VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

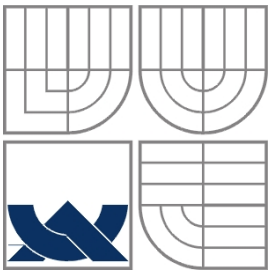
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

DOLOVÁNÍ DAT V PROSTŘEDÍ DB SERVERU ORACLE A MS SQL SERVERU

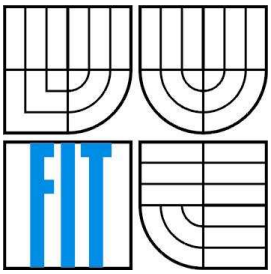
BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE Martin Opršal
AUTHOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

DOLOVÁNÍ DAT V PROSTŘEDÍ DB SERVERU ORACLE A MS SQL SERVERU

DATA MINING ON ORACLE DATABASE SERVER AND MS SQL SERVER

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN OPRŠAL

VEDOUCÍ PRÁCE
SUPERVISOR

ING. LUKÁŠ STRYKA

BRNO 2007

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2007/2008

Zadání bakalářské práce

Řešitel: **Opršal Martin**
Obor: Informační technologie
Téma: **Dolování dat v prostředí DB serveru Oracle a MS SQL Serveru**
Kategorie: Databáze

Pokyny:

1. Seznamte se s problematikou získávání znalostí z databází.
2. Seznamte se s podporou pro dolování dat v prostředí Oracle a MS SQL Serveru.
3. Navrhněte ukázkové aplikace pro implementaci dolování dat v prostředí DB serveru Oracle a MS SQL Server.
4. Aplikace realizujte a jejich funkčnost ověřte na vhodném vzorku dat.
5. Zhodnoťte dosažené výsledky a porovnejte obě prostředí. Diskutujte další možná rozšíření systému.

Literatura:

- Han, J., Kamber, M.: *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers. 2001.
- *Oracle Data Mining*. Dokumentace dostupná v online knihovně Oracle.
- Bain T., et.al.:*SQL Server 2000 Data Warehousing with Analysis Servives*

Při obhajobě semestrální části projektu je požadováno:

- Body 1 a 2.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Stryka Lukáš, Ing.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2007
Datum odevzdání: 14. května 2008

L.S.

doc. Ing. Jaroslav Zendulka, CSc.
vedoucí ústavu

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami

1. **Pan**

Jméno a příjmení: **Martin Opršal**
Id studenta: 78820
Bytem: Ludmily Konečné 1, 639 00 Brno
Narozen: 13. 11. 1985, Brno
(dále jen "autor")

a

2. **Vysoké učení technické v Brně**

Fakulta informačních technologií
se sídlem Božetěchova 2/1, 612 66 Brno, IČO 00216305
jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....
(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
bakalářská práce

Název VŠKP: Dolování dat v prostředí DB serveru Oracle a MS SQL Serveru
Vedoucí/školitel VŠKP: Stryka Lukáš, Ing.
Ústav: Ústav informačních systémů
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:

tištěné formě počet exemplářů: 1
elektronické formě počet exemplářů: 2 (1 ve skladu dokumentů, 1 na CD)

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizování výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti:
   - ☐ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

.....	.....
Nabyvatel	Autor

Abstrakt

Bakalářská práce se zabývá problematikou získávání znalostí z dat. Práce je zaměřena na získávání pravidel z relačních databází na serverech Microsoft SQL a Oracle Data mining serveru. V praktické části je popsán návrh aplikace pro dolování asociačních pravidel pod oběmi servery. V aplikacích jsou použity programovací jazyky asp.NET,C# pro Microsoft SQL server a Java pro Oracle server.

Klíčová slova

Dolování dat z databází, získávání znalostí z dat, asociační pravidla, Apriori, Bayesovská klasifikace, Shluková analýza, Oracle Data Mining server 10.2, Oracle Data Mining server 11g, Microsoft SQL 2005.

Abstract

This bachelor‘s thesis deals with issue of knowledge discovery in databases. This document is focused in getting rules from relation databases based on Microsoft SQL server or Oracle Data mining server. The practical part of this document is about design applications that run on both servers. These applications are programmed in asp.NET, C# for Microsoft SQL server and Java for Oracle server.

Keywords

Knowledge discovery in data, data mining in database, association rules, Apriori, Bayes‘s classification, Clustering, Oracle Data Mining server 10.2, Oracle Data Mining server 11g, Microsoft SQL 2005

Citace

Opršal Martin: Dolování dat v prostředí DB serveru MS SQL a Oracle. Brno, 2008, bakalářská práce, FIT VUT v Brně.

Dolování dat v prostředí DB serveru MS SQL a Oracle

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením ing.Lukáše Stryky
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Jméno Příjmení
Datum

Poděkování

Děkuji tímto ing. Lukáši Strykovi za pomoc a podporu při řešení tohoto projektu.

Obsah

Úvod

Díky velkému nárůstu dat v databázích, přestala být data přehledná a byla snaha tyto data pochopit. Kvůli tomu vzniklo odvětví dolování znalostí z databází, které se zabývá vypracováním metod, které nám dávají možnost tyto rozsáhle data studovat a posléze z nich odvozovat závěry.

Hlavním propagátorem rozvoje získávání znalostí byl marketing, potažmo obchod, který dokázal nashromáždit data o transakcích velice rychle, tyto data sami o sobě nemají žádnou vypovídající hodnotu. Z pohledu marketingu však bylo třeba získat z těchto dat závěry, pro lepší a efektivnější rozhodování, potažmo lepšímu řízení celé firmy. Tedy například jedná-li se o cestovní kancelář, po prostudování dat můžeme určit odběratele nabídkového katalogu dovolených, například pro adrenalinové dovolené zvolíme raději skupinu do 30-let, protože právě ti jezdí na tyto dovolené častěji nežli lidé v důchodovém věku. Postupem času se dolování znalostí rozšířilo i do dalších odvětví.

Tato práce slouží jako úvod do problematiky dolování dat, používá k tomu dnes dva nejrozšířenější databázové servery Microsoft SQL server a Oracle Data Mining server, na těchto serverech implementuje aplikace, které z dat získají asociační pravidla, která jsou spolu s dalšími nejpoužívanějšími algoritmy popsána v teoretické části.

1 Základní pojmy

Získávání znalostí z databází (KDD – Knowledge Discovery in Databases) – obecný iterativní a interaktivní proces získávání užitečných znalostí z dat. Jedná se o netriviální získávání implicitních, dosud neznámých, pochopitelných a potenciálně užitečných znalostí z databází.

Dolování dat (Data Mining) – konkrétní aplikace nebo algoritmy pro získávání vztahů z dat.

Data je množina F faktů

Vzorek je výraz E v jazyku L popisující fakta v podmnožině $F_E \subset F$. E se nazývá vzorek, je-li jednodušší než výčet faktů z F_E.

Proces je obvykle vícekrokový. Zahrnuje přípravu dat, vyhledávání vzorků, vyhodnocení znalostí a zjemnění včetně iterace po modifikaci. Je netriviální.

Platnost Získaný vzorek by měl být platný i pro nová data s jistým stupněm určitosti.

Novost vzorku může být měřena vzhledem ke změnám v datech nebo znalostech.

Potenciální užitečnost Vzorky by měly vést k nějakým užitečným akcím.

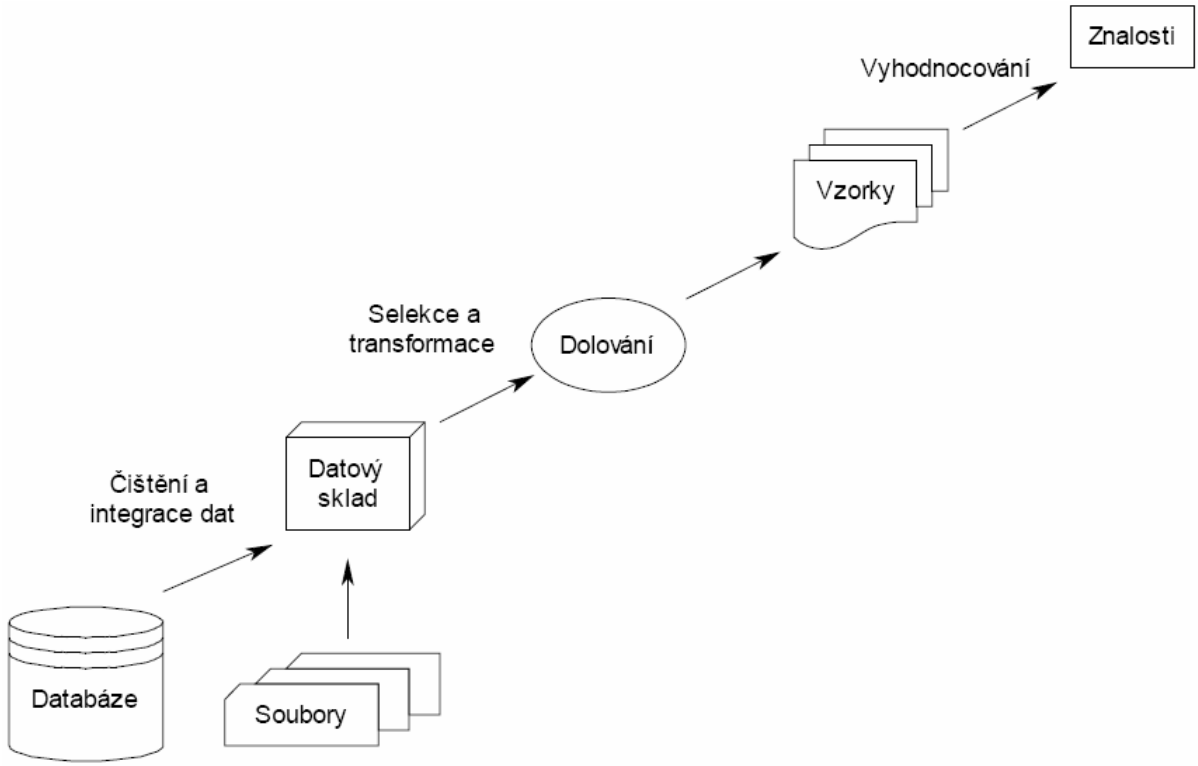
Pochopitelnost Cílem KDD je získat vzorky pochopitelné člověku, aby lépe poznal zkoumaná data.

Zajímavost je obecná vlastnost vzorku kombinující platnost, novost, užitečnost a jednoduchost.

Znalost Vzorek $E \in L$ je nazýván znalost, když pro nějaký daný práh i platí, že znalost vzorku je větší než tento zvolený práh.

Někdy se pod pojmem dolování dat označuje celý proces získávání znalostí, dolování dat a získávání znalostí jsou tedy synonyma. Někdy se do procesu získávání znalostí zahrnuje i předzpracování, vzorkování a transformace vstupních dat získaných z databáze. [3]

1.1 Proces získávání znalostí



Obr 1. – Proces získávání znalostí

1. **Čištění dat** (data cleaning) – slouží k odstranění šumu a nekonzistentních dat
2. **Integrace dat** (data integration) – kde mohou být kombinovány vícenásobné zdroje dat. Trendem se stává vyčištění a integraci dat začlenit jako předzpracování, jehož výsledek se uloží do datového skladu.
3. **Selekce dat** (data selection) – kde jsou data relevantní pro úlohu získávána z databáze
4. **Transformace dat** (data transformation) – kde jsou data transformována nebo sdružována do tvaru vhodného pro dolování zahrnující operace agregace a shrnutí.
5. **Dolování dat** (data mining) – jádro procesu, ve kterém jsou na data aplikovány metody vedoucí k získání vztahů mezi daty.
6. **Vyhodnocení vztahů** (pattern evaluation) – identifikace vztahů representující potenciálně zajímavé znalosti na základě vhodných metrik.
7. **Prezentace znalostí** (knowledge presentation) – kde jsou použity vizualizační a prezentační techniky k prezentování získaných znalostí uživateli. [3]

2 Algoritmy pro dolování dat

Zde uvedu nejpoužívanější metody pro dolování dat. Ve svých aplikacích jsem později použil asociační pravidla, díky čemuž se jim zde budu věnovat trošku víc.

2.1 Asociační pravidla

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

Termín asociační pravidla široce zpopularizoval počátkem 90.let Agrawal v souvislosti s analýzou nákupního Košíku. Při této analýze se zjišťuje, jaké druhy zboží si současně kupují zákazníci v supermarketu (např. pivo a parky). Jde tedy o hledání vzájemných vazeb (asociací) mezi různými položkami sortimentu prodejny. Přitom není upřednostňován žádný speciální druh zboží jako závěr pravidla. [1]

2.1.1 Formální zápis asociačních pravidel

Nechť $I=\{ i_1, i_2, i_3,...\}$ je množina literálů nazývaných položky,

D je množina transakcí, kde každá transakce T je množina položek taková, že $T \subseteq I$. Množství položek se neuvažuje – zajímá nás pouze přítomnost / nepřítomnost položky v transakci.

Ke každé transakci přísluší unikátní identifikátor *TID*. Nechť X je množina položek. Říkáme, že transakce T obsahuje X tehdy a jen tehdy, pokud $X \subseteq T$.

Asociační pravidlo je implikace tvaru $X \Rightarrow Y$, kde $X \subset T$, $Y \subset T$ a $X \cap Y = \varnothing$.

Pravidlo $X \Rightarrow Y$ má **podporu (support)** ***supp*** v množině transakcí D, jestliže *supp%* transakcí v D obsahuje množinu položek $X \cup Y$.

Pravidlo $X \Rightarrow Y$ platí v množině transakcí se **spolehlivostí (confidence)** ***conf***, jestliže *conf%* transakcí, které obsahují X obsahuje také Y. Spolehlivost je vlastně podmíněná pravděpodobnost, tedy pro pravidlo $X \Rightarrow Y$ je to pravděpodobnost, že transakce obsahuje Y za předpokladu, že obsahuje X.

Číselně ji lze vyjádřit jako: $conf(X,Y) = \frac{\text{supp}(X \cup Y)}{\text{supp}(X)}$.

Pomocí těchto dvou parametrů (podpory a spolehlivosti) lze určit, jak statisticky významné pravidlo jsme z databáze získali, tj. jak je pro nás zajímavé. Obyčejně volíme nějakou spodní mez minimální podpory a minimální spolehlivosti, pomocí kterých se vylučují ta pravidla, která pro nás nejsou tak zajímavá. [3]

Pro ilustraci zde uvedu příklad, mějme databázi D definovanou touto tabulkou:

TID	Položky
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

množina položek: I={1,2,3,4,5},

množina transakcí: D={{1,3,4},{2,3,5},{1,2,3,5},{2,5}}

supp (3⇒5) = 2 / 4 *100=50%

conf (3⇒5) = (2/4) / (3/4) *100 = 2/3 * 100 =66%

Pravidlo 3⇒5 má podporu supp = 50%, protože množina položek {3,5} je obsažena v transakcích 200 a 300 – polovina počtu transakcí, a platí se spolehlivostí 66% protože položka 3 je ve 3 transakcích, z nichž ve dvou je i 5.

2.1.2 Algoritmus Apriori

Apriori je algoritmus pro získávání frekventovaných množin položek pro booleovská asociační pravidla. Název algoritmu je založen na faktu, že algoritmus využívá předchozí znalosti (prior knowledge) vlastností frekventovaných množin s využitím tzv. *Apriori podmínky* (viz. algoritmus Apriori_gen). Apriori využívá iterativní přiblížení známé jako úrovňově rozumné hledání (level-wise search), kde *k-množiny* (množiny mající právě k-prvků) jsou využívány k zjišťování (k+1)-množin. Pro zlepšení výkonu algoritmu lze pro vyhledávání kandidátů v transakci využít hashovací strom, což je strom jehož vnitřní uzly jsou hashovací tabulky. Datové složky těchto tabulek jsou odkazy na synovské uzly daného uzlu. Listové uzly tohoto stromu obsahují samotné kandidáty (kandidátní množiny, tj. potenciálně frekventované množiny). Funkce *subset* prohledává tento strom, pokud jsme došli až do listu, pak jsme nalezli kandidáta obsaženého v aktuální transakci, jinak pokračujeme do hloubky na uzel, jenž vznikl hashováním položky, která následuje po položce následující. [3]

Samotný algoritmus:

Vstupem je databáze *D*, minimální hodnota podpory min_supp

Výstupem je *L* – frekventované množiny v *D*

Metoda:

1) L_1 = nalezni_frekventované_1-množiny(*D*);
2) **for** (*k*=2; $L_{k-1} \neq \emptyset$; *k*++) {
3) C_k=apriori_gen(L_{k-1},*min_supp*); //generování nových kandidátů

4) **for each** transakce $t \in D$ { //projdi D pro zjištění počtů výskytů
5) C_t=subset(C_k, t); // kandidátské množiny obsažené v transakci t
6) **for each** kandidát $c \in C_t$
7) *c.počet_výskytů*++;
8) }
9) Lk={c ∈ C_k | c.počet_výskytů ≥ min_supp};
10) }
11) **return** $L=U_k L_k$;

Apriori_gen – procedura pro generování k-množin

Pracuje ve dvou krocích – *slučovacím* a *vylučovacím*. Slučovací krok vytváří z výsledku předchozího kroku (frekventované (k-1)–množiny) nové kandidáty na frekventované k-množiny. Nová k-množina je výsledkem sloučení dvou (k-1)-množin, které se liší pouze v jednom prvku. Vstupem jsou tedy dvě lexikograficky seřazené množiny. Jsou to množiny M_1={*p[1],p[2],...,p[k-1]*} a M_2={*q[1],q[2],...,q[k-1]*}, pro které platí: *p[1]=q[1],...,p[k-2]=q[k-2]* a pouze *p[k-1]* ≠ *q[k-1]*, výsledkem je nová k-množina { *p[1],p[2],...,p[k-1],q[k-1]*}, jejíž prvky jsou rovněž lexikograficky seřazeny. Druhý krok je vylučovací, z výsledné množiny jsou odstraňovány k-množiny, jejichž některá (k-1)-podmnožina se nevyskytuje ve frekventovaných (k-1)-množinách, protože podpora jakékoliv množiny nemůže být větší než podpora jakékoliv její podmnožina, což je tzv. *Apriori podmínka*. [3]

procedure aprori_gen(L_{k-1}; frekventovaná (k-1)-množina; *min_supp*)

1) **for each** množina $l_1 \in L_{k-1}$
2) **for each** množina $l_2 \in L_{k-1}$
3) **if** $((l_1[1]=l_2[1]) \wedge (l_1[2]=l_2[2]) \wedge ... \wedge (l_1[k-2]=l_2[k-2]) \wedge (l_1[k-1]=l_2[k-1]))$ {
4) $c=l_1 \cup l_2$; //slučovací krok – generování kandidátů
5) **if** (má_nefrekventovanou_podmnožinu(c,L_{k-1}))
6) odstraň *c*: // vylučovací krok, odstraňování nefrekventovaných kandidátů
7) **else** přidej *c* do C_k;
8) }
9) **return** C_k;

2.2 Rozhodovací stromy

Způsob reprezentování znalostí v podobě rozhodovacích stromů je dobře znám z řady oblastí. Vzpomeňme jen na nejrůznější „klíče k určování“ různých živočichů nebo rostlin známých z biologie. Indukce rozhodovacích stromů patří k nejznámějším algoritmům z oblasti symbolických

metod strojového učení. Při tvorbě rozhodovacího stromu se postupuje metodou rozděl a panuj. Trénovací data se postupně rozdělují na menší a menší podmnožiny (uzly stromu) tak, aby v těchto podmnožinách převládaly příklady jedné třídy. Na počátku tvoří celá trénovací data jednu množinu, na konci máme podmnožiny tvořené příklady téže třídy. Pro tvorbu stromů se používá obecný algoritmus **TDIDT**. [1]

1. zvol jeden atribut jako kořen dílčího stromu
2. rozděl data v tomto uzlu na podmnožiny podle hodnot zvoleného atributu a přidej uzel pro každou podmnožinu
3. existuje-li uzel, pro který nepatří všechna data do téže třídy, pro tento uzel opakuj postup od bodu 1, jinak algoritmus ukonči

2.3 Shluková analýza

Neboli anglicky Clustering, je proces rozdělování objektů do tříd na základě podobnosti mezi objekty. Snažíme se aby objekty ve shluku si byli podobny maximálně naopak prvky různých shluků se podobaly minimálně. Podobnost objektů se posuzuje na základě hodnot jednotlivých atributů objektu. Často je shluková analýza použita jako předzpracování dat pro další algoritmy. Shlukování je, z hlediska strojového učení, učení bez učitel, tedy nevyžaduje žádné předdefinované třídy ani žádnou trénovací množinu příkladu, objekty o kterých víme do jaké třídy patří. [5]

2.3.1 K-means

Asi nejznámější algoritmem je algoritmus K-means který rozděluje množinu objektu do několika předem určených tříd. Je založen na fiktivním centrálním bodu, tedy každá třída je reprezentována tímto bodem, ze začátku se střed určí náhodně, při dalších krocích se počítá jako střed mezi objekty které do třídy patří, samozřejmě každý objekt může patřit pouze do jedné třídy.

Ukázka algoritmu:

1. vyber k(*počet shluků*) objektu z *trénovaných dat* jako počáteční středové body
2. přiřaď všechny objekty z *trénovaných dat* do shluků, podle největší podobnosti ke středovým bodům
3. znovu vypočítej středy shluků
4. pokud nastala změna ve shlucích vrat se k bodu 2 jinak algoritmus ukonči.

Opět pro ukázku uvedu příklad, mějme množinu bodů {-5,-2,0,1,2,5,6} a tyto body chceme rozdělit do 3 shluku {a,b,c}, jako počáteční středové body jsme náhodně vybrali body: -2,1,2. V dalším kroku vypočítáme vzdálenosti bodu od jednotlivých shluku tedy pro bod -5 je vzdálenost k shluku a=3, b=6 a c=7, tedy bod se přiřadí do shluku a.

Shluk a: body {-5,-2} a nový střed (-5 - 2) / 2 = -3,5

Shluk b: {0,1} a nový střed = 0,5

Shluk c: {2,5,8} a nový střed = 5

Opět znovu přiřadíme body ke shlukům

Shluk a: {-5,-2} a nový střed = -3,5

Shluk b: {0,1,2} a nový střed = 1

Shluk c: {5,8} a nový střed = 6,5

A opět znovu přiřadíme body ke shlukům, vyjde stejné přiřazení jako v předešlém kroku, tedy algoritmus ukončíme.

Algoritmus pracuje dobře pokud jsou data poměrně dobře vzdálené, nevýhodou metody je že musíme předem znát počet shluků.

2.4 Bayesovská klasifikace

Metody bayesovské klasifikace vycházejí z Bayesovy věty o podmíněných pravděpodobnostech. Ačkoliv tedy jde o metody pravděpodobnosti, jsou intenzivně studovány v souvislosti se strojovým učením a uplatňují se rovněž v systémech pro dobývání znalostí.

Bayesův vztah pro výpočet pravděpodobnosti, že do třídy C náleží prvek X má podobu

$$P(C_i \mid X) = \frac{P(X \mid C_i)P(C_i)}{P(X)}$$

Kde $C_1, C_2 .. C_m$ (C_i) značí jednotlivé třídy, do kterých jsou prvky klasifikovány.

Kde X značí testovaný vzorek, který má být do nějaké třídy přiřazen.

Vzhledem k tomu, že nás nezajímá konkrétní hodnota můžeme zanedbat jmenovatel P(X) a budeme hledat maximum z $P(X| C_i) P(C_i)$, kde $P(C_i)$ je pravděpodobnost, že libovolný prvek patří do třídy C_i. $P(X| C_i)$ je pravděpodobnost, že libovolný prvek ze třídy C_i bude mít stejné hodnoty atributů jako prvek X.

Pro ilustraci zde uvedu příklad, použiji jednoduchou úlohu z poskytování úvěrů. Předpokládáme že banka půjčí 2/3 lidí tedy P(půjčit) = 0,667 a naopak P(nepůjčit) = 0,333. Dále předpokládáme, že vysoký příjem mělo 91% klientu, kterým banka půjčila a naopak nízký příjem mělo 88% klientům, kterým banka nepůjčila, tedy

P(vysoký příjem | půjčit) = 0,91

P(vysoký příjem | nepůjčit) = 0,12

P(nízký příjem | půjčit) = 0,09

P(nízký příjem | nepůjčit) = 0,88

Posuzujeme klienta s vysokým příjmem. Bude větší pravděpodobnost, že banka půjčí nebo že nepůjčí?

P(vysoký příjem | půjčit) P(půjčit) = 0,91 * 0,667 = 0,607

P(vysoký příjem | nepůjčit) P(nepůjčit) = 0,12*0,333 = 0,040

Z toho vyplívá, že banka klientovi půjčí. Pokud by bylo víc atributu nežli výška příjmu (např. majetek) postupovalo by se obdobně s tím rozdílem že výsledná pravděpodobnost by se vypočítala s mezikrokem kde se spočítá součin dílčích pravděpodobností, tedy

P(X | půjčit) = P(velký majetek | půjčit) × P(vysoký příjem | půjčit)

P(X | půjčit) P(půjčit) = pravděpodobnost [1,4]

3 MS SQL server

Aktuální je MS SQL server 2008, ale pro svou práci jsem použil starší server MS SQL server 2005. Veškera dokumentace je popsána v [10], stránky jsou i v češtině, ale bohužel nejsou přeloženy všechny její části, takže jsem zvolil plně anglickou dokumentaci.

3.1 Dolování dat v MS SQL

Microsoft SQL server nabízí několik služeb, database server pro OLTP databáze, analysis server pro OLAP, dolování dat a vytváření multidimenzionálních kostek a několik další služeb (reporting server, Integration server, atd.). Analysis server nabízí pro vytváření dolovacích modelu jazyk DMX, jedná se o rozšíření jazyku SQL, nebo komunikací pomocí XML potažmo SOAP, bohužel veškeré úkoly se musí vytvářet psaním skriptů. Právě komunikaci pomocí XML využívá Microsoft Visual Studio 2005, které díky uživatelsky příjemnějšímu prostředí umožňuje definovat model dolování dat pomocí intuitivních průvodců, stejně jako zobrazovat výsledky.

3.1.1 Dolování dat pomocí DMX jazyka

Jedná se o komunikaci klient-server kdy klient posílá požadavky na server, který požadavek povede a pošle zpět klientovi odpověď.

První krok k vytvoření modelu je příkaz *CREATE MINING MODEL [jméno modelu] (seznam sloupců s parametry)*. Tento příkaz vytvoří na serveru nejen model ale i strukturu, ve které se model nachází. Platí, že jedna struktura může mít víc modelů z jednoho databázového zdroje, lze tedy do struktury vytvářet modely, které se odlišují nastavením algoritmů, ale všechny modely budou mít stejná data. Druhý krok vytváření modelu je právě nahrání dat do vytvořené struktury příkazem *INSERT INTO [jméno modelu] (výčet sloupců) VALUES zdroj dat*. Zdroj dat lze nadefinovat příkazem *OPENROWSET(poskytovatel, připojovací řetězec, výběr dat)*. Tento příkaz se připojí na databázový server a vybere z něj dat. Je však nutné, aby server měl povoleno přistupovat k datům pomocí ad-hoc komunikace, která je při počátečním nastavení vypnutá. Tím máme vytvořený model a můžeme z něj přečíst výsledek pomocí příkazu *select * from [jméno struktury] nebo [jméno struktury].content*. Kde druhá, výše zmíněná, tabulka pak obsahuje již setříděná pravidla, tabulka obsahuje veškeré výsledky ze struktury a rozdělení algoritmů je pomocí sloupce *node_type*. [6,10]

Ukázka vytvoření modelu a vyčtení výsledků:

```
Create Mining Model MovieAssociation (
      Customer_id long key,
      Gender text discrete predict,
```

```
      Marital_Status text discrete predict
)
Using Microsoft_Association_Rules (Minimum_support = 0.02,
Minimum_probability = 0.40)

Insert into MovieAssociation (Customer_id,Gender,Marital_status)
OPENROWSET ('MSDataShape', 'data
provider=SQLOLEDB;server=myserver;UID=login;pwd=mypass', 'Select
Customer_id,Gender,Marital_status from NewCustomers')

Select node_caption, node_probability, node_support from
MovieAssociation.content where node_type = 8
```

3.1.2 Dolování pomocí XML

Druhý způsob dolování dat je postaven na komunikaci pomocí XML souboru a taktéž je postaven na principech klient-server. Tento způsob jsem si posléze vybral k vytvoření ukázkové aplikace.

Vytvoření modelu se skládá z více kroků stejně jako při jeho vytváření pomocí DMX jazyka. Nejprve je potřeba vytvořit zdroj dat data source, který slouží k nadefinování spojení mezi analysis servem (na kterém vytváříme model) a database serverem (na kterém jsou uložena data). Dalším krok je vytvořeni tzv. pohledu (anglicky: data source view). Zde v XML nadefinujeme sloupce z databázové tabulky nebo pohledu používat, které chceme používat. Pro připojení k database serveru používáme komunikaci nadefinovanou v data source. Poté následují 2 kroky, které lze spojit do jednoho XML. Jedná se o vytvoření struktury, ve které se nachází modely a vytvoření modelu ve struktuře. Oba požadavky se dají samozřejmě poslat odděleně. Nový model se dá rovněž přidat do již existující struktury. Co se týče vytváření struktury tak opět nadefinujeme vyčet sloupců pro modely ve struktuře a v modelu pak určíme ke sloupcům role. Tím máme model nadefinovaný, chybí pouze poslední věc a to ho spustit, nebo, pokud se změnila tabulka dat např. přidáním sloupců, ho stejným XML přebudovat a následně spustit.

Server na jakýkoliv výše uvedený dotaz vrací XML s potvrzením provedení, popřípadě s chybou. Oba vrácené XML dokumenty mají stejnou strukturu s tím rozdílem, že při chybě se ve struktuře objeví tag se jménem *Message* a jeho potomci, tagy *Error*, jsou jednotlivé chyby, které při vykonávání příkazu nastali. Nejdůležitější je atribut *Description*, ve kterém je popis chyby. Pokud dotaz proběhl v pořádku, je tag *Message* prázdný. Ukázky XML souborů jsou v příloze.

3.1.3 Microsoft Visual Studio 2005

Komerční program určený nejen pro dolování dat. Po připojení k analysis serveru (File->Open->Analysis Services Database) a po výběru databáze, se v levé části obrazovky objeví stromová struktura serveru. Kde je možné vytvářet objekty, které jsou popsány v kapitole 3.1.2 a i další, které jsem však pro své potřeby nepotřeboval. Při vytváření objektů uživatele provede grafický průvodce, pomocí kterého se dají veškeré parametry nastavit.

3.2 Analýza aplikace

Úkolem aplikace bude demonstrovat proces získávání znalostí z databáze v prostředí Microsoft SQL Server 2005.

3.2.1 Neformální specifikace

Aplikace bude mít za úkol, postupně zpracovávat požadavky od uživatele a vytvořit na serveru data source, data source view a model se strukturou. Tyto požadavky bude postupně odesílat na server, který je vykoná a aplikace si následně zažádá o výsledky, které zobrazí formou asociačních pravidel. Jako výstup aplikace bude použita webová klientská aplikace.

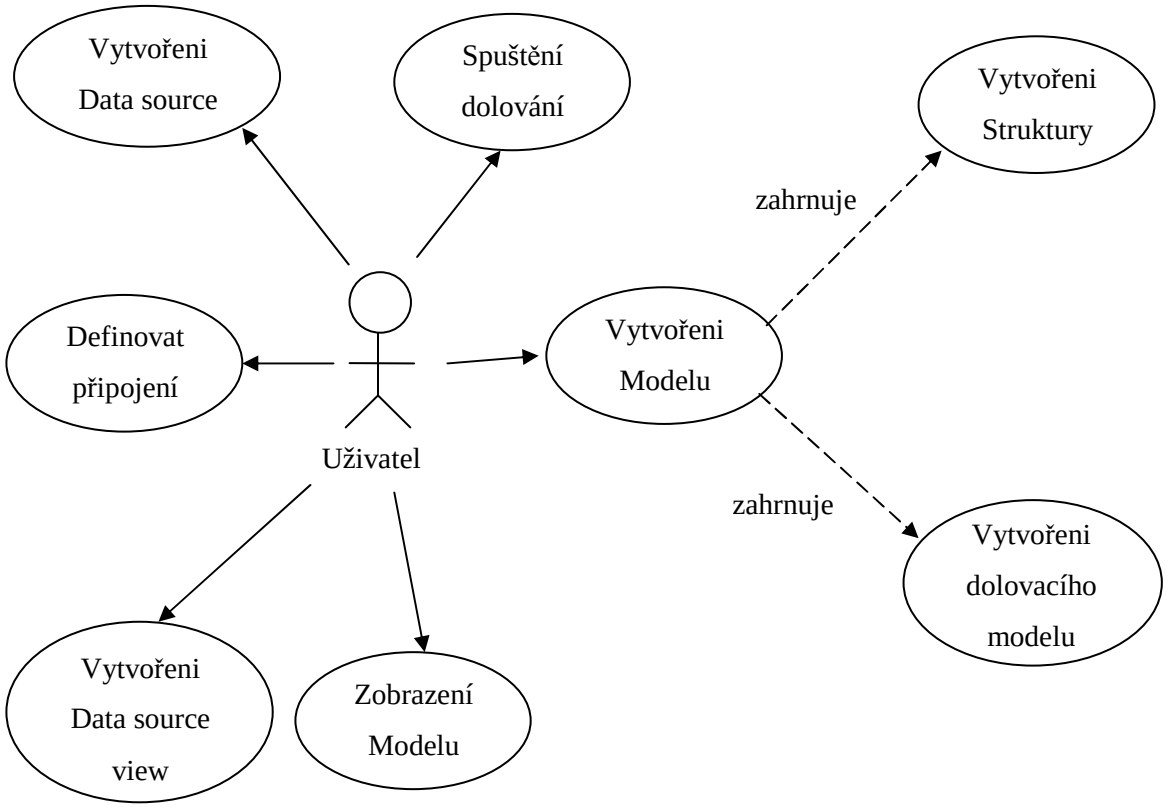
3.2.2 Ovládání programu

Pro ovládání programu je počítáno s internetovým prohlížečem. Jedná se o ukázkovou aplikace, proto bude moci uživatel ovlivnit pouze nezbytné parametry, složitější nastavení bude moci být předáno v URL stránky. Samotná aplikace bude mít intuitivní ovládání a bude se snažit navádět uživatele ke správnému pořadí provedení úloh. Jako výstup aplikace bude sloužit opět jen klientská webová stránka.

3.2.3 Výběr technologie pro implementaci programu

Aplikace bude napsána v kombinaci jazyku C# a asp.NET. Asp.NET je jazyk určený pro webové služby. C# je objektově orientovaný jazyk. Asp.NET slouží jako grafické prostředí pro C#. Tyto jazyky jsem rozhodl vybrat kvůli dobré podpoře komunikace s MS SQL serverem a také, do budoucna, pro jednoduchý přístup k aplikaci pomoci webové služby, potažmo i nezávislosti na klientském operačním systému.

3.2.4 Diagram použití



Obr. 2 – Diagram případů použití

3.3 Použité technologie

Kromě níže zmíněných technologií jsem použil i základy HTML, kaskádové styly a XML, hlouběji je nebudu popisovat, není to předmětem práce.

3.3.1 Asp.NET a C#

Nejprve bude zmíněn **Asp.NET.** Jedná se o jazyk určený pro implementaci webových aplikací. Pro svoji funkčnost potřebuje Microsoft IIS server (webový server). **C#** (taktéž C sharp) je moderní objektově orientovaný jazyk, založený na syntaxi podobné jazyku C++. Asp.NET slouží pouze jako grafické rozhraní pro celý program, které veškeré podměty od uživatele předává do C#.

Webová stránka je tvořena formuláři, které obsahují jednotlivé prvky, jako jsou tlačítka nebo textová pole, používané pro předávání parametrů do C#. Parametry se předávají metodou POST (popřípadě i GET), tedy každý parametr je poslán v HTTP paketu na server. Naopak C# se k

jednotlivým prvkům nadefinovaných v asp.NET chová jako k objektům a je možné do nich rovnou zapsat použitím některé z metod. Naopak C# se k jednotlivým prvkům nadefinovaných v asp.NET chová jako k objektům a je možné do nich rovnou psát použitím některé z metod.

3.3.2 Adomd client a další knihovny

Veškeré informace v kapitole jsem čerpal z [8,9], kde lze nalézt podrobnější informace o třídách. Zde jsem uvedl pouze ty, které jsem potřeboval pro tvorbu aplikace.

Jelikož C# obsahuje pouze třídy pro práci s databázovým serverem a ne s analysis serverem, musel jsem se poohlédnout po externí knihovně, která by poskytovala metody pro připojení k analysis serveru. Tyto podmínky splňuje právě Microsoft.AnalysisServices.AdomdClient.dll.

Třídy, které pro potřeby aplikace použiji, jsou AdomdConnection pro navázání a udržování spojení k serveru, AdomdCommand pro odesílání příkazu na server a AdomdDataReader pro přečtení odpovědi ze strany serveru. Knihovna obsahuje ještě další třídy pro tvorbu dolovacího modelu, OLAP analýzu atd. Díky tomu, že jsem se rozhodl celý model vytvořit ručně pomocí XML, jsem nemusel tyto třídy dál zkoumat a ani používat.

Pokud se jedná o externí knihovnu, musí být na webu přidána do podsložky bin umístěné v kořenovém adresáři, odtud totiž asp.NET, potažmo C#, načítá jakékoliv knihovny.

Microsoft.AnalysisServices.AdomdClient.AdmomdConnection

AdmomdClient(string connectionString) je konstruktor třídy pro navázání připojení k analysis serveru, kde connectionString obsahuje povinný parametr „Data Source=“ s názvem serveru. dále pak parametr „Initial Catalog=“ pro výběr databáze na serveru (tento parametr nemusí být u všech příkazů uveden, protože často je databáze vybírána v XML dotazu). Všechny parametry jsou oddělené středníkem. Příklad připojení k serveru vypadá následovně: „Data Source=localhost;Initial Catalog=datamining“. Konstruktor může být volán i s jinými parametry (bližší informace lze nalézt v [4]). Metoda Open() slouží k otevření spojení se serverem, metoda Close() pak slouží k ukončení spojení se serverem. Návratový typ obou metod je void.

Microsoft.AnalysisServices.AdomdClient.AdmomdCommand

Základem třídy je opět konstruktor, který se stará o vytvoření a odeslání příkazu na server. Obvykle se používá s parametry AdomdCommand(string commandText,AdomdConnection connection), kde commandText je text příkazu, který chceme vykonat na serveru (v mém případě ve formátu XML), a kde connection je spojení, které jsme navázali pomocí předchozí třídy. Konstruktor může opět obsahovat i jiné parametry. Třída dále obsahuje metodu ExecuteReader(), která má za úkol přečíst výsledek, návratový typ je AdomdDataReader (popsáno níže). Další užitečnou metodou je ExecuteXmlReader(), která opět přečte výsledek, ale vrátí objekt typu XmlReader (tedy xml). Tato

třída je vhodná právě proto, že server odpovídá na požadavek ve stejném formátu, v jakém byl poslán, tedy pokud je posláno XML, odpoví XML.

Microsoft.AnalysisServices.AdomdClient.AdomdDataReader

Tato třída slouží pro přečtení výsledku. Pomocí metody NextResult() s návratovým typem boolean se postupně posouváme po řádcích odpovědi. Jednotlivé hodnoty získáme pomocí metod getChar, getFloat, apod, které jako parametr mají index sloupce např. v příkazu select. Návratová hodnota koresponduje s typem metody, tedy Char, Float, atd.

System.Xml.XmlReader

Poslední, zde zmíněná třída už patří mezi vestavěné knihovny, jedná se o XML parser. Z této třídy používám metodu ReadToFollowing(string tagName) která přesune kurzor na element určený pomoci tagName, pokud takový element nenalezne vrací false. Druhá metoda je getAttribute(string name) tato metoda přečte atribut určený pomocí name a vrátí objekt typu string.

3.4 Popis implementace

Uživatelské rozhraní je vytvořeno pomocí HTML. Celá aplikace se nachází na jediné stránce a v závislosti na stisku konkrétního tlačítka jsou volány příslušné metody.

3.4.1 Struktura

V této části jsou popsány třídy, které slouží především pro odeslání příkazu (požadavku) na server. Jednotlivé ukázky příkazů jsou popsány v příloze.

public void Page_Load(object sender,EventArgs e)

Jedná se o metodu, která se zavolá při každém načtení stránky, jejím úkolem je nastavení a vytváření připojení k serveru, které si potom jednotlivé další metody otevírají samy a po vykonání požadavku si je i samy zavřou.

public void data_Source(object sender,EventArgs e)

Tato metoda má jediný úkol, a to vytvořit data source na serveru pomocí specifikovaného připojovacího řetězce.

public void data_Source_View(object sender,EventArgs e)

Slouží k vytvoření data source view na serveru, k výběr sloupců z databázové tabulky slouží metody data_Source_View_addInt(string jméno) a data_Source_View_addString(string jméno, int délka).

public void create_Model(object sender,EventArgs e)

Je volána jako reakce na tlačítko „Create model“, vytvoří strukturu a model zavoláním metody create_Mining_Model.

public void build_Model(object sender,EventArgs e)

Zavolá se při stisku tlačítka „Process Model“, na server se odešle požadavek na zpracování modelu, je nezbytné, aby byl vyplněn název modelu.

public void show_Rules(object sender,EventArgs e)

Slouží k zobrazení výsledku modelu, je volána jako reakce na stisk tlačítka „Show association ruless“. Nejdříve zkontroluje jestli jsou všechny parametry zadány, pokud některý chybí doplní za něj počáteční hodnotu. Potom odešle příkaz na server, po obdržení výsledků vypíše prvních několik pravidel (podle zadaných kriterii) na obrazovku.

private string create_Mining_Model(string name)

První pomocná funkce která slouží k vytvoření modelu. Volá se při create_Model. Opět je potřeba definovat sloupce k tomu slouží *addColumn_Mining_Model(string name,bool key,bool predict)*.

private boolean get_xml_results(string name), get_xml_results()

Pomocná metoda která odešle XML příkaz na server a po přijetí odpovědi zjistí, zda nedošlo k chybě. Parametr name slouží k výpisu chyby, kdy se před chybu dá jméno akce, při které chyba nastala. Tato metoda také otevírá a uzavírá spojení se serverem.

private string rules_format(string rules)

Slouží k zformátování pravidel do tvaru IF podmínka THEN výsledek

private string float_sqlformat(string s)

V SQL příkazech se používá formát float s „.“ namísto s „,“.

private void print_error(string error)

K tisku chyb do HTML stránky.

Další pomocné metody

Metody *create_Model_addIntKey(string name)*, *private string create_Model_addInt(string name)*, pro specifikaci struktury modelů, *data_Source_View_addString(string name, int length)*, *data_Source_View_addInt(string name)* pro přidání sloupců do data soure view. Všechny metody zde napsané jsou typu string. Používají se pro výběr sloupců z tabulky, pro určité XML příkazy.

3.4.2 Uživatelské rozhraní

Návrh rozhraní

V první řadě jsem se snažil minimalizovat počet informací, které musí uživatel zadávat a to zejména kvůli jednoduchosti výsledné aplikace. Naopak u tlačítek jsem zvolil opačný postup, na stránku jsem dal všechna nezbytná, aby bylo uživateli jasnější, jak celý proces funguje a co je potřeba vytvořit, i když většinou se jedná jen o stisknutí tlačítka, které ani nepotřebuje žádné vstupní parametry.

Ze začátku jsem počítal pouze s odlišením menu a prostoru pro zobrazení výsledku, postupem času jsem se rozhodl pro oddělení informací sloužících pro připojení k serverům a toto nastavení jsem přesunul do horní části stránky. V menu jsem se držel zásady, že tlačítka jsou umístěna v pořadí, ve kterém jsou uživatelem používána. Podobně je to pak i u zadávání informací, kdy umístění textboxů je vždy těsně nad tlačítkem, kde se poprvé musí zadat. Pokud tedy otevře uživatel stránku a bude postupovat při zadávání informací od shora dolů, podaří se mu úspěšně vytvořit model.

Jako jazyk aplikace byla zvolena angličtina zejména kvůli jednoduššímu zobrazování chyb, SQL server hlásí chyby zásadně v angličtině. Pokud bych je chtěl zobrazovat v češtině, nastal by nesnadný překlad.

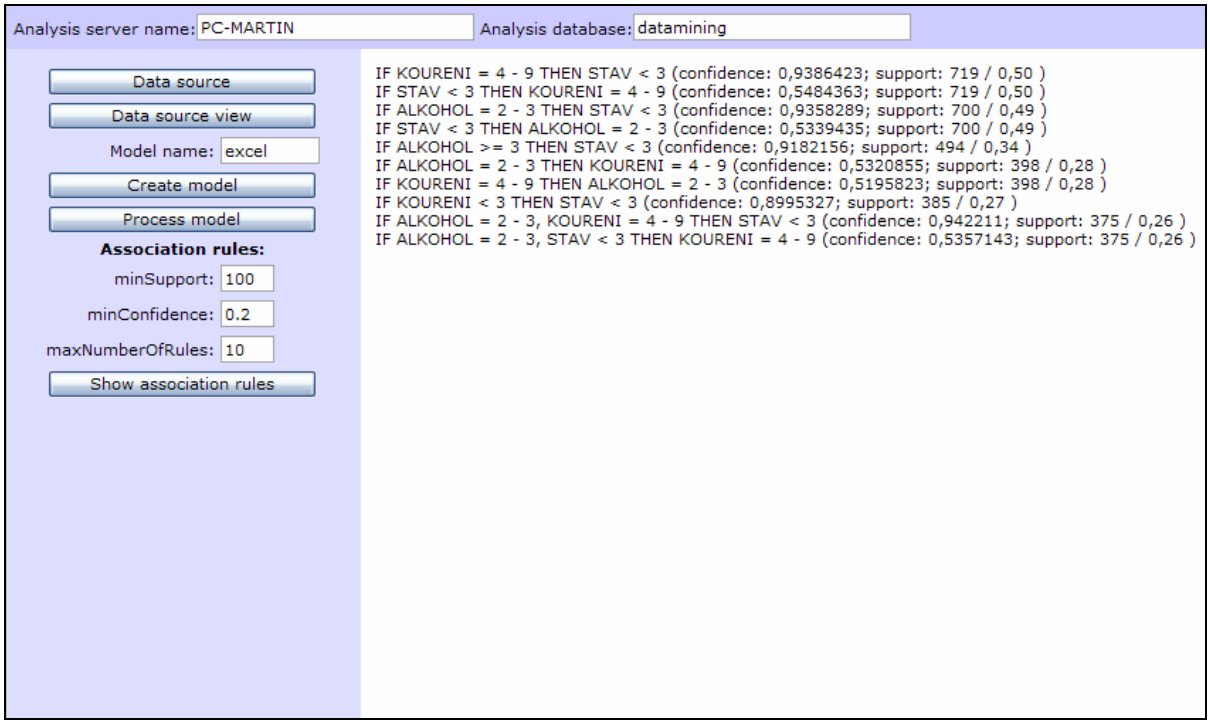
Pro tlačítka jsem zvolil standardní prvek `asp:Button` a pro pole `asp:TextBox` . Pro vystup na obrazovku jsem nepoužil žádný formulářový prvek, ale text vypisuji přímo do HTML kódu.

Ovládání

Jak jsem již zmínil dříve, menu je uspořádáno tak, aby uživatele navádělo intuitivně přes vytvářrní modelu až po zobrazení výsledků. Pokud uživatel některý krok přeskočí, aplikace mu to dovolí, ale ze serveru se vrátí chybové hlášení, které vede k tomu, že se akce neprovede. Bylo by samozřejmě možné uživateli zakázat přeskakovat kroky pomocí zamrznutí některých tlačítek, ale nerozhodl jsem se tak, kvůli jednoduché myšlence. Pokud už některý z objektů je vytvořen na serveru a uživatel například chce vytvořit druhý model, musel by znovu vytvářet něco, co už na serveru je a taktéž to ani většinou nejde. Proto zobrazuji jen chyby, které souvisí se špatným zadáním od uživatele. Výjimkou je nastavení algoritmu, kdy pokud uživatel nezadá žádné vstupy se použiji výchozí a tyto hodnoty se do polí doplní samy.

Možnost jak nastavit parametry database serveru, tedy serveru odkud se berou data, je předávat tyto parametry metodou GET, tedy v URL stránky. Kde dbschemaname je typ tabulky, dbtablename je název tabulky, connstringdb připojovací řetězec k database serveru, datasourcename pro název data sourc na analysis serveru tedy například:

Default.aspx?dbschemaname=dbo&dbtablename=Excel&connstringdb=Provider=SQLNCLI.1;Data Source=PC-MARTIN;Uid=sa;Pwd=aaa;Initial Catalog=DATA_TO_MINE&datasourcename=datatomine



Obr. 3 – Ukázka aplikace

3.5 Dosažené výsledky

Zdroj dat

Jako zdroj dat jsou použil průzkum STULONG, více o průzkumu v literatuře [7]

Jako vstupní data jsem uvažoval sloupce počet vykouřených cigaret (koureni), četnost pití alkoholu (alkohol), věk, vzdělání, doba kouření (dobakour), výška, váha, četnost pití čaje (caj) a kávy (kava).

Výpis pravidel

Ze vstupních dat jsem získal těchto 10 pravidel s největší podporou.

1. IF KOURENI = 4 - 9 THEN DOBAKOUR >= 9 (confidence: 0,9530026; support: 730 / 0,51)
2. IF DOBAKOUR >= 9 THEN KOURENI = 4 - 9 (confidence: 0,7628004; support: 730 / 0,51)
3. IF ALKOHOL = 2 - 3 THEN DOBAKOUR >= 9 (confidence: 0,6604278; support: 494 / 0,34)
4. IF DOBAKOUR >= 9 THEN ALKOHOL = 2 - 3 (confidence: 0,5161964; support: 494 / 0,34)
5. IF CAJ = 5 - 6 THEN DOBAKOUR >= 9 (confidence: 0,6384106; support: 482 / 0,34)
6. IF DOBAKOUR >= 9 THEN CAJ = 5 - 6 (confidence: 0,5036573; support: 482 / 0,34)
7. IF KAVA = 2 - 3 THEN DOBAKOUR >= 9 (confidence: 0,7169518; support: 461 / 0,32)
8. IF DOBAKOUR >= 9 THEN KAVA = 2 - 3 (confidence: 0,4817137; support: 461 / 0,32)
9. IF ALKOHOL = 2 - 3 THEN CAJ = 5 - 6 (confidence: 0,5601604; support: 419 / 0,29)

10. IF CAJ = 5 - 6 THEN ALKOHOL = 2 - 3 (confidence: 0,5549669; support: 419 / 0,29)

Z prvního pravidla lze přečíst, že kuřák který vykouří 15 a více cigaret denně kouří 11 a více let. Toto pravidlo se v databázi vyskytuje 730krát (51%) a spolehlivost pravidla je 95%. Druhé pravidlo je podobné prvním s rozdílem spolehlivosti pouze 76%, tedy pokud dotázaný kouří 11 a více let tak pouze 76% z nich vykouří 15 a více cigaret. Třetí pravidlo říká, že pokud někdo pije alkohol občas nebo pravidelně, kouří více jak 11 let s podporou 34% a spolehlivostí 66%. Páté pravidlo říká, že každý kdo vypije alespoň jeden šálek čaje kouří 11 a více let s podporou 34% a spolehlivostí 64%. Deváté pravidlo říká, pokud někdo pije alkohol tak vypije i 1 a více šálků čaje, s podporou okolo 29% a spolehlivostí 56%.

3.6 Další rozvoj aplikace

První možný rozvoj je přidat i další znalosti k vytvoření modelu. Také by uživatel mohl nadefinovat data, která pochází z database serveru, pomocí výběru tabulky a sloupců pro algoritmus. Celá aplikace by pak mohla být více kroková kde v prvním kroku se připojí na zmiňovaný database server, tam vybere dat, potom přes výběr algoritmu se dostane na stránku nynější aplikace kde se celý model vytvoří. Postupně by se v těchto krocích vytvářely na serveru objekty které by byli potřeba.

4 Oracle Database

Nejnovějším serverem je Oracle Data Mining 11.1, pro mou práci jsem použil tento i starší 10.2 server.

4.1 Dolování dat v Oracle

Opět se jedná o komunikaci klient-server. Klient slouží k vytvoření popisu modelu a dat. Tyto požadavky posílá na server, který je zpracuje a pošle výsledky zpět klientovi, který je zobrazí či jinak zpracuje. Klient nijak nezasahuje do průběhu dolovacích algoritmů, neboť to vše má na starosti server.

Podobně jako u Microsoft SQL serveru zůstává vytvořený model na serveru, tedy opět klient nemusí znovu vytvářet popis modelu a spouštět ho vícekrát, ale stačí se dotázat pouze na výsledky předešlého získávání znalostí.

4.1.1 Oracle Data Miner

Jedná se o nástroj od formy Oracle pro dolování dat v prostředí Oracle Data Mining server, celá aplikace je napsána v programovacím jazyce Java, pro svou funkčnost potřebuj mít Javu nainstalovanou. Ke každé verzi serveru je i nový Data Miner, většinou jsou si vzhledově jednotlivé verze podobné. Po spuštění, se aplikace dotáže na spojení ke serveru. Při úspěšném přihlášení se otevře hlavní okno, v levé části je umístěno hlavni navigační menu, pro mé potřeby byli nejdůležitější odrážky data sources, kde jsou umístěny data, Models, kde jsou výsledné modely a Mining Activities, zde jsou umístěny popisy a nastavení vytvoření modelů. Po výběru Activity->Build se objeví průvodce k vytvoření modelu.

4.1.2 Java a Oracle Java Data Mining API

Java

Jedná se o moderní objektově orientovaný jazyk, není závislý na platformě, na které je použit, protože využívá tzv. virtuální stroj. To je v podstatě mezivrstva mezi operačním systémem a aplikací. Veškerý kód je přeložen do virtuálního stroje a ten s ním dále pracuje. Více informací o Javě v literatuře [18]

Oracle Java Data Mining API

Je aplikační rozhrání, které slouží pro dolování dat. Obsahuje třídy pro vytvoření spojení se serverem, vytvoření modelu, přečtení modelu a další, které jsem pro potřeby mé práce tolik nevyužil. Je distribuováno ve více balících.

Server 10.2:

```
$APLICATION_DIRECTORY/lib/jdm.jar
$APLICATION_DIRECTORY/lib/ojdm_api.jar,xdb.jar
$APLICATION_DIRECTORY/lib/ojdbc14.jar
$APLICATION_DIRECTORY/lib/connector.jar
$APLICATION_DIRECTORY/lib/orai18n.jar
$APLICATION_DIRECTORY/lib/orai18n-mapping.jar
$APLICATION_DIRECTORY/lib/xmlparserv2.jar
```

Server 11.1:

```
$APLICATION_DIRECTORY/lib/jdm.jar
$APLICATION_DIRECTORY/lib/ojdm_api.jar
$APLICATION_DIRECTORY/lib/xdb.jar
$APLICATION_DIRECTORY/lib/ojdbc5.jar
$APLICATION_DIRECTORY/lib/connector.jar
$APLICATION_DIRECTORY/lib/orai18n.jar
$APLICATION_DIRECTORY/lib/orai18n-mapping.jar
$APLICATION_DIRECTORY/lib/xmlparserv2.jar
```

Problémem při distribuci balíků je hlavně jejich verze, každá knihovna je určena pro konkrétní verzi serveru. Často tedy vzniká problém v nekompatibilitě verzí mezi serverem a klientem. Dalším problémem je i shánění správných verzí knihoven. Jedním z řešení, je použit knihovny z Oracle Data Mineru, který by měl mít u sebe správné knihovny. Druhou variantou je extrakce knihoven přímo z nainstalovaného Oraclu.

4.1.3 Rozdíl mezi verzemi serveru

Rozdíl ve vytváření modelu je nejrazantnější trošku nelogicky mezi verzemi 10.1 a 10.2. Verze po 10.2 ,tedy nyní nejnovější verze 11.1, jsou ve zdrojovém kódu podobn0, stejně tak i 10.1 a předešlá 9i verze se skoro neliší, samozřejmě je potřeba vyměnit knihovny pro Javu při použití různých verzí. Zde popíšu hlavně rozdíl mezi verzí 10.1 a 10.2.

Výběr metod k dolování dat se v obou verzích příliš neliší, to co je podstatné na server 10.2 je možná zaměnitelnost modelů vytvořených pomocí javy a PL/SQL. Dalším rozdílem je, že v 10.2 (narozdíl od 10.1) nejsou podporována logická data, namísto toho se doporučuje používat databázových pohledů. Samozřejmě se změnou verze přišlo i přejmenování knihoven.

Rozdíl mezi verzemi serveru v Javě

Filozofii verze 10.2 je, že veškeré objekty se získávají z tzv. *factory* z vytvořeného připojení, díky čemuž se již nepoužívá na vytvoření objektu klíčové slovo *new* (10.1), ale přes *javax.datamining.connection.getFactory("jméno knihovny")*, díky čemuž se objeví v programu jediné *new* a to při vytváření spojení na server. Na ukázku příkaz na vytvoření Fyzických dat (Physical Data Specifikation / Set)

Tedy namísto kódu **v 10.1**

```
LocationAccessData lad =  new
LocationAccessData("MINING_DATA_BUILD_V","DMUSER");
PhysicalDataSpecification pds = new
NonTransactionalDataSpecification(lad);
```

se **v 10.2** napíše kód takto, kde m_dmeConn je spojení se serverem

```
PhysicalDataSetFactory pdsFactory = (PhysicalDataSetFactory)
m_dmeConn.getFactory( "javax.datamining.data.PhysicalDataSet" );
PhysicalDataSet buildData = m_pdsFactory.create
("MINING_DATA_BUILD_V",false);
m_dmeConn.saveObject( "nbBuildData", buildData, true );
```

Další rozdílem je ukládání nastavení, v 10.2 se s nastavením modelu (fyzická data, nastavení algoritmu, atd.) pracuje pouze přes symbolická jména, kdežto u 10.1 se u některých používá symbolické jméno jinde se pracuje s objektem, jako ukázku lze opět vybrat předešlý příkaz.

4.1.4 Popis tříd použitých pro dolování dat

Veškeré informace použité v této kapitole jsem čerpal z [12,13], kde lze nalézt podrobnější a přesnější znění příkazů.

OraConnectionFactory

Objekt této třídy je jako jediný vytvořen pomocí klíčového slova new. Jedná se o třídu ze které pomocí metody *getConnectionSpec()* dostaneme objekt *ConnectionSpec*, popsán níže. Další důležitá metoda je *getConnection(ConnectionSpec connSpec)* která vrací objekt typu *Connection*, opět je popsán níže.

ConnectionSpec

Třída která se stará o nastavení parametrů pro připojení k serveru, pomocí metody *setURI(string ConnectionString)*. *ConnectionString* je ve tvaru jdbc:oracle:thin:@[adresa serveru]:[port]:[SID], pro mou aplikaci jsem použil tento řetězec tvaru: *„jdbc:oracle:thin:@pcuifs1:1523:STUD“*. Dále mé

třída další metody na zadání přihlašovacího jména a hesla, *setName(string name)* a *setPassword(string password)*.

Connection

Třída se stará a o správu připojení k serveru, podporuje také ukládání a příjem pojmenovaných objektů (jako např. Fyzická data, popis algoritmu, výsledný model, atd.). Nejpodstatnější metodou je *getFactory(string objectName)* pomocí které se získávají *factory* k vytváření dalších objektů. Druhou metodou je *saveObject(string name, MiningObject object, boolean replace)*, která ukládá objekt se jménem *name*, specifikaci určenou pomoci *object*. Další metody slouží na odeslání a přijetí modelu na a ze serveru *retrieveObject(),execute()* obě mají víc možností parametrů.

PhysicalDataSet

Objekt této třídy dostaneme z *PhysicalDataSetFactory* pomocí metody *create(java.lang.String uri, boolean importMetaData)* kde *uri* je název tabulky na serveru, ze které chceme data získat. Pomocí metody *addAttribute(PhysicalAttribute attribute)* slouží k přidání specifikace některého ze sloupců.

AssociationSettings

Třída sloužící k nastavení algoritmu, získáme ji opět z *AssociationSettingsFactory* voláním metody *create()*. K nastavení algoritmu slouží metody *setMinSupport(double minSupport)*, *setMinConfidence(double minConfidence).*

BuildTask

Opět získáme ze třídy *BuildTaskFactory* voláním metody *create(String buildData, String buildSettingsName, String modelName)*, kde *buildData* je objekt vytvořený pomocí třídy *PhysicalDataSet*, *buildSettingsName* je nastavení algoritmu a *modelName* jméno modelu které nám vrátí server.

ExecutionHandle a ExecutionStatus

Objekty slouží k odeslání modelu na server, *ExecutionHandle* získáme ze třídy *Connection* pomocí metody *execute(string buildTaskName)* a *ExecutionStatus* potom z *ExecutionHandle* pomocí metody *waitForCompletion(int time)*. Metodou *getState()* z *ExecutionStatus* získáme zda vše proběhlo v pořádku a *getDescription()* v případě chyby tuto chybu vrátí.

AssociationModel

Objekt této třídy je model který získáme ze serveru, opět ho dostaneme ze třídy *Connection* pomocí metody *retrieveObject(string modelName).*

4.2 Analýza aplikace

Úkolem aplikace bude demonstrovat získávání znalostí z databázi pod serverem firmy Oracle.

4.2.1 Neformální specifikace

Po zpracování požadavků od uživatele vytvoří model, který odešle na server, zažádá si o výsledky ze serveru, které zpracuje a poté zobrazí na obrazovku. Uživatel bude moci definovat základní parametry pro dolovací algoritmus a parametry pro připojení na server.

4.2.2 Ovládání programu

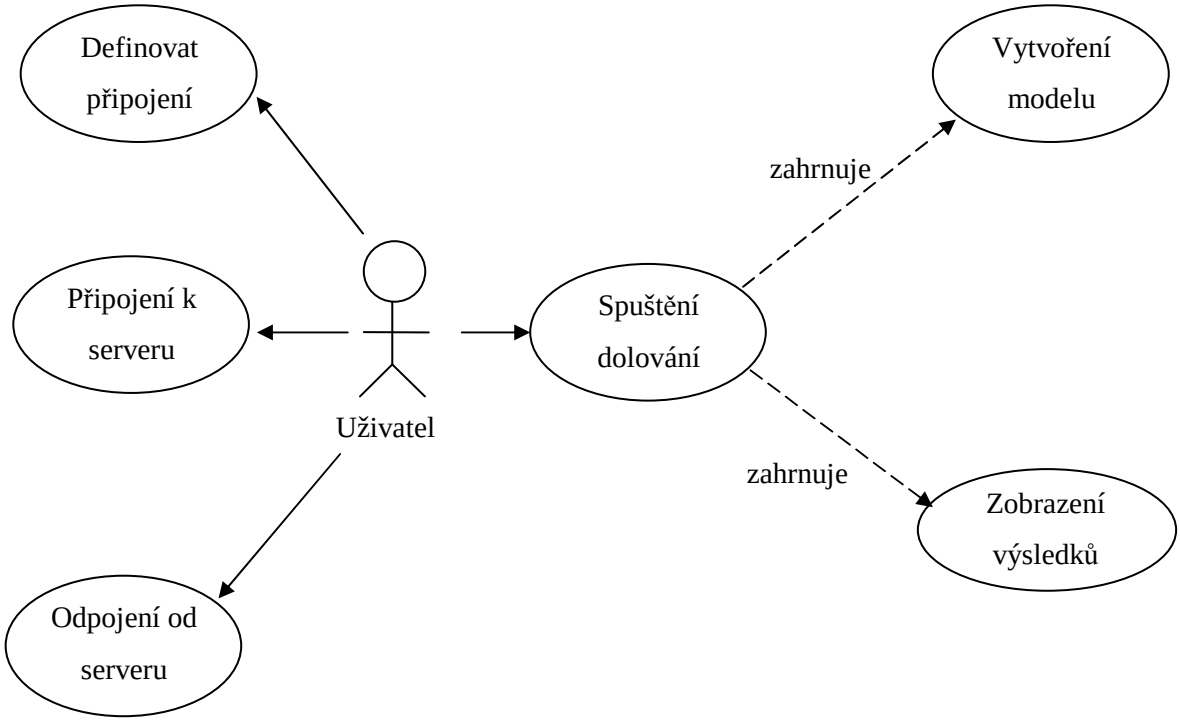
Vstup bude realizován pouze přes uživatelské grafické rozhraní, s jiným ovládáním se nepočítá. Podobně jako u Microsoft SQL serveru, slouží program pro demonstraci dolování dat, tedy je kladen důraz na jednoduchost namísto komplexnosti, díky čemuž bude ovládání jednoduché, snadno pochopitelné a mnoho parametrů nebude moci uživatel ovlivnit. Jako jediný výstup se opět počítá pouze s výpisem na obrazovku aplikace.

4.2.3 Použité technologie

Pro tvorbu aplikace jsem zvolil jazyk Java, který je popsán výše, jako grafické rozhraní budou složit prvky z balíku Awt a Swing.

Pomoci API rozhraní se aplikace bude připojovat k serveru Oracle Datamining 11g, pro tento server jsem se rozhodl především kvůli tomu, že se jedná o nejnovější server od firmy Oracle. Samozřejmě po obměně knihoven by bylo možné aplikaci použít i na starším serveru verze 10.2.

4.2.4 Diagram použití



Obr 4. – Diagram použití

4.3 Popis implementace

Aplikace je vytvořena v Javě, používá grafické rozhraní k ovládání programu.

4.3.1 Struktura programu

Třída *Main()* dědí ze třídy *JFrame* a při spuštění vytvoří instanci sama sebe, v Main() jsou tyto další metody.

public Main()

Jejím úkolem je inicializovat aplikaci, to znamená vytvořit grafické rozhraní se všemi prvky, přiřadit jednotlivým tlačítkům akce a parametry. Taktéž vybrat tlačítka která budou aktivní.

void serverConnect()

Metoda která slouží pro vytvoření spojení k serveru, výsledný objekt, který reprezentuje spojení uloží do proměnné typu *Connection*. Pokud se připojení nezdařilo vypíše chybové hlášení, jinak nastaví tlačítka do stavu připojeno.

void serverDisconnect()

Zruší připojení navázané pomocí metody *serverConnect()* a podobně jako tato metoda nastaví tlačítka do stavu odpojeno.

void setButtonsToConnected(),void setButtonsToDisconnected()

Slouží k nastavení tlačítek do stavu připojeno nebo odpojeno podle volané metody.

void apriori()

Metoda která je stavebním kamenem celé aplikace, vytváří model, posílá model na server a po přijetí výsledného modelu ho zobrazí. Nejprve se přečtou vstupní parametry, pokud některý z nich není nastaven zvolí se standardní hodnota. Potom se smažou všechny dříve vytvořené modely, pomocí metody *removeObject().*

Tady skončili před přípravy a metoda začíná vytvářet model, prvně vytvoří popis fyzických dat pomocí třídy *PhysicalDataSet*, poté vytvoří popis nastavení algoritmu, v mém případě pomocí třídy *AssociationSettings*, kde pomocí metod popsaných v kapitole 4.1.4 se zpracují vstupy od uživatele. Dalším krokem ve vytváření modelu, je spojení specifikací pomocí třídy *BuildTask*, objekt této třídy se potom odešle na server, k tomu slouží metoda *execute()*. Server vrátí výsledný model jako objekt typu *AssociationModel*, který zobrazíme, na to slouží především třídy *OraRulesFilter*, *OraAssociationRule* a další.

4.3.2 Grafické uživatelské rozhraní

Grafické rozhraní je vytvořeno z knihoven Awt a Swing,

Návrh rozhraní

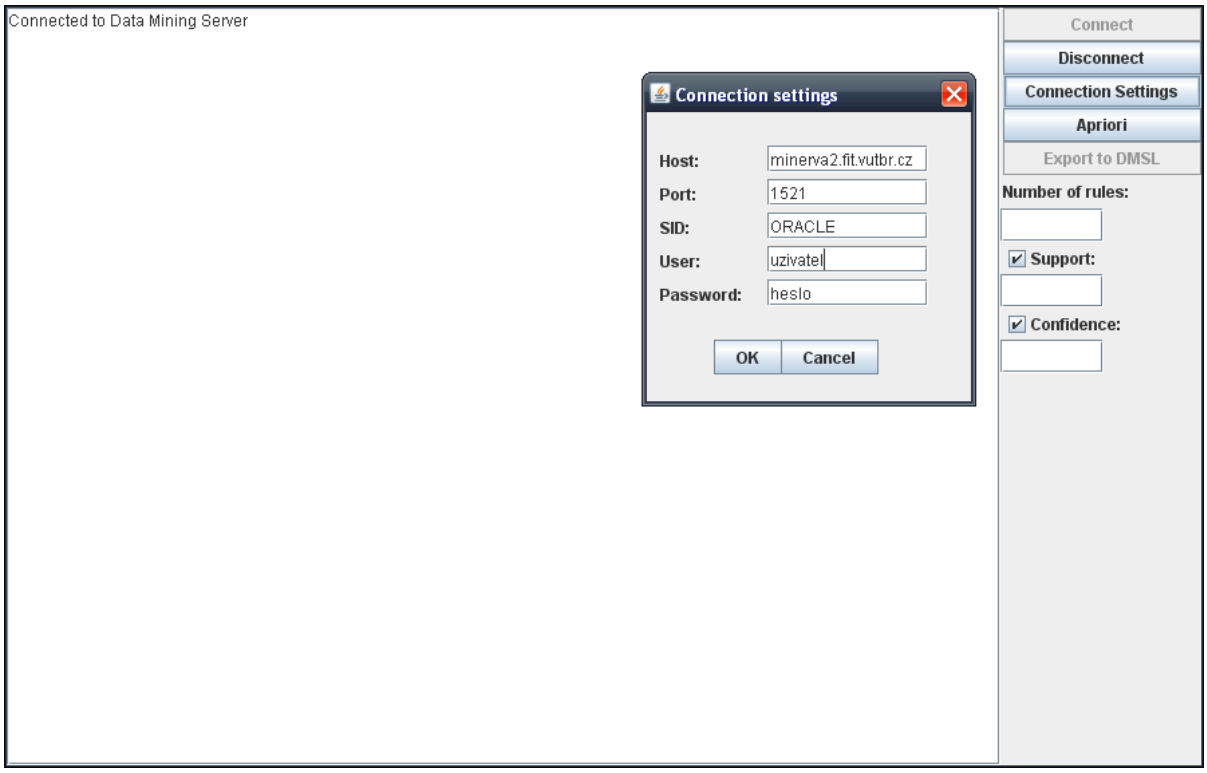
Opět jsem se snažil celý proces získávání znalostí uživateli zjednodušit, díky čemuž je opět v aplikaci nezbytné minimum ovládacích prvků.

Základem grafického rozhraní je textové pole do kterého se vypisují nejen výsledek celého dolování dat, ale i chyby a informace o vytváření modelu. Pro toto textové pole jsem zvolil komponentu typu *JTextArea*, dalšími komponenty v aplikaci jsou vstupní textová pole typu *JTextField* a samozřejmě tlačítka *JButton*, která jednotlivé události spouští.

Díky tomu, že vytváření dolovací úlohy je v jediné metodě, tak i počet tlačítek je v aplikaci minimální, pro usnadnění automaticky blokuji nesmyslné operace, např. nejde vytvořit model bez připojení k serveru.

Ovládání

Prvním krokem pro vytvoření modelu je nastavení připojení k serveru, po stisku tlačítka *Connection settings* se uživateli objeví nové okno kde je vyzván k nastavení parametrů připojení. Zbytek procesu se již odehrává v hlavním okně aplikace, po stisku tlačítka *Connect* se server připojí k serveru, po úspěšném provedení se odblokují zbylá tlačítka, *Disconnect* slouží k odpojení od serveru a nejpodstatnější tlačítko *apriori* provede dolování a zobrazí výsledky, zbylé tři textová pole slouží k nastavení algoritmu. Pokud je rámeček k zatržení nad těmito poli nezatržený použijí se výchozí hodnoty pro toto nastavení, taktéž pokud je pole prázdné.



Obr. 5 – Ukázka aplikace v Javě

4.4 Další rozvoj aplikace

V první řadě se mě nepodařilo zprovoznit ani ukázkové příklady od firmy Oracle jak na serveru 11 tak ani na serveru 10.2, tedy celá aplikace nevytvoří model na serveru, bez problému se připojí a model odešle na server, který zahlásí chybu, zajímavé je že každý ze serveru zahlásí jinou chybu.

Možný rozvoj aplikace je výběr vstupních dat pomocí průvodce, který by se na server připojil a nabídl uživateli vybrat tabulku a sloupce. Další možný rozvoj je rozšířit aplikaci o moduly k uplatnění jiných algoritmů nežli apriori, taktéž by se dali výsledky exportovat do DMSL dokumentu.

4.5 Získané znalosti

Na ukázku přikládám získané znalosti z modelu vytvořeného pomocí PL/SQL na Oracle 11.1.
Znalosti, založené jestli si zákazník přeje produkt firmy. Ukázáno je prvních 5 zákazníků, pouze pátý zákazník si produkt přeje.

CUST_ID	PREDICTION	PROBABILITY
100001	0	.78700680952651525
100001	1	.21299319047348478
100002	0	.72166995976221537
100002	1	.27833004023778457
100003	0	.89584296216320425
100003	1	.10415703783679578
100004	0	.89484096182014439
100004	1	.10515903817985559
100005	0	.13508666695008609
100005	1	.86491333304991391

5 Porovnání obou prostředí

V porovnání serverů se musím přiklonit na stranu Microsoft, kde komunikace pomocí XML a SOAP má podle mě daleko větší uplatnění v tvorbě různých aplikacích. Na druhou stranu má dlouhý seznam softwarových požadavků, které se musí splnit při instalaci. Taktéž co měli napsané v dokumentaci opravdu po splnění nastavení fungovalo. A implementace aplikace byla bez problému.

S řešením od firmy Oracle jsem se poprvé seznámil v předmětu IDS, v tu dobu jsme se serverem byl maximálně spokojený, nepracoval jsem však s dolováním dat. Teď si i tak myslím, že řešení je dobré, celý model je na vytvoření jednoduší jak v MS SQL, kdyby opravdu fungoval tak jak je popsáno v dokumentaci. Právě tam bych hledal vinu toho, že aplikace neskončila podle mých představ, proto bych si dovolil kritizovat firmu Oracle za neúplnost dokumentace. Už několikrát jsem slyšel o neaktuálnosti manuálu kdy i bugy známe nějakou dobu, nejsou v dokumentaci popsané a nejedná se jen o dolování dat. Podstatné bude s čím přijde nový MS SQL 2008 v porovnání s Oracle 11g.

Závěr

Teoretickou částí dolování dat se zabývají první 2 kapitoly, kde jsem se snažil podat pojmy tak, aby je pochopili i začátečníci, pro pokročilé uživatele pak slouží odkazy do literatury. Popsal jsem nepodstatnější algoritmy pro dolování dat s příklady, které by měli sloužit k snadnějšímu pochopení.

Při implementaci aplikace pro server Microsoft SQL server jsem měl jediný problém a to najít vhodnou knihovnu, kterou bych se připojil na server. Po nalezení knihovny AdomdClient, už práce pokračovala svižným tempem a v průběhu jsem nenarazil na žádný větší problém. Asp.NET a C# jsou jazyky, které jsem před tímto projektem neznal a jsem s nimi maximálně spokojený. V porovnání asp a php se kterým jsem pracoval dříve, mi příjde asp lepší, díky větší robustnosti, které do něj přináší C#.

Podařilo se mi realizovat aplikaci, která demonstruje získávání asociačních pravidel z databáze. Aplikace samozřejmě nemůže být na úrovni komerčních produktů, ale pro ukázku postupu dolování dat na serveru Microsoft, bohatě postačí.

Při druhé aplikaci pro Oracle, jsem byl už dopředu upozorněn na různé verze knihoven, díky čemuž jsem si pohlídal aktuální verze a větší problém jsem s nimi neměl. Bohužel na problém jsem narazil tehdy, když jsem se snažil zprovoznit ukázkové příklady od firmy Oracle, které nefungovali ani na jedné verzi serveru. Psaní programu postupovalo díky tomu dost pomalu a ke konci se mě ani žádné pravidla nepovedlo vydolovat, i když čas který jsem tomu věnoval byl několikrát vyšší jak v případě MS SQL. V programovacím jazyku Java jsem pracoval předtím a mám s ním dobré zkušenosti. Podle mě se jedná o jeden z nejlepších objektově orientovaných jazyků, je jednoduchá na pochopení a narozdíl od C++ se nemusíme starat na jakým OS bude aplikace fungovat.

Porovnání obou prostředí je trošku subjektivní právě kvůli výsledkům tvorby obou aplikací, díky čemuž se mi pod MS SQL pracovalo lépe. Samozřejmě nelze jednoznačně určit lepší server, oba systémy mají určitě své pro a proti.

Literatura

[1] Berka, P. *Dobývání znalostí z databází*. Praha, SNTL 1996.

[2] Lacko, L. *Datové skaldy analýza OLAP a dolování dat*. Brno, Computer Press 2003

[3] Stryka, L. *Modul dolování asociačních pravidel*, FIT VUT 2003.

[4] Krásný, M. *Dolování asociačních pravidel v prostředí DB serveru Oracle*, FIT VUT 2006

[5] Zendulka, J. a kol. *Získávání znalostí z databází*, opora k předmětu FIT VUT Brno 2006

[6] ZhaoHui Tang, Maclenna J., *Data mining with SQL server*, Wiley Publishing, Inc. 2005

[7] Průzkum STULONG [online],[cit. 2008-05-12]
URL: http://www.fit.vutbr.cz/~stryka/ds//stulong_popis/index.htm

[8] MSDN library - .NET Framework Class Library [online],[cit. 2008-05-05]
URL: http://msdn.microsoft.com/en-us/library/ms229335.aspx

[9] MSDN library - Microsoft.AnalysisServices.AdomdClient Namespace [online],[cit. 2008-05-05]
URL: http://msdn.microsoft.com/en-us/library/microsoft.analysisservices.adomdclient.aspx

[10] MSDN library – Microsoft SQL server 2005 [online],[cit. 2008-05-05]
URL: http://msdn.microsoft.com/en-us/library/bb418498.aspx

[11] Converting to the ODM 10.2 Java API [online],[cit. 2008-05-05]
URL: http://youngcow.net/doc/oracle10g/datamine.102/b14340/java_moving.htm

[12] Overview for javax.datamining [online],[cit. 2008-05-05]
URL: http://www.oracle.com/technology/products/bi/odm/jsr-73/index.html

[13] Oracle Data Mining Java API Reference 10g Release 2 (10.2) [online],[cit. 2008-05-05]
URL: http://download-east.oracle.com/docs/cd/B19306_01/datamine.102/b14341/index.html

[14] Oracle Data Mining Java API Reference 11g Release 1 (11.1) [online],[cit. 2008-05-12]
URL: http://download.oracle.com/docs/cd/B28359_01/datamine.111/b28132/toc.htm

[15] Oracle Data Mining Application Developer's Guide, 10g Release 2 (10.2) [online],[cit. 2008-05-12]
URL: http://download.oracle.com/docs/pdf/B14340_01.pdf

[16] Oracle Data Mining Application Developer's Guide, 11g Release 1 (11.1) [online],[cit. 2008-05-12]
URL: http://download.oracle.com/docs/cd/B28359_01/datamine.111/b28131.pdf

[17] Oracle Data Mining Documentation [online],[cit. 2008-05-12]
URL: http://www.oracle.com/technology/documentation/datamining.html

[18] Java [online],[cit. 2008-05-12] **URL:** http://java.sun.com/

[19] Sample schema Scripts and Object Destripction
URL: http://download.oracle.com/docs/cd/B28359_01/server.111/b28328/scripts.htm

Seznam příloh

Přílohy

Příklady XML příkazů na vytvoření modelu:

Vytvoření data source:

```
<Create xmlns="http://schemas.microsoft.com/analysisservices/2003/engine">
<ParentObject>
  <DatabaseID>dataminingDOTNET</DatabaseID>
</ParentObject>
<ObjectDefinition>
<DataSource xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ddl2="http://schemas.microsoft.com/analysisservices/2003/engine/2"
xmlns:ddl2_2="http://schemas.microsoft.com/analysisservices/2003/engine/2/
2" xsi:type="RelationalDataSource">
  <ID>DATATOMINE</ID>
  <Name>DATATOMINE</Name>
  <ConnectionString>Provider=SQLNCLI.1;Data Source=PC-
MARTIN;Uid=sa;Initial Catalog=DATA_TO_MINE</ConnectionString>
  <ImpersonationInfo>
    <ImpersonationMode>ImpersonateServiceAccount</ImpersonationMode>
  </ImpersonationInfo>
  <Timeout>PT0S</Timeout>
</DataSource>
</ObjectDefinition>
</Create>
```

Vytvoření data source view:

```
<Create xmlns="http://schemas.microsoft.com/analysisservices/2003/engine">
  <ParentObject>
    <DatabaseID>dataminingDOTNET</DatabaseID>
  </ParentObject>
  <ObjectDefinition>
     <DataSourceView xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ddl2="http://schemas.microsoft.com/analysisservices/2003/engine/2"
xmlns:ddl2_2="http://schemas.microsoft.com/analysisservices/2003/engine/2/
2">
        <ID>DATATOMINEview</ID>
        <Name>DATATOMINEview</Name>
        <Annotations>
            <Annotation>
              <Name>http://schemas.microsoft.com/DataWarehouse/Designer/
1.0:SchemaRestriction</Name>
            </Annotation>
            <Annotation>
              <Name>http://schemas.microsoft.com/DataWarehouse/Designer/
1.0:RetrieveRelationships</Name>
              <Value>true</Value>
            </Annotation>
         </Annotations>
         <DataSourceID>DATATOMINE</DataSourceID>
         <Schema>
             <xs:schema id="DATATOMINEview" xmlns=""
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:msdata="urn:schemas-
microsoft-com:xml-msdata" xmlns:msprop="urn:schemas-microsoft-com:xml-
msprop">
```

```
              <xs:element name="DATATOMINEview" msdata:IsDataSet="true"
msdata:UseCurrentLocale="true">
              <xs:complexType>
                 <xs:choice minOccurs="0" maxOccurs="unbounded">
                    <xs:element name="dbo_Excel"
msprop:FriendlyName="Excel" msprop:DbSchemaName="dbo"
msprop:DbTableName="Excel" msprop:TableType="Table">
                       <xs:complexType>
                          <xs:sequence>
                             <xs:element name="ICO"
msprop:FriendlyName="ICO" msprop:DbColumnName="ICO" type="xs:int" />
                             <xs:element name="STAV"
msprop:FriendlyName="STAV" msprop:DbColumnName="STAV" type="xs:int" />
                             <xs:element name="KOURENI"
msprop:FriendlyName="KOURENI" msprop:DbColumnName="KOURENI" type="xs:int"
/>
                             <xs:element name="ALKOHOL"
msprop:FriendlyName="ALKOHOL" msprop:DbColumnName="ALKOHOL" type="xs:int"
/>
                             <xs:element name="ROKNAR"
msprop:FriendlyName="ROKNAR" msprop:DbColumnName="ROKNAR" type="xs:int" />
                          </xs:sequence>
                       </xs:complexType>
                    </xs:element>
                 </xs:choice>
              </xs:complexType>
           </xs:element>
        </xs:schema>
     <diffgr:diffgram xmlns:msdata="urn:schemas-microsoft-com:xml-
msdata" xmlns:diffgr="urn:schemas-microsoft-com:xml-diffgram-v1" />
   </Schema>
  </DataSourceView>
 </ObjectDefinition>
</Create>
```

Vytvoření modelu a struktury

```
<Create xmlns="http://schemas.microsoft.com/analysisservices/2003/engine">
    <ParentObject>
        <DatabaseID>dataminingDOTNET</DatabaseID>
    </ParentObject>
    <ObjectDefinition>
        <MiningStructure xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ddl2="http://schemas.microsoft.com/analysisservices/2003/engine/2"
xmlns:ddl2_2="http://schemas.microsoft.com/analysisservices/2003/engine/2/
2">
            <ID>Model</ID>
            <Name>Model</Name>
            <Annotations>
                <Annotation>

<Name>http://schemas.microsoft.com/DataWarehouse/Designer/1.0:CaseTableNam
e</Name>
                    <Value>dbo_Excel</Value>
                </Annotation>
            </Annotations>
            <Source xsi:type="DataSourceViewBinding">
                <DataSourceViewID>DATATOMINEview</DataSourceViewID>
            </Source>
            <Language>1033</Language>
```

```
<Collation>Czech_CI_AS</Collation>
<Columns>
    <Column xsi:type="ScalarMiningStructureColumn">
        <ID>ICO</ID>
        <Name>ICO</Name>
        <IsKey>true</IsKey>
        <Type>Long</Type>
        <Content>Key</Content>
        <KeyColumns>
            <KeyColumn>
                <DataType>Integer</DataType>
                <DataSize>-1</DataSize>
                <Source xsi:type="ColumnBinding">
                    <TableID>dbo_Excel</TableID>
                    <ColumnID>ICO</ColumnID>
                </Source>
            </KeyColumn>
        </KeyColumns>
    </Column>
    <Column xsi:type="ScalarMiningStructureColumn">
        <ID>STAV</ID>
        <Name>STAV</Name>
        <Type>Long</Type>
        <Content>Discretized</Content>
        <KeyColumns>
            <KeyColumn>
                <DataType>Integer</DataType>
                <DataSize>-1</DataSize>
                <Source xsi:type="ColumnBinding">
                    <TableID>dbo_Excel</TableID>
                    <ColumnID>STAV</ColumnID>
                </Source>
            </KeyColumn>
        </KeyColumns>
    </Column>
    <Column xsi:type="ScalarMiningStructureColumn">
        <ID>KOURENI</ID>
        <Name>KOURENI</Name>
        <Type>Long</Type>
        <Content>Discretized</Content>
        <KeyColumns>
            <KeyColumn>
                <DataType>Integer</DataType>
                <DataSize>-1</DataSize>
                <Source xsi:type="ColumnBinding">
                    <TableID>dbo_Excel</TableID>
                    <ColumnID>KOURENI</ColumnID>
                </Source>
            </KeyColumn>
        </KeyColumns>
    </Column>
    <Column xsi:type="ScalarMiningStructureColumn">
        <ID>ALKOHOL</ID>
        <Name>ALKOHOL</Name>
        <Type>Long</Type>
        <Content>Discretized</Content>
        <KeyColumns>
            <KeyColumn>
                <DataType>Integer</DataType>
                <DataSize>-1</DataSize>
                <Source xsi:type="ColumnBinding">
```

```
                        <TableID>dbo_Excel</TableID>
                        <ColumnID>ALKOHOL</ColumnID>
                    </Source>
                </KeyColumn>
            </KeyColumns>
        </Column>
        <Column xsi:type="ScalarMiningStructureColumn">
            <ID>ROKNAR</ID>
            <Name>ROKNAR</Name>
            <Type>Long</Type>
            <Content>Discretized</Content>
            <KeyColumns>
                <KeyColumn>
                    <DataType>Integer</DataType>
                    <DataSize>-1</DataSize>
                    <Source xsi:type="ColumnBinding">
                        <TableID>dbo_Excel</TableID>
                        <ColumnID>ROKNAR</ColumnID>
                    </Source>
                </KeyColumn>
            </KeyColumns>
        </Column>
    </Columns>
    <MiningModels>
        <MiningModel>
            <ID>Model</ID>
            <Name>Model</Name>
            <Algorithm>Microsoft_Association_Rules</Algorithm>
            <Columns>
                <Column>
                    <ID>ICO</ID>
                    <Name>ICO</Name>
                    <SourceColumnID>ICO</SourceColumnID>
                    <Usage>Key</Usage>
                </Column>
                <Column>
                    <ID>STAV</ID>
                    <Name>STAV</Name>
                    <SourceColumnID>STAV</SourceColumnID>
                    <Usage>Predict</Usage>
                </Column>
                <Column>
                    <ID>KOURENI</ID>
                    <Name>KOURENI</Name>
                    <SourceColumnID>KOURENI</SourceColumnID>
                    <Usage>Predict</Usage>
                </Column>
                <Column>
                    <ID>ALKOHOL</ID>
                    <Name>ALKOHOL</Name>
                    <SourceColumnID>ALKOHOL</SourceColumnID>
                    <Usage>Predict</Usage>
                </Column>
                <Column>
                    <ID>ROKNAR</ID>
                    <Name>ROKNAR</Name>
                    <SourceColumnID>ROKNAR</SourceColumnID>
                    <Usage>Predict</Usage>
                </Column>
            </Columns>
            <Language>1033</Language>
```

```
                    <Collation>Czech_CI_AS</Collation>
                </MiningModel>
            </MiningModels>
        </MiningStructure>
    </ObjectDefinition>
</Create>
```

Zpracování modelu

```
<Batch xmlns="http://schemas.microsoft.com/analysisservices/2003/engine">
<Parallel>
<Process xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ddl2="http://schemas.microsoft.com/analysisservices/2003/engine/2"
xmlns:ddl2_2="http://schemas.microsoft.com/analysisservices/2003/engine/2/
2">
  <Object>
    <DatabaseID>dataminingDOTNET</DatabaseID>
    <MiningStructureID>Model</MiningStructureID>
    <MiningModelID>Model</MiningModelID>
  </Object>
  <Type>ProcessFull</Type>
  <WriteBackTableCreation>UseExisting</WriteBackTableCreation>
</Process>
</Parallel>
</Batch>
```

Chybová odpověd ze serveru

```
<return xmlns="urn:schemas-microsoft-com:xml-analysis">
<results xmlns="http://schemas.microsoft.com/analysisservices/2003/xmla-
multipleresults">
  <root xmlns="urn:schemas-microsoft-com:xml-analysis:empty">
    <Exception xmlns="urn:schemas-microsoft-com:xml-analysis:exception" />
    <Messages xmlns="urn:schemas-microsoft-com:xml-analysis:exception">
      <Error ErrorCode="3239313412" Description="Errors in the metadata
manager. Either the mining structure with the ID of 'Plat_AS' does not
exist in the database with the ID of 'datamining', or the user does not
have permissions to access the object." Source="Microsoft SQL Server 2005
Analysis Services" HelpFile="" />
    </Messages>
  </root>
</results>
</return>
```