi úrovněmi souladu (Entry/Intermediate/Full). SQL1999 vyznačující se objektově orientovanými rysy, se stal dalším krokem. V současnosti vývoj dospěl až k SQL2003, jenž podporuje OLAP i XML. Bouřlivý vývoj jazyka měl za následek fakt, že se vyvinula řada dialektů jazyka, jejichž společným základem je ale SQL/92 (minimálně úroveň Entry).

SQL/MM (SQL Multimedia and Application Packages) je ISO/IEC mezinárodní normalizační projekt. Jako SQL, tak i SQL/MM je standard složený z mnoha částí s tím rozdílem, že různé části jsou na sobě poměrně nezávislé. Aby tyto části spolu mohly efektivně komunikovat, je implementována první část standardu, známá jako tzv. rámec (framework). Poskytuje definice běžných konceptů užívaných v dalších částech. Zvláště popisuje možnosti, jejichž prostřednictvím ostatní části využívají strukturované uživatelsky definované typy k definici typů té dané části. Ostatní části jsou využívány např. k interpretaci full-textových dat, prostorových souřadnic, statického obrazu či dolování dat. Ne všechny části SQL/MM jsou již komerčně úspěšnými, podstatné ale je, že poskytují určitý obecný rámec pro práci s touto problematikou.

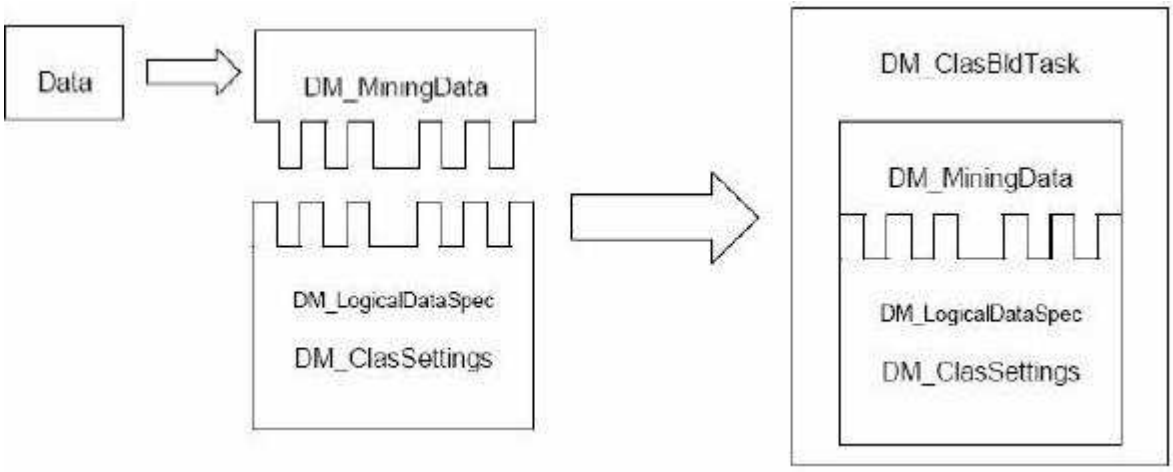
4.2 Standard SQL/MM DM

Problematice dolování dat je věnována šestá část standardu SQL/MM [3]. Motivací vzniku této části byl fakt, že databázové systémy by měly být schopny integrovat aplikace pro dolování z dat standardně tak, aby umožnily koncovému uživateli vykonávat toto dolování snadně. Dolování z dat se stalo součástí moderního řízení dat i přirozeným rozšířením původní funkčnosti poskytované SQL. Z těchto důvodů se stalo nutností integrovat modul pro dolování z dat do SQL/MM.

SQL/MM DM (Data Mining) poskytuje API pro data mining aplikace k přístupu dat z SQL/MM pomocí relačních databází. To definuje strukturovaného uživatele – definované typy včetně přidružených metod pro podporu dolování dat. Uživatelské typy lze rozdělit do pěti oblastí – typy specifikující dolovací data, typy pro vytváření dolovacích modelů, typy pro nastavení různých atributů dané dolovací techniky, typy pro aplikaci modelu na vstupní data a typy pro testování výsledků. Tato část standardu poskytuje standardizované rozhraní pro dolovací algoritmy, které mohou být vrstveny na objektově relačním databázovém systému. Asociační pravidla, shlukování, klasifikace a regrese jsou typickými dolovacími úlohami, které standard podporuje.

4.3 Uživatelem definované typy

Standard SQL/MM DM je souborem definic uživatelem definovaných typů pro dolování z dat, je v něm popsáno rozhraní a chování metod jednotlivých typů. Typy obsažené v šesté části standardu se týkají pouze dolování z dat, nikoliv jiných fází dolování znalostí. Příkladem použití typů jsou např. datové sklady. Použití datových skladů typicky potřebuje flexibilně vykonat různé úlohy na různých datových množinách. Z toho důvodu je ve standardu navrženo mnoho typů, které mezi sebou vytváří různé souvislosti a vztahy.



Obr. 4 Příklad vztahů mezi uživatelskými typy

Obrázek č. 4 ukazuje typy definované standardem pro určení všech informací potřebných k trénování (trénovací fáze u klasifikace je vybrána jako příklad spolupráce (vytváření vztahů) mezi jednotlivými typy). Cílem spolupráce je definování úlohy dolování obsahující všechny potřebné informace k trénování klasifikačního modelu, tato začíná ze strany dat, nebo ze strany nastavení. Jestliže prvním krokem je definování nastavení pro dolování z dat, pak je to umožněno množnou polí – logická datová specifikace. Dále je stanovena reprezentace reálných dat (*DM_MiningData*), která doplňuje definici objektu (*DM_ClusBldTask*). Pokud se začne definicí reprezentace reálných dat, může být logická datová specifikace jednoduše odvozena od těchto dat, v nichž se doluje, a použita k definici nastavení dolování z dat. Na závěr se nastavení dolování z dat spolu s logickou datovou specifikací a reprezentace reálných dat připojí k objektu úlohy klasifikace.

4.3.1 Typy pro přípravu dat

Pod pojmem typy pro přípravu dat míním uživatelské typy standardu, které nesouvisejí pouze s metodami pro získávání asociačních pravidel [2]. Tyto typy neposkytují žádnou metodu, která by umožňovala získat jakoukoliv informaci týkající se dolování. Mohou být použity jen pro definování metadat potřebných pro pozdější zpracování, reprezentují informace reálného světa. Jedná se o typy:

4.3.1.1 Typ DM_LogicalDataSpec

Typ je abstrakcí pro množinu dolovacích polí identifikovaných pomocí jejich jména. Každé z polí obsahuje také přidružený logický typ pole (0 – *DM_Categorical*, 1 – *DM_Numerical*). Typ *DM_LogicalDataSpec* je úvodem k reprezentaci vstupních dat potřebných při trénovacím, testovacím nebo aplikačním běhu. Definuje tyto metody:

a) metoda *DM_addDataSpecFld (CHARACTER VARYING (DM_MaxFieldAliasLength))*

Úkolem této metody je vložení další položky pole do objektu *SELF*. Vstupní parametr obsahuje název pole. Pokud je hodnota parametru shodná s nějakou hodnotou pole obsaženou v objektu *SELF*, pak je vyvolána výjimka *SQL/MM Data Mining exception – field already defined.*

b) metoda *DM_remDataSpecFld (CHARACTER VARYING (DM_MaxFieldAliasLength))*

Tato metoda umožní odstranit pole shodné s hodnotou obsaženou v parametru metody. Pokud tento parametr obsahuje hodnotu, které není ekvivalentní s žádnou hodnotou v objektu *SELF*, je vyvolána výjimka *SQL/MM Data Mining exception – field not defined in data specification.*

c) metoda *DM_getNumFields ()*

Metoda navrací počet polí obsažených v objektu *SELF*.

d) metoda *DM_getFldName (INTEGER)*

Vrací pole obsažené v objektu *SELF* na pozici, kterou udává vstupní parametr typu integer. Pokud je tento parametr záporné hodnoty, nebo větší než hodnota vrácená funkcí *DM_getNumFields*, pak je generována výjimka *SQL/MM Data Mining exception – parametr out of range.*

e) metoda *DM_setFldType (CHARACTER VARYING (DM_MaxFieldAliasLength), SMALLINT)*

Účelem této metody je nastavení typu pole specifického jména. Pokud není hodnota prvního parametru ekvivalentní s nějakým jménem pole obsaženého v objektu *SELF*, je vyvolána výjimka *SQL/MM Data Mining exception – field not defined in data specification.*

f) metoda *DM_getFldType (CHARACTER VARYING (DM_MaxFieldAliasLength))*

Tato metoda vrací typ pole specifického jména uvedeného jako parametr metody. Jestliže hodnota vstupního parametru neodpovídá žádnému jménu pole v objektu *SELF*, je vyvolána výjimka *SQL/MM Data Mining exception – field not defined in data specification.*

g) metoda *DM_isCompatible (DM_LogicalDataSpec)*

Úkolem metody je zjistit, zda je kompatibilní hodnota vstupního parametru s objektem *SELF*. Pokud jsou všechny položky objektu *SELF* obsaženy v hodnotě vstupního parametru, pak vrací hodnotu 1 (tedy true), v opačném případě 0.

4.3.1.2 Typ DM_MiningData

Tento typ představuje abstrakci pro data reálného světa, která jsou uložena v tabulce nebo pohledu. Hodnota tohoto typu reprezentuje metadata pro přistup k reálným tabulkám pro pozdější trénování, testovaní nebo aplikaci. Standardem jsou popsány následující metody:

a) metoda *DM_defMiningData (CHARACTER VARYING (DM_MaxTableNameLength))*

Tato metoda je konstruktor typu. Navrací hodnotu typu *DM_MiningData* korespondující se zdrojovou tabulkou určenou jménem, které je zadáno parametrem metody. Pokud jméno neoznačuje platnou tabulku, je vyvolána výjimka *SQL/MM Data Mining exception – invalid table name.*

b) metoda *DM_setFldAlias (CHARACTER VARYING (DM_MaxFieldNameLength), CHARACTER VARYING (DM_MaxFieldNameLength))*

Metoda nastavuje alias pole obsaženého v objektu *SELF*. V jejím rámci je prováděna kontrola vstupních parametrů, na základě výsledku kontroly jsou pak metodou vyvolány patřičné výjimky. U prvního parametru se kontroluje, zda je jeho hodnota ekvivalentní s názvem pole obsaženém v objektu *SELF*. Pokud tomu tak není, je metodou vyvolána výjimka *SQL/MM Data Mining exception – invalid field name.* Dále se kontroluje, zda hodnota druhého parametru není rovna NULL, a zda je shodná s jménem nebo aliasem pole obsaženém s objektu *SELF*. Pokud tomu tak je, pak je vyvolána výjimka *SQL/MM Data Mining exception – alias already in use.*

c) metoda *DM_genDataSpec ()*

Tato metoda má za úkol vytvořit objekt typu *DM_LogicalDataSpec*.

4.3.2 Typy pro dolování asociačních pravidel

Pro každou z dolovacích technik existuje alespoň jeden definovaný typ. V této kapitole jsou uvedeny ty, které souvisejí s asociačními pravidly.

4.3.2.1 Typ DM_RuleModel

Tento typ reprezentuje modely, které jsou výsledkem hledání asociačních pravidel. Pro veřejné užití poskytuje následující rozhraní:

a) metoda *DM_impRuleModel (CHARACTER LARGE OBJECT(DM_MaxContentLength))*

Tato metoda je konstruktorem typu *DM_RuleModel*. Importuje model asociačních pravidel vytvořený podle standardu PMML (Predictive Model Markup Language), který je uložen v jediném vstupním parametru typu CLOB (Character Large Object). Pokud vstupní proměnná neobsahuje dokument XML (vytvořený podle PMML), pak hodnota typu *DM_RuleModel* nabývá NULL.

b) metoda *DM_expRuleModel ()*

Druhá metoda typu exportuje model do dokumentu XML vytvořeného podle standardu PMML Association Rules DTD (Document Type Description).

c) metoda *DM_getNumRules ()*

Vrací počet asociačních pravidel obsažených v importovaném modelu.

d) metoda *DM_getRuleBldTask ()*

Poslední metoda vrací hodnotu typu *DM_RuleBldTask*, která je použita k vytvoření modelu pro asociační pravidla.

4.3.2.2 Typ DM_RuleSettings

Tento typ popisuje nastavení, které je využito při generování modelu asociačních pravidel. Obsahuje následující metody:

a) metoda *DM_impRuleSettings (CHARACTER LARGE OBJECT (DM_MaxContentLength))*

První metoda typu je opět jejím konstruktorem. Má jeden vstupní parametr typu CLOB, ze kterého se provádí import nastavení modelu asociačních pravidel. Pokud tento parametr neobsahuje vyhovující reprezentaci typu *DM_RuleSettings*, je standardem předepsáno vyvolání výjimky *SQL/MM Data Mining exception – invalid import format*.

b) metoda *DM_expRuleSettings ()*

Druhá metoda typu exportuje hodnotu typu CLOB reprezentující nastavení modelu asociačních pravidel, který je obsažený v objektu *SELF*. Tato reprezentace je závislá na implementaci.

c) metoda *DM_setMinSupport (DOUBLE PRECISION)*

Tato metoda nastavuje hodnotu minimální podpory asociačních pravidel v hodnotě nastavení (*DM_RuleSettings*). Pokud je podpora záporná, nebo větší než sto, je voláno hlášení *SQL/MM Data Mining exception – parametr out of range*.

d) metoda *DM_getMinSupport ()*

Vrací hodnotu minimální podpory, která byla specifikována jako parametr hledání asociačních pravidel. Muže dosahovat dvou typů hodnot. V případě, že se *SELF* rovná volání *SELF.DM_setMinSupport(v)* pro nějakou hodnotu v (typu DOUBLE PRECISION), pak v. V opačném případě nabývá hodnoty NULL, což indikuje, že numerická hodnota minimální podpory bude determinována až v data mining běhu.

e) metoda *DM_setMinConf (DOUBLE PRECISION)*

Tato metoda nastavuje hodnotu minimální spolehlivosti asociačních pravidel pro objekt typu *SELF*. Pokud je spolehlivost záporná, nebo větší než sto, je voláno hlášení *SQL/MM Data Mining exception – parametr out of range*.

f) metoda *DM_getMinConf ()*

Vrací hodnotu minimální spolehlivosti pro asociační pravidla objektu *SELF*. Může dosahovat stejných typů hodnot, jako při volání *DM_getMinSupport()*.

g) metoda *DM_setMaxLength (INTEGER)*

Specifikuje maximální délku asociačního pravidla. Pokud je její hodnota menší než dva, je vyvolána výjimka *SQL/MM Data Mining exception – parametr out of range*. V opačném případě vrací metoda hodnotu objektu, pro který byla volána s tím, že se někam uloží hodnota délky pravidla. Tu pak vrací metoda *DM_getMaxLength()*.

h) metoda *DM_getMaxLength ()*

Metoda vrací maximální délku pravidla. Hodnota NULL v tomto případě znamená, že délka asociačního pravidla není omezena.

i) metoda *DM_useRuleDataSpec (DM_LogicalDataSpec)*

Tato metoda specifikuje vstupní hodnotu typu *DM_LogicalDataSpec* pro objekt *SELF* (resp. hodnotu *DM_RuleSettings*). Hodnota *DM_LogicalDataSpec* určuje platné hodnoty *DM_MiningData*, která mohou být zpracovány v trénovaní fázi.

j) metoda *DM_getRuleDataSpec ()*

Vrací logickou datovou specifikaci *DM_LogicalDataSpec* definovanou pro nastavení asociačních pravidel.

k) metoda *DM_setGroup (CHARACTER VARYING (DM_MaxFieldAliasLength))*

Metoda specifikuje pole použité k identifikaci skupiny (např. nákupní transakce) pro dolování asociačních pravidel. Typ vstupního parametru představuje maximální délku řetězců reprezentujících jméno pole, kromě jejich logické datové specifikace. Pokud je hodnota *groupField* (vstupni hodnota) rovna NULL, pak hodnota typu *DM_RuleSettings* neobsahuje tuto položku. V případě, že není v *SELF* obsažena žádná logická datová specifikace, je vyvolána výjimka *SQL/MM Data Mining exception – no logica data specification defined*. Jestliže není obsaženo pole s názvem *groupField* v logické datové specifikaci *SELF*, je generována *SQL/MM Data Mining exception – field not defined in data specification*. Další excepce, *SQL/MM Data Mining exception – field not categorical*, je volána tehdy, když je předešlé splněno, typ pole pro *groupField* je definován, ale nejedná se o definici kategorického typu. Jsou-li všechny výše uvedené podmínky splněny, obsahuje hodnota typu *DM_RuleSettings* položku s názvem *groupField*, určenou k seskupování.

l) metoda *DM_getGroup ()*

Vrací jméno pole skupiny definované pro nastavení pravidel.

4.3.2.3 Typ DM_RuleBldTask

Tento typ reprezentuje informaci o úloze hledání asociačních pravidel, především pak o vstupních datech a nastavení parametrů. Poskytuje metodu pro výpočet (vytvoření) modelu asociačních pravidel.

a) metoda *DM_defRuleBldTask (DM_MiningData, DM_RuleSettings)*

   Jedná se o konstruktor typu *DM_RuleBldTask*. Objekt tohoto typu je pak dán hodnotami vstupních parametrů typu *DM_MiningData* a *DM_RuleSettings*).

b) metoda *DM_getRuleTrnData ()*

   Vrací hodnotu typu *DM_MiningData*, která reprezentuje trénovací data pro úlohu asociačních pravidel (podle dané hodnoty typu *DM_RuleBldTask*)

c) metoda *DM_getRuleSettings ()*

   Metoda vrací hodnotu typu *DM_RuleSettings*, jež reprezentuje nastavení úlohy hledání asociačních pravidel.

d) metoda *DM_buildRuleModel ()*

   Tato metoda na výstupu vrací specifikovanou hodnotu typu *DM_RuleModel*. Tato nabývá hodnoty NULL, pokud *SELF* neobsahuje platnou reprezentaci typů *DM_RuleSettings* a *DM_MiningData*. V případě, že volání metody *DM_buildRuleModel()* je úspěšné, potom je hodnota typu *DM_RuleModel* určena informací obsaženou v *SELF*. Jestliže provedení této akce úspěšné není, vyvolá se výjimka *SQL/MM Data Mining exception – model computation failed*.

5 Aplikace v prostředí Oracle

Při práci s objekty pomocí vývojových prostředí typu Microsoft Visual Studio .NET může dojít k problémům způsobeným samotnou prací s objektovou databází, jelikož pro práci s objekty nejsou ještě úplně definované standardizované metody. V souladu se zadáním diplomového projektu je proto efektivnější při vytváření aplikace použít nástroj pro vytváření uživatelského rozhraní Oracle Forms [8]. Integraci s prostředím Oracle podporuje také vybraný programovací jazyk PL/SQL.

S vydáním databáze Oracle8 (nyní je aktuální verze 10.2) se umožnilo využití objektů (uživatelem definovaných typů) v databázi. To byl důležitý průlom, protože vznikla možnost definovat vlastní třídy, známé jako typy objektu, v databázi. Třídy v systému Oracle, stejně jako v objektových programovacích jazycích, mohou obsahovat jak metody, tak atributy. Typ objektu se vytváří za pomoci příkazu *CREATE [OR REPLACE] TYPE*. Pokud jsou pro daný typ objektu definovány metody, pak se jejich chování definuje pomocí příkazu *CREATE TYPE BODY*. Metody můžeme rozlišit na statické a členské. Ke statickým metodám lze přistoupit přímo, např. *create_object()*, kdežto k členským pouze přes objekt, např. *object1.create_item()*. Jako u klasických objektů, tak i u objektů v databázi Oracle lze použít typovou dědičnost (od verze Oracle 9i), což dovoluje vytvářet hierarchii typů. Pokud je tedy požadováno, aby mohl být určitý typ děděný, musí být použita klauzule *NOT FINAL*. Další vlastnost, která může být buď povolena, nebo zakázána, je možnost vytváření objektu daného typu. Pokud se u definice typu objektu užije klauzule *NOT INSTANTIABLE*, nemůže být vytvořen objekt tohoto typu. Tento typ objektu je pak použit pouze jako abstraktní typ. Důležité je zmínit možnost vytvoření vlastního konstruktoru k inicializaci atributů objektu. Konstruktor je funkcí typu objektu, která je deklarována klíčovými slovy *CONSTRUCTOR FUNCTION*. Typy objektu s výše definovanými vlastnostmi se využívají k definici sloupců v relačních nebo objektových tabulkách. Dále je možnost využít unikátního identifikátoru a následně se odkazovat na objekty v jiných tabulkách.

5.1 Jazyk PL/SQL

PL/SQL je rozšířením jazyka SQL o konstrukční tvary programovacích jazyků [9]. Manipulace s daty a dotazovací prvky jsou obsaženy uvnitř procedurálních jednotek kódu. PL/SQL je vysoce výkonný jazyk na provádění transakcí, který nabízí vyšší produktivitu, výkon, či integraci s Oracle.

SQL se stal standardním databázovým jazykem, protože je flexibilní, mocný, a snadno naučitelný. Několik příkazů, jako je *SELECT*, *INSERT*, *UPDATE* nebo *DELETE*, umožňuje snadnou manipulaci s daty v databázi.

PL/SQL je jazyk s blokovou strukturou. To znamená, že základní jednotky (procedury, funkce, nepojmenované bloky), které tvoří program, jsou logickými bloky, které mohou obsahovat libovolný

počet vnořených bloků. Deklarace jsou lokální vzhledem k bloku, a nejsou mimo něj dostupné. Blok má tři části: deklarační, část s příkazy a část s obsluhou výjimek. PL/SQL rozšiřuje možnosti nástrojů jako je Oracle Forms a Oracle Reports. PL/SQL blok lze například použít v triggerech v Oracle Forms a zvýšit tak produktivitu práce. Bez PL/SQL musí Oracle zpracovávat dotazy jeden po druhém. Každý SQL příkaz způsobuje nový dotaz na server, což zatěžuje síť. Při použití PL/SQL lze naráz poslat na server celý PL/SQL blok, a provést tak naráz několik SQL příkazů a velmi tak snížit zatížení sítě. Aplikace napsané v PL/SQL jsou portabilní na libovolný systém, na kterém běží Oracle, není je třeba upravovat pro nová prostředí. Lze psát portabilní PL/SQL knihovny, které jsou použitelné v různých prostředích. Integrací s Oracle je míněna podpora atributů jako *%TYPE* a *%ROWTYPE*.

Předdefinovanými datovými typy v PL/SQL jsou tyto: binary_integer, pls_integer, number, natural, naturaln, positive, positiven, dec, decimal, double precision, integer, int, numeric, real, smallint, float, char, character, long, raw, long raw, rowid, varchar2, string, varchar, boolean, date, mlslabel.

6 Oracle Data Mining

Oracle Data Mining (ODM) představuje podporu databáze Oracle umožňující do ní zahrnout techniky datového dolování. Není tedy nutné přenášet data k analýze mimo databázi, což značně zefektivňuje práci. ODM podporuje rozhraní jazyků Java a PL/SQL (v dalším textu se budu věnovat zejména podpoře pro jazyk PL/SQL z důvodu implementace diplomové práce právě v tomto jazyce) a následující dolovací techniky a jejich algoritmy:

- Důležitost atributů – identifikace nejvlivnějších atributů pro výsledek predikce (např. podle ceny)
- Klasifikace a predikce – ohodnocení a rozdělování atributů do oddělených tříd a predikce komu náleží
- Regrese – funkční aproximace a predikce proměnlivých (obvykle numerických) hodnot
- Shlukování – hledání přirozeného třídění dat do skupin
- Asociační pravidla – hledání zároveň se vyskytujících položek nákupního košíku (navržení kombinací výrobků nebo lepšího umístění v regálech)
- Extrakce rysů – redukce rozsáhlé neuspořádané datové množiny do reprezentace nových atributů (vytváření nových atributů kombinací původních)
- Text mining – kombinování dat a textu ke zlepšení modelů, klasifikace a shlukování dokumentů

ODM užívá data, které se skládají z tabulek uložených v databázi. Řádky datové tabulky představují případy, záznamy nebo příklady, sloupce pak jednotlivé atributy. Atributy mohou být dvou typů, a to numerického nebo kategorického. Každý atribut udržuje nějakou informaci. Dále jsou definovány určité podmínky pro vstupní data – formát datové tabulky, typ dat sloupců a typ atributů (viz výše). ODM nepodporuje všechny datové typy jako Oracle, datové typy sloupců tak musejí být jedním z těchto: NUMBER, CHAR, VARCHAR2, BFILE, CLOB, XMLTYPE, URITAPE. Rozlišujeme pak data připravená nebo nepřipravená. O připravených datech hovoříme tehdy, pokud jsou uživatelem provedeny určité datové transformace požadované dolovacím algoritmem (data pro dolování dat pomocí PL/SQL musí být připravena). Poslední nutnou podmínkou ODM je formát dat tabulky. ODM data mohou být jedním z následujících formátů:

- Případ jako jednoduchý záznam – každý případ je uložen jako jeden řádek, obvykle se používá v relačních databázích. K identifikaci každého záznamu není nutný unikátní klíč, který je ale zapotřebí ke spojení případů s výsledky trénování.
- Případ jako vícenásobný záznam – každý případ může obsahovat více položek, obvykle se používá pro data s mnoha atributy. Oracle databáze podporuje existenci maximálně 1000 sloupců (atributů). Každý případ je rozdělen do několika řádků a uložen v tabulce se sloupci

sequence ID, *atribute name* a *value*. Takovýto formát také označujeme jako transakční, transakce T o N položkách je popsána N řádky tabulky se stejnou hodnotou ID.

6.1 ODM a asociační pravidla

Jak bylo zmíněno v kapitole 3, existuje několik charakteristik asociačních pravidel. ODM počítá dvě základní z této množiny, tedy podporu (počet objektů splňujících předpoklad i závěr) a spolehlivost (podmíněná pravděpodobnost závěru, pokud platí předpoklad). ODM využívá právě tato statistická měřítka k seřazení pravidel a potažmo predikci.

Asociační modely jsou navrženy k využívání řídkých (sparse) dat. Pod pojmem řídká data rozumíme taková data, která obsahují pouze malý počet atributů s nespecifikovanou hodnotou v jakémkoliv řádku tabulky. Příkladem může být problém nákupního košíku. V daném obchodě existuje asi tisíc výrobků, ale průměrný nákup obsahuje pouze okolo dvaceti položek. To znamená, že záznam transakce má pouze 20 atributů, které nenabývají hodnoty null. Z celkového počtu 1000 atributů to představuje hustotu 2%. Tato hustota je typická pro problémy nákupního košíku nebo zpracování textu. Data s významně větší hustotou vyžadují velmi velký prostor pro hledání asociací. Asociační modely zpracovávají hodnoty null jako řídká data, chybějící hodnoty algoritmus nezpracovává. Přítomnost outlierů (bludných kamenů, tedy hodnot hodně vzdálených od normálního rozsahu v datové množině) při generování do šířky způsobuje koncentraci většiny dat pouze v několika množinách (extrémním případem je existence pouze jedné množiny). Výsledkem pak může být významně omezená schopnost modelu detekovat rozdíly v numerických atributech. Například všechny numerické atributy pro výši důchodu mohou patřit do jedné množiny (bin), kromě jedné hodnoty (outliner), která je v množině jiné. Výsledkem pak je, že by tady nebyla žádná pravidla reflektující rozdílnou úroveň důchodů. Všechna pravidla obsahující důchody budou pouze reflektovat rozsah první množiny, který odpovídá důchodu celé populace. V těchto případech je nutné použít ostrou (clipping) transformaci pro práci s outlinery.

6.1.1 ODM a algoritmus apriori

ODM využívá pro práci s asociačními pravidly algoritmus apriori (viz kapitola 3.6). Problém dolování asociačních pravidel je tak rozdělen na dva podproblémy:

- Nalezení všech kombinací položek (častých vzorů, frekventovaných množin položek, frequent itemsets), jejichž podpora je větší než určená mez minimální podpory
- Využití frekventovaných množin pro generování požadovaných pravidel. ODM podporuje pouze pravidla s jednou implikací.

Počet frekventovaných množin je kontrolován pomocí parametru minimální podpory, počet vygenerovaných pravidel pomocí počtu frekventovaných množin položek a parametru spolehlivosti.

Pokud je spolehlivost nastavena na příliš vysokou hodnotu, mohou se v modelu vyskytovat frekventované množiny, ale už ne asociační pravidla.

Algoritmus apriori nedokáže efektivně zpracovávat tyto případy:

- Hledání asociací zahrnující vzácné události
- Hledání asociací v hustých datových množinách s velkým počtem atributů
- Spojitá data

Aby byl algoritmus schopen hledat asociace zahrnující vzácné události, musí být spuštěn s velmi malou hodnotou podpory. To by ale znamenalo potencionální zničení některých vygenerovaných množin položek, speciálně v případech s velkým množstvím položek, což může značně prodloužit čas provedení algoritmu. Proto metoda dolování asociačních pravidel není doporučována pro hledání asociací obsahujících vzácné události při velkém počtu položek. Tento problém odpadá použitím např. klasifikačních modelů.

6.1.2 ODM moduly DBMS_DATA_MINING_TRANSFORM a DBMS_DATA_MINING

Oracle poskytuje mnoho PL/SQL programových balíčků pro rozšíření možností práce s databází. Pod pojmem balíček rozumíme kolekci souvisejících objektů společně uložených v databázi. Těmito objekty jsou zejména procedury, funkce, proměnné, konstanty, kurzory a výjimky. Při vytváření databázových aplikací tak může programátor využít řady předimplementovaných modulů, což značně zefektivňuje jeho práci. Pro realizaci typů definovaných standardem SQL/MM DM pro asociační pravidla jsou vhodné moduly DBMS_DATA_MINING_TRANSFORM a DBMS_DATA_MINING. První z nich poskytuje množství utilit pro přípravu dolovacích dat tak, aby je druhý modul mohl korektně využít pro samotné dolování dat.

6.1.2.1 Modul DBMS_DATA_MINING_TRANSFORM

Tento balíček obsahuje řadu utilit pro datovou transformaci, které připraví data pro dolování z nich. Jakmile jsou takto data připravena, je možné je využít pro vytváření dolovacích modelů a zacházení s nimi. DBMS_DATA_MINING_TRANSFORM je open-source PL/SQL balíček, což umožňuje nejenom využívání předprogramovaných rutin ale i vyvíjení vlastních na základě public source licence.

Hlavním důvodem, který stál za návrhem tohoto modulu, je fakt, že SQL je dostatečně silný nástroj na to, aby dokázal efektivně provádět většinu běžných dolovacích transformací. Nicméně SQL dotazy vykonávající transformace mohou být velmi dlouhé, proto je žádoucí mít vhodné rutiny, které s tímto dokážou pomoci. Cílem tohoto modulu je tedy poskytnutí odpovídajícího servisu pro generování dotazů týkajících se většiny běžných dolovacích transformací, stejně jako připravení rámce, který může být jednoduše rozšířen pro implementaci ostatních transformací.

DBMS_DATA_MINING_TRANSFORM podporuje tyto transformační metody: normalizace, transformace chybějí hodnoty, binning, winsorizing a clipping. V balíčku je implementováno celkem 23 transformačních podprogramů. Z důvodu transparentnosti je definován pouze jeden datový typ, *Column_List* (datový typ varray(1000) of varchar2(32)). Jedná se o seznam názvů sloupců reprezentujících dolovací atributy. Pro definování datové transformace, je třeba následovat tyto kroky:

1. Operace CREATE – slouží k vytvoření tabulky transformačních definic s přednastavenou množinou sloupců
2. Operace INSERT – naplní vytvořenou tabulku definicemi podle zvolených atributů
3. Operace XFORM – vytvoří náhled na tabulku

6.1.2.2 Modul DBMS_DATA_MINING

Tento balíček je jakýmsi rozhraním pro ODM, jeho prostřednictvím je možno navrhnout dolovací model, testovat jej a aplikovat na data v databázi. Modul vytváří dolovací model pro dolovací techniky užívající specifikovaný dolovací algoritmus. Algoritmus může být ovlivněný užitím jeho odpovídajícího nastavení. Vytvoření modelu je v rozhraní PL/SQL synchronní. Poté, co je vytvořený model vytrénován, lze k němu přistupovat jak „single-user“, tak „multi-session“. Jméno ODM modelu musí splňovat některé požadavky. Délka jeho názvu nesmí překročit 25 znaků, podporovanými znaky jsou alfanumerické, podtržítko (_), znak dollaru ($) a #. Stejně jako tabulka v databázi, i model má své úložiště. Podoba a tvar skladiště jsou uživateli nepřístupné, může ale nahlížet na obsah modelu.

Pokud jde o konstanty, tyto používá ODM ke specifikaci dolovací techniky, jejího algoritmu a dalších detailů asociačního modelu. DBMS_DATA_MINING podporuje klasifikaci, regresi, asociaci, shlukování a extrakci rysů. Technika dolování je specifikována parametrem procedury CREATE_MODEL a každá může být implementována jedním či více algoritmy. Pro každou dolovací techniku je specifikován standardní algoritmus (viz výše, pro asociační pravidla je to algoritmus apriori), ale lze jej přetížit prostřednictvím explicitní tabulky nastavení. Každý algoritmus má jedno nebo více nastavení, které ovlivňuje způsob, jak bude model vytvořen, je možné využít i standardní nastavení.

Tabulka nastavení je jednoduchá relační tabulka s jasnou strukturou, musí obsahovat dva sloupce tohoto tvaru: *setting_name* varchar2(30) *setting_value* varchar2(30). Hodnota uvedená v této tabulce přetěžuje standardní hodnotu algoritmu. Hodnoty v prvním sloupci představují jednu nebo více konstant definovaných v modulu DBMS_DATA_MINING, jedná se o *asso_max_rule_length* (nastavení maximální délky pravidla využité v asociačním modelu), *asso_min_confidence* (minimální spolehlivost pro asociační model) a *asso_min_support* (minimální

podpora). Každá hodnota druhého sloupce je předdefinovaná konstanta nebo platná numerická hodnota korespondující s vlastním nastavením.

Modul dále obsahuje řadu funkcí, které vrací specifické informace o modelu pro zvolenou dolovací techniku. Tyto tabulkové funkce přijímají na vstupu jméno dolovacího modelu a na výstup posílají požadované informace jako kolekci řádků tabulky. Všechny mají jednotné názvosloví – GET_n, kde n identifikuje typ hledané informace. Navíc využívají pipelining, což umožňuje, že každý volaný řádek výstupu je čten ze skladiště modelu, aniž by se muselo čekat na generování celé tabulky. Virtuální tabulka vrácená GET funkcemi je objektem, jiný objekt definuje řádky a některé sloupce tabulky nabývají též objektového datového typu, který definuje vložené tabulky.

V balíčku je definováno 23 podprogramů. Z hlediska této diplomové práce jsou zajímavé zejména tyto:

- Procedura CREATE_MODEL – vytvoří dolovací model
- Procedura DROP_MODEL – smaže dolovací model
- Procedury EXPORT_MODEL a IMPORT_MODEL – přesunutí modelu z jednoho schématu do jiného nebo z jedné databázové instance do další
- Funkce GET_ASSOCIATION_RULES – vrací pravidla z asociačního modelu
- Funkce GET_DEFAULT_SETTINGS – vrací standardní nastavení všech dolovacích metod a algoritmů
- Funkce GET_FREQUENT_ITEMSETS – vrací časté vzory z asociačního modelu
- Funkce GET_MODEL_SETTINGS – vrací nastavení použité pro vytvoření dolovacího modelu
- Funkce GET_MODEL_SIGNATURE – vrací informace o atributech použitých k vytvoření dolovacího modelu
- Procedura RENAME_MODEL – přejmenuje model
- Funkce GET_MODEL_DETAILS – vrací obsah modelu (podoba a tvar jeho skladiště je uživateli neprůhledný)

6.1.2.3 Dolování dat

Jak již bylo zmíněno výše, ODM dolovací úlohy v PL/SQL rozhraní zahrnují vytvoření dolovacího modelu, jeho testování a následnou aplikaci na data v databázi. Vývojová technologie pro dolování z dat používající moduly DM_DATA_MINING_TRANSFORM a DM_DATA_MINING má dvě fáze. Nejdříve se provádí analýza problému a dat, poté se přechází k vývoji dolovací aplikace (v našem případě asociační). Vytváření probíhá v těchto krocích:

1. Příprava trénovacích a testovacích dat prostřednictvím rutin modulu DBMS_DATA_MINING_TRANSFORM a PL/SQL (případně samostatného SQL) tak, jak požaduje vybraná dolovací technika a algoritmus. Pokud je již vytvořen prediktivní model, připraví se testovací množina dat. Aby byly

výsledky dolování smysluplné, je podstatné, aby trénovací, testovací a vyhodnocovací datové množiny byly připraveny identickým způsobem. V této práci se předpokládá, že dolovací data jsou již připravena, proto tato část není implementována.

2. Příprava tabulky nastavení, která přetěžuje původní nastavení dolovacího algoritmu. To znamená, že místo původního nastavení pracuje algoritmus s hodnotami uloženými v tabulce nastavení. Tento krok je ale v zásadě nepovinný, pokud je programátor se standardním nastavením spokojen. V našem případě je ale dolovací model vytvářen i pomocí tabulky nastavení *Settings_table*, která obsahuje tři řádky udávající hodnoty minimální podpory, minimální spolehlivosti a maximální délky asociačních pravidel.
3. Vytvoření dolovacího modelu (konkrétně pro asociační pravidla viz kapitola 6.1.2.4 – procedura CREATE_MODEL).
4. V případě prediktivního modelu (klasifikace, regrese) provádění jeho testování na přesnost a další atributy. Testování probíhá jeho aplikací na testovací data (tzn. vyhodnocování testovacích dat) a výpočtem metrik u výsledků aplikace. Asociační pravidla nemají předdefinované testovací metriky, mohou se ale vyhodnocovat dvě nepřímé míry úspěchu modelování. První představuje počet vygenerovaných pravidel, je možné se vyvarovat nalezení nevhodných pravidel k interpretaci zvětšením hranice minimální podpory. Druhou mírou může být důležitost pravidel, tedy vysoká minimální spolehlivost determinuje vysokou kvalitu pravidel.
5. Aplikace modelu na nová data ke generování předpovědí a vzorů dat. Asociační pravidla mají tu specifičnost, že aplikace není vyžadována a po vytvoření modelu se přistupuje přímo k získávání znalostí (častých vzorů pomocí GET_FREQUENT_ITEMSETS a asociačních pravidel díky GET_ASSOCIATION_RULES, více viz kapitola 6.1.2.4).

6.1.2.4 Metody použité při implementaci

Procedura CREATE_MODEL

Tato metoda vytváří dolovací model pro odpovídající dolovací techniku, v našem případě tedy model pro asociační pravidla. Na vstupu přijímá osm parametrů, z nichž jsou první čtyři povinné. Syntaktický zápis vypadá následovně:

```
DBMS_DATA_MINING.CREATE_MODEL(
    model_name              IN VARCHAR2,
    mining_function         IN VARCHAR2,
    data_table_name         IN VARCHAR2,
```

```
case_id_column_name      IN VARCHAR2,
target_column_name       IN VARCHAR2 DEFAULT NULL,
settings_table_name      IN VARCHAR2 DEFAULT NULL,
data_schema_name         IN VARCHAR2 DEFAULT NULL,
settings_schema_name     IN VARCHAR2 DEFAULT NULL);
```

První parametr specifikuje jméno vytvářeného modelu, které musí respektovat určité syntaktické podmínky (viz výše). Další parametr představuje ODM konstantu pro reprezentaci dolovací techniky (tedy konstanta *association*). Parametr *data_table_name* udává jméno tabulky nebo pohledu obsahující trénovací data. Poslední povinný parametr určuje identifikující (klíčový) sloupec trénovací tabulky. Pátý parametr nabývá pro asociační pravidla hodnoty NULL, *settings_table_name* udává tabulku nastavení. Poslední dva parametry jsou z našeho hlediska nastaveny také na hodnotu NULL.

Data poskytovaná všem následujícím operacím musí odpovídat datům vstupujícím do procedury CREATE_MODEL co se týče jejich schématu a relevantního obsahu. Jsou-li například data pro CREATE_MODEL předzpracována, musí stejně tak být předzpracována i pro proceduru APPLY a to prostřednictvím statistik z CREATE_MODEL data pre-processing. Procedura dále vytváří množinu tabulek k uložení vzorů a informací, které představují dolovací model pro odpovídající algoritmus. Jména těchto tabulek mají společný prefix DM$.

Procedura DROP_MODEL

Metoda umožňuje zrušit existující dolovací model specifikovaný jeho názvem pomocí syntaxe:

```
DBMS_DATA_MINING.DROP_MODEL (model_name in varchar2);
```

Funkce GET_ASSOCIATION_RULES

Tato tabulková funkce vrací pravidla z asociačního modelu. Uživateli je umožněno specifikovat filtrovací kritéria pro vyvolání pouze určité podmnožiny asociačních pravidel. Tato kritéria mohou zlepšit provádění funkce. Pokud je počet pravidel příliš velký, nejvíce dosažené výsledky zlepšuje vstupní parametr *topn*. Tento i všechny ostatní vstupní parametry, s výjimkou prvního, jsou nepovinné. Syntaktický zápis funkce vypadá takto:

```
DBMS_DATA_MINING.GET_ASSOCIATION_RULES (
model_name          IN VARCHAR2,
topn                IN NUMBER DEFAULT NULL,
rule_id             IN INTEGER DEFAULT NULL,
min_confidence      IN NUMBER DEFAULT NULL,
min_support         IN NUMBER DEFAULT NULL,
max_rule_length     IN INTEGER DEFAULT NULL,
min_rule_length     IN INTEGER DEFAULT NULL,
```

sort_order IN DMSYS.ORA_MINING_VARCHAR2_NT DEFAULT NULL,
antecedent_items IN DYSYS.ORA_MINING_VARCHAR2_NT DEFAULT NULL,
consequent_items IN DYSYS.ORA_MINING_VARCHAR2_NT DEFAULT NULL)
RETURN DM_RULES PIPELINED;

kde typ DMSYS.ORA_MINING_VARCHAR2_NT je definován jako tabulka z VARCHAR2(4000).

Parametr	**Popis**
model_name	Jméno dolovacího modelu.
topn	Vrací n pravidel seřazených podle spolehlivost a poté podle podpory, obojí sestupně. Pokud je specifikováno třídění, vrchních n pravidel je odvozeno až toto třídění proběhne.
rule_id	Identifikátor pravidla k vrácení. Jestliže je tento parametr uveden, nespecifikují se ostatní filtrovací parametry.
min_confidence	Navrátí se pravidla se spolehlivostí větší nebo rovné této hodnotě.
min_support	Navrátí se pravidla s podporou větší nebo rovné této hodnotě.
max_rule_length	Vrátí pravidla s délkou menší nebo rovnou této hodnotě. Délka pravidla se vztahuje k počtu položek pravidla. Například pravidlo A $\Rightarrow$ B (když A, pak B) má 2 položky.
min_rule_length	Vrátí pravidla s délkou větší nebo rovnou této hodnotě.
sort_order	Setřídí pravidla podle hodnot v jednom nebo více vrácených sloupcích tabulky (*ASC* pro vzestupné pořadí, *DESC* pro sestupné). Např. specifikování sestupného třídění podle sloupců *NUMBER_OF_ITEMS* a *RULE_CONFIDENCE* bude vypadat takto: DMSYS.ORA_MINING_VARCHAR2_NT (‘NUMBER_OF_ITEMS DESC‘, ‘RULE_CONFIDENCE DESC’). Standardně jsou výsledky nejdříve tříděny sestupně podle spolehlivosti, poté také sestupně podle podpory.
antecedent_items	Vrátí pravidla s těmito položkami v roli předchůdce.
consequent_items	Vrátí pravidla s těmito položkami v roli následníka.

Tabulka 2 : Popis jednotlivých vstupních parametrů funkce GET_ASSOCIATION_RULES

Voláním této funkce dostaneme hodnotu DM_RULES, která reprezentuje množinu řádků typu DM_RULE. Tyto řádky mají následující sloupce:

(rule_id INTEGER,
antecedent DM_PREDICATES,
consequent DM_PREDICATES,
rule_support NUMBER,
rule_confidence NUMBER,

```
antecedent_support      NUMBER,
consequent_support      NUMBER,
numer_of_items          INTEGER)
```

Druhý a třetí sloupec vrací vložené tabulky typu DM_PREDICATES. Pro řádky tohoto typu jsou definovány tyto sloupce:

```
(attribute_name         VARCHAR2(30)
conditional_operator    CHAR(2)       /*= , <>, <, >, <=, >= */
attribute_num_value     NUMBER,
attribute_str_value     VARCHAR2(4000),
attribute_support       NUMBER,
attribute_confidence    NUMBER)
```

7 Popis implementace

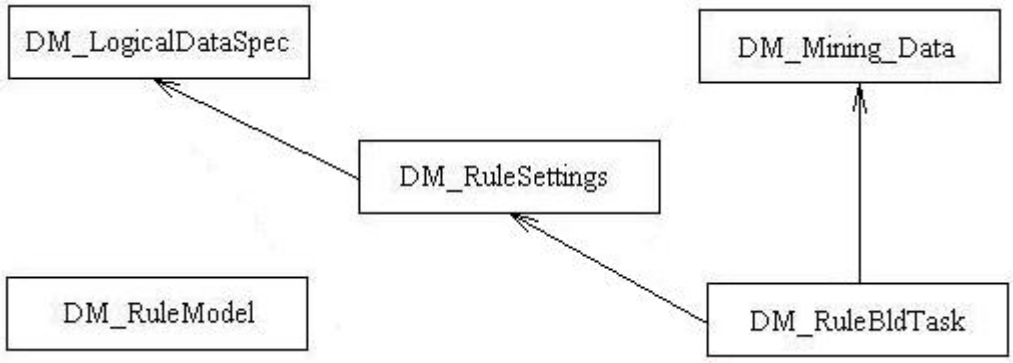
7.1 Implementace typů standardu

Předmětem samotné implementace je vytvoření jakéhosi můstku mezi standardem SQL/MM DM a prostředky Oracle Data Mining, které již obsahují metody pro dolování dat. Pro úspěšnou realizaci je nutné respektovat požadavky obou stran. ODM klade důraz zejména na přípravu dat pro samotné dolování, dokáže pracovat pouze s diskrétními hodnotami. Asociační modely jsou navrženy pro práci s řídkými daty (běžně hustota menší než 1%), navíc je vhodné odstranit duplicitní výskyty položek (např. dvě krabice mléka při jednom nákupu). Pokud databáze obsahuje položky s daty numerickými, nebo kategorickými s velkou kardinalitou, je nutné provést binning (např. quantile binning). Na druhé straně standard vyžaduje rozdělení celého procesu dolování (příprava a analýza dat, vytvoření dolovacího modelu, jeho trénování a samotné dolování, tedy získávání frekventovaných množin položek a z nich asociačních pravidel) do předem definovaných metod. Cílem kapitoly o implementaci je právě popis těchto metod a způsobu jejich naprogramování.

Standard specifikuje pro dolování asociačních pravidel několik vzájemně provázaných typů. Každý z nich řeší určitou logickou část procesu, přičemž lze ale najít jeden společný aspekt, tedy že obsahují jediný atribut – *DM_content*. Tento je typu CLOB (Character Large OBject) specifikovaného parametrem *DM_MaxContentLength*, což je implementačně závislá maximální délka atributu *DM_content* asociačních metod. Datový typ CLOB [6] ukládá jak singlebyte tak multibyte znaková data a dosahuje velikosti (4 GB -1)*velikost bloku databáze. Tudíž je možné ukládat veškerá potřebná data, která patří k danému typu, do atributu *DM_content*. Tento datový typ je vhodný zejména jako skladiště nestrukturovaných dat, nebo dat se strukturou neobvyklou pro databáze, jako je například formát XML. Jelikož nebyly metody pro práci s tímto formátem implementovány (a nestrukturované ukládání dat je v tomto případě krajně nevhodné), do globálního atributu se vždy ukládá jen specifická informace pro daný typ, tedy název zdrojové tabulky (*DM_LogicalDataSpec*, *DM_MiningData*) nebo identifikátor instance typu (ostatní typy), pomocí nichž se pak přistupuje k ostatním datům uložených v pomocných tabulkách (viz níže) mimo atribut *DM_content*.

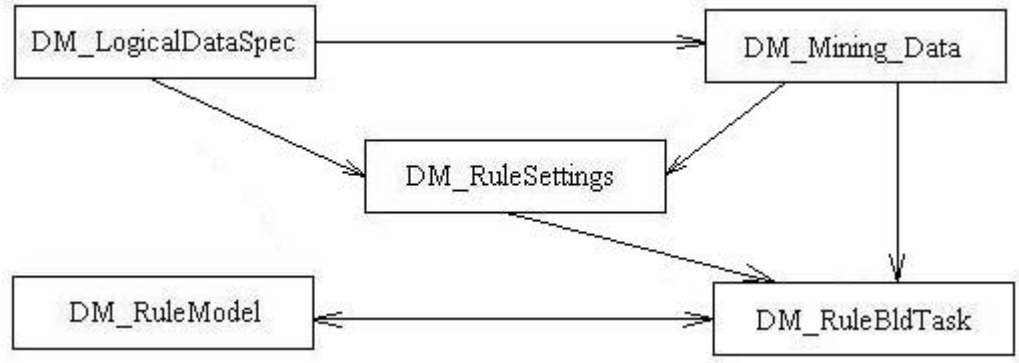
Dalším společným znakem může být závislost, kterou standard předepisuje pro jednotlivé typy vhodné pro získávání asociačních pravidel. Na základě těchto vztahů pak bylo určeno pořadí implementace jednotlivých typů, které je respektováno i v této kapitole při jejím popisování. Je možné rozdělit tuto závislost na dva druhy:

1. Typ potřebuje pro práci data jiného typu – následující obrázek představuje takovouto datovou závislost. Šipka ukazuje na typ, který je využíván typem, od kterého jde šipka.



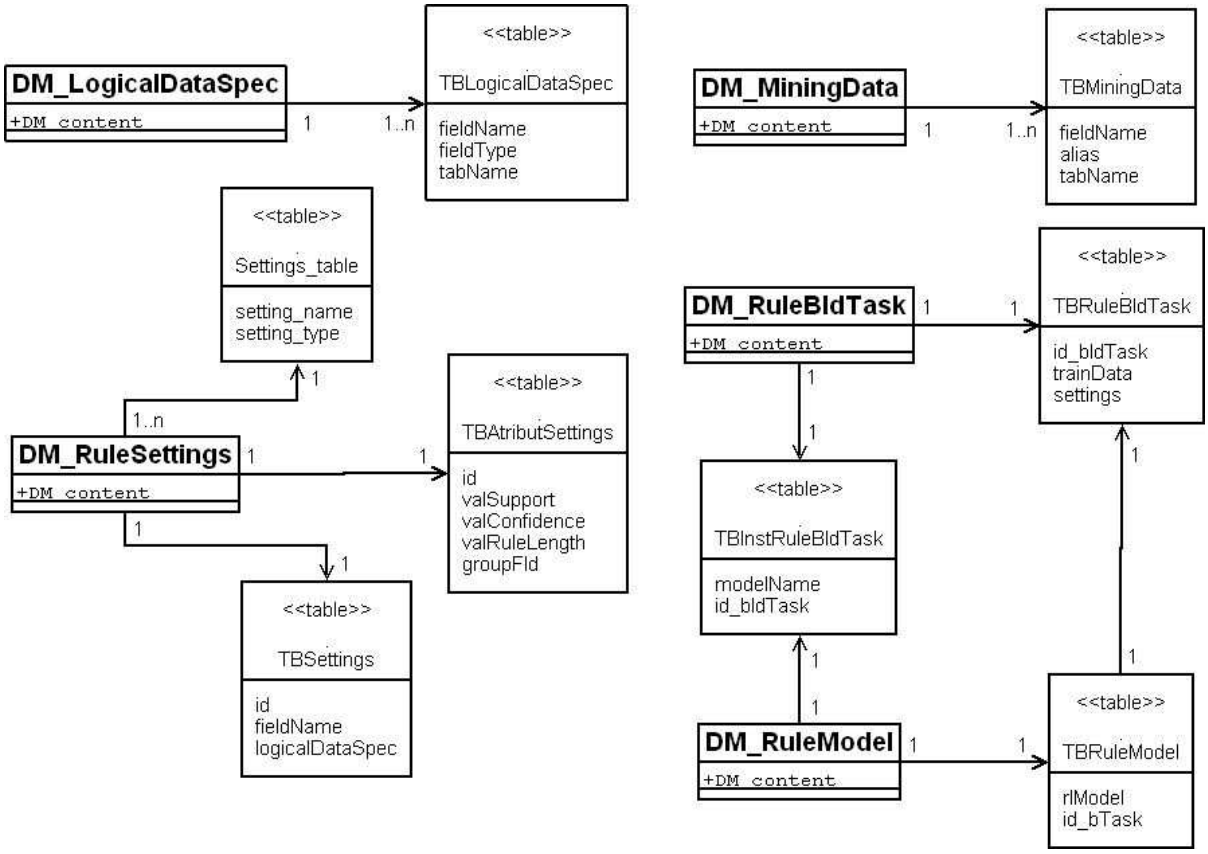
Obr. 5 Typová závislost prvního druhu

2. Typ vytváří data jiného typu – následující obrázek představuje takovouto typovou závislost. Šipka ukazuje na typ, jehož metoda vrací instanci typu, od kterého šipka směřuje. Z obrázku je vidět, že mezi typy *DM_RuleModel* a *DM_RuleBldTask* vzniká nepříjemné zacyklení. U každého z těchto typů existuje metoda, která vrací instanci typu toho druhého. Řešení tohoto problému je popsáno v kapitolách věnovaných implementaci těchto typů.



Obr. 6 Typová závislost druhého druhu

Pro každý typ byla navržena jedna nebo více pomocných tabulek, ve kterých se uchovávají jeho specifická data. Na Obr. 7 můžeme vidět vztahy mezi typy a jejich tabulkami. Tyto asociace jsou tvořeny právě pomocí hodnoty globálního atributu *DM_content*, ve kterém je uložena specifická informace, podle které se rozlišují záznamy patřící jednotlivým objektům.



Obr. 7 Diagram

Stejně jako jsou specifikovány požadavky ze stran standardu a ODM, narazil jsem při implementaci také na určité omezení ze strany jazyka PL/SQL. Totiž jestliže má statická metoda vracet objekt, jehož je metodou, musí to být podle jazyka reference na tento objekt. Tímto bych ale musel změnit předepsaný standard, proto jsem zvolil následující variantu. Místo statické metody s funkcí konstruktoru vytvářím přímo metodu konstruktoru typu. Tento konstruktor musí být stejného jména jako typ a vrací přímo instanci tohoto typu (ne referenci na typ jako statická metoda). V případě, že typ nemá definovaný konstruktor nebo nebyla implementována metoda, která má funkci konstruktoru, musel být konstruktor vytvořen dodatečně.

Implementace daného typu je vždy složena ze dvou souborů. Jedním z nich je hlavičkový soubor s deklarací typu, jeho konstruktoru a jednotlivých metod. Dále jsou v něm vytvářeny pomocné tabulky, se kterými typ pracuje. Druhý soubor, s přídomkem _body, obsahuje samotnou implementaci odpovídajícího typu.

7.2 Typ DM_LogicalDataSpec

Typ *DM_LogicalDataSpec* je úvodem k reprezentaci vstupních dat potřebných při trénovacím, testovacím nebo aplikačním běhu. Představuje abstrakci pro množinu data mining polí

identifikovaných pomocí jejich jména. Druhou informací, se kterou tento typ nakládá, je typ pole. Pro uložení těchto dat jsem vytvořil pomocnou tabulku *TBLogicalDataSpec*, která obsahuje tři sloupce. Do prvního s názvem *fieldName* se ukládá jméno pole, do druhého sloupce *fieldType* je přesměrován typ pole. Aby bylo možné rozlišit data pro jednotlivé instance typu *DM_LogicalDataSpec*, ukládá se do třetího sloupce (*tabName*) název zdrojové tabulky. Tento název je unikátní informací typu, proto je současně uložen i do globálního atributu *DM_content*. Primární klíč jsem definoval jako složený ze sloupců *tabName* a *fieldName*.

Z důvodu omezení jazyka PL/SQL (viz kapitola 7.1) je první metodou tohoto typu konstruktor. Nese stejný název jako typ a hodnota předávaná jeho vstupním parametrem *tableName* představuje jméno zdrojové tabulky. Jedná se o identifikující informaci, proto je uložena do atributu *DM_content*.

DM_addDataSpecFld je první metodou typu. Umožňuje přidání jednoho pole do instance typu, fyzicky se tedy přidá toto pole do pomocné tabulky *TBLogicalDataSpec* s tím, že je odstíněno pomocí sloupce *tabName*. Metodu jsem implementoval jako proceduru a podle standardu je přijímán na vstupu jediný parametr *fieldName*, který představuje jméno pole, jež se má přidávat. V příkazové části se nejprve provádí kontrola, zda toto pole již v dané instanci neexistuje. Pokud ano, je vyvoláno příslušné chybové hlášení. Stejně tak je kontrolováno, jestli vstupní parametr opravdu obsahuje nějakou hodnotu. Jsou-li obě tyto podmínky splněny, provede se přidání pole do tabulky.

Další členskou metodou typu je *DM_remDataSpecFld*. Tato je opět navržena jako procedura se vstupním parametrem *fieldName* specifikujícím pole k odstranění z tabulky. Pokud parametr obsahuje nějakou hodnotu a pokud je tato hodnota obsažena v tabulce *TBLogicalDataSpec*, jsou z ní prostřednictvím příkazu *delete from* odstraněny informace identifikované touto hodnotu (*fieldName*, *fieldType* i *tabName*), tedy celý řádek tabulky.

Následující metodou, kterou definuje standard pro typ *DM_LogicalDataSpec*, je *DM_getNumFields*. Tato funkce vrací počet polí tabulky pro objekt *SELF*, což je zjištěno pomocí jediného příkazu *select count(*)*.

DM_getFldName má za úkol pomocí vstupního parametru *position* nalézt název odpovídajícího pole, který je vrácen na výstupu funkce. Nejdříve se kontroluje kvalita vstupního parametru, tedy je-li jeho hodnota větší než nula a zároveň menší nebo rovna počtu polí tabulky *TBLogicalDataSpec* pro daný objekt *SELF*. Jestliže vstupní parametr podmínkami neprojde, je vyvoláno chybové hlášení podle standardu. V opačném případě se pomocí kurzoru prochází tabulka odpovídající objektu *SELF* tak dlouho, až se dojde na řádek, jehož pozice odpovídá hodnotě parametru *position*. Při této akci jsem využil SQL atribut *%ROWCOUNT*. Z nalezeného řádku pak funkce vrací název pole.

Metodou, která slouží k nastavení typu pole, je *DM_setFldType*. Protože nevrací žádnou hodnotu, navrhl jsem ji jako proceduru, která podle standardu přijímá dva vstupní parametry. Pomocí prvního s názvem *fieldName* je nalezeno pole s tímto názvem, jehož typ je pak pomocí hodnoty druhého parametru *miningType* změněn. Příkazová část opět začíná kontrolou, zda je vstupní název pole obsažen v tabulce *TBLogicalDataSpec* pro objekt *SELF*. Díky SQL příkazu *update* se pak

nastaví hodnota typu odpovídajícího pole a to buď na hodnotu 0 (*DM_Categorical*) nebo 1 (*DM_Numerical*).

Další metoda provádí opačnou operaci, než její předchůdkyně. Pomocí *DM_getFldType* můžeme zjistit typ pole specifikovaného vstupním parametrem *fieldName*. Jestliže je *fieldName* validním parametrem (a není tak nutné vyvolat odpovídající výjimky), nalezne se jeho hodnota v pomocné tabulce objektu *SELF* a na výstupu funkce je pak vrácen typ této hodnoty.

Poslední deklarovanou metodou typu je *DM_isCompatible*, která má za úkol porovnat objekt *SELF* s jiným objektem stejného typu, který je přijímán na vstupu prostřednictvím parametru *dataSpec*. Máme tedy dva objekty typu *DM_LogicalDataSpec* a máme zjistit, zda jsou kompatibilní. V deklarační části jsem inicializoval proměnnou *isCompatible*, která představuje výstup funkce, na hodnotu 1. Tato hodnota znamená, že jsou obě instance kompatibilní a v příkazové části pak jen kontroluji možnosti, kdy tomu tak není *(isCompatible* v tom případě nabude hodnoty 0). Prvním takovýmto případem je, že vstupní parametr je NULL. Dále pokud je počet polí pro objekt *dataSpec* menší než počet polí pro objekt *SELF*, nemohou být v *dataSpec* obsažena všechna pole *SELF*, objekty tedy opět nejsou kompatibilní. V případě, že tomu tak není a počty polí pro oba objekty jsou si rovny nebo počet polí *dataSpec* je větší než počet polí *SELF*, kontrolují se pomocí kurzoru jednotlivé řádky objektu *SELF* a příkazem *select* se k nim hledá odpovídající záznam v *dataSpec*. Jestliže je zjištěn jeden případ, kdy nejsou odpovídající data v *dataSpec* nalezena, znamená to znovu, že oba objekty nemohou být kompatibilní. Jsou-li všechny výše uvedené podmínky splněny, můžeme říct, že jsou obě instance kompatibilní a funkce vrací hodnotu 1.

7.3 Typ DM_MiningData

Typ *DM_MiningData* představuje abstrakci pro data reálného světa, která jsou uložena v tabulce nebo pohledu. Hodnota tohoto typu reprezentuje metadata pro přístup k reálným tabulkám pro pozdější trénování, testovaní nebo aplikaci. Představuje tedy data, ze kterých se bude dolovat. Důležitými informacemi pro tento typ jsou jména sloupců a jejich aliasy, z čehož vyplívá, že tabulka *TBMiningData* pro uložení dat tohoto typu má tři sloupce. Prvním je *fieldName* připravený pro název pole, tedy sloupce reálné tabulky s fyzickými daty. Pro druhou informaci slouží sloupec *alias* a pro rozlišení jednotlivých instancí typu je opět připraven sloupec *tabName*. Primární klíč této tabulky je navržen stejně jako u předešlého typu, tedy složený klíč sloupci *fieldName* a *tabName*. Do globálního atributu *DM_content* je ukládán název reálné tabulky.

Standardem je sice navržena jako konstruktor metoda *DM_defMiningData*, z důvodu omezení jazyka PL/SQL popsaného v kapitole 7.1 jsem ale byl nucen ji přejmenovat na *DM_MiningData*. Toto je ale jediná změna oproti definici ve standardu, ostatní části konstruktoru jsou implementovány v souladu s ním. Konstruktor tak na vstupu přijímá parametr *tableName* určující jméno tabulky, ze které se budu dolovat. Hodnota tohoto parametru je vzápětí uložena do *DM_content* a určuje základní

vlastnost konstruktoru, tedy zjištění, zda opravdu existuje tabulka s názvem *tableName*. K tomuto ověření jsem využil předdefinované SQL atributy *ALL_TABLES*, *ALL_TAB_COLUMNS*, *TABLE_NAME* a příkazu *select count(*)*. Není-li nalezena žádná tabulka odpovídajícího názvu, je na výstup posláno chybové hlášení. V opačném případě se přistupuje ke zpracování tabulky. Toto se provádí díky kurzoru, jehož prostřednictvím se postupně prochází všechny položky příkazu *select*, který nalezl všechny názvy sloupců dané tabulky. Tyto názvy jsou ukládány do tabulky metadat *TBMiningData* jak do sloupce *fieldName*, tak do sloupce *alias*. Když je doplněn i poslední sloupec názvem vstupního parametru, je konstruktorem vytvořena platná instance typu *DM_MiningData*.

První členskou metodou typu je *DM_setFldAlias*, která uživateli umožní nastavit alias pole. Implementoval jsem ji jako proceduru se dvěma vstupními parametry. První *dataField* určuje jméno pole, jehož alias se má přenastavit, druhý *aliasName* nese novou hodnotu aliasu. Aby se tato akce mohla provést, musí se splnit několik podmínek. Nejprve je testován parametr *dataField*. Je-li jeho hodnota NULL, vrátí se objekt *SELF* nezměněný. Není-li jeho hodnota nalezena v tabulce *TBMiningData*, vyvolá se odpovídající SQL/MM data mining výjimka. Naopak jestliže *dataField* existuje, ale *aliasName* je NULL, nastaví se hodnota *aliasName* pro odpovídající pole na hodnotu parametru *dataField*. Dále pokud je hodnota *aliasName* již v tabulce uvedena, neprovede se žádná akce. Neplatí-li žádná z výše zmíněných možností, nastaví se alias pole s názvem *dataField* na hodnotu *aliasName*.

DM_genDataSpec představuje další metodu typu. Jejím předmětem je vytvoření objektu typu *DM_LogicalDataSpec* podle metadat v *TBMiningData* pro objekt *SELF*. Standard metodě, kterou jsem navrhl jako funkci, nespecifikuje žádný vstupní parametr. V příkazové části je tak nejprve vytvořen objekt typu *DM_LogicalDataSpec* pomocí konstruktoru tohoto typu. Následně jsou díky kurzoru, který prochází všechny záznamy v tabulce *TBMiningData*, a metody *DM_addDataSpecFld* typu *DM_LogicalDataSpec* přidávány tyto záznamy do tabulky *TBLogicalDataSpec*, čímž jsou přiřazeny do nově vytvořeného objektu, který je pak funkcí vrácen.

7.4 Typ DM_RuleSettings

Typ *DM_RuleSettings* je navržen k popisu nastavení, které je použito při vytváření modelu pro asociační pravidla, čímž nakládá s poměrně mnoha informacemi. Tentokráte jsem si tak nevystačil pouze s jednou pomocnou tabulkou jako u předešlých typů, ale navrhl jsem celkem tři. První z nich s názvem *TBAtributSettings* uchovává data, která jsou vytvářena metodami typu. Obsahuje celkem pět sloupců. Primárním klíčem je první sloupec *id* umožňující rozlišit jednotlivé instance tohoto typu. Pro každou instanci se pak v ostatních sloupcích uchovávají hodnoty minimální podpory, minimální spolehlivosti, maximální délky asociačního pravidla a název pole pro identifikaci skupiny (např. nákupní transakce). *Settings_table* představuje tabulku nastavení s hodnotami k přetěžování standardního dolovacího algoritmu apriori. Implementována je přesně tak, jak je popsána v ODM,

tedy se dvěma sloupci *setting_name* a *setting_value*. Do prvního sloupce jsou prostřednictvím níže popsaných metod typu ukládány ODM identifikátory pro nastavení algoritmu (*asso_min_support*, *asso_min_confidence*, *asso_max_rule_length*), do druhého jejich hodnoty. Primárním klíčem tabulky je první sloupec. Poslední tabulka *TBSettings* je využívána k uložení implementačních dat typu, aby mohly být kteroukoliv metodou okamžitě použity. Obsahuje tři sloupce, primární klíč *id* představuje rozlišení jednotlivých instancí typu *DM_LogicalDataSpec*. Druhý sloupec *fieldName* umožňuje ukládání názvů polí instancí typu *DM_LogicalDataSpec* uvedených v posledním sloupci *logicalDataSpec*.

Jakožto konstruktor typu je standardem definována metoda *DM_impRuleSettings*. Z důvodů uvedených v kapitole 7.1 jsem však navrhl konstruktor vlastní, *DM_RuleSettings* (navíc implementace neobsahuje podporu formátu XML, tudíž nejsou členské metody *DM_impRuleSettings* a *DM_expRuleSettings* řešeny). Jeho předmětem je vygenerování identifikačního čísla instance tohoto typu, proto není nutné, aby na vstupu přijímal jakýkoliv parametr. Pomocí příkazu *settings_sequence.nextval* nad systémovým katalogem vygeneruje nový identifikátor, který je pak uložen do globálního atributu *DM_content*.

První implementovanou členskou metodou typu *DM_RuleSettings* je procedura pro nastavení hodnoty minimální podpory asociačních pravidel *DM_SetMinSupport*. Na vstupu přijímá parametr *support*, který právě hodnotu podpory specifikuje. Na začátku těla procedury se nejprve zkontroluje, zda tento parametr leží ve standardem definovaném intervalu. Poté se hledá v tabulce *TBAtributSettings*, jestli již dříve byla minimální podpora nastavena. Pokud ano, pomocí příkazu *update* se změní odpovídající číslo na nové, pokud ne, provede se *insert* hodnoty support do tabulky. V obou případech se na stejný řádek ukládá i identifikátor instance typu. Minimální podpora se současně posílá i do tabulky *Settings_table* na řádek s ODM identifikátorem *asso_min_support*. Předtím je ale nutné vstupní parametr vydělit stem, jelikož ODM definuje hodnotu minimální podpory v intervalu (0,1), zatímco standard v intervalu (0,100).

Protiváhou předešlé procedury je funkce *DM_getMinSupport*, která vrací hodnotu nastavenou metodou *DM_setMinSupport*. Jestliže se hodnota podpory pro daný objekt *SELF* v tabulce *TBAtributSettings* nalezne, je poslána na výstup funkce. Pokud tomu tak není, je vrácena hodnota NULL znamenající, že se minimální podpora bude nastavovat až při aplikačním běhu.

Stejné dvojice metod definuje standard i pro nastavení/získání hodnot minimální spolehlivosti, maximální délky asociačních pravidel a názvu groupového pole. Při jejich implementaci jsem šel podobnou cestou jako u předešlé dvojice, proto při popisu způsobu implementace skočím rovnou až k metodě *DM_useRuleDataSpec*. Procedura přijímá vstupní parametr typu *DM_LogicalDataSpec*, který představuje požadovanou logickou datovou specifikaci pro nastavení dolovacího modelu. Jestliže jeho hodnota projde testem, zda není náhodou rovna NULL, hledá se v tabulce *TBLogicalDataSpec* instance zadaná vstupním parametrem *logicalDataSpec*. Příkazem select se kontroluje, jestli hodnota ve sloupci *tabName* souhlasí s hodnotou atributu *DM_content* objektu. Je-li

vše v pořádku, jsou všechny názvy polí z *TBLogicalDataSpec* patřící objektu *logicalDataSpec* zkopírovány do tabulky *TBSettings*. Tuto operaci jsem udělal pomocí kurzoru, který navíc ještě do *TBSettings* vkládá identifikační číslo uložené v atributu *DM_content*, podle kterého se na závěr do posledního sloupce uloží i vstupní parametr. Procedura je opět doplněna metodou *DM_getRuleDataSpec*, která vrací hodnotu nastavené logické datové specifikace, tedy právě poslední sloupec tabulky *TBSettings* podle identifikátoru v *SELF.DM_content*.

7.5 Typ DM_RuleBldTask

Stěžejním smyslem typu *DM_RuleBldTask* je fakt, že poskytuje metodu pro vytvoření asociačního modelu. Typ reprezentuje informace důležité pro dolovací model – vstupní data a nastavení. Tato data jsou uchovávána v pomocné tabulce *TBRuleBldTask*, jejímž primárním klíčem je první sloupec *id_bldTask* pro ukládání hodnot identifikujících data objektu *SELF*. Sloupec s názvem *trainData* je druhým v tabulce a ukládají se do něho vstupní data typu *DM_MiningData*. Nastavení dolovacího algoritmu je uloženo do třetího sloupce *settings* s daty typu *DM_RuleSettings*. Typ pracuje ještě s jednou tabulkou s názvem *TBInstRuleBldTask*, jež slouží k uchovávání instancí typu. Obsahuje dva sloupce, do prvního *modelName* se ukládá identifikátor úlohy dolování a do druhého *id_bldTask* identifikátor instance typu. Primárním klíčem je první sloupec tabulky.

První metodou typu je opět jeho konstruktor. Standardem je definován jako *DM_defRuleBldTask*, ale z důvodů uvedených v kapitole 7.1 jsem ho přejmenoval na *DM_RuleBldTask*. Konstruktor přijímá na vstupu dvojici parametrů, tedy *trainData* (typ *DM_MiningData*) a *settings* (typ *DM_RuleSettings*). Oba jsou nejprve prověřeny, zda jejich hodnoty nejsou NULL (ošetřeno patřičnými chybovými hlášeními), a poté se přistoupí k ověření kompatibility logických datových specifikací obou parametrů. Toto se provádí stejným způsobem, jaký jsem implementoval v metodě *DM_isCompatible* typu *DM_LogicalDataSpec*, proto popis přeskočím. Jestliže se zjistí, že jsou obě specifikace kompatibilní, vygeneruje se díky příkazu *bldTask_sequence.nextval* nad systémovým katalogem identifikační číslo úlohy. Toto se jednak uloží do atributu *DM_content* objektu *SELF*, je ale také využito na identifikaci vytvořeného dolovacího modelu a uloží se spolu s hodnotami *trainData* a *settings* do tabulky *TBRuleBldTask*.

První členskou metodou typu je *DM_getRuleTrnData*. Navrhl jsem ji jako funkci, která vrací trénovací data typu *DM_MiningData* (viz konstruktor typu). Pomocí příkazu *select count(*)* nad tabulkou *TBRuleBldTask* se zjišťuje, jestli už vůbec byla nějaká vstupní data načtena. Pokud je výsledek této operace kladný, je na výstup funkce poslána hodnota ze sloupce *trainData* pro odpovídající *id_bldTask*.

DM_getRuleSettings vrací druhou podstatnou informaci, tedy nastavení. Je implementována stejným způsobem jako předešlá, jen na výstup posílá hodnotu sloupce *settings* (typ *DM_RuleSettings*) tabulky *TBRuleBldTask*.

Jak již bylo zmíněno výše, jádrem tohoto typu je metoda *DM_buildRuleModel*, která vrací hodnotu typu *DM_RuleModel* jakožto nově vytvořený dolovací model pro asociační pravidla. Nejdříve se pomocí dvou předešlých metod získají hodnoty trénovacích dat a nastavení. Pokud nejsou obě NULL, hledá se nad systémovým katalogem sloupec, který je primárním klíčem tabulky trénovacích dat. Poté se už přistoupí pomocí příkazu *dbms_data_mining.create_model* k vytvoření dolovacího modelu (viz kapitola 6.1.2.4). Do jeho prvního specifikovaného parametru *model_name* je uložena hodnota identifikátoru úlohy asociačních pravidel z atributu *DM_content*. Pro parametr *mining_function* je uvedena ODM konstanta *dbms_data_mining.association* specifikující typ dolovací úlohy, tedy asociační pravidla. Následující *parametr data_table_name* udává jméno tabulky s trénovacímy daty a *case_id_column_name* deklaruje klíčový sloupec této tabulky (získaný na začátku metody nad systémovým katalogem). Poslední implementovaný parametr je *settings_table_name* pro název tabulky nastavení přetěžující standardní dolovací algoritmus. Tímto postupem je vytvořen dolovací model, jehož název je předán konstruktoru typu *DM_RuleModel* (viz Obr. 6 Typová závislost druhého druhu). V těle funkce se ještě před vytvořením modelu zapíší hodnoty do tabulky instancí *TBInstRuleBldTask*.

7.6 Typ DM_RuleModel

Typ *DM_RuleModel* reprezentuje modely, které jsou výsledkem hledání asociačních pravidel. Základní informací, se kterou typ nakládá, je identifikace asociačního modelu. Pro uchování tohoto druhu dat slouží tabulka *TBRuleModel* složená ze dvou sloupců. Klíčovým sloupcem je opět první s názvem *rlModel* pro rozlišení dat jednotlivých instancí typu *DM_RuleModel*. Do druhého sloupce *id_bTask* se ukládají hodnoty identifikující instance typu *DM_RuleBldTask*. Tento sloupec je cizím klíčem odkazujícím na tabulku *TBRuleBldTask*, ve které se tyto identifikátory uchovávají. Do atributu *DM_content* se opět ukládá identifikační číslo instance typu. Do implementace tohoto typu nejsou z výše uvedených důvodu zahrnuty členské metody *DM_impRuleModel* a *DM_expRuleModel*.

Konstruktor *DM_RuleModel* jsem byl nucen z důvodů uvedených v kapitole 7.1 vytvořit dodatečně. Na vstupu přijímá parametr *id_modelName*, který specifikuje instanci typu *DM_RuleBldTask* a který je zároveň identifikátorem vytvořeného dolovacího modelu metodou *DM_buildRuleModel* (viz kapitola 7.5), v jejímž rámci je volán. V příkazové části konstruktoru je nejdříve nad systémovým katalogem vygenerováno číslo instance, jež je následně uloženo do globálního atributu *DM_content*. Současně je, spolu s hodnotou vstupního parametru, uchováno do tabulky *TBRuleModel*.

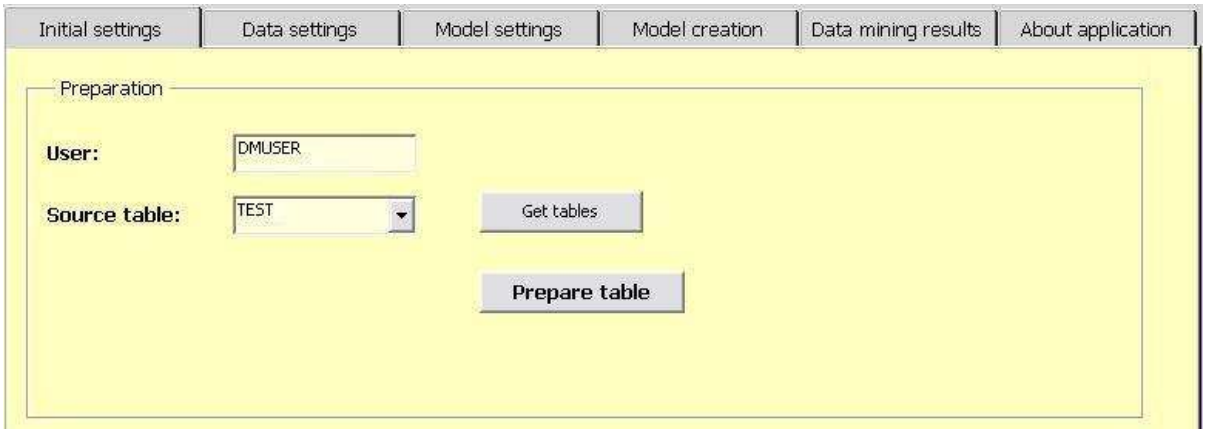
Členská metoda pro získání počtu asociačních pravidel má název *DM_getNumRules*. Ve funkci je nejdříve pomocí příkazu *select* nad tabulkou *TBRuleModel* zjištěn název dolovacího modelu, který slouží jako vstupní parametr ODM metody *dbms_data_mining.get_association_rules*. Tato metoda dokáže nad tabulkou identifikovanou názvem modelu zjistit počet nalezených asociačních pravidel.

Aby bylo možné získat instance typu *DM_RuleBldTask*, definuje standard poslední členskou metodu typu *DM_getRuleBldTask*. Právě tato metoda způsobuje zacyklení mezi typy *DM_RuleBldTask* a *DM_RuleModel* (viz Obr. 6), proto jsem byl při implementaci nucen změnit hlavičku funkce tak, aby metoda nevracela přímo instanci typu *DM_RuleBldTask*, ale referenci na tuto instanci, čímž byl problém se zacyklením vyřešen. V přízové části metody se nejprve příkazem *select* nad tabulkou *TBRuleModel* získá jméno dolovací úlohy, pomocí kterého je pak v tabulce *TBInstRuleBldTask* vyhledán identifikátor instance typu. Díky tomuto identifikátoru se následně v tabulce *TBRuleBldTask* načtou data ze zbytku řádku, která se využijí jako vstupní parametry konstruktoru *DM_RuleBldTask*, jehož hodnota je poslána na výstup funkce.

7.7 Implementace ukázkové aplikace

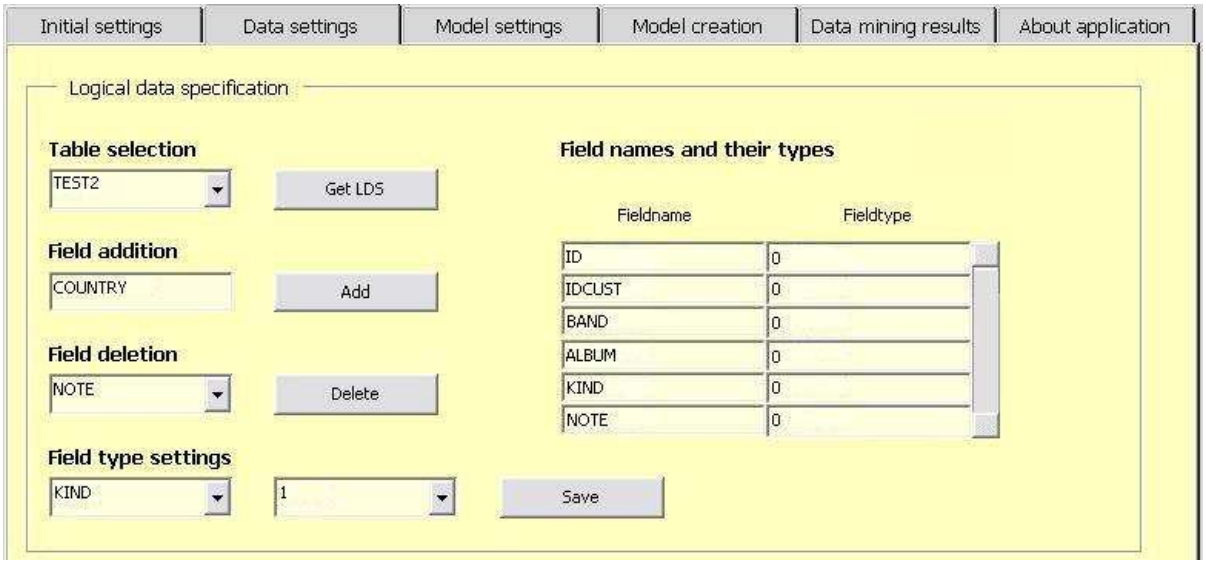
Funkčnost výše popsaných typů standardu SQL/MM DM pro asociační pravidla je ověřena na ukázkové aplikaci vytvořené pomocí Oracle Forms Builderu 6. Poté co jsem navrhl grafickou podobu, mohl jsem přistoupit k implementaci funkčnosti jednotlivých částí programu. Přitom jsem zjistil, že Forms Builder 6 nedokáže přímo využívat typy a jejich metody uložené v databázi. Proto bylo nutné vytvořit pomocné procedury a tabulky, které jsou pak volány z aplikace. O těchto procedurách se zmiňuji níže při popisu funkcí jednotlivých tlačítek. Pokud jde o pomocné tabulky, byla pro každý uživatelský typ (s výjimkou *DM_RuleBldTask*) vytvořena jedna. Každá obsahuje vždy dva sloupce, první nese název zdrojové tabulky (pro datové typy) nebo název nastavení, resp. modelu, do druhého se ukládají instance daného typu. Primárním klíčem je první sloupec tabulky. Byly tak vytvořeny následující tabulky: *PTBLogicalDataSpec (tabName, LDS), PTBMiningData (tabName, MD), PTBRuleSettings (setName, RS), PTBRuleModel (modelName, RM)*. Pro uchování výsledků dolování asociačních pravidel byla ještě vytvořena tabulka *PTBMiningResults*. Obsahuje pět sloupců, primárním klíčem je první sloupec *rule_id*, který představuje identifikátor nalezeného pravidla. Dalšími sloupci jsou *antecedent* (levá část pravidla), *consequent* (pravá část pravidla), *rule_support* (podpora pravidla), *rule_confidence* (spolehlivost pravidla).

První strana aplikace představuje počáteční nastavení. Uživateli umožňuje zadat jeho jméno, přičemž se po stisku tlačítka *Get tables* načtou do roletového menu s označením *Source table* všechny tabulky, které vytvořil. Vybere-li pak z něj některou a zmáčkne tlačítko *Prepare table*, vytvoří se instance typů *DM_LogicalDataSpec* a *DM_MiningData*. Toho se dosáhne prostřednictvím pomocné procedury *PrepareData*, která na vstupu přijímá název tabulky (vybrané v menu *Source table*). V příkazové části je pak volán konstruktor typu *DM_MiningData*, čímž se vytvoří jeho instance, a díky metodě tohoto typu *DM_genDataSpec* se vytvoří i instance typu *DM_LogicalDataSpec*. Obě takto vytvořené instance jsou následně uloženy do pomocných tabulek *PTBLogicalDataSpec*, resp. *PTBMiningData*.



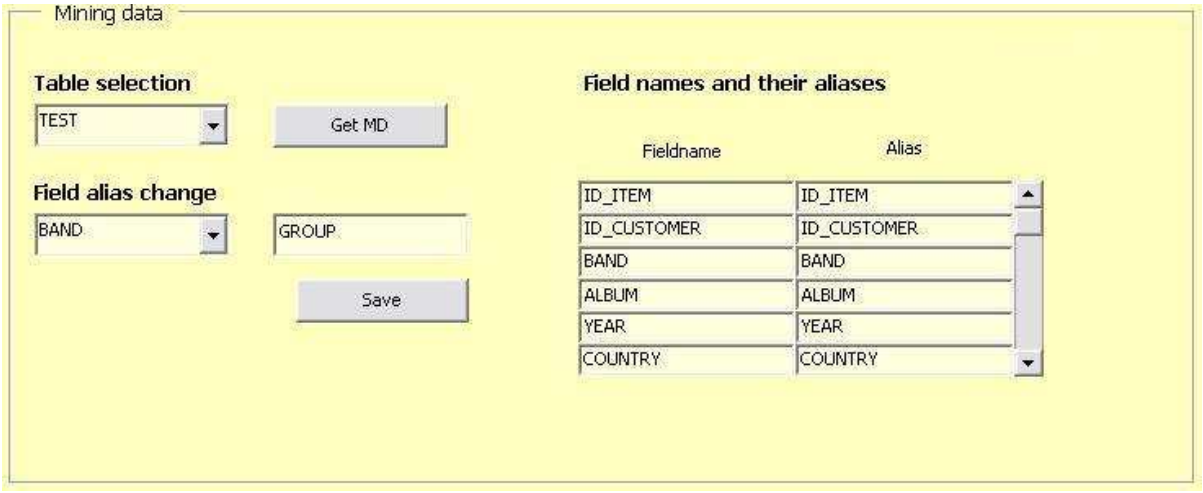
Obr. 8 Počáteční nastavení

Další stránka aplikace je *Data settings*, která slouží k manipulaci s objekty typů *DM_LogicalDataSpec* a *DM_MiningData*. Je rozdělena do dvou rámců. V prvním *Logical data specification* se provádí metody typu *DM_LogicalDataSpec*. Pro výběr tabulky, která se bude zpracovávat je určena část *Table selection*. Po stisku tlačítka *Get LDS* se do roletového menu načtou názvy zdrojových tabulek, které jsou uloženy v tabulce *TBLogicalDataSpec*. Je-li následně vybrána některá z nabízených tabulek, zobrazí se její obsah vpravo v tabulce *Field names and their types*. Uživatel dále může přidat nové pole do tabulky, nastavit jeho typ nebo vybrané pole smazat. Přidání pole umožňuje část *Field addition*. Poté co je zadán název nového pole a zmáčknuto tlačítko *Add*, je volána procedura *AddField*, která na vstupu přijímá název používané tabulky (specifikované v oddílu *Table selection*) a název pole k přidání. Nejprve se z tabulky *PTBLogicalDataSpec* načte specifikace, do které je pomocí metody *DM_addDataSpecFld* pole přidáno. Podobný postup je i při mazání pole (část *Field deletion*), kdy je po výběru pole z roletového menu a stisku tlačítka *Delete* volána procedura *RemoveField*, která využívá metody *DM_remDataSpecFld*. V sekci *Field type settings* může uživatel vybrat název pole a hodnotu jeho typu, kterou chce nastavit. Po stisku tlačítka *Save* je volána procedura *SetType*, která používá metodu *DM_setFldType* dané instance typu *DM_LogicalDataSpec*.



Obr. 9 Logická datová specifikace

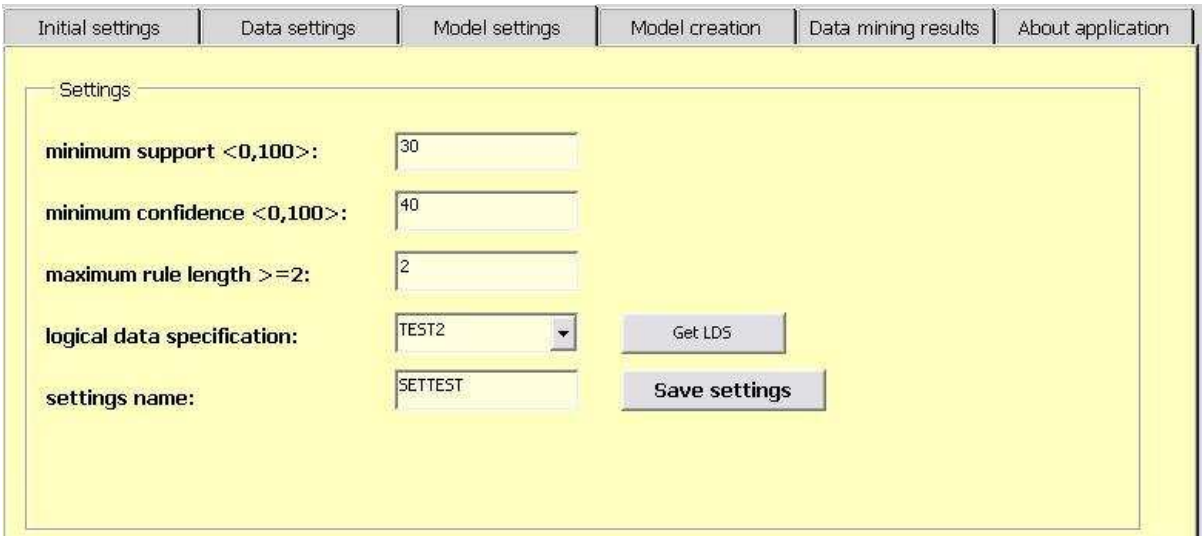
Mining data je druhým rámcem stránky *Data settings*. Pomocí tlačítka *Get MD* se do roletového menu načtou všechny názvy tabulek, uložených v *TBMiningData*. Je-li pak některá vybrána, zobrazí se vpravo v tabulce *Field names and their aliases*. Pokud si uživatel přeje změnit alias některého pole, může tak učinit v části *Field alias change*. Musí pak vybrat název pole a zadat jeho nový alias. Po stisku tlačítka *Save* se volá procedura *ChangeAlias*, která díky metodě *DM_setFldAlias* změní alias daného pole instance typu *DM_MiningData*.



Obr. 10 Dolovací data

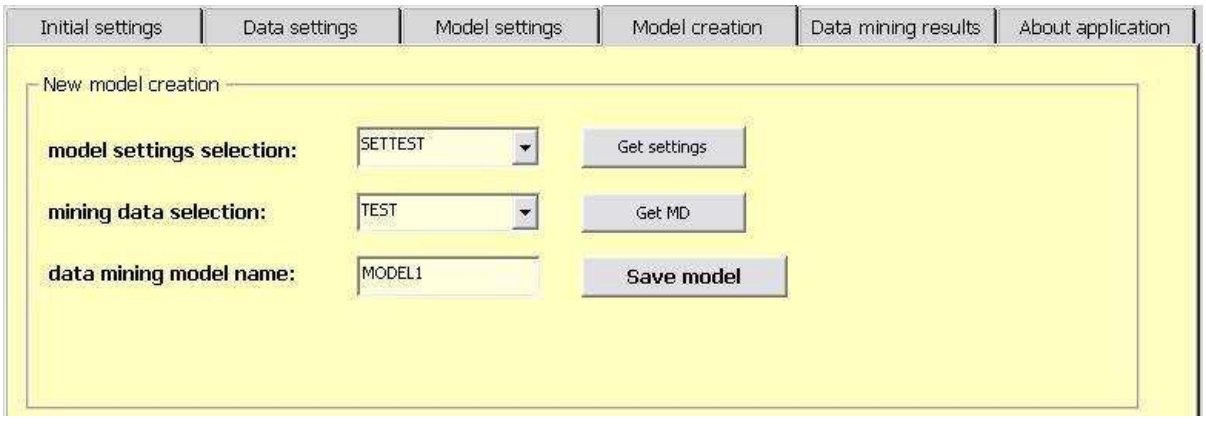
Následující stránkou aplikace je *Model settings*. Ta umožňuje zadat hodnoty minimální podpory, minimální spolehlivosti a maximální délky pravidla. Uživatel následně musí vybrat logickou datovou specifikaci a zadat název nastavení. Specifikované nastavení dolovacího modelu pak uloží prostřednictvím tlačítka *Save settings*, které volá proceduru *CreateSettings*. V jejím rámci je volán

konstruktor typu *DM_RuleSettings*, datová specifikace se uloží pomocí metody *DM_useRuleDataSpec*. Hodnoty dolovacích atributů jsou reflektovány metodami *DM_setMinSupport*, *DM_setMinConf* a *DM_setMaxLength*.



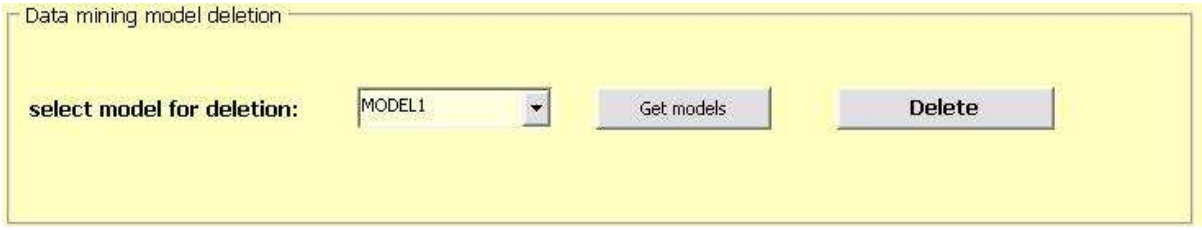
Obr. 11 Nastavení dolovacího modelu

Model creation je další stránkou programu. První rámec *New model creation* poskytuje uživateli možnost definovat nový dolovací model. Je-li z roletových menu vybráno nastavení i dolovací data a zadá-li uživatel do pole s označením *data mining model name* název nového modelu, stiskem tlačítka *Save model* se akce provede. Je pak volána procedura *CreateModel*, v jejímž rámci se vytvoří instance typu *DM_RuleBldTask*, jejíž metoda *DM_buildRuleModel* nový model vytvoří. Název modelu a instance typu *DM_RuleModel* se následně uloží do tabulky *PTBRuleModel*. Funkčnost vytvoření modelu se bohužel nepodařilo ve Form builderu dokázat. Po stisku tlačítka *Save model* vždy vyskočila interní systémová chyba ORA-40101. Pro její vyřešení doporučovala technická podpora na serveru oracle.com kontaktovat Oracle Support Service. Z časových důvodů byla tedy funkčnost ověřena voláním procedury *CreateModel* z konzole SQL*Plus, přičemž se nový dolovací model korektně vytvořil.



Obr. 12 Vytvoření nového dolovacího modelu

Druhý rámec *Data mining model deletion* umožňuje smazání existujícího dolovacího modelu. Po stisku tlačítka *Get models* se do roletového menu načtou z tabulky *PTBRuleModel* všechny názvy dolovacích modelů. Poté, co uživatel některý z modelů vybere a stiskne tlačítko *Delete*, je volána procedura *DeleteModel*, která pomocí ODM procedury *DROP_MODEL* smaže existující dolovací model. Záznamy o modelu jsou odstraněny i z pomocných tabulek *PTBRuleModel* a *TBInstRuleBldTask*. Funkčnost této části implementace byla také ověřena prostřednictvím konzoly SQL*Plus. Volání procedury *DeleteModel* se specifikovaným názvem modelu proběhlo bez chyb.



Obr. 13 Smazání existujícího dolovacího modelu

Poté, co jsou provedena všechna výše popsaná nastavení, lze přistoupit k samotnému dolování asociačních pravidel, což nabízí stránka *Data mining results*. Uživatel je nejprve vyzván aby z roletového menu uvozeného textem *model selection* vybral model, ze kterého si přeje získávat pravidla (všechny existující modely jsou opět načteny po stisku tlačítka *Get models*). Pokud si pouze přeje zobrazit počet nalezených asociačních pravidel, může zmáčknout tlačítko *Get* (uvozené textem *number of found association rules*) a v roletovém menu se zobrazí odpovídající číslovka. Roletové menu bylo vybráno z důvodu uchovávání jednotlivých hodnot počtu asociačních pravidel, uživatel má totiž na této stránce možnost přenastavovat dolovací atributy. Po stisku tlačítka *Get* je volána procedura *NumberRules*, jež na vstupu přijímá parametr specifikující název vybraného dolovacího modelu. V podprogramu se spočítá počet řádků tabulky s výsledky hledání asociačních pravidel a tato hodnota se uloží na odpovídající místo tabulky *PTBRuleModel*. Právě tato hodnota je zobrazena v roletovém menu. Chce-li uživatel jít dál a vidět podrobnosti o výsledcích dolování, je pro něj připraveno tlačítko *Display rules*, které zobrazí do tabulky pod ním hodnoty identifikačního čísla asociačního pravidla, levou a pravou stranu pravidla i hodnoty minimální podpory a spolehlivosti. Navíc má možnost specifikovat nové hodnoty atributů, nevedly-li původní k očekávaným výsledkům. Lze nastavit počet pravidel, které si přeje v tabulce zobrazit, hodnoty minimální podpory a spolehlivosti, maximální délky pravidla a specifikovat levou či pravou stranu pravidla. Po stisku tlačítka *Display rules* je volána procedura *GetRules* s výše zmíněnými vstupními parametry. V jejím rámci je do kurzoru načten blok tabulky s výsledky dolování specifikovaný následujícím příkazem:

SELECT rule_id, antecedent, consequent, rule_support, rule_confidence FROM TABLE (DBMS_DATA_MINING.GET_ASSOCIATION_RULES

(modelName, topn, NULL, min_confidence, min_support, max_rule_length,

DMSYS.ORA_MINING_VARCHAR2_NT('RULE_CONFIDENCE DESC',

'RULE_SUPPORT DESC'),

DMSYS.ORA_MINING_VARCHAR2_NT(antecedent),

DMSYS.ORA_MINING_VARCHAR2_NT(consequent)));

V rámci kurzoru se následně procházejí jednotlivé řádky a ukládají do pomocné tabulky *PTBMiningResults*. Hodnotami z této tabulky je naplněna i tabulka na této stránce. Tím je proces získávání asociačních pravidel dokončen.

Tato stránka aplikace se bohužel nepodařila ve Form builderu rozjet, výsledky hledání asociačních pravidel tedy byly kontrolovány v konzole. Pro zobrazení počtu asociačních pravidel příkazem:

SELECT COUNT(*) FROM TABLE

(DBMS_DATA_MINING.GET_ASSOCIATION_RULES(modelName)).

Pro zobrazení nalezených asociačních pravidel byl použit příkaz:

*SELECT * FROM TABLE*

(DBMS_DATA_MINING.GET_ASSOCIATION_RULES(modelName)).

Pro konkrétní vyhledávání skupiny asociačních pravidel pak pomocí příkazu:

*SELECT * FROM TABLE(DBMS_DATA_MINING.GET_ASSOCIATION_RULES*

('1', 20, NULL, .5, .1, 4, 2, DMSYS.ORA_MINING_VARCHAR2_NT

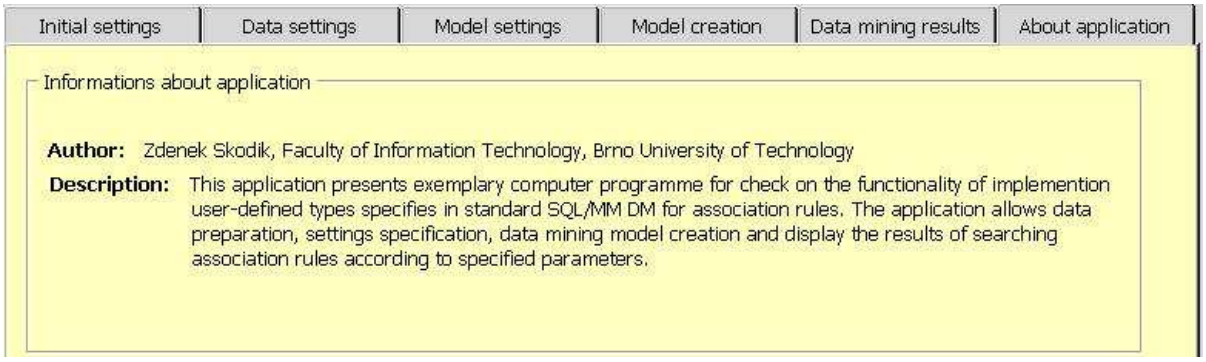
('RULE_CONFIDENCE DESC', 'RULE_SUPPORT DESC'),

DMSYS.ORA_MINING_VARCHAR2_NT('ID_CUSTOMER'),

DMSYS.ORA_MINING_VARCHAR2_NT('KIND')));

V tomto případě si přejeme zobrazit vrchních 20 pravidel modelu 1 seřazených podle hodnoty minimální spolehlivosti a poté i podle hodnoty minimální podpory. Naleznou se pravidla, jejichž levá strana je specifikována řetězcem *ID_CUSTOMER* a pravá řetězcem *KIND* (celkový počet položek pravidla tady může být až 4). Pravidla musí mít alespoň spolehlivost 0.5 a podporu 0.1.

Aplikace ještě obsahuje jednu stránku About application, která nese informace o autorovi a popis účelu programu.



Obr. 15 O aplikaci

8 Závěr

V diplomové práci jsem se zabýval problematikou získávání znalostí z databází. Hlavním úkolem pak byla implementace uživatelských typů předepsaných standardem SQL/MM DM pro asociační pravidla. Aby jej bylo možné dosáhnout, musel jsem nejdříve nastudovat teorii dolování z dat a následně se zaměřit na úlohu získávání asociačních pravidel. Poté jsem mohl postoupit ke standardu SQL/MM DM, odpovídajícím uživatelem definovaných typům a rozboru jejich metod. Přitom jsem počítal s co největší podporou Oracle pro dolování nazvanou Oracle Data Mining, z hlediska cíle diplomové práce byl vhodný zejména modul DBMS_DATA_MINING. Nyní již bylo možné přistoupit k návrhu implementace jednotlivých typů a jejich metod, který jsem pak implementoval v prostředí jazyka PL/SQL. Funkčnost implementace byla ověřena na ukázkové aplikaci vytvořené pomocí nástroje Form Builder 6, obsaženého v Oracle Database 10g release 2, z důvodu efektivnější manipulace s objekty v databázi. Zatímco implementace typů respektovala jejich popis ve standardu SQL/MM DM, ukázková aplikace byla vytvořena v souladu s teorií procesu získávání znalostí a funkcí jednotlivých metod typů. Při realizaci jsem řešil některé problémy, které jsou v diplomové práci zmíněny.

Práce na diplomovém projektu mi přinesla mnoho praktických zkušeností při prací s databází Oracle, získal jsem hlubší znalosti jazyka PL/SQL a seznámil jsem se s vývojovým nástrojem pro tvorbu grafických rozhraní Form Builder 6. Nejen z tohoto pohledu byla tato diplomová práce pro mě osobně velkým přínosem. Rozšířil jsem si znalosti z této oblasti oboru informačních technologií, což považuji za velmi užitečné pro mou další práci. Dolování dat je rozsáhlé a problematické téma, neboť se dolovací algoritmy neustále vyvíjí v důsledku rostoucího objemu dat, která naše civilizace produkuje. Z toho důvodu se v poslední době používá dolování dat stále častěji pro efektivní získávání informací.

Jako další pokračování projektu vidím v nepoužívání pomocných tabulek pro uchovávání informací, se kterými jednotlivé typy pracují. Tato data by bylo možné ukládat do globálního atributu DM_content pomocí neimplementovaných metod pro import/export dat ve formátu XML, pro který již Oracle poskytuje také dobrou podporu. Tímto by i odpadl problém s mazáním informací spojených s instancí určitého typu, nemusely by být mazány záznamy z pomocných tabulek, jak jsem to činil já, ale stačilo by pouze zrušit instanci daného typu. Stejně tak by bylo možné rozšířit ukázkovou aplikaci tak, aby pracovala se všemi implementovanými metodami typů, které jsem do aplikace nezahrnul, jelikož nebyly z uživatelského pohledu získávání asociačních pravidel podstatné. Aby byl vytvořen opravdu efektivní dolovací systém, mohl by obsahovat i jiné dolovací úlohy, než jen asociační pravidla, různé problémy si totiž vyžadují různý přístup.

Literatura

[1] BERKA P. *Dobývání znalostí z databází*. Vydání 1., Praha, Academia 2003

[2] ŠKODÍK Z. *Aplikační rozhraní pro dolování z dat SQL/MM – asociační pravidla*. Ročníkový projekt, Brno, VUT – Fakulta informačních technologií, 2005

[3] SQL Multimedia and Application Packages - Part 6: Data Mining. Standard ISO/IEC 13249-6:2002

[4] Oracle Corporation. *Oracle Database Application Developer´s Guide – Fundamentals*. 10g Release 2 Part No. B14251-01, 2005

[5] Oracle Corporation. *Oracle Database Application Developer´s Guide – Large Objevte*. 10g Release 2 (10.2) Part No. B14249-01, 2005

[6] Oracle Corporation. *Oracle Database Reference*. 10g Release2 (10.2) Part No. B14237-02, 2005

[7] Oracle Corporation. *Oracle Data Mining Concepts*. 10g Release 2 (10.2) Part No. B14339-01, 2005

[8] Oracle Corporation. *Oracle Forms Developer and Oracle Reports Developer*. Release 6i Part No. A73073-02, 2000

[9] Oracle Corporation. *PL/SQL User's Guide and Reference*. 10g Release 2 (10.2) B14261-01, 2005

[10] Oracle Corporation. *Oracle Database SQL Reference*. 10g Release 2 (10.2) Part No. B14200-02, 2005

[11] Hemiš, P. *Implementace standardu SQL/MM DM pro shlukování*. Diplomová práce, Brno, VUT – Fakulta informačních technologií, 2006

Seznam příloh

Příloha 1. DVD se zdrojovými texty typů, aplikace a podobou dokumentace (viz readme.txt)