VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

PODPORA ROZHODOVÁNÍ V CRM SYSTÉMU

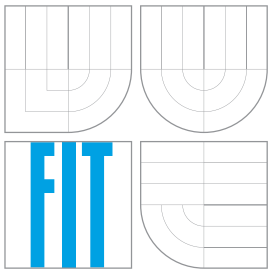
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. MICHAL FILUS

BRNO 2011

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

PODPORA ROZHODOVÁNÍ V CRM SYSTÉMU

DECISION MAKING SUPPORT IN CRM SYSTEMS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. MICHAL FILUS

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. OTA JIRÁK

BRNO 2011

Abstrakt

Práce se zabývá problematikou podpory rozhodování v CRM systémech. Cílem práce bylo navrhnout modul pro podporu rozhodování v CRM systému použitím analytických nástrojů od skupiny Pentaho. Teoretická část práce obsahuje popis datových skladů a získávání znalostí z databází se zaměřením na analytické operace a podporu rozhodování. Dále obsahuje stručný popis CRM systémů a možné aplikace podpory rozhodování v těchto systémech. Praktická část práce se zabývá architekturou CRM systému CRMminer a popisuje modul pro podporu rozhodování v tomto systému.

Abstract

The thesis deals with a decision making support in CRM systems. The goal of this thesis was to design a module for decision making support in the CRM system using Pentaho analysis tools. The theoretical part contains a description of the data warehousing and data mining with a focus on analytic operations and decision making support. It also contains brief description of CRM systems and possible application of decision making support in these systems. The practical part deals with the description of architecture of CRM system CRMminer and describes the decision making support module in this system.

Klíčová slova

Podpora rozhodování, získavání znalostí, OLAP, CRM systémy.

Keywords

Decision making support, data mining, OLAP, CRM systems.

Citace

Michal Filus: Podpora rozhodování v CRM systému, diplomová práce, Brno, FIT VUT v Brně, 2011

Podpora rozhodování v CRM systému

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Ota Jiráka. Další informace mi poskytl Ing. Petr Chmelař. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Michal Filus
19. května 2011

Obsah

Kapitola 1

Úvod

Dnešné trhové prostredie sa nesie v duchu globalizácie. Je ovplyvnené rozvojom komunikačných a informačných technológií. V takomto prostredí prestávajú fungovať klasické obchodné prístupy a spoločnosti sú donútené prispôsobiť sa za účelom dosiahnutia svojich obchodných cieľov na jednotlivých trhoch. Riešením je ponúkať produkty a služby, ktoré sú variabilné a schopné reflektovať aktuálne požiadavky zákazníkov. Predpokladom pre takúto ponuku je však schopnosť dokonale poznať svojich zákazníkov a správne identifikovať ich potreby.

CRM systémy (*Customer Relationship Management*) sú systémy pre manažment vzťahov so zákazníkmi. Umožňujú spoločnostiam zhromaždiť veľké množstvo informácií z jednotlivých oddelení obchodu na jedno miesto, spraviť ich použiteľnými pre svojich zamestnancov a postaviť tak zákazníka do stredu záujmu spoločnosti. Dôležitou súčasťou kvalitných CRM systémov je ich analytická časť, ktorej úlohou je poskytnúť manažérom spoločnosti podporu pri ich rozhodovaní pomocou metód analýzy dát a techník získavania znalostí.

Tento dokument sa zaoberá problematikou aplikácie podpory rozhodovania v CRM systémoch. Cieľom tejto práce je navrhnúť univerzálny modul, ktorý umožní spracovať údaje uložené v databáze a splniť definované požiadavky manažérov na získanie sumárnych informácií z rôznych modulov aplikácie. Na tento účel využíva voľne dostupné nástroje pre analýzu dát *Mondrian*[20] a *WEKA*[14] od skupiny *Pentaho*. Tieto nástroje sú integrované priamo do aplikácie. V práci je popísaný spôsob ich integrácie a riešenie problémov s tým spojených.

V úvode práce je popísaná stručná história spracovania dát. Vysvetľuje pojem multidimenzionálneho modelu uloženia dát, popisuje problematiku dátových skladov a nakoniec analýzu dát pomocou *OLAP*.

Metódy získavania znalostí umožňujú objaviť v dátach nové, skryté informácie, ktoré nie sú klasickým prístupom spracovania dát viditeľné. Vysvetlením procesu získavania znalostí z databáz a jeho etáp sa zaoberá tretia kapitola práce. Popisuje proces prípravy dát, vybrané typy modelov pre dolovanie, spôsob vyhodnotenia získaných výsledkov a poukazuje na možnosti použitia týchto techník pri reálnych problémoch.

Piata kapitola je venovaná problematike CRM systémov. Obsahuje ich základnú charakteristiku a popisuje obecnú štruktúru aplikácie. Na túto kapitolu úzko naväzuje kapitola o aplikovaní podpory rozhodovania v tomto type systémov, ktorá popisuje použitie OLAP a metód získavania znalostí v analytickej časti CRM systémov.

Nástroje pre analýzu dát boli integrované do CRM systému s názvom *CRMminer*. Šiesta kapitola obsahuje popis technológií pri implementácií tohto systému. Je uvedený dôvod

ich použitia a prínos z hľadiska výslednej kvality CRM systému. Popisuje architektúru aplikácie s dôrazom na tie aspekty návrhu, ktoré boli dôležité pre integrovanie a správnu funkčnosť uvedených analytických nástrojov. Pre získanie základného prehľadu o štruktúre CRM systému obsahuje niektoré z vývojových diagramov.

Siedma kapitola práce popisuje návrh modulu pre podporu rozhodovania. Obsahuje stručný popis nástrojov Mondrian a WEKA, spôsob ich integrácie do aplikácie. Ďalej obsahuje príklad použitia modulu v reálnom nasadení systému, na ktorom je demonštrovaná jeho funkčnosť. Záver kapitoly sa venuje zhodnoteniu vlastností modulu a poukazuje na jeho možné rozšírenia do budúcnosti.

Kapitola 2

Analýza dát

Pred samotným popisom analýzy dát je potrebné objasniť pojmy databázový systém a relačná databáza. Databázový systém (anglicky *database management system*, *DBMS*), alebo systém riadenia bázy dát, pozostáva z kolekcie navzájom prepojených dát, nazývaných databáza a skupinou softvérových aplikácií pre správu a prístup k dátam. Tieto aplikácie zahŕňajú postupy pre definíciu databázových štruktúr, pre ukladanie dát, pre súbežný, zdieľaný alebo distribuovaný prístup k dátam a na zabezpečenie ich konzistencie a uložených informácií aj v prípade zlyhania systému alebo pokusu o neoprávnený prístup [13].

Relačná databáza je tvorená množinou tabuliek, z ktorých každá má priradený vlastný jedinečný názov. Každá tabuľka sa skladá z množiny atribútov (stĺpcov alebo polí) a obvykle ukladá veľký súbor n-tic (záznamov alebo riadkov). Každý záznam v relačnej tabuľke reprezentuje objekt štandardne identifikovaný pomocou unikátneho kľúča a popísaný množinou hodnôt atribútov. Relačná databáza je modelovaná pomocou *ER modelu* (*Entition-Relationship Model*). ER model reprezentuje dáta v databáze ako súbor entít a ich vzťahov [13]. K dátam uloženým v relačnej databáze sa pristupuje pomocou databázových dotazov, ktoré sú tvorené jazykom pre dotazovanie nad relačnou databázou SQL. Užívateľ systému môže zadávať tieto dotazy priamo alebo pomocou grafického užívateľského rozhrania, cez ktoré sa dajú nadefinovať požadované atribúty a obmedzenia na týchto atribútoch, viac na [13].

V súčasnosti každá spoločnosť používa pre svoju činnosť nejaký druh softvéru pre podporu podnikania. Sú to napríklad ekonomické systémy pre účtovníctvo, skladové hospodárstvo alebo systémy pre manažment zákazníkov CRM a iné. Používaním týchto systémov sa zhromažďujú údaje. Niektoré z týchto dát sú veľmi cenné, často však zostávajú nevyužité, pretože sú uložené vo forme, ktorá ich činí nedostupnými pre účely získavania informácií. A tak, aj napriek existencií údajov, sa vedúci pracovníci k potrebným informáciám nemusia dostať. A potom ani najlepší manažér nedokáže urobiť zodpovedajúce rozhodnutia. Z tohto dôvodu bolo potrebné nájsť možnosti, ako dostupné informácie spracovať a dať ich prehľadným spôsobom k dispozícií manažérom. Jednou z ciest, ako dosiahnuť efektívne zhodnotenie týchto informácií, je zavedenie systémov pre analytické spracovanie dát *OLAP* (*Online Analytical Processing*) [13].

V nasledujúcej kapitole sa nachádza popis informačných systémov pre manažérov *EIS*, ktoré môžu byť považované za predchodcov dnešných OLAP systémov. Druhá časť kapitoly je venovaná práve samotným OLAP systémom.

2.1 Informačné systémy pre manažérov

EIS (*Executive Information Systems*, slovensky podnikové informačné systémy) bol prvý pokus ako umožniť manažérom dotazovanie do databázy za účelom analýzy dát. Jednalo sa o systémy s jednoduchým užívateľským rozhraním, ktorých úlohou bolo odtieniť užívateľa od syntaxi SQL jazyka a od nutnosti poznať štruktúru databázy s ktorou chcel pracovať. Princíp tohto oddelenia spočíval v tom, že v menu aplikácie boli k dispozícii vopred pripravené dotazy, ktoré sa následne prevádzali do jazyka SQL.

Výhodou toho prístupu bola jednoduchosť. Nevýhoda týchto systémov však spočívala v tom, že ak analytik alebo manažér potreboval získať informácie, ktorým nevyhovoval žiadny z pripravených dotazov, opäť bol nútený konzultovať riešenie s programátorom. EIS boli teda užívateľsky priateľské, ale málo flexibilné nástroje pre analýzu dát v databázach, viac na [4].

Tieto systémy sú stále častejšie nahradzované systémami *DSS* (*Decision-Support Systems*) pre podporu rozhodovania. DSS systémy poskytujú taktickú a strategickú úroveň rozhodovania. Poskytujú vedúcim pracovníkom výsledky pomerne zložitých analýz, viac na [13].

2.2 OLAP

Klasické transakčné databázy (označované ako *Online Transaction Processing*, *OLTP*) sú charakteristické tým, že k dátam, ktoré obsahujú, pristupuje v rovnakom čase veľké množstvo užívateľov, ktorí načítavajú, ukladajú alebo prevádzajú jednoduché analytické operácie nad týmito dátami. Teoreticky by bolo možné prevádzať analýzu dát aj nad OLTP databázami. Problémom je, že údaje uložené v transakčných databázach by mali byť v normalizovaných tabuľkách, čo znamená veľa atomických, relačne zviazaných tabuliek. Analýza veľkého množstva takto uložených dát by tak bola neefektívna a časovo náročná. Z týchto dôvodov bola zavedená technológia organizácie údajov ako *multidimenzionálny dátový model* [13].

2.2.1 Dátové sklady

V súvislosti s multidimenzionálnym modelom je potrebné objasniť pojem *dátových skladov*. Dátové sklady a technológia OLAP sú založené na multidimenzionálnom dátovom modely [8]. Dátový sklad je možné definovať ako podnikovo štruktúrovaný depozitár subjektovo orientovaných, integrovaných, časovo premenlivých, historických dát použitých na získavanie informácií a podporu rozhodovania. V dátovom sklade sú uložené atomické a agregované dáta [13].

Dátový sklad vzniká spojením niekoľkých heterogénnych zdrojov dát. Existuje niekoľko prístupov k vytváraniu dátových skladov. Väčšinou sa jedná o dlhodobý a náročný proces, ktorý prebieha iteratívne v etapách. Tieto takzvané prírastkové metódy môžu prebiehať smerom zhora-dolu, keď sa vytvára dátový sklad na základe požiadaviek užívateľov alebo metódou zdola-nahor keď majú prioritu údaje pred obchodným ziskom.

Dáta, z ktorých je dátový sklad vytvorený, pochádzajú z rôznych zdrojov a ich formát často nie je vyhovujúci. Príprava a zavedenie údajov je dôležitou súčasťou každého riešenia dátového skladu. Príprava dát zahŕňa ich extrakciu, čistenie a transformáciu do požadovanej podoby. Tento proces prípravy dát sa nazýva *ETL* (*Extraction*, *Transformation*, *Loading*). Skladá sa z extrakcie dát z rôznych zdrojov, ich transformácie a premiestnenia do dátového

skladu. Hlavným cieľom ETL procesov je centralizácia údajov, môžeme ich nazvať aj ako dátové pumpy pre dátový sklad, viac na [13]. Podrobnejší popis prípravy dát je popísaný v kapitole venovanej získavaniu znalostí z databáz.

2.2.2 Popis technológie

OLAP bol riešením, ktorý ponúkol užívateľom flexibilitu, rýchlosť a príjemné, intuitívne ovládanie. Jedná sa o technológiu uloženia dát v databáze, ktorá umožňuje usporiadať veľké objemy dát tak, aby boli jednoducho prístupné manažérom, analytikom alebo iným užívateľom, ktorí sa zaoberajú analýzou obchodných trendov a výsledkov. Výsledkom analýzy sú súhrny a reporty, ktoré slúžia manažérom ako podklady pre ich rozhodovanie a to v oblasti riadenia firmy, riadenia technologických a ekonomických procesov a podobne [13, 4].

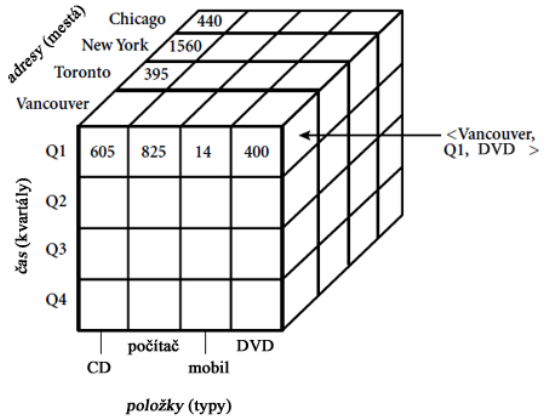
Technológiu OLAP je možné charakterizovať ako:

- multidimenzionálny koncept uloženia a manipulácie s dátami,
- intuitívna manipulácia s dátami,
- práca s dátami heterogénnych dátových zdrojov,
- prevádzanie konverzie dát,
- použitie analytických metód,
- klient / server architektúra,
- podpora užívateľského pohľadu,
- ukladanie výsledkov OLAP mimo zdrojové dáta,
- dynamická manipulácia s riedkymi maticami,
- spracovanie chýbajúcich hodnôt,
- neobmedzený počet dimenzií a úrovní hierarchií [4].

2.2.3 Dátové kocky

Základom technológie OLAP je pohľad na dáta ako na mnohorozmernú tabuľku nazývanú dátová kocka. Táto je výsledkom analýzy a agregácie dát. Pre výpočet dátovej kocky je potrebné vykonať veľké množstvo výpočtov a agregácií a to v reálnom čase. Každá dátová kocka má niekoľko dimenzií. Na rozdiel od geometrickej kocky môže mať multitimenzionálny databázový model viac ako tri dimenzie. Príkladom typického trojdimenzionálneho modelu môže byť kocka s dimenziami čas, miesto a predaný produkt ktorá je zobrazená na obrázku 2.1 [13].

Databázu prevedieme na dátovú kocku tak, že jednotlivé sledované atribúty budú tvoriť dimenziu kocky. Bunky potom odpovedajú jednotlivým záznamom v databáze. Takýto spôsob uloženia umožňuje používanie rôznych pohľadov na dáta ako sú natáčanie kocky alebo prevádzanie rezov. Veľký počet buniek však môže zostať prázdnych, plytvá sa teda zbytočne miestom.



Obrázek 2.1: Príklad dátovej kocky, upravené podľa [8].

Tabulka 2.1: Záznamy databázy predaj, spracované podľa [4].

dátum	produkt	mesto	množstvo
10.1.2009	skrutky	Praha	241
10.1.2009	matice	Praha	61
10.1.2009	skrutky	Brno	17
10.1.2009	podložky	Brno	42
10.2.2009	skrutky	Praha	92
10.2.2009	podložky	Praha	27

V tabuľke 2.1 sú zobrazené záznamy z databázy predaj. Obsahuje informácie o počte predaných položiek s rozličných druhov tovaru v rôznych mestách v určitom čase. V nasledujúcej tabuľke 2.2 je ukázaný prevod databázy na dátovú kocku. Mnohé položky v kocke sú prázdne. Matematicky sa takýto typ štruktúry nazýva *riedka matica*.

Dátová kocka obsahuje dáta z operačných databáz a aj ich čiastkové súhrny. Tieto súhrny umožňujú rýchlu odozvu na nepredpripravené dotazy užívateľa a flexibilitu systému. Samotná práca s dátovou kockou spočíva v rôznom natáčaní (anglicky *pivot*), prevádzaní rezov (anglicky *slice*), výberu určitých častí (anglicky *dice*) a zobrazovaní rôznych agregovaných hodnôt. Hodnoty atribútov je možné združovať do *hierarchií*, ktoré môžu mať viacej úrovní. Hierarchie sa používajú pri operáciách *roll-up* a *roll-down*. Operácia roll-up znamená prechod na vyššiu, obecnejšiu úroveň, zobrazované údaje majú podobu súhrnov. Pri roll-down dochádza naopak ku prechodu na nižšiu úroveň pohľadu, ponúka podrobnejší pohľad na dáta, viac na [4].

Tabulka 2.2: Prevod databázy na dátovú kocku, spracované podľa [4].

	Praha			Brno		
	skrutky	matice	podložky	skrutky	matice	podložky
10.1.2009	241	61		17		
10.2.2009	92		27			

Základný multidimenzionálny model má podobu n–rozmernej kocky. Z hľadiska implementácie existuje niekoľko možností, akými je možné túto štruktúru uložiť. Odlišnosť jednotlivých implementácii je v práci s riedkymi dátami a v efektívnosti uloženia dát. V zásade rozlišujeme dva prístupy [4]:

- *Hyperkocka* – jedna veľká kocka, ktorá obsahuje nástroje pre prácu s riedkymi dátami. Jej výhodou je jednoduchá štruktúra a zrozumiteľnosť pre užívateľa.
- *Multikocka* – viac navzájom prepojených menších kociek ktoré obsahujú len niekoľko dimenzií. Jej výhodou je efektívne uloženie dát.

2.2.4 ROLAP

Nevýhodou uloženia dát v dátovej kocke sú zvýšené nároky na dátový server, ktoré vznikajú potrebou rýchleho prístupu k dátam. Tieto nároky viedli k zavedeniu *relačného OLAP* (*ROLAP*) miesto štandardného OLAP. OLAP prístup je niekedy nazývaný aj ako *MOLAP* (*multidimenzionálny OLAP*) [4].

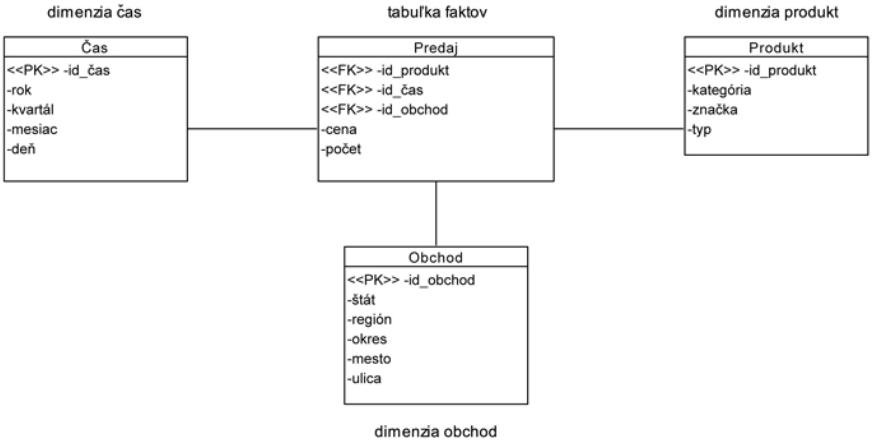
ROLAP prístup sa hodí pre stredne veľké, statické aplikácie. Príkladom môžu byť súhrny historických dát za nejaké obdobie. Tieto výpočty zaberajú určitý čas a práve z tohto dôvodu nie je tento prístup vhodný pre statické aplikácie.

Pri použití ROLAP prístupu sa dotazy OLAP prevádzajú do klasických SQL dotazov. Preto je vhodný pre rozsiahle aplikácie, kde sú využívané transakčné dáta. Výhodou je možnosť spracovávania rozsiahlych dát za použitia existujúcich databázových technológii. Existujú rôzne možnosti implementácie systému ROLAP:

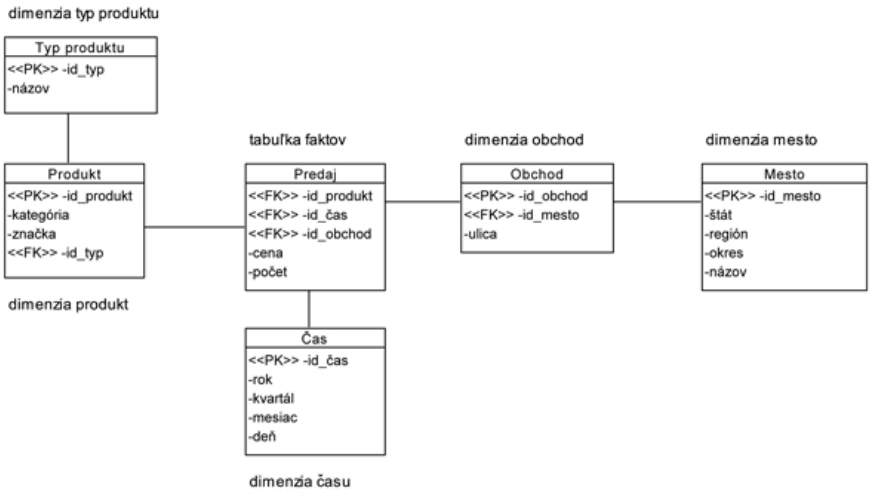
- *schéma hviezdy* (obrázok 2.2),
- *schéma vločky* (obrázok 2.3),
- *súhvezdie faktov* (obrázok 2.4).

Pri prístupe schéma hviezdy sa vychádza z jednej centrálnej tabuľky, takzvanej tabuľky faktov. Táto obsahuje zložený primárny kľúč, kde je jeden segment kľúča pre každú dimenziu a detailné alebo aj agregované dáta. Pre každú dimenziu existuje jedna tabuľka, ktorá obsahuje údaje na rôznej úrovni príslušnej hierarchie. Túto tabuľku nazývame tabuľka dimenzií. Kvôli prehľadnosti musí táto tabuľka obsahovať aj záznam o úrovni hierarchie, ktorej sa daný záznam týka. Výhodou hviezdy je jej zrozumiteľnosť, ľahké definovanie hierarchií, jednoduché metadáta a rýchlosť prístupu. Nevýhodou sú problémy s veľkými tabuľkami dimenzií a predpoklad, že pracujeme so statickými dátami, ktoré nie sú aktualizované v reálnom čase.

Alternatívou ku schéme hviezdy je schéma snehovej vločky. Pri tomto prístupe sa pracuje s normalizovanými tabuľkami dimenzií tak, že každá tabuľka nejakej dimenzie ukazuje na odpovedajúcu agregovanú tabuľku faktov. Tabuľky dimenzií obsahujú jediný primárny kľúč pre danú úroveň dimenzie spolu s odkazom na najbližšieho rodiča v hierarchii dimenzií. Pri použití snehovej vločky nie je potrebné používať indikátor úrovne v hierarchii. V každej tabuľke sú údaje len s jednej úrovne. Výhody tejto koncepcie sa prejavia pri používaní dotazov, ktoré sa týkajú agregovaných hodnôt. Vločka priamo zachytáva uloženie dát a tie potom zaberajú menej priestoru. Model je jednoduchý na pochopenie pre návrhárov, náročný na pochopenie pre manažérov. Nevýhodou je zložitá údržba, nárast počtu tabuliek v databáze a spomalenie rýchlosti odozvy.



Obrázek 2.2: Schéma hviezdy, vytvorené podľa [8].

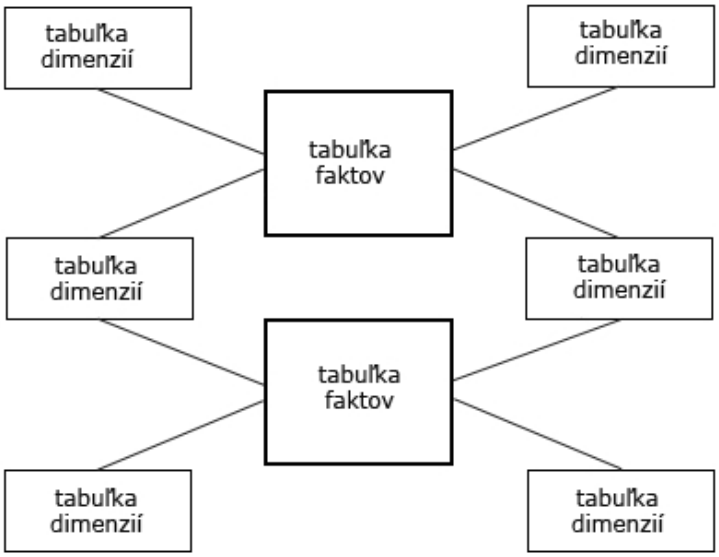


Obrázek 2.3: Schéma vločky, vytvorené podľa [8].

Posledným typov schémy multidimenzionálnej databázy je súhvezdie faktov. Pri tomto type uloženia dát existujú viaceré tabuľky faktov, ktoré zdieľajú rovnaké tabuľky dimenzií [8].

2.2.5 Zobrazenie dát

Jednou z možností, ako zobrazovať mutidimenzionálne dáta užívateľsky prehľadným spôsobom je *kontingenčná tabuľka* (anglicky *Pivot Table*). Kontingenčná tabuľka má na rozdiel od klasickej tabuľky niekoľko špeciálnych vlastností. Napríklad umožňuje určitú rotáciu, teda výmeny riadkov a stĺpcov, kombinácie a hierarchickú štruktúru riadkov a stĺpcov. Názvy riadkov a stĺpcov sú tvorené dimenziami, bunky tabuľky tvoria údaje z tabuľky faktov. Dáta, ktoré tabuľka zobrazuje, sú výsledkami multidimenzionálnych dotazov vytvorených pomocou jazyka MDX, jeho popis sa nachádza v ďalšej podkapitole, viac na [13].



Obrázek 2.4: Súhvezdie faktov, vytvorené podľa [8].

2.2.6 Jazyk MDX

MDX (anglicky *Multidimensional Expressions*) je jazyk pre dotazovanie v multidimenzionálnych databázach. Dá sa považovať za ekvivalent jazyka SQL, ktorý je používaný pre dotazovanie v relačných databázach. Pomocou dotazu MDX sa vypíše vybraná podmnožina dát z dátovej kocky do kontingenčnej tabuľky. Základná štruktúra príkazov je nasledovná:

Príklad 2.1: Základná štruktúra MDX dotazov.

```
SELECT <<zoznam osí>> ON COLUMNS,
<< zoznam osí >> ON ROWS,
<< zoznam osí >> ON PAGE
FROM <<kocka>>
WHERE <<rezy>>
```

Pri vytváraní MDX dotazov je potrebné poznať štruktúru dátovej kocky s ktorou sa pracuje. Pomocou klauzulí `ON COLUMNS` a `ON ROWS` sa definujú dimenzie, ktoré budú zobrazované v stĺpcoch a riadkoch tabuľky. Rovnakým spôsobom je možné nadefinovať, ktoré údaje z tabuľky faktov budeme chcieť pri zobrazení vidieť. Ak nie je žiada zadaná, OLAP pracuje s hodnotami faktu, ktorý je definovaný ako predvolený pre zvolenú dátovú kocku, viac na [13].

Príklad 2.2: Ukážka dotazu v jazyku MDX.

```
select {[Measures].[Odhad hodin], [Measures].[Realne hodiny]} ON COLUMNS,
{([Zakaznik].[Brno], [Projekt].[eshop])} ON ROWS
from [Task]
```

Ukážka reálneho dotazu je uvedená na príklade 2.2. V stĺpcoch výslednej tabuľky sa

nachádzajú fakty o odhadovaných a reálne strávených hodinách na projekte. V riadkoch tabuľky sú nadefinované dimenzie zakazník a projekt. Výsledkom tohto dotazu je informácia, koľko hodín celkovo sa strávilo na projektoch pre elektronický obchod, ktoré boli vypracované pre zákazníkov z Brna, v závislosti na tom, koľko hodín bol pôvodný odhad na vývoj.

Kapitola 3

Získavanie znalostí z databáz

V dnešnej dobe majú spoločnosti uložené v databázach svojich informačných systémov veľké informačné bohatstvo. Databázové tabuľky môžu obsahovať veľký počet záznamov, ktoré sú rôzne usporiadané, členené. Problémom získania týchto informácii, ich správnym pochopením a spracovaním sa zaoberá proces získavania znalostí z databáz.

Získavanie znalostí je proces analýzy dát z rôznych perspektív a ich premena na užitočné informácie. Je to extrakcia zaujímavých informácií, ktoré sú netriviálne, skryté, predtým neznáme a môžu byť potenciálne užitočné. Z matematického a štatistického hľadiska ide o hľadanie korelácii, teda vzájomných vzťahov alebo vzorov v údajoch. Na tento účel využíva štatistické metódy a metódy z oblasti umelej inteligencie. Pomáha sledovať, analyzovať trendy a predvídať udalosti, viac na [4, 8].

Proces získavania znalostí , zobrazený na obrázku 3.1, sa skladá z niekoľkých etáp:

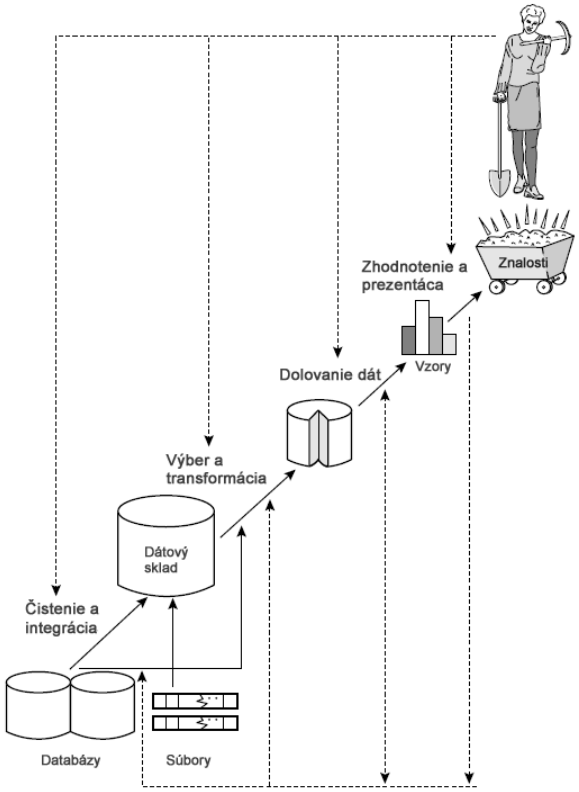
- definícia problému,
- výber vhodných dát,
- výber vhodného algoritmu a modelu,
- predspracovanie dát,
- dolovanie dát,
- interpretácia znalostí.

Jedná sa o iteratívny proces, každá z týchto etáp ma definované svoje kroky a ciele. Podrobný popis jednotlivých etáp sa nachádza v nasledujúcich podkapitolách.

Najväčší dôraz je kladený na prípravu dát pre analýzu a na interpretáciu výsledných znalostí. Pri príprave dát sa obvykle z dát uložených v zložitej štruktúre, akou je napríklad dátový sklad, vytvára jedna tabuľka obsahujúca relevantné údaje o sledovaných objektoch. Pod relevantnými údajmi rozumieme hodnoty rôznych atribútov. Sledované objekty predstavujú napríklad zákazníci, klienti banky, údaje o predaji a podobne. Pri interpretácii sa nájdené znalosti hodnotia z pohľadu koncového užívateľa. Tieto znalosti je potrebné prezentovať vhodnou formou, aby sa stali užitočnými.

3.1 Definícia problému

Impulzom pre zahájenie procesu získavania znalostí je nejaký reálny problém a cieľom je získať čo najviac relevantných informácii vhodných k jeho riešeniu. Prvým krokom pri riešení



Obrázek 3.1: Proces získavania znalostí, prevzaté a upravené z [8].

problému je zostavenie riešiteľského tímu. Tento by sa mal skladať z experta na riešenú problematiku, experta na dáta v organizácii a nakoniec experta na metódy získavania znalostí.

Prvou úlohou tímu je špecifikácia problému, ktorý je potrebné riešiť v súvislostiach získavania znalostí. To zahŕňa napríklad zisťovanie vzťahov medzi rôznymi položkami, definovanie skupín položiek, problém závisí od oblasti použitia získavania znalostí.

Po špecifikácii je potrebné získať všetky dostupné dáta, ktoré môžu byť použité pre riešenie problému. Znamená to posúdiť všetky dostupné dáta a zvážiť, či odpovedajú danému problému. Náročnosť získania dát je nepriamo úmerná úrovni dátovej základni, ktorá je k dispozícii [4].

3.2 Príprava dát

Príprava dát je najzložitejším a časovo najnáročnejším krokom celého procesu získavania znalostí z databáz a má kľúčový význam pre úspech danej aplikácie. V tomto kroku sa vyžaduje najväčšia miera spolupráce s expertom z danej aplikačnej oblasti.

Cieľom predspracovania je najskôr vybrať z dostupných dát tie údaje, ktoré sú relevantné pre zvolenú úlohu získavania znalostí. Je potrebné dôkladne porozumieť problémom a dostupným dátam z cieľovej aplikačnej oblasti.

Po zvolení správnych dát nasleduje snaha reprezentovať tieto údaje takým spôsobom, aby boli vhodné pre spracovanie zvoleným algoritmom pre získavanie znalostí. Získané dáta

musia byť uspôsobené pre jednotlivé analytické procedúry, preto pri predspracovaní musíme poznať požiadavky jednotlivých metód na podobu vstupných dát.

Dáta, ktoré chceme použiť pre analýzu pomocou techník získavania znalostí, sú často nekompletné, obsahujú šum a sú nekonzistentné. Nekompletnosť dát spôsobujú napríklad chýbajúce hodnoty atribútov alebo agregované údaje, kde by bolo potrebné poznať detailné hodnoty položiek a nie len ich súhrny. Šum v dátach je spôsobovaný chybnými alebo odľahlými hodnotami, rozdielnym formátom dát a podobne. Príkladom vzniku šumu sú neošetrované vstupy pri ukladaní dát z formulárov do databázy. Z nekonzistentnosťou sa stretávame pri integrácií dát z viacerých zdrojov, kde môže vzniknúť problém s odlišným pomenovaním, kódovaním, jedná sa o problém redundancie dát.

Cieľovým stavom predspracovania pre väčšinu metód je reprezentácia dát prostredníctvom jednej tabuľky, ktorá zachytáva hodnoty atribútov objektov. Prostriedkom k dosiahnutiu tohto cieľa potom budú rôzne metódy čistenia dát, integrácie a transformácie dát, selekcie a výberu atribútov [4, 8].

3.2.1 Sumárne popisné charakteristiky dát

Pre úspešné predspracovanie dát je nevyhnutné, aby sme získali celkový obraz o dátach, ktoré máme k dispozícií. Techniky popisnej charakteristiky dát môžu byť použité na identifikáciu ich typických vlastností a zvýraznenie dát, ktoré by mali byť považované za šumové.

Dáta je možné charakterizovať pomocou rôznych štatistických mier. Z pohľadu výpočtu nad súborom dát ich môžeme rozdeliť na *distributívne*, *algebraické* a *holistické miery*, viď [8]. Pre mnoho úloh predspracovania je potrebné poznať *strednú hodnotu dát* a ich *rozptyl*. Strednú hodnotu súboru dát je možné určiť pomocou *aritmetického priemeru*, *mediánu* alebo *modusu*. Rozptyl dát sa určuje napríklad pomocou *miery rozsahu hodnôt*, ktorý sa určuje ako rozdiel maximálnej a minimálnej hodnoty dát, pomocou *kvartilov* alebo pomocou *medzikvartilovej vzialenosti IQR*, viac na [8].

Okrem stĺpcových, koláčových a čiarových grafov sa pre zobrazenie charakteristík dát používajú niektoré špeciálne typy grafov ako *krabicový graf* (anglicky *boxplot*), *histogramy*, *kvantilový graf*, *q-q graf*, *bodový graf* a iné, viac na [8].

3.2.2 Čistenie dát

Reálne dáta majú tendenciu byť neúplné, hlučné, a nekonzistentné. Čistenie dát je proces úpravy dát, ktorého cieľom je doplnenie chýbajúcich hodnôt, vyhladenie šumu pri identifikácii odľahlých hodnôt a riešenie nekonzistencie a redundancie dát. Pozostavá z dvoch krokov. Prvým krokom je detekcia nezrovnalostí, kedy sa pomocou *metadát* snažíme pochopiť dáta a identifikovať prípadné nezrovnalosti. Čím viac informácií tieto metadáta obsahujú, tým je proces detekcie jednoduchší a efektívnejší. Druhým krokom je odstránenie nezrovnalostí, kedy sa snažíme objavené chyby v dátach odstrániť. Riešenie jedného problému v dátach však môže podnietiť vznik ďalších, preto sa jedná o *iteratívny proces* [8].

Chýbajúce hodnoty

Pri spracovaní objektov s chýbajúcimi hodnotami je možné použiť niektorý z nasledujúcich spôsobov:

- ignorovať objekt s nejakou chýbajúcou hodnotou,
- doplniť chýbajúcu hodnotu manuálne,

- nahradiť chýbajúcu hodnotu globálnou konštantou alebo priemerom hodnôt atribútu,
- nahradiť chýbajúcu hodnotu priemerom hodnôt atribútov objektov, ktoré patria to rovnakej triedy,
- doplnenie chýbajúcich hodnôt pomocou použitia niektorého algoritmu pre modelovanie. Atribút s chýbajúcimi hodnotami sa považuje za cieľový atribút a chýbajúce hodnoty sú potom doplnené na základe modelu, viac na [8, 4].

Šum v dátach

Šum je náhodná chyba, alebo odchýlka v meranej veličine. Na odstránenie šumu v spracovávaných dátach sa používajú nasledujúce techniky vyhladzovania dát:

- *Plnenie* (*binning*) – technika vyhladenia numerických dát na základe okolitých hodnôt, rozdeluje obor hodnoť atribútov do košov (*bins*) a následne prevádza lokálne vyhladenie nahradením hodnôt v koši ich priemerom, mediánom alebo hraničnou hodnotou.
- *Regresia* – hodnoty atribútu sú doplnené na základe regresnej krivky. Využíva sa predikcia hodnoty atribútu na základe hodnôt jedného (*lineárna regresia*) alebo viacerých (*viacnásobná lineárna regresia*) susedných atribútov objektu.
- *Zhlukovanie* – technika pre hľadanie odľahlých hodnôt. Zoskupuje hodnoty atribútu do zhlukov. Tie, ktoré nie sú zaradené v žiadnom zhluku, sú identifikované ako odľahlé hodnoty.

Niektoré z techník vyhladzovania sa používajú pri redukcií dát. Použitím plnenia redukujeme počet možných hodnôt, ktoré môže atribút nadobudnúť. Tieto techniky je možné použiť aj k diskretizácií hodnôt numerických atribútov, keď sa ich hodnoty nahradia intervalmi, viac na [8].

3.2.3 Integrácia a transformácia dát

Proces získavania znalostí často vyžaduje integrovať dáta, ktoré pochádzajú z viacerých zdrojov. Tieto dáta musia byť následne transformované do formy vhodnej pre ďalšie spracovanie.

Integrácia

Pri integrácií je potrebné riešiť celý rad problémov ako konflikt schémy, konflikt identifikácie a problém redundancie dát. Pri konflikte schémy sa rieši integrácia metadát do iných metadát, kde je potrebné správne spárovať jednotlivé atribúty z rozličných zdrojov, tieto môžu mať rozdielny názov alebo tvar. Konflikt identifikácie súvisí s rozpoznaním konkrétnych objektov na základe jednoznačného identifikátora pričom tento sa môže líšiť v závislosti od zdroja dát. Redundancia súvisí s výskytom rovnakých atribútov v rôznych zdrojoch dát a s odvodenými atribútmi, ktoré sú nadbytočné a ich hodnoty je možné odvodiť na základe hodnôt iných atribútov. Niektoré redundancie je možné riešiť použitím korelačnej analýzy, kde sa skúma vzájomný vzťah dvoch atribútov, ako silne implikuje jeden atribút druhý. Pre numerické atribúty sa používa *korelačný koeficient* (*Pearsonov koeficient*), pre kategorické atribúty sa používa *λ2-test dobrej shody*, viac na [8].

Transformácia

Cieľom transformácie dát je upraviť dáta do podoby vhodnej pre získavanie znalostí. Transformácia dát môže zahŕňať nasledujúce kroky:

- *Vyhladenie* – odstránenie šumu z dát.
- *Agregácia* – agregácia detailných údajov.
- *Generalizácia* – nahradenie na základe konceptuálnej hierarchie.
- *Normalizácia* – mapovanie numerických hodnôt do špecifického intervalu (*min-max*, *z-skóre*).
- *Konštrukcia atribútov* – vytváranie nových atribútov, ktoré sú vhodnejšie pre proces dolovania. Ich hodnoty sú odvodené z hodnôt iných atribútov a tieto sú potom nahradené novo vytvorenými atribútmi, viac na [8].

3.2.4 Redukcia dát

Techniky redukcie dát používame na získanie redukovanej množiny dát, ktorá má vzhľadom na pôvodnú množinu menší objem, ale zachováva charakter a integritu pôvodných dát. Použitím redukovaných dát pri získavaní znalostí by sme mali docieliť zefektívnenie procesu získavania znalostí a dosiahnuť rovnaký, alebo takmer rovnaký výsledok dolovania, aký by bol dosiahnutý použitím kompletnej množiny dát.

Existuje niekoľko stratégií redukcie dát:

- *Agregáce dátovej kocky* – agregovanie údajov, používané pri dátových skladoch.
- *Výber podmnožiny atribútov* – vylúčenie málo relevantných atribútov, len potrebné atribúty pre dolovanie.
- *Redukcia dimenzionality* – kódovanie dát pri zachovaní možnosti prevádzať operácie nad množinou dát (*PCA*, *vlnková transformácia*)
- *Redukcia počtu hodnôt* – nahradenie dát parametrickým modelom alebo neparametrickým ako napríklad zhlukovanie, vzorkovanie a iné.
- *Diskretizácia a generalizácia konceptuálnej hierarchie* – redukcia počtu rôznych hodnôt atribútov, hodnoty atribútov sú nahradené intervalmi alebo pojmami z nejakej konceptuálnej hierarchie, viac na [8].

3.3 Modely

Použitie analytických metód je jadrom celého procesu získavania znalostí. Vstupom do tejto etapy získavania znalostí sú vhodným spôsobom predspracované dáta a jej výstupom sú znalosti. Metódy získavania znalostí je možné rozdeliť podľa rôznych kritérií. Z hľadiska druhu dolovaných znalosti je rozdelenie nasledovné:

- *Charakterizácia a diskriminácia* – popis konceptu/triedy, základná forma popisného získavania znalostí. Popísanie súboru dát stručným sumárnym spôsobom, prezentácia zaujímavých vlastností dát.

- *Asociačná a korelačná analýza* – hľadanie vzťahov medzi atribútmi, hľadanie frekventovaných vzorov, asociácií a korelácií nad množinou položiek (napríklad analýza nákupného košíka).
- *Klasifikácia a predikcia* – klasifikácia sa používa pre predpovede podľa kategorických, diskrétnych hodnôt, napríklad zaradenie zákazníkov do určitých tried na základe hodnôt atribútov. Predikcia sa používa pre predpovede spojitých hodnôt atribútov, napríklad predpoveď počtu predaných produktov potenciálnym zákazníkom na základe ich príjmu.
- *Zhluková analýza* – pri tejto technike získavania znalostí nie sú vopred známe triedy. Cieľom je nájsť zhluky, ktoré budú tieto triedy predstavovať. Príkladom môže byť rozdelenie zákazníkov spoločnosti do tried na základe ich vlastností a následné prispôsobenie ponuky.
- *Analýza odľahlých hodnôt* – detekcia podvodov, analýza objektov ktoré sa vymykajú normálnemu chovaniu.
- *Analýza trendov a vývoja* – regresné analýzy, dolovanie sekvenčných vzorov, viac na [8].

V nasledujúcich kapitolách sú uvedené vybrané metódy pre získavanie znalostí z databáz, je vysvetlený ich základný princíp a možné použitie.

3.3.1 Rozhodovacie stromy

Štruktúra rozhodovacieho stromu sa skladá z uzlov a listov. Pri klasifikácií prechádzame stromom zhora-nadol a vykonávame predpísané testy v jednotlivých uzloch. Uzly obsahujú testy atribútov a listy obsahujú ohodnotenie. Výsledky jednotlivých testov nám potom určia, ktorou cestou v strome budeme ďalej pokračovať.

Základný algoritmus spočíva v tvorbe rozhodovacieho stromu metódou rozdeľ a panuj. Trénovacie dáta sa rozdeľujú na menšie a menšie podmnožiny (uzly stromu) tak, aby v týchto podmnožinách prevládali príklady tejto triedy. Na počiatku sa všetky trénovacie dáta nachádzajú v jednej množine. Na konci sú vytvorené podmnožiny, ktoré obsahujú trénovacie dáta rovnakej triedy.

Algoritmus vytvárania rozhodovacieho stromu je závislý na metódach hľadania vhodných atribútov, to znamená atribútov s najvyššou rozlišovacou schopnosťou. Existuje niekoľko metód, ako tieto atribúty získať, medzi najznámejšie patria metódy *ID3*, *C4.5* a metóda *Gini indexu*.

Algoritmus ID3 je založený na výpočte *informačného zisku* (*entropie*) pre jednotlivé atribúty. Vhodnosť atribútu sa určuje podľa hodnoty *Gain(A)* atribútu kde A predstavuje testovaný atribút. Hľadá sa atribút, kde je hodnota *Gain(A)* atribútu najväčšia alebo inak povedané ten, ktorý má najnižší informačný zisk. Vzorec pre výpočet *Gain(A)* je nasledovný:

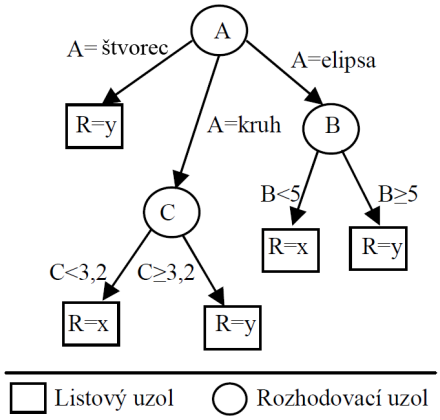
$$Gain(A) = Info(S) - Info_A(S) \tag{3.1}$$

Hodnota *Info(S)* predstavuje očakávanú informáciu potrebnú pre klasifikáciu daného vzorku. Hodnota $Info_A(S)$ predstavuje očakávanú informáciu pre klasifikáciu založenú na delení podľa atribútu A. Hotnota S reprezentuje množinu všetkých trénovacích dát, pre podrobnejšie vysvetlenie viď [27, 8].

Metóda rozhodovacích stromov je vhodná pre kategoriálne typy dát [4]. Numerické atribúty je potrebné upraviť, pretože môžu obsahovať veľký počet hodnôt, čím sú pre vytvorenie stromu nevhodné. Nie je možné urobiť pre každú hodnotu samostatnú vetvu. Tento problém sa štandardne rieši použitím intervalov pri predspracovaní dát. Diskretizácia atribútov však môže byť priamo súčasťou systémov pre tvorbu rozhodovacích stromov. Jednou z metód je rozdelenie hodnôt atribúty do dvoch skupín použitím deliaceho bodu *split-point*. Hľadá sa vhodná deliaca hodnota split-point. Simuluje sa vytvorenie uzlu a hľadá sa hodnota s najmenším informačným ziskom, viac na [8].

Nevýhodou metódy ID3 je to, že hodnota *Gain* je ovplyvnená počtom hodnôt skúmaného atribútu, kde atribút s väčším počtom hodnôt je zvýhodňovaný. Vylepšenie prináša metóda C4.5, ktorá nahradením hodnoty *Gain* hodnotou *GainRatio*. Metóda Gini indexu rieši problém zložitých výpočtov logaritmov pri metóde ID3 nahradením hodnoty informačného zisku Gini indexom. V tomto prípade je rozhodovací strom binárny, viac na [8].

Príklad rozhodovacieho stromu sa nachádza na obrázku 3.2.



Obrázek 3.2: Príklad rozhodovacieho stromu, prevzaté z [12].

Problémom vytvárania rozhodovacích stromov môže byť ich *preučenie*. K tomuto javu dochádza pri snahe bezchybne klasifikovať všetky trénovacie dáta. Strom sa stáva rozsiahly, málo zrozumiteľný. Bezchybne dokáže klasifikovať trénovacie dáta, nedokáže však spracovať dáta zaťažené šumom. Riešením je prerezávanie stromu, jeho redukcia. Redukcia je možná úpravou stávajúceho algoritmu, alebo zjednodušením a úpravou už vytvoreného stromu. Existujú dva hlavné prístupy na prerezanie stromu, *prepruning* a *postpruning*. Pri prepruningu sa už pri vytváraní dbá na to, aby strom neobsahoval vetvy s malým významom. Pri postpruningu sa vetvy s malým významom odstraňujú až po vytvorení stromu, viac na [8, 12].

3.3.2 Asociačné pravidlá

Asociačné pravidlá ako technika dolovania v dátach je použiteľná pri riešení problémov spojených s analýzou nákupného košíka, objavovaním skrytých závislostí v položkách a podobne. Cieľom je spravidla hľadanie vzájomných väzieb medzi rôznymi položkami v dátach.

Asociačné pravidlo je implikácia vo forme $Ant \Rightarrow Suc$, kde ľavá strana Ant značí predpoklad a pravá strana Suc značí záver pravidla. Základnými charakteristikami asociačných pravidiel sú:

- *Podpora* – označuje počet objektov ktoré splňujú predpoklad aj záver pravidla.
- *Spoľahlivosť* – je podmienená pravdepodobnosťou záveru, ak súčasne platí predpoklad.
- *Pokrytie* – podmienená pravdepodobnosť predpokladu, ak platí záver.
- *Kvalita* – vážený súčet spoľahlivosti a pokrytia.

Medzi ďalšie charakteristiky patrí *závislosť*, *zaujímavosť*, *kauzálna podpora*, *kauzálna spoľahlivosť* a iné, viď [4].

Asociačné pravidlá sú generované z takzvaných *frekventovaných množín*. Frekventovaná množina je množina položiek, ktoré spĺňajú podmienku minimálnej podpory. Pri generovaní sa využíva takzvaná *apriory vlastnosť*, ktorá hovorí o tom, že každá podmnožina frekventovanej množny musí byť frekventovaná. Existuje niekoľko algoritmov na generovanie frekventovaných množín. Z najznámejších sú to algoritmy *Apriory* a *FP strom* (anglicky *Frequent-Pattern tree*). Pri algoritme Apriory sa generujú kombinácie kandidátov. Je to výpočetne náročný proces, pri ktorom je počet kombinácií exponenciálne závislý na počte atribútov. Tento problém rieši algoritmus FP strom, kde sa množiny vytvárajú bez generovania kandidátov metodológiou „rozdeľ a panuj“, viac na [4, 8].

Asociačné pravidlá pracujú s booleovskými hodnotami a preto je potrebné upraviť spracovávanú databázu na správny tvar. Transakčnú databázu je možné jednoducho previesť na booleovskú databázu do tvaru, keď jednotlivé atribúty vlastne zodpovedajú položkám vyskytujúcim sa v transakčnej databáze a nadobúdajú hodnoty *0* alebo *1*. Cieľom je prispôsobiť algoritmus, ktorý funguje pre hľadanie asociačných pravidiel v booleovských databázach (teda transakčné dáta) na zložitejšie dátové typy. Na tento účel slúžia *kvantitatívne asociačné pravidlá*. Hlavná myšlienka spočíva v tom, že sa databázy s numerickými a kategorickými atribútmi transformujú tak, aby bolo možné použiť podobný postup ako pre booleovské databázy. Pritom sa najprv numerické a kategorické atribúty transformujú na booleovské. Symbolické alebo numerické atribúty A s niekoľko málo diskrétnymi hodnotami $w_1, \ldots, w_k$ možno transformovať na k booleovských atribútov $A_1, \ldots, A_k$. Numerické atribúty s veľkým rozsahom hodnôt sa najprv diskretizujú a ďalší postup transformácie je potom rovnaký ako pre kategorické atribúty, viac na [4, 8].

3.3.3 Rozhodovacie pravidlá

Rozhodovacie pravidlá sú používané na klasifikáciu objektov do tried. Ľavú stranu pravidiel tvorí predpoklad, ktorý je kombináciou vytvorenou s kategórií vstupných atribútov. Pravú stranu tvorí zaradenie príkladu do triedy. Použitím programovej konštrukcie pre vetvenie by sme mohli syntax pravidla napísať ako *IF Predpoklad THEN Trieda*.

Hľadaním rozhodovacích pravidiel sa zaoberajú algoritmy *učenia s učiteľom*. Cieľom je naučiť sa zaraďovať príklad do rôznych tried. Jednou z možností je vytvorenie pravidiel z rozhodovacieho stromu. Ďalším spôsobom je použitie algoritmu *pokrývania množín*. Jeho základným princípom je metóda „oddeľ a panuj“, Snahou je získať pravidlá, ktoré pokrývajú príklady hľadaného konceptu a tieto príklady oddeliť od iných príkladov rovnakého konceptu a od príkladov inej triedy. V priestore atribútov sa potom postupne vytvoria oblasti, ktoré budú obsahovať príklady iba z jednej triedy. Vyjadrovacia sila je rovnaká ako pri rozhodovacích stromoch, získané oblasti majú podobu mnohorozmerných hranolov rovnobežných so súradnicovými osami. Niektoré algoritmy sú schopné priraďovať pravidlám hodnotu (pravdepodobnosť, váhu). Vytvárajú takzvané *pravdepodobnostné pravidlá*.

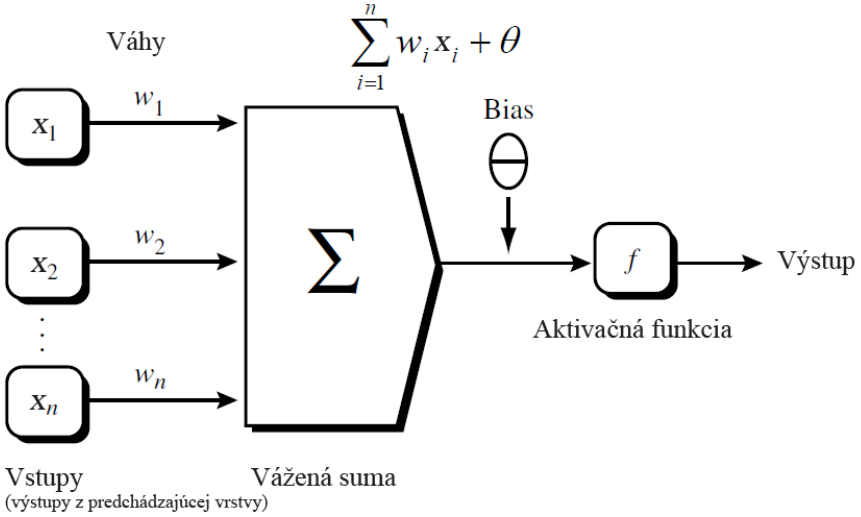
Chýbajúce hodnoty je potrebné väčšinou ošetriť vo fáze predspracovania dát. Niektoré algoritmy pre tvorbu rozhodovacích pravidiel však umožňujú pracovať aj s takými hodnotami. Numerické atribúty je potrebné rovnako, ako aj v prípade rozhodovacích stromov, diskretizovať. Podkapitola bola spracovaná podľa [4].

3.3.4 Neurónové siete

Neurónová sieť je výpočtový model, zostavený na základe abstrakcie vlastností biologických nervových systémov. Základným prvkom neurónovej siete je *model neurónu* (*perceptron*), zobrazený na obrázku 3.3. Vo všeobecnosti má niekoľko vstupov od iných neurónov alebo zo vstupných dát a jeden výstup. Vstupy neurónov predstavujú hodnoty x_1, x_2, ... x_n. Hodnoty w_1, w_2, ... w_n sa nazývajú váhy, reprezentujú synapsie u biologického neurónu. *Bias* je vnútorná konštanta daného neurónu. Výstup neurónu tvorí suma:

$$\sum_{i=1}^{n} w_i x_i + \theta \tag{3.2}$$

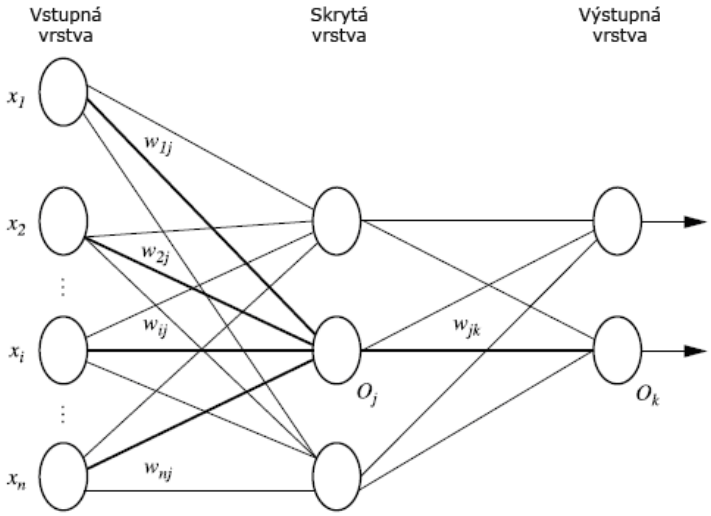
Tento výsledok je ešte transformovaný aktivačnou funkcniou, táto môže mať spojitý alebo skokový charakter. Zjednodušene povedané, neurón prijíma kladné alebo záporné podnety od iných neurónov a keď súhrn týchto podnetov prekročí daný prah, sám sa aktivuje. Výstup neurónu môže viesť na vstupy ďalších neurónov alebo môže tvoriť výstupnú hodnotu, na základe ktorej bude prevedená klasifikácia daného vzorku. Samotná neurónová sieť je zložená z viacerých vrstiev s rôznym počtom navzájom poprepájaných neurónov, viac na [8, 4].



Obrázek 3.3: Model umelého neurónu, prevzaté a upravené z [8].

Pre potreby získavania znalostí z databáz sa používa algoritmus *spätného šírenia chýb* (anglicky *Backpropagation*). Tento algoritmus využíva *viacvrstvové dopredné neurónové siete*, príklad takejto siete je na obrázku 3.4. Sieť sa skladá zo vstupnej vrstvy, vstupom sú sledované atribúty. Potom nasleduje skrytá vrstva. Skrytých vrstiev môže byť viacero, výstup z jednej sa privedie na vstup nasledujúcej. Výstupná vrstva následne dodá predpoveď

pre vstupné n-tice atribútov. Samotný algoritmus je iteratívny proces, pracuje so skupinou trénovacích dát. V každej iterácii sa zisťuje chyba medzi získanou hodnotou a aktuálnou hodnotou. Podľa chyby sa nastavia váhy jednotlivých neurónov a nasleduje ďalšia iterácia. Chyba je šírená dozadu, medzi jednotlivými vrstvami. Po určitom počte iterácií bude chyba konvergovať a učenie sa zastaví. Nie je to však zaručené.



Obrázek 3.4: Dopredná neurónová sieť, prevzaté a upravené z [8].

Neurónové siete sú v oblasti získavania znalostí jedným z najpoužívanejších nástrojov pre klasifikáciu alebo predikciu [4]. Sú vhodnou alternatívou ku rozhodovacím stromov a pravidlám v situáciách, keď nie je potrebná zrozumiteľnosť nájdených znalostí.

Neurónové siete sú vhodné pre prácu s numerickými dátami. Kategoriálne atribúty je potrebné upravovať, štandardným riešením tohto problému je *binarizácia*. Podstatou binarizácie je vytvorenie takého počtu binárnych atribútov, koľko možných hodnôt mal pôvodný atribút. Často je prevádzaná v rámci algoritmu pre učenie neurónovej siete.

Výhodou neurónových sietí je ich tolerancia na prácu s dátami so šumom a schopnosť zaradiť vzory, na ktoré nebola naučená. Nevýhodou je však nezrozumiteľnosť pre ľudí. Je ťažké pochopiť čo sa skrýva za hodnotami nastavených váh a v skrytých vrstvách. Neurónové siete je vhodné použiť tam, kde máme málo znalostí o vzťahoch medzi atribútmi a triedami, viac na [8].

3.3.5 Ďalšie prístupy

Okrem uvedených modelov je potrebné spomenúť aj ďalšie modely, u ktorých existuje pravdepodobnosť ich použitia v aplikácií pri nasadení v reálnom prostredí. S pomedzi klasifikačných metód je to *Bayessovská klasifikácia*. Táto metóda je založená na podmienenej pravdepodobnosti $P(X|Y)$. Hovorí o tom, aká je pravdepodobnosť že nastane jav X keď vieme, že nastal jav Y. Medzi jej výhody patrí jednoduchá implementácia a v mnohých prípadoch podáva dobré výsledky. Jej najväčšou nevýhodou je predpoklad, že jednotlivé atribúty sú na sebe nezávislé, čo v praxi väčšinou nie je pravdou. Riešením tohto problému je vytvorenie takzvaných *Bayessovských sietí*. Sieť tvorí orientovaný acyklický graf, uzly reprezentujú jednotlivé atribúty a obsahujú svoju tabuľku pravdepodobností. Závislosť medzi

atribútmi je reprezentovaná hranami grafu. Pri vytváraní týchto sietí je potrebná asistencia človeka so znalosťou v danej oblasti.

Ďalšou často používanou metódou, napríklad pre segmentáciu zákazníkov, je metóda *zhlukovej analýzy*. Pri tejto metóde nie sú dopredu známe triedy, tieto sú až výsledkom tvorenia zhlukov. Zhluk je množina objektov, ktoré sú si navzájom podobné. Existuje niekoľko druhov metód na tvorenie zhlukov, napríklad sú to hierarchické metódy, metódy založené na hustote, mriežke, modeloch a iné. Základnou požiadavkou na metódu tvorenie zhlukov je veľká podobnosť objektov v rámci triedy a minimálna podobnosť objektov medzi triedami. Zhlukovanie je možné použiť pri analýze odľahlých hodnôt, napríklad pri detekcií podvodov, neobvyklého chovania a podobne.

Poslednou metódou, ktorá bude spomenutá, je *regresná analýza*. Táto metóda sa používa pri úlohách zameraných na predikciu, kedy sú atribútom priraďované hodnoty obecne spojitého charakteru. Princíp metódy spočíva v predpovedaní hodnoty cieľového atribútu na základe hodnôt iných atribútov. Medzi najznámejšie varianty metódy v rámci regresnej analýzy patria *lineárna jednoduchá regresia*, *lineárna viacnásobná regresia* a *nelineárna regresia*. Pri výpočte hodnoty atribútu pri lineárnej regresií sa používa rovnica priamky $y = ax + b$, kde hodnoty a, b sú vypočítané pomocou *metódy najmenších štvorcov* [8]. Pri viacnásobnej regresií sa používa zovšeobecnenie tejto metódy pre viaceré atribúty. Nelineárna regresia sa často rieši transformáciou na lineárnu regresiu. Príkladom použitia metód regresnej analýzy môže byť napríklad predpoveď teploty vzduchu na základe ďalších údajov, alebo predpovedanie výšky splátky úveru osoby na základe jeho príjmu a podobne.

Pre ďalšie modely získavania znalostí a podrobnejšie informácie o spomenutých modeloch viď [8].

3.4 Vyhodnotenie získaných výsledkov

Vyhodnotenie získaných výsledkov je posledným a dôležitým krokom v celom procese získavania znalostí. V tejto fáze sa hlavne jedná o interpretáciu a ocenenie znalostí, ktoré sme dolovaním v dátach získali a ich prínos pre koncového užívateľa. V prípade deskriptívnych úloh nás zaujíma novosť, zaujímavosť, užitočnosť a zrozumiteľnosť. Ideálnym výsledkom sú znalosti, ktoré svojou povahou poskytnú pre zákazníka z danej aplikačnej oblasti nový pohľad na problematiku.

Vo fáze vyhodnotenia teda dochádza k hodnoteniu znalostí, ktoré sme získali za použitia zvolených modelov. Toto robíme pomocou testovania modelov. Pri hľadaní znalostí pre potreby klasifikácie sa postupuje obvykle metódou učenia s učiteľom. Vychádzame z toho, že máme k dispozícii prípady o ktorých vieme, že do danej triedy patria. Vyhodnotenie modelov potom prevádzame na základe toho, ako sa získané znalosti zhodujú s informáciami od učiteľa. Pri testovaní sa používa trénovacia množina dát a existuje mnoho variant možností, aké dáta z trénovacej množiny použijeme pre učenie a aké pre testovanie. Sú to napríklad *krížová validácia* (anglicky *cross validation*), kedy sa náhodne rozdelia dáta do množín a iteratívne sa jedna množina použije na testovanie a ostatné na trénovanie, alebo testovanie v celých trénovacích dátach, *leave-one-out* (jeden príklad pre testovanie, ostatné pre učenie) a ďalšie. Pri klasifikácií môže dôjsť k takzvanému preučeniu, kedy je model úspešný na celej množine trénovacích dát, nie je však schopný klasifikácie nových prípadov. Správnosť klasifikácie sa určuje napríklad pomocou *krivky učenia*, ktorá dáva do súvislosti počet príkladov v trénovacej množine a počet správne zaradených príkladov. Pri použití viacerých algoritmov pre získavanie znalostí nad rovnakými dátami sme schopní

porovnávať výsledky testovania jednotlivých modelov a následne vyhodnotiť, ktorý model je najlepší.

Vo fáze porozumenia dátam hrá dôležitú úlohu vizualizácia. Existuje veľa možností vizualizácie získaných znalostí od najjednoduchšieho textového zobrazenia až po vizualizáciu pomocou rôznych typov grafov, viac na [4, 8].

3.5 Oblasti použitia

Oblasť použitia techník získavania znalostí je veľmi rozsiahla. Pomocou získavania znalostí je možné študovať, pochopiť a následne zlepšiť prakticky akýkoľvek proces vo vzájomne veľmi odlišných oblastiach. Podmienkou je možnosť zhromažďovania údajov o týchto procesoch. Príkladom vhodného aplikovania získavania znalostí môže byť problém poskytnutia úveru v banke. Ako podklad slúžia údaje o transakciách klientov, platobnej disciplíne, demografické údaje a záznamy o všetkých dosiaľ poskytnutých úveroch. Manažér banky sa následne rozhoduje o poskytnutí úveru s cieľom minimalizovať riziko. Ďalším prípadom použitia môže byť medicínska oblasť, kde sa na základe rôznych kritérií lekári rozhodujú o ďalšom postupe pri liečbe pacienta. Berú pri tom do úvahy rizikové faktory a záznamy o výsledkoch tisícov vyšetrení v minulosti. Hľadávajú sa skryté vzťahy medzi jednotlivými diagnózami a podobne. Výhody tejto technológie je možné použiť napríklad aj v energetike na predikciu zaťaženia distribučnej siete alebo predikciu odberu elektriny.

Výsledkom dolovania informácii by mali byť informácie na podporu rozhodovania, pričom ich využívaním by sa mal dosiahnuť merateľný ekonomický efekt. Proces prípravy údajov, ich vyčistenia, návrhu a overenia modelu môže byť značne časovo, a teda aj ekonomicky náročný, preto je tento efekt žiaduci.

3.6 OLAM

OLAM (*on-line analytical mining*) je technológia integrácie OLAP systémov so systémami pre získavanie znalostí. Výhody použitia dátových skladov ako zdroja dát pre dolovanie sú nasledovné:

- vysoká kvalita dát v dátových skladoch,
- dostupné informácie z rôznych zdrojov, systematická infraštruktúra dátových skladov,
- možnosť použitia OLAP operácií pri spracovaní dát,
- možnosť online výberu vhodnej metódy získavania znalostí podľa potreby.

OLAM server pracuje v architektúre systémov priamo nad dátovou kockou, kde pre získavanie dát spolupracuje s OLAP serverom. Výsledky dolovania sú zobrazované štandardne pomocou grafického užívateľského rozhrania, viac na [8].

Kapitola 4

CRM systémy

Pre pochopenie významu *CRM systémov* je potrebné uviesť aké dôvody viedli k ich zavedeniu do spoločností. *Marketing* je možné definovať ako spoločenský a riadený proces, ktorým jednotlivci a skupiny získavajú to čo potrebujú, prostredníctvom tvorby, ponuky a zámeny hodnotových produktov s ostanými. Marketing prešiel počas histórie vývojovými etapami. Tieto etapy môžeme rozdeliť na historický, klasický a moderný marketing [5]. Klasický marketing je postavený na takzvanom *marketingovom mixe*, ktorý sa skladá z premenných produkt, cena, reklama a miesto. Snahou firmy je potom „namixovať“ tieto premenné tak, aby firma dosiahla svoje ciele na jednotlivých trhoch. Takto postavený marketingový model dobre fungoval v neglobalizovanom tržnom prostredí. Rozvoj komunikačných a informačných technológií však podnietil uvoľňovanie tržných bariér a došlo k obmedzeniu marketingového mixu. Firmy mohli začať oslovovať zákazníkov aj mimo svoj región a rozšíriť svoje pôsobenie na širšie oblasti.

Základnou prekážkou klasického marketingu sú premenné produkt, ktorý ak je pripravený na globálny trh, bude ťažko spĺňať lokálne požiadavky zákazníkov a rovnako nie je možné presne vymedziť miesto predaja. Optimálnym riešením sa javilo vytvorenie takzvanej skladačky, ktorú predstavoval komplexný produkt variabilne zložený z niekoľkých základných komponentov a poprípade doplnkových služieb. S takto variabilným produktom mohli lokálny marketingový manažéri reflektovať požiadavky zákazníkov. Aby vytvorené produkty mohli trvale odrážať aktuálne požiadavky, bolo potrebné vytvoriť systém, ktorý by umožňoval rýchlu spätnú väzbu a jej kvalitné analytické spracovanie. To bol dôvod ktorý podnietil vznik CRM systémov, viac na[5].

4.1 Základná charakteristika CRM

CRM (anglicky *Customer Relationship Management*) je stratégia, ktorá sa používa na získavanie informácii o potrebách zákazníkov a ich správaní. Je to interaktívny proces, ktorého cieľom je dosiahnutie optimálnej rovnováhy medzi firemnou investíciou a uspokojením zákazníckych potrieb. Optimálna rovnováha predstavuje maximálny zisk oboch strán. Cieľom je vytvorenie pevnejších vzťahov medzi spoločnosťou a jej zákazníkmi. Dobré vzťahy so zákazníkmi sú jadrom obchodného úspechu. CRM nám umožní lepšie porozumieť ich potrebám, všimnúť si tieto potreby a včas na ne reagovať. Táto stratégia však závisí za získaní veľkého počtu informácií o zákazníkoch a trendoch na trhu a to v konečnom dôsledku umožní predávať produkty a ponúkať služby efektívnejšie.

Častým problémom spoločností je zdieľanie informácií. Spoločnosť má zvyčajne o svo-

jich zákazníkoch dostatok údajov, problémom ale je, že mnohokrát nie sú zdieľané a sú k dispozícii iba pre niektoré pracovné funkcie a konkrétne osoby, alebo sa nachádzajú na viacerých miestach a je zložité sa k nim dostať. Ak chce napríklad obchodník vedieť o aktuálnom stave riešenia problémov s poskytovaním služieb pre konkrétneho zákazníka, musí kontaktovať osobu, ktorá pozná tieto informácie a čakať na odpoveď. Často tieto informácie obchodník potrebuje ihneď ako odpoveď na otázku nového potenciálneho zákazníka.

V súlade so zásadami moderného marketingu starostlivosť o zákazníkov zahrňuje nasledovné, prevzaté z [5]:

- Trvalú aktualizáciu zákazníckych potrieb, motivácií a zvykov.
- Kvantifikáciu prínosov základných funkcií CRM a to marketingových, predajných a servisných aktivít.
- Využívanie zákazníckych znalostí a skúseností pri inovácií produktov.
- Integrácií marketingu, predaja a zákazníckej podpory v jednotný celok.
- Využívanie moderných nástrojov umožňujúcich podporu zákazníckych potrieb a kvantifikáciu prínosov CRM.
- Trvalé udržovanie rovnováhy medzi marketingovými, predajnými a servisnými aktivitami s cieľom maximalizácie zisku.

CRM systém umožňuje spoločnostiam postaviť zákazníka do stredu svojho záujmu a zhromaždiť na jedno miesto všetky potrebné informácie, ktoré sa k nemu vzťahujú a umožniť oprávneným osobám v rámci organizácie prístup k týmto informáciám. V praxi to znamená, že obchodníci jednoduchým spôsobom pristupujú ku všetkým informáciám, ktoré ovplyvňujú vzťah so zákazníkmi ako sú rozhovory, emailové správy, sťažnosti, vyjadrenia a ďalšie informácie, ktoré boli získané alebo zaslané zákazníkom v priebehu doterajšej spolupráce.

Z hľadiska funkcionality by sme mohli funkčnosť CRM systému rozdeliť do troch oblastí, viď obrázok 4.1. Prvú oblasť môžeme nazvať *kolaboratívne CRM*. Táto oblasť sa stará o integráciu všetkých komunikačných kanálov medzi spoločnosťou a zákazníkmi. Zahrňuje napríklad manažment kontaktov, aktivity spoločnosti prostredníctvom internetu a zákaznícke centrum. Druhou oblasťou je *operatívne CRM*, ktorá sa zaoberá návrhom, plánovaním a prevádzaním každodenných aktivít, ktoré súvisia s marketingom, predajom a službami. Poslednou oblasťou je oblasť *analytického CRM*. Táto oblasť sa zaoberá aplikáciou podpory rozhodovania za účelom optimalizácie vzťahov so zákazníkmi a zvýšenia zisku. Je podrobnejšie popísaná v kapitole 5 o aplikácií podpory rozhodovania v CRM, viac na [26].

V súvislosti CRM systémami je vhodné spomenúť pojem *manažment znalostí* (anglicky *Knowledge Management*). Manažment znalostí je proces, ktorým organizácie vytvárajú hodnotu prostredníctvom intelektuálneho a znalostného kapitálu. Jedná sa predovšetkým o výmenu znalostí medzi zamestnancami, oddeleniami a firmami. S tým súvisí aj problém zavedenia CRM systémov do spoločnosti, kde je potrebné, aby si zamestnanci uvedomili, že zdieľaním informácií sa dosiahne prínos pre všetkých. Úspešnosť koncepcie CRM je priamo závislá na funkčnej koncepcií manažmentu znalostí, viac na [5].

<table>
<tr><td>Kolaboratívne CRM</td><td colspan="2">Správa kontaktov</td><td colspan="2">eCRM/Internet</td><td colspan="2">Interakcia so zákazníkom</td></tr>
<tr><td rowspan="3">Operačné CRM</td><td rowspan="3">Automatizácia marketingu</td><td>Vývoj kampane</td><td rowspan="3">Automatizácia predaja</td><td>Správa objednávok</td><td rowspan="3">Automatizácia služieb</td><td>Helpdesk</td></tr>
<tr><td>Rozbehnutie kampane</td><td>Podpora predaja</td><td>Správa sťažností</td></tr>
<tr><td>Kontrola kampane</td><td>Konfigurácia produktov</td><td>Správa servisných požiadaviek</td></tr>
<tr><td>Analytické CRM</td><td colspan="2">Marketingové analýzy</td><td colspan="2">Analýzy predaja</td><td colspan="2">Analýzy služieb</td></tr>
</table>

Obrázek 4.1: Klasifikácia funkcionality CRM systémov, upravené podľa [26].

4.2 Štruktúra CRM systémov

CRM systémy sa väčšinou skladajú z jednotlivých modulov, ktoré sú navzájom poprepájané a naviazané na záznamy o obchodných partneroch. Medzi obecné moduly je možné zaradiť, viac na [5]:

- evidencia obchodných partnerov a kontaktov,
- obchodné prípady a príležitosti,
- modul pre marketing,
- správa projektov,
- moduly pre komunikáciu,
- plánovanie,
- analýza a vyhodnotenie.

Počet a druh modulov v rôznych systémoch je odlišný, často závisí od požiadaviek zákazníka, pre ktorého je CRM systém určený. Medzi údaje, ktoré by mali byť spracovávané pomocou CRM systémov napríklad patria:

- reakcie na kampane a reklamu,
- záznamy práce na projektoch,
- záznamy o predajoch a nákupoch,
- informácie o účte,
- webové registračné údaje,
- záznamy o servise a podpore,
- demografické údaje,
- webové údaje o predaji.

Rovnako ako pri moduloch, aj pri type spracovávaných údajoch závisí od konkrétnej implementácie CRM systému a od požiadaviek zákazníka.

4.3 Aktuálny trend vývoja CRM systémov

V minulých rokoch bolo nasadenie týchto systémov výsadou veľkých spoločností, ktoré mali dostatok financií na ich nákladnú implementáciu a prevádzkovanie. V súčasnej dobe existujú riešenia dostupné stredným aj malým firmám. Vďaka vysokému počtu dodávateľov je možné si vyberať zo širokej ponuky obecných aj špecificky zameraných CRM systémov podľa jednotlivých odvetví.

V posledných rokoch sa značne rozšírili takzvané open-source systémy pod rôznymi platformami, ktoré je možné stiahnuť zadarmo a bez obmedzení používať. Ďalšou možnosťou je použitie CRM systémov, ktoré nie je potrebné inštalovať priamo v spoločnosti. Stačí si zaplatiť účet u externého poskytovateľa služieb, nastaviť požadované moduly a systém je pripravený pre použitie. Nevýhodou tohto spôsobu riešenia je fakt, že svoje údaje poskytujeme tretej strane.

Primárnym cieľom CRM systémov je získanie čo najväčšieho množstva informácií. Vďaka koncepcie CRM sú centralizované a pri ich správnom zhodnotení pomocou aplikácie metód analýzy dát a metód získavania znalostí môžu spoločnosti priniesť informačné výhody, ktoré jej umožnia uspieť pri jej ďalšom pôsobení na trhu. Preto by moderný CRM systém mal obsahovať aspoň základné nástroje pre podporu rozhodovania, ktoré by zrozumiteľným a jednoduchým spôsobom manažérom poskytovali cenné informácie a umožnil tak získať výhody nad konkurenciou. Podporou rozhodovania a jej aplikácie v CRM systémoch sa zaoberá nasledujúca kapitola 5.

Kapitola 5

Aplikácia podpory rozhodovania v CRM systémoch

Úlohou podpory rozhodovania v CRM systémoch je nájsť odpovede na otázky ako zvýšiť príjmy, znížiť náklady, zvýšiť lojalitu zákazníkov spoločnosti a navrátiť tak investície venované do zavedenia CRM systému. V súčasnej dobe je zákaznícka základňa pre marketing a predaj značne široká a komplikovaná. Mnohé CRM systémy obsahujú nástroje, ktoré sú schopné odhaliť a upozorniť na rôzne prahové hodnoty na základe výpočtov. CRM systémy obsahujúce nástroje na podporu rozhodovania môžeme nazvať ako *analytické CRM systémy* [21].

Aplikácia podpory rozhodovania v CRM systémoch by mala manažérom poskytovať podporu vo všetkých troch fázach životného cyklu zákazníka:

- Získanie zákazníkov – pri tejto fáze je potrebné identifikovať možnosti trhu a premeniť ich na budúcich zákazníkov. Zahrňuje zameranie marketingového úsilia na správnu skupinu zákazníkov, identifikáciu tejto skupiny.
- Zvýšenie hodnoty stávajúcich zákazníkov – *cross selling* (*krížový predaj*). Krížový predaj je druh predajnej stratégie. Jej cieľom je ponuka ďalších produktov alebo služieb existujúcim zákazníkom spoločnosti. Zámerom je snaha uspokojiť čo najviac zákazníckych potrieb širokou ponukou ďalších produktov, ktoré môžu, ale nemusia súvisieť s pôvodným nákupom zákazníka. Pri tejto fáze sa jedná hlavne o identifikáciu potrieb zákazníkov a ich segmentáciu.
- Zachovanie dobrých zákazníkov – modely pre identifikáciu ziskových a neziskových zákazníkov. Zameranie nielen na udržanie dobrých zákazníkov, ale aj na tých, ktorí by mohli byť zisková v budúcnosti. Spôsoby ako sledovať vzťahy so zákazníkom je napríklad monitoring sťažností, servisných zásahov, zmeny oproti štandardnému chovaniu a podobne, viac na [6].

Podporu rozhodovania v CRM môžeme rozdeliť do dvoch oblastí a to OLAP analýza a techniky získavania znalostí v dátach [21].

5.1 Použitie techník OLAP

Pre zvládnutie náročnejších úloh spoznania zákazníkov a ich potrieb sú vytvárané systémy analytického CRM a na CRM zamerané prvky nástrojov business intelligence budované

nad dátovými skladmi. Tieto je možné chápať ako určité podstatné rozšírenie operatívneho CRM, sú relevantné pre riadenie a rozhodovanie na vyšších úrovniach. OLAP poskytuje retrospektívne, dynamické poskytovanie informácií na viacerých úrovniach. Umožňuje analyzovať trendy a vzory správania zákazníkov na základe historických dát. Pomáha pri zhrnutí a porovnávaní dát a tak podporuje proces rozhodovania. Hlavnou devízou OLAP je poskytovanie multidimenzionálneho pohľadu na dáta na rôznych úrovniach. Ako príklad použitia OLAP v analýze CRM je zobrazovanie predaja produktov, kedy sa na predaj nazerá z dimenzie času, dimenzie produktov, dimenzie miesta predaja a dimenzie zákazníkov, viac na [21, 11].

5.2 Použitie metód získavania znalostí

Získavanie znalostí na rozdiel od OLAP analýz, ktoré poskytujú informácie na základe minulosti, umožňujú analyzovať minulosť a poskytnúť predpovede do budúcnosti. Techniky získavania znalostí sa snažia odhaliť užitočné vzory a údaje, ktoré sú z dôvodu veľkého objemu a komplexnosti dát CRM systému skryté [21].

Základné kroky procesu získavania znalostí pre efektívne CRM systémy reflektujú obecný postup pri získavaní znalostí a sú nasledovné:

- Definícia obchodného problému - špecifikujeme cieľ, problém na ktorý sa chceme zamerať.
- Vytvorenie databázy pre dolovanie - príprava dát pre dolovanie, ktoré sa viažu k danému cieľu, úlohe.
- Preskúmanie dát - cieľom fázy je analýza dát.
- Príprava dát pre dolovanie.
- Vytvorenie modelu - výber modelu, trénovanie a testovanie.
- Zhodnotenie modelu - zhodnotenie dosiahnutých výsledkov.
- Nasadenie modelu do aplikácie a testovanie výsledkov pri reálnom použití, viac na [6].

Úlohy, ktoré spadajú do oblasti získavania znalostí, sa podľa zamerania dajú rozdeliť na úlohy zamerané na predikciu a úlohy zamerané na segmentáciu. Účelom prediktívnych modelov je odhad hodnoty veličiny na základe dostupných údajov. V prípade CRM systémov sa často jedná o predikciu chovania klientov. Pomocou prediktívnych modelov sa riešia napríklad nasledujúce úlohy:

- identifikácia zákazníkov, u ktorých hrozí riziko, že prerušia obchodné vzťahy a prejdú ku konkurencií,
- výber správnej cieľovej skupiny pre marketingovú kampaň propagujúca určitý produkt alebo službu,
- určenie, ktorý produkt má byť ponúkaný určitej skupine klientov,
- vyhodnotenie úspešnosti marketingovej kampane, viac na [11].

Pri vytváraní predikčných modelov sa používajú metódy regresnej analýzy, rozhodovacie stromy, neurónové siete a iné.

Druhým typom úloh v oblasti získavania znalostí sú segmentačné úlohy. Segmentácia patrí medzi základné nástroje marketingového riadenia. Pre efektívne vykonávanie činností, ktoré je možné zaradiť do riadenia vzťahov so zákazníkmi, v spoločnostiach s veľkým počtom klientov prirodzene vzniká potreba dostupnosti vhodnej segmentácie klientov. Segmentačné modely podporujú lepšie poznanie chovania a potrieb klientov a umožňujú lepšie zamerať marketingovú komunikáciu. K tvorbe segmentačných modelov je možné použiť metódy zhlukovej analýzy. Môžeme však naraziť na problém s určením potrebných vstupných parametrov pre tvorenie zhlukov. Výsledný model tak môže byť nepresný a ťažko interpretovateľný. Riešením je použitie metódy rozhodovacích stromov, kde sa sledovaná vlastnosť nastaví ako cieľová premenná, podrobnejšie na [11].

Použitia techník získavania znalostí v CRM systémoch má niektoré špecifiká, ktoré je potrebné pri tomto procese zohľadniť:

- Pre získanie netriviálnych výsledkov je takmer vždy potrebné kombinovať rôzne techniky získavania znalostí. Dáta je potrebné preskúmať z rôznych uhlov a pohľadov na ich rôzne aspekty.
- Zaujímavé výsledky dolovania dát sú podmienené integráciou dát z rôznych zdrojov, v prípade CRM sú to kombinácie údajov z rôznych modulov.
- Dáta pre dolovanie sú často heterogénne, demografické údaje, časové dáta, textové a prípadne aj multimediálne dáta.
- Pri CRM systémoch sa často pracuje s osobnými údajmi osôb, preto je potrebné pri dolovaní dbať na ich ochranu pred zneužitím.
- Získané znalosti je nutné overiť reálnym nasadením v aplikácií. Vzory sú často považované za hypotézy a musia byť podrobené štatistickými testami pre potvrdenie výsledkov, prevzaté z [24].

Mnohé dnešné CRM systémy sú implementované ako webové aplikácie. Pri tomto spôsobe nasadenia je jednoduché integrovať CRM so systémami *B2C* (*Bussiness to Customer*). Systémy B2C predstavujú rôzne formy online obchodovania prostredníctvom internetu. Tieto systémy môžu zdieľať databázu zákazníkov s CRM a otvoriť tak nové možnosti ich spoznávania. Príkladom môže byť využitie logovacích záznamov B2C aplikácie, keď sa sleduje a ukladá pohyb prihláseného klienta na jednotlivých častiach webu. Z týchto záznamov je potom možné zistiť, o ktoré produkty a služby má klient záujem a prispôsobiť podľa toho individuálnu ponuku, ktorá sa následne zašle napríklad prostredníctvom reklamnej pošty.

Kapitola 6

CRM systém CRMminer

Aplikácia dostala pracovný názov CRMminer. Je primárne určená pre spoločnosti s menším a stredným počtom zamestnancov, pri jej návrhu sa dbalo na čo najväčšiu univerzálnosť vzhľadom na pracovné odvetvia, v ktorom bude nasadená. Pri nasadení produktu sa počíta s jeho napojením na systémy elektronického obchodu a prípadne systémy pre účtovníctvo, kde budú zhodnotené jeho možnosti pre podporu rozhodovania.

Základnou technickou požiadavkou na výslednú aplikáciu je nezávislosť od operačného systému a nezávislosť na použitej databázovej platforme. Dôvodom tejto požiadavky je skutočnosť, že mnoho klientov si praje z rôznych dôvodov použiť iný typ databázy a aplikácia by mala umožniť túto zmenu bez výrazných zásahov do jej štruktúry.

Aplikácia je typu tenký klient. To znamená, že bude umiestená na servery a klient sa bude pripájať prostredníctvom klienta, ktorý v tomto prípade tvorí webový prehliadač. Výhodou tohto prístupu je centralizácia administrácie a tým pádom jej zjednodušenie, zjednotenie správy a výrazná redukcia nákladov, keďže nemusí byť aplikácia inštalovaná na každej pracovnej stanici. Nevýhodou je naopak potreba sieťového pripojenia pre spojenie zo serverom. To môže znamenať určité obmedzenie pri práci mimo kanceláriu. V dnešnej dobe bezdrôtových pripojení na internet však nie je tento problém taký výrazný. Aplikácia môže byť používaná v prostredí internetu a intranetu v menších podnikových sieťach.

6.1 Použité technológie

V nasledujúcich podkapitolách sú popísané technológie, ktoré boli použité pri vývoji samotnej aplikácie CRM miner a zároveň aj modulu pre podporu rozhodovania. Pri každej technológií je uvedený jej základný popis, ktorý je nutný pre pochopenie prečo a na základe akých vlastností bola použitá v aplikácií.

Výsledná aplikácia je vytváraná ako produkt, ktorý bude ponúkaný klientom pre správu zákazníkov v spoločnosti s možnosťou napojenia na ich interné systémy. Požiadavky klienta môžu byť veľmi rozmanité nielen čo sa týka samotnej funkčnosti aplikácie ale aj prostredia, v ktorom bude výsledky produkt nasadený. Pri vývoji aplikácie bol kladený veľký dôraz na použitie open-source riešení. Dôvodom je jednoduchosť použitia, spravidla sa jedná o overené riešenia a zvyčajne existuje komunita užívateľov, ktorá sa stará o ďalší vývoj, podporu, dokumentácie a podobne.

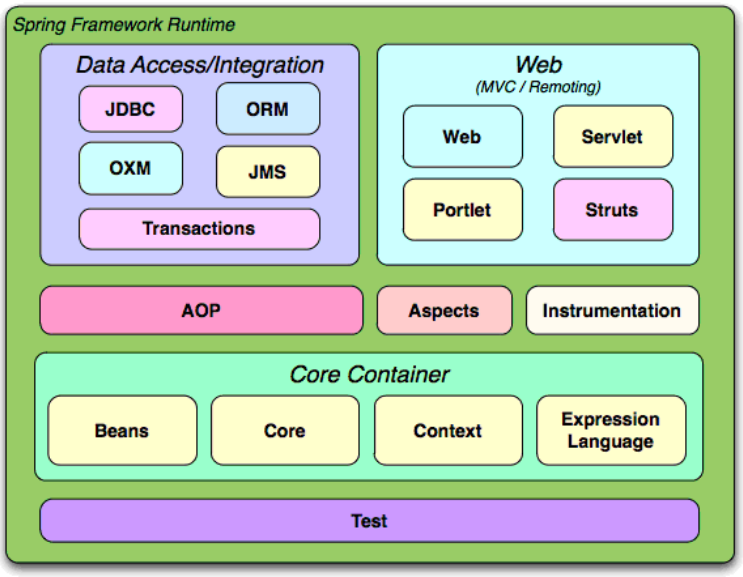
Na základe uvedených požiadaviek na aplikáciu, bola ako implementačné prostredie zvolená technológia *Java EE* (*Java Enterprise Edition*). Najväčšou výhodou Javy je jej nezávislosť na platforme, široká komunita vývojárov a veľké množstvo voľne dostupných

knižníc ktoré významne uľahčujú a zrýchľujú vývoj aplikácií. Technológia je založená na takzvaných *Java Servletoch*. Servlety sú spracovávané pomocou kontajnera servletov, ktorý spolupracuje s webovým serverom. Výkonnosť servletov je zaručená tým, že vytvárajú iba jediný proces a užívateľské dotazy sú spracované pomocou menej náročného vlákna, ktoré spravuje virtuálny stroj Javy (*Java Virtual Machine*), viac na[7].

Pri implementácii bol zvolený návrhový vzor *MVC* (anglicky *Model View Controller*). Tento vzor umožňuje rozdelenie aplikácie do vrstiev a to na vrstvu spracovania požiadaviek, vrstvu s aplikačnou logikou a nakoniec vrstvu pre samotné zobrazovanie. Pred zahájením implementácie sa bolo potrebné rozhodnúť pre použitie vhodného *rámca* (anglicky *framework*), ktorý by výrazne uľahčil a zrýchlil vývoj. Pre Javu EE ich existuje veľké množstvo, každý z nich je určený na iný typ aplikácií a každý má svoje výhody a slabé stránky. Medzi najznámejšie patria rámce *JSF*, *Struts*^{TM}, *Spring* a iné. S pomedzi dostupných rámcov som sa rozhodol použiť rámec Spring. Jeho presnejší popis a dôvod prečo je použitý v aplikácií sú uvedené v nasledujúcej sekcií.

6.1.1 Spring

Spring je *aplikačný rámec*, ktorý poskytuje komplexnú podporu pre vývoj aplikácií v jazyku Java. Veľká väčšina existujúcich aplikačných rámcov sa sústreďuje na podporu len určitej vrstvy v architektúre aplikácie. Príkladom môže byť obľúbený MVC rámec *Struts*, ktorý uľahčuje vývoj webovej prezentačnej vrstvy aplikácií, alebo populárny objektovo-relačne mapovací nástroj *Hibernate*^{TM}, orientujúci sa na perzistenčnú vrstvu. Rámec Spring naproti tomu konzistentným spôsobom podporuje všetky vrstvy aplikácií, od prezentačnej vrstvy, cez aplikačnú vrstvu až k dátovej a perzistenčnej vrstve. Výnimkou však nie je ani napríklad vrstva webových služieb. Moduly rámca Spring sú zobrazené na obrázku 6.1, viac na [16].



Obrázek 6.1: Architektúra rámca Spring, prevzaté z [22].

Spring je primárne označovaný ako takzvaný *IoC kontajner*. Aplikuje koncept inverzie kontroly (anglicky *Inversion of Control*). Návrhový vzor inverzia kontroly predstavuje

možnosť presunutia zodpovednosti za vytvorenie a previazanie spolupracujúcich objektov z aplikácie do rámca.

V Springu sa objekty, ktoré tvoria jadro našej aplikácie a ktoré su spravované s IoC kontajnerom, nazývaju *Beany*. Bean je objekt, ktorého inštanciu vytvára, upravuje a spracuje Spring IoC kontajner. Inak je bean jednoducho iba jeden z mnohých objektov v aplikácií. Beany a závislosti medzi nimi sú definované v konfiguračných súboroch, ktoré používa kontajner.

Spring je navrhnutý tak, aby „nezavadzal“, čo znamená, že logika kódu aplikácie všeobecne nie je závislá na rámci. V integračných vrstvách (napríklad pre prístup k databáze) existujú niektoré závislosti na technológii pre prístup k dátam a nutnosti použiť potrebné knižnice. Tieto závislosti je však ľahké izolovať od zvyšku aplikácie.

V minulosti sa Springu často vyčítala jeho zložitá konfigurácia pomocou XML súborov. Od verzie *2.5* je tento problém odstránený pridaním podpory pre *anotácie*, viac na [18]. Zavedením anotácií sa výrazne zredukovalo množstvo konfigurácie, ktorá sa preniesla priamo do tried aplikácie. Nevýhodou anotácií je, že konfigurácia je decentralizovaná, čo v niektorých prípadoch môže spôsobiť horšiu orientáciu v aplikácií. Ideálnym riešením je vhodná kombinácia obidvoch spôsobov.

Veľkou výhodou rámca Spring je jeho modularita, kompatibilita s inými typmi rámcov a celkovo jeho jednoduché prepojenie na iné technológie. Z množstva jeho modulov je možné používať len tie, ktoré skutočne potrebujeme a naopak zakomponovať moduly tretích strán bez nutnosti meniť samotnú štruktúru rámca [22].

Aplikácia CRMminer je navrhnutá ako webová aplikácia typu klient - server. Pre webovú vrstvu aplikácie je použitý modul *Spring MVC*, ktorý je priamo súčasťou rámca Spring. Dôvodom pre jeho použitie je jeho jednoduchosť a vstavaná podpora pre asynchrónne volanie metód na servery (*AJAX*, viac na [22]). Pre tvorbu šablón webových stránok som sa rozhodol použiť technológiu *Java Server Pages*, viac na [7]. Spring poskytuje sadu knižníc značiek (anglicky *tagov*), ktoré zjednodušujú mnohé časté operácie, akými sú napríklad vytváranie formulárov a mapovanie hodnôt, vetvenie programu a iné.

Bezpečnosť aplikácie je zaistená pomocou kontrolného rámca *Spring Security*. Je to silný a vysoko prispôsobiteľný rámec na overovanie a kontrolu vstupu. Je považovaný za štandard pre zabezpečenie aplikácií založených na rámci Spring. Zameriava sa na dve hlavné oblasti zabezpečenia a to na autentizáciu a autorizáciu (alebo povolenie prístupu). Na úrovni autentizácie podporuje širokú škálu overovacích modelov, umožňuje napríklad overovanie pomocou LDAP, čo má význam pre nasadenie aplikácií vo väčších spoločnostiach.

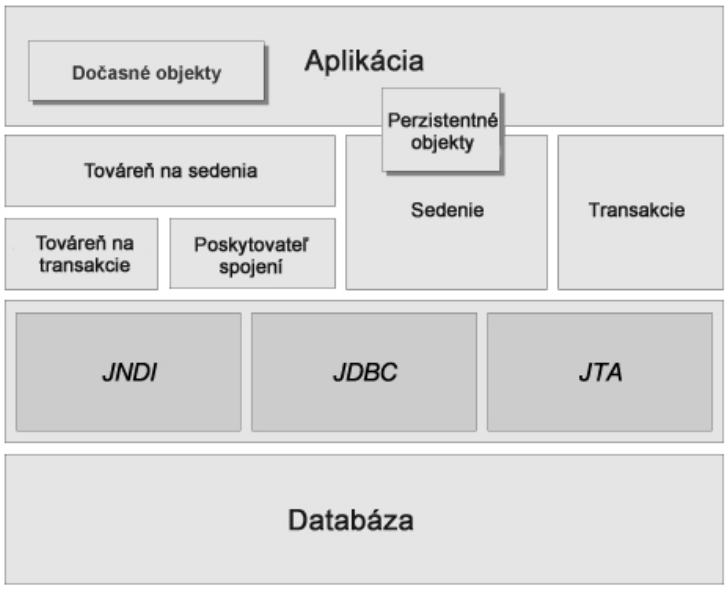
Inštalácia kontrolného rámca do aplikácie je jednoduchá, rámec je poskytovaný ako sada knižníc ktoré je potrebné integrovať do aplikácie. Konfigurácia je centralizovaná do jedného konfiguračného súboru, kde sú uvedené prístupy pre jednotlivé role užívateľov. Rámec definuje základnú štruktúru databázy užívateľov, ktorá zohľadňuje nastavenie prístupových údajov a oprávnení pre jednotlivé funkčné bloky aplikácie, viď [23].

Spring je vhodný na vytváranie veľkých, robustných aplikácií kde je potrebné zaistiť prehľadnú stavbu, náväznosť jednotlivých modulov a jednoduchú udržovateľnosť aplikácie. Z toho dôvodu som sa ho rozhodol použiť pre aplikáciu CRMminer, kde bude potrebné kombinovať množstvo rôznych technológií a zaistiť ich bezproblémovú komunikáciu medzi sebou.

6.1.2 Hibernate

Hibernate je rámec pre objektovo-relačné mapovanie určený pre aplikácie postavené na Jave. Je to súhrn nástrojov, ktoré umožňujú vývojárom používať *POJO* (*Plain Old Java Object*, viď [19]) formát doménových objektov v ich aplikáciách. Rozširuje možnosti štandardného objektovo-relačného mapovania o ďalšiu užitočnú funkcionalitu. Termín objektovo-relačné mapovanie objektov sa vzťahuje na techniku mapovania dát od objektovej reprezentácie modelu do relačnej reprezentácie modelu dát a naopak [10].

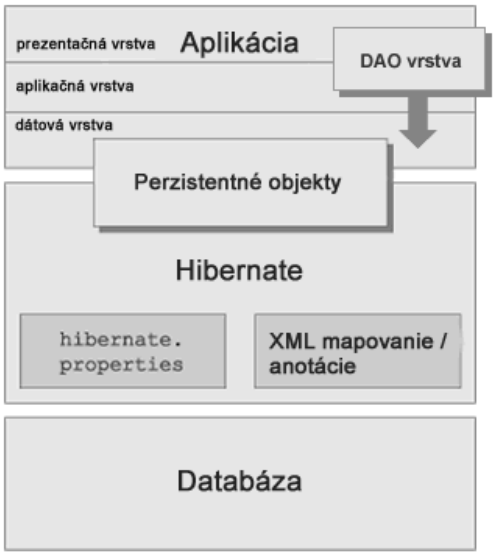
Rámec Hibernate je nezávislý od použitej databázy a aktuálne podporuje všetky najpoužívanejšie relačné databázové systémy. Z obchodného hľadiska je nezávislosť aplikácie od databázového systému potrebná, pretože v praxi sa často stáva, že klient je zmluvne viazaný na konkrétny typ systému a pri vývoji aplikácie na to musí zohľadniť. V prípade Hibernate stačí integrovať ovládač na používanú databázu v podobe Java knižnice (*Java Database Connector JDBC*) a nastaviť typ používanej databázy v konfigurácií. Architektúra Hibernate je zobrazená na obrázku 6.2, pre podrobnejší popis obrázku viď [10].



Obrázek 6.2: Architektúra Hibernate, vytvorené podľa [10].

Integrácia Hibernate do aplikácie zahrňuje nastavenie parametrov spojenia s relačnou databázou a samotné mapovanie doménových objektov na databázové tabuľky. Pôvodne boli jednotlivé doménové objekty mapované na databázové tabuľky pomocou konfiguračných súborov, ktoré používali jazyk XML na popis typu atribútov a vzťah jednotlivých entít medzi sebou. Tento starší spôsob sa mohol zdať prehľadný, vytváranie týchto súborov však predstavovalo množstvo zbytočného kódu. Novšie verzie Hibernate podporujú používanie anotácií, ktoré umožňujú preniesť konfiguráciu priamo do javovských tried, takzvaných entít. Entity majú formát POJO objektov. Pre každý atribút je možné nadefinovať viazanosť na jeho ekvivalent v databázovej tabuľke a určiť mapovanie na ďalšie entity, ak to vyplýva z definície schémy. Hibernate ďalej umožňuje pomocou anotácií kontrolovať aj hodnoty položiek, je to napríklad dĺžka textových reťazcov, správny formát dátumov alebo kontrola správneho formátu telefónneho čísla.

Štandardná aplikácia používajúca Hibernate je založená na trojvrstvovej architektúre. Jedná sa o prezentačnú, aplikačnú a dátovú vrstvu, zobrazené na obrázku 6.3. Dátová vrstva je tvorená množinou tried, ktoré sa starajú o manipuláciu s perzistentnými objektmi nad databázou. Táto vrstva je reprezentovaná vrstvou pristupujúcou k dátam, tzv. *DAO vrstvou* (anglicky *Data Access Objects*). Tvorí rozhranie medzi aplikáciou a rámcom Hibernate, s ktorým komunikuje ako jediná v rámci aplikácie, viac na [10].



Obrázek 6.3: Architektúra aplikácie s Hibernate, vytvorené podľa [16].

Hlavné výhody rámca Hibernate, ako sú podpora anotácií, nezávislosť na typu databázy, podpora transakcií, jednoduchosť použitia a skutočnosť, že je priamo podporovaný a vstavaný do rámca Spring, boli dôvodom, prečo je vhodný pre použitie a bol použitý v aplikácií CRMminer.

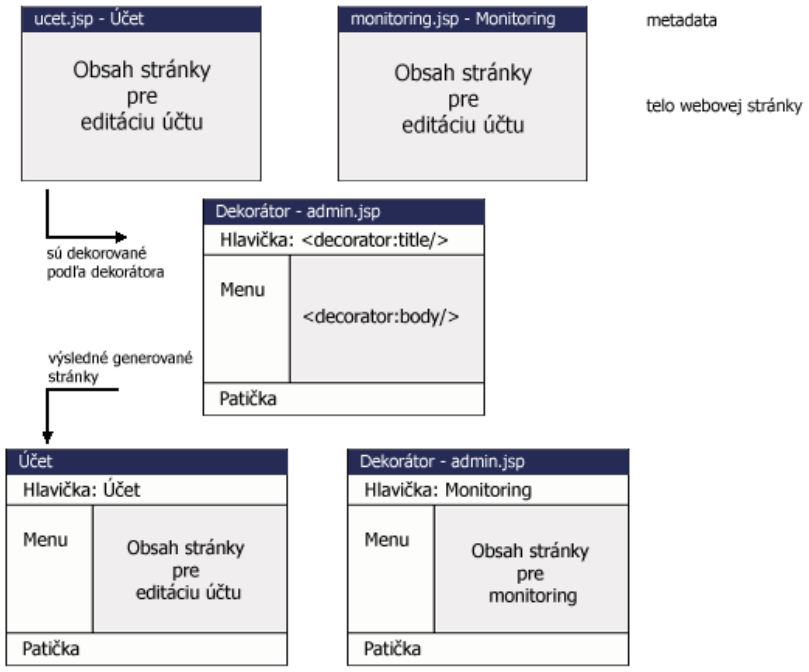
6.1.3 Sitemesh

SiteMesh je rámec určený pre dekoráciu webových stránok a vytváranie šablón. Je určený pre systémy, v ktorých je potrebné vytvárať a následne spravovať veľký počet stránok, ktoré musia mať konzistentý vzhľad, štruktúru a navigáciu.

SiteMesh zachytí všetky požiadavky na statické alebo dynamicky generované HTML stránky, smerované na webový server, analyzuje stránky, získava vlastnosti a údaje z obsahu a generuje zodpovedajúcu výslednú stránku podľa danej šablóny. Tú nahradí za tú pôvodnú, proces je zobrazený na obrázku 6.4.

SiteMesh je vytvorený pomocou Java EE, JSP a XML technológie. Vďaka tomu je ideálny pre použitie s aplikáciami nad platormou Java. Môže však byť integrovaný s inými architektúrami, ako je *CGI*, *PHP* a iné.

Integrácia Sitemesh do aplikácie pozostáva s vytvorenia takzvaných *dekorátorov*, iným slovom šablón webových stránok. Tieto šablóny tvoria dizajn aplikácie, predstavujú kostru stránok, odkazujú na použité kaskádové štýly, javascriptové súbory a ostatné potrebné súbory pre aplikáciu. Následne je potrebné nastaviť, ktoré dekorátory budú použité na



Obrázek 6.4: Princíp fungovania rámca Sitemesh, vytvorené podľa [17].

konkrétne stránky, alebo celé moduly aplikácie. Na to slúži špeciálny konfiguračný súbor v jazyku XML. V uvedenom príklade 6.2 je zobrazené použitie dvoch dekorátorov a to šablóny pre administráciu, ktorá bude použitá pre URL adresy, ktoré začínajú na `admin` a druhú šablónu `web.jsp`, ktorá slúži pre verejnú časť aplikácie. Pri použití Sitemesh v spolupráci s aplikačným rámcom Spring, alebo iným rámcom založeným na MVC architektúre, programátor narazí na problém s rozpoznávaním URL adries, ktoré priamo neodkazujú na konkrétnu šablónu, napriklad `/admin/ucet/editacia`. Tieto adresy sú nasmerované do koreňového adresára webovej aplikácie čo je samozrejme chybné. Riešením je vytvoriť vlastnú verziu triedy, ktorá sa stará o spracovanie adries a bude tieto adresy vyhodnocovať správne, viac na [15].

Príklad 6.1 obsahuje ukážku jednoduchého dekorátora webovej stránky. Zdrojový kód v príklade 6.2 ukazuje, akým spôsobom sú priradené dekorátory pre určité sekcie webu. V uvedenom prípade je to webová verejná a administračná časť, pre podrobnejší popis viď [17].

Hlavná šablóna pre administráciu systému CRM miner bude rozdelená na niekoľko logických celkov a na kartách jednotlivých modulov sa bude meniť len obsahová časť. Ostatné funkčné celky budú nemenné. Preto je použitie rámca Sitemesh v tejto aplikácií veľmi užitočné. Pri akejkoľvek zmene vzorovej šablóny sa zmena ihneď prejaví na všetkých podstránkach aplikácie. V praxi klient zvyčajne požaduje úpravu dizajnu administrácie podľa svojich požiadaviek a pomocou Sitemesh je ich možné jednoducho splniť. Vývoj aplikácie je taktiež výrazne urýchlený, pretože pri vytváraní šablón nie je potrebné znovu zbytočne kopírovať funkčné bloky, stačí vytvoriť len niektorú časť a Sitemesh ostatné časti vhodne doplní do výslednej podoby webovej stránky.

Príklad 6.1: Ukážka jednoduchého dekorátora webovej stránky.

```
<%@ taglib uri="http://www.opensymphony.com" prefix="decorator" %>
<html>
   <head>
       <!--  css, JavaScript, ...!>
       <decorator :head />
     <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
 </head>
 <body>
    <div id="header">Obsah hlavičky webovej stránky</div>
    <div id="menu">Hlavné menu</div>

      <!-- doplnovana cast z ostatnych webovych stranok!>
      <decorator :body />

    <div id="footer">Patička webovej stránky</div>
 </body>
</html>
```

Príklad 6.2: Priradenie dekorátorov pre určité sekcie webu.

```
<decorators defaultdir="WEB-INF/decorator-config">
    <decorator name="admin" page="admin.jsp">
      <pattern>/admin/*</pattern>
  </decorator>
  <decorator name="public" page="web.jsp">
      <pattern>/public/*</pattern>
  </decorator>
</decorators>
```

6.1.4 Maven

Maven je nástroj určený pre zostavenie a správu projektov, primárne určený pre projekty založené na jazyku Java. Slúži na podobný účel ako nástroj *Apache Ant*, ale je založený na iných konceptoch a pracuje odlišným spôsobom. Hlavným cieľom Maven je umožniť vývojárom pochopiť kompletný stav vývoja projektu a získať prehľad v čo najkratšom čase. Cieľom Maven je pokrytie nasledujúcich piatich oblastí:

- zjednodušenie procesu zostavenia aplikácie,
- zavedenie jednotného systému zostavovania aplikácie,
- poskytovanie kvalitných informácií o projekte,
- poskytovanie usmernení pre osvedčené postupy rozvoja,
- poskytnutie transparentného pridávania nových funkcií.

Tabulka 6.1: Štandardná štruktúra aplikácie pod Maven, spracované podľa [3].

Adresár	Popis
Koreňový adresár	pom.xml a iné potrebné súbory
src/main/java	adresár pre kompilovateľné Java súbory
src/main/webapp	adresár pre súbory potrebné k webovej časti aplikácie, obsahuje JSP šablóny, CSS, obrázky atď.
src/main/config	adresár pre konfiguračné súbory
src/main/resources	adresár pre zdroje, preklady, nastavenia
src/test/java	adresár pre testovacie Java triedy
src/test/resources	adresár pre konfiguračné súbory pre testy

Základným princípom fungovania Mavenu je popísanie projektu pomocou takzvaného *Project Object Model*, ďalej POM. Tento model popisuje softvérový projekt nie len z pohľadu jeho zdrojového kódu, ale poukazuje aj na závislosti na externých knižniciach, popisuje proces zostavovania aplikácie a rôznych funkcií s tým spojených. Umožňuje napríklad spúšťanie testov, zbieranie informácií o zdrojových kódoch a iné, viac na [3].

Maven je postavený na modulárnej architektúre a funguje na princípe volania jednotlivých *zásuvných modulov* (anglicky *pluginov*). Pluginy rozširujú možnosti Maven o užitočné funkcie, alebo tvoria spojenie s inými typmi technológií, ktoré chceme použiť pri správe projektov. Maven sa stará iba o dodanie a spustenie pluginov. Nemá žiadne vlastné grafické užívateľské rozhranie a predvolene funguje ako konzolová aplikácia ovládaná pomocou príkazového riadku. Pluginy tak môžu využívať všetky nástroje, ktoré dokážu komunikovať pomocou štandardných vstupov.

Maven definuje obecnú štruktúru aplikácie, ktorá je zobrazená v tabuľke 6.1. Táto štruktúra je doporučovaná, je možné ju však zmeniť podľa vlastných potrieb. Zmeny štruktúry aplikácie, definovanie knižníc a registrácia pluginov potrebných pre zostavenie aplikácie je možné nadefinovať v konfiguračnom súbore `pom.xml`, ktorý predstavuje už spomenutý model POM. Súbor popisuje projekt ako objekt pomocou jednoduchej XML štruktúry, kde sú nadefinované všetky spomenuté vlastnosti. Tento konfiguračný súbor sa nachádza v koreňovom adresári projektu. Pri rozsiahlejších projektoch môže existovať viac POM súborov. Budú vytvárať hierarchiu, kde sa využíva dedičnosť vlastností súborov. Projekt je aj takto možné zostaviť jediným príkazom, viac na [3].

Výhodou Maven je distribúcia projektov bez potrebných knižníc. Tieto sú definované v POM súbore a Maven pri zostavovaní projektu automaticky vyhľadá a nainštaluje potrebné knižnice. S tým je spojené aj riešenie možných závislostí medzi jednotlivými knižnicami a automatické doplnenie o chýbajúce knižnice. Vyhľadávanie prebieha v *úložiskách* (anglicky *repository*). Existuje globálne voľne prístupné Maven úložisko, ktoré je prednastavené ako hlavný zdroj. Potom existuje mnoho ďalších menších aj väčších serverov, ktoré majú svoje vlastné úložiská a odkazy na ne je možné dodatočne pridať do POM súboru.

Maven je v súčasnej dobe rozšírený a množstvo projektov je distribuovaných práve prostredníctvom Mavenu, napríklad aj moduly rámca Spring, viď 6.1. Podporujú ho aj dve často používané prostredia pre vývoj aplikácií nad Java platformou, *Eclipse* a *Netbeans*, ktoré ho integrujú priamo do svojich projektov a zjednodušujú tak vývoj.

Pri implementácií CRM systému CRMminer bolo potrebné kombinovať veľa rôznych technológii a vytvoriť z nich jeden funkčný celok. Každá z nich mala svoje potrebné knižnice a pomocné súbory. Preto bolo nutné dodržovať presnú a prehľadnú štruktúru projektu, aby sa aj napriek zložitosti, ktorá vznikne integráciou softvéru tretích strán do aplikácie, bolo možné v projekte zorientovať. Maven je na tento účel vhodným riešením. Spočiatku je časovo náročnejšie nastaviť všetky požadované vlastnosti objektového modelu POM pre potreby aplikácie, následne však tento strávený čas prevážia už len výhody spojené s jeho použitím. Ako príklad je možné uviesť rozšírenie do Maven, ktoré pri zostavovaní projektu podľa nadefinovaných parametrov zostaví rôzne verzie aplikácie pre vývoj alebo distribúciu. Ďalším vhodným nástrojom je možnosť vygenerovania a nastavenia databázy pri inštalácií projektu alebo napojenie na systémy pre správu zdrojových kódov.

6.2 Architektúra aplikácie

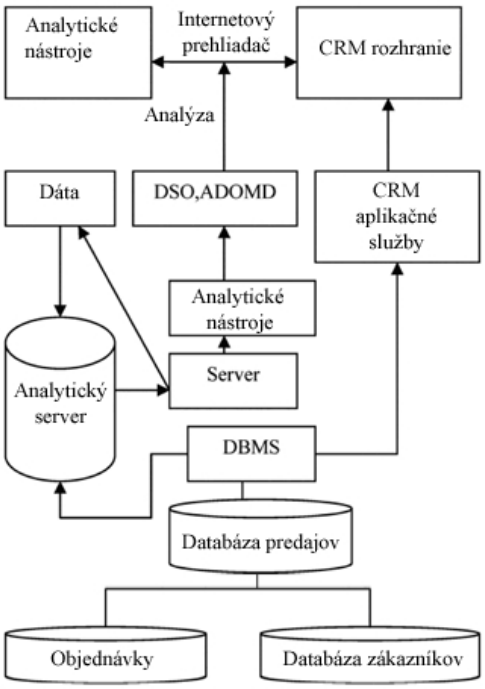
Pri návrhu aplikácie bolo potrebné vyriešiť problém integrácie externých modulov, ktoré budú ďalším zdrojom dát a doplnia aplikáciu o funkcie pre podporu rozhodovania. Ako najvhodnejšie sa javili dva spôsoby riešenia. Prvým spôsobom by bolo vytvorenie viacerých nezávislých aplikácií, ktoré by medzi sebou komunikovali pomocou webových služieb. CRM systém by tak komunikoval s externým systémami a zaisťoval by len konečný výstup pre užívateľa. Tento spôsob by bol jednoduchší na implementáciu, ponúkala by sa možnosť použitia už existujúceho CRM systému, v ktorom by bolo potrebné doplniť knižnice a urobiť len malé zmeny na prezentačnej vrstve. Nevýhodou tohto spôsobu by bola obmedzená možnosť použitia modulov na niektorých vrstvách aplikácie a nutnosť upraviť jej štruktúru pre správne zobrazenie dát. Z obchodného hľadiska je nevýhodou zložitejšia distribúcia systému pre zákazníkov, keďže aplikácia netvorí jednotný celok.

Druhým spôsobom je priama integrácia modulov do architektúry aplikácie a vytvorenie komplexného modulu pre podporu rozhodovania ako je to popisované v práci CRM systémy založené na OLAP [1]. CRM systémy zvyčajne obsahujú veľké objemy dát pochádzajúce z rôznych zdrojov, ktoré však majú pri nesprávnej interpretácií pre používateľa systému len malú informačnú hodnotu. Preto je výhodnejšie navrhnúť systém s integrovanými nástrojmi. Nevýhodou tohto riešenia je zložitejšia implementácia vzhľadom na nutnosť kombinovať rôzne technológie a zaistenie ich bezproblémovej komunikácie. Ukážka modelu CRM systému, ktorý je založený a priamo integruje nástroje na podporu rozhodovania je na obrázku 6.5.

Architektúra aplikácie musí umožňovať začlenenie externých aplikácií pre podporu rozhodovania a to takým spôsobom, aby neexistovala žiadna priama závislosť od konkrétnej verzie týchto aplikácií a existovala jednoduchá možnosť pridania nových modulov podľa potreby. Samotná funkčnosť modulov musí byť pre ostatné vrstvy aplikácie odtienená. Pre tieto vrstvy sa musí jednať iba o štandardný zdroj dát, ktoré sú im poskytované v nadefinovanom formáte a tieto sú ďalej spracované.

Architektúra aplikácie CRMminer nadväzuje na štruktúru, ktorá je definovaná aplikačným rámcom Spring. Rámec je použitý na všetkých vrstvách aplikácie, inicializácia jednotlivých vrstiev a ich komunikácia medzi sebou je záležitosťou rámca.

Aplikácia pracuje v troch vrstvách. Sú to vrstvy prezentačná, aplikačná a dátová, zobrazené na obrázku 6.6. Najnižšie sa nachádza dátová vrstva, ktorej úlohou je práca s dátami. Primárne je tvorená množinou DAO objektov (viď kapitolu 6.1.2). V rámci celej aplikácie je táto vrstva jediná, ktorá manipuluje s perzistentnými dátami nad databázou. Skladá sa z viacerých modulov, každý z týchto modulov môže používať iný zdroj dát. Všetky moduly,



Obrázek 6.5: CRM založené na OLAP, upravené podľa [1].

ktoré pracujú v tejto vrstve musia implementovať jednotné rozhrania. Výhoda takéhoto prístupu spočíva v tom, že vyššie vrstvy, ktoré využívajú služby dátovej vrstvy, získavajú dáta vždy v rovnakom formáte a skutočný pôvod dát je pre ne transparentný.

Hlavným zdrojom dát je relačná databáza, nad ktorou pracuje objektovo-relačný rámec Hibernate. Ten tvorí samostatný modul vrstvy a zabezpečuje všetky základné operácie nad dátami t.j. vytváranie, načítavanie, editáciu a mazanie dát.

Aplikačná vrstva je tvorená pomocou služieb, v aplikácií sa tieto služby nazývajú *manažéri*, iným slovom manažéri služieb. V manažéroch je skrytá samotná logika aplikácie. Manažér je tvorený triedou, ktorá implementuje rozhranie obsahujúce množinu požadovaných metód pre dosiahnutie potrebnej funkcionality. Pri implementácií využíva DAO objekty dátovej vrstvy, združuje informácie z viacerých zdrojov, spracuje ich, pričom používa externé moduly a vytvára z dát požadovaný formát, ktorý je jeho výstupom pre vyššie vrstvy aplikácie. Samotná logika je pre vyššie vrstvy odtienená. Aplikačná vrstva ďalej pozostáva s modulu pre OLAP dotazovanie nad databázou pomocou serveru *Mondrian* a modulu pre operácie spojené so získavaním znalostí, ktorý využíva knižnicu nástrojov pre dolovanie dát *WEKA*. Tieto dva moduly tvoria jadro modulu pre podporu rozhodovania a poskytujú dáta pre triedy manažérov, ktoré ich spracujú pre potreby prezentačnej vrstvy. Podrobnejší popis obidvoch modulov a spôsob, akým boli začlenené do aplikácie, sa nachádza v kapitole 7.

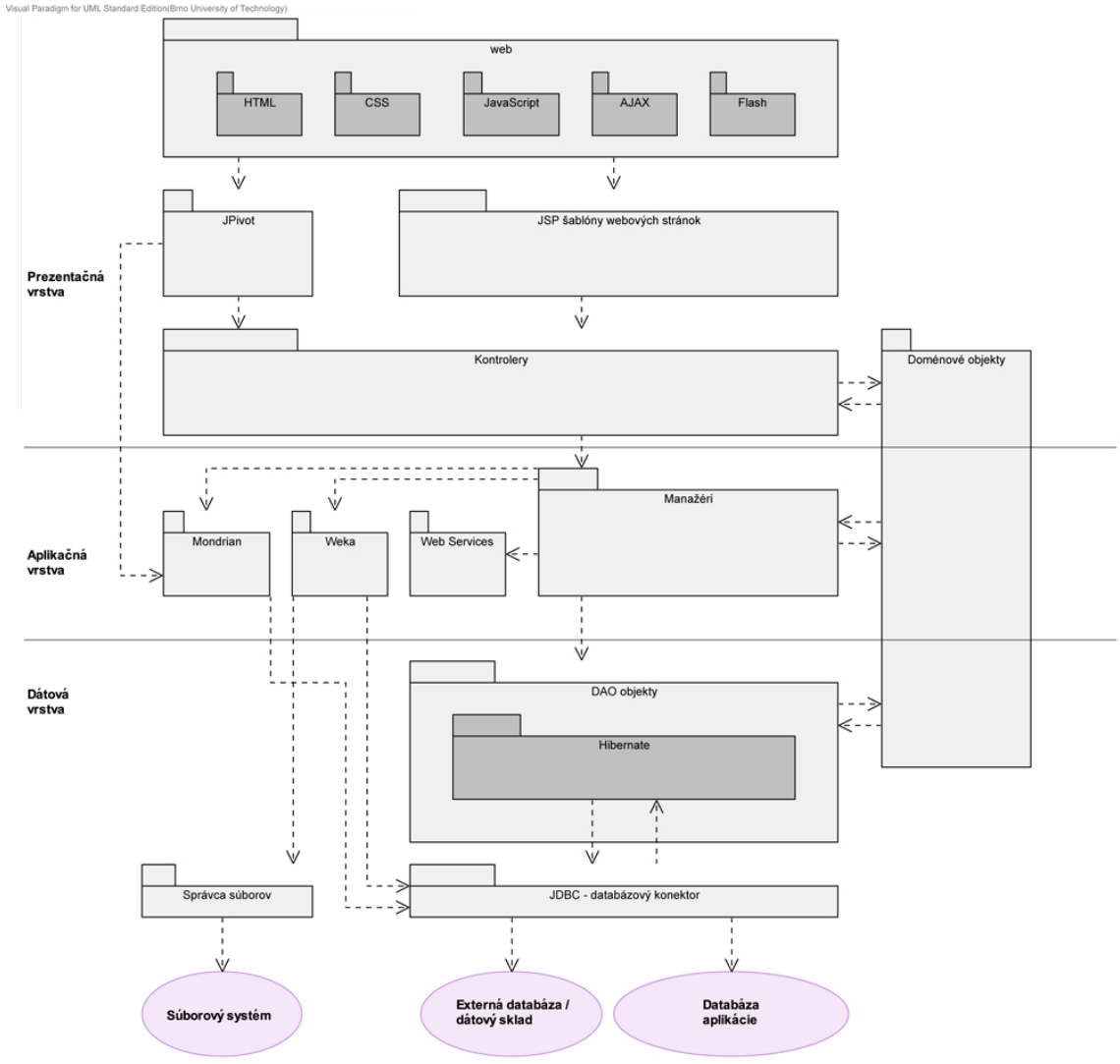
Poslednou vrstvou je prezentačná vrstva, ktorú reprezentuje MVC modul rámca Spring. Skladá sa z vrstvy kontrolérov, ktoré využívajú služby manažérov pre získanie potrebných dát a tie ďalej predávajú vrstve pre zobrazovanie. Vrstva pre zobrazovanie je tvorená pomocou JSP (viď [7]) šablón v spojení s dekoračným rámcom SiteMesh. Je vhodne doplnená o technológie *jQuery* (rámec pre jazyk *javascript*) a *FusionGraph* (podrobnejší popis v sekcií

7.1.2)

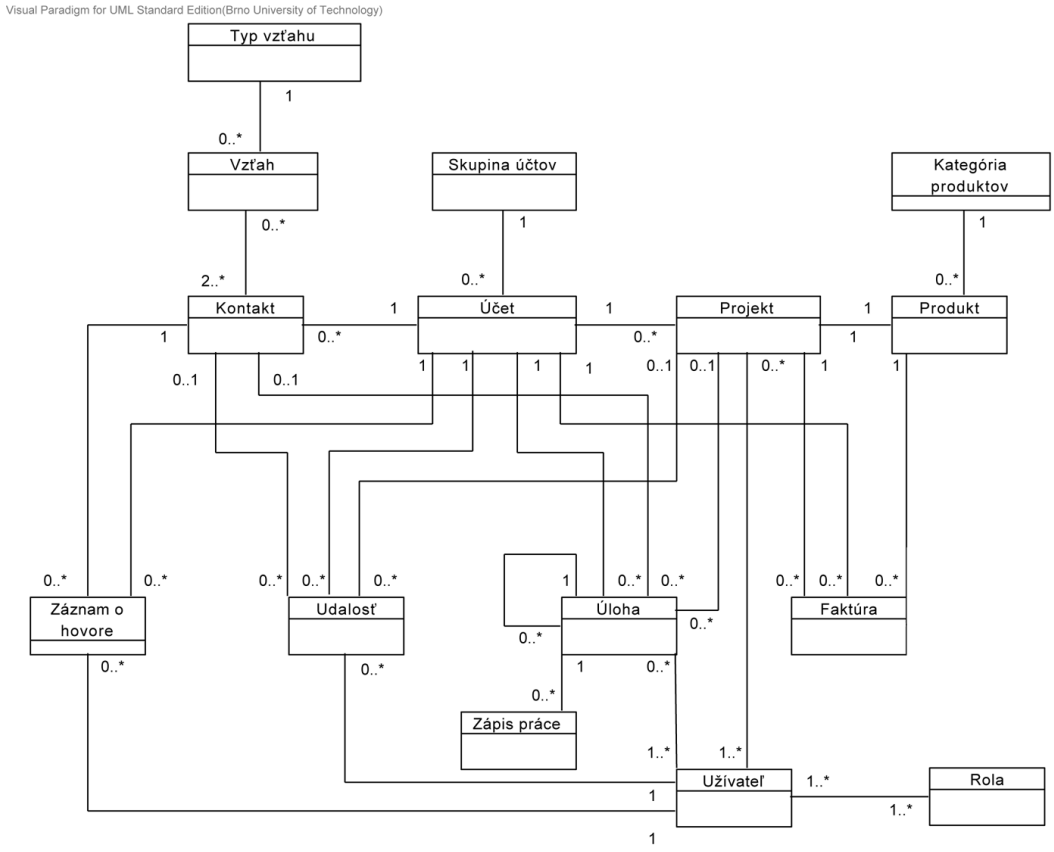
Všetky potrebné nastavenia aplikácie sú uložené v nezávislých konfiguračných súboroch. Nastavenia sú logicky oddelené na konfiguráciu samotnej aplikácie a konfiguráciu externých modulov.

6.3 Diagram tried

Na obrázku 6.7 sa nachádza diagram tried aplikácie. V reálnej aplikácií sú všetky moduly CRM systému na seba úzko naviazané. Uvedený diagram je pre prehľadnosť zjednodušený a nie sú na ňom vyobrazené všetky vzájomné vzťahy. Na základný prehľad o moduloch však postačuje. Podrobnejší popis jednotlivých modulov sa nachádza v sekcií 6.5.



Obrázek 6.6: Architektúra aplikácie CRMminer.



Obrázek 6.7: Diagram tried aplikácie CRMminer.

6.4 Diagram prípadov použitia

Na obrázku 6.8 je zobrazený diagram prípadov použitia aplikácie CRMminer. Základná varianta systému rozlišuje štyri role užívateľov.

Na najnižšej úrovni sa nachádza rola „zákazník“. Pri niektorých projektoch sa vytvorí systémový účet pre zákazníka a ten následne prostredníctvom tohto účtu môže sledovať prácu na svojich projektoch, zadávať a schvalovať úlohy v rámci projektu.

Nasleduje rola hlavného manažéra. Manažér potrebuje mať prehľad vo všetkých oblastiach CRM systému, preto má práva pre správu všetkých modulov okrem modulu pre správu užívateľov.

Špecifickým typom užívateľa je užívateľ s nastavením práv. Pri tejto roli administrátor prostredníctvom modulu správy užívateľov nadefinuje pre konkrétneho užívateľa práva vstupu k jednotlivým modulom. Tento užívateľ tak môže nadobudnúť rovnaké práva ako má administrátor.

Rola administrátor predstavuje užívateľa s najväčším právami v rámci celej aplikácie, má prístup do všetkých modulov.

6.5 Popis modulov

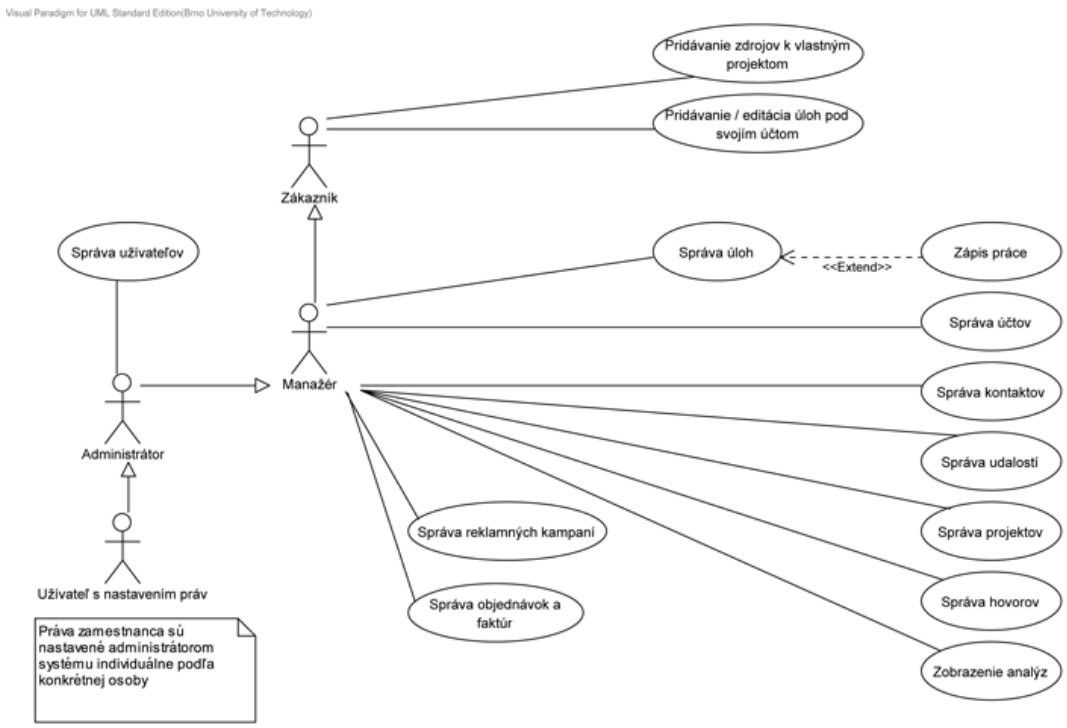
Hlavným modulom aplikácie je modul pre správu účtov. Účet nemusí reprezentovať len zákazníka, môže sa jednať o obchodného partnera, konkurenčnú firmu, distribútora a iné. Každá spoločnosť alebo súkromná osoba, ktorá je nejakým spôsobom vo vzťahu s firmou využívajúcou CRM musí mať založený svoj účet. Bez vlastného účtu nie je možné ukladať k osobe alebo firme žiadne ďalšie informácie. Účty môžu byť zoskupované do skupín, podľa zamerania alebo potrieb spoločnosti. Tieto skupiny je možné nadefinovať v administrácií.

Modul kontakty nadväzuje na modul účtov. K účtu sa môžu vzťahovať viaceré kontakty, tieto je možné prehľadne zobrazovať a filtrovať. Pri kontakte sa uchovávajú všetky štandardné informácie o adrese, kontaktoch ale aj údaje na sociálne siete a iné. Návrh databázy zohľadňuje rozšírenie modulu o uchovávanie vzájomných vzťahov medzi kontaktmi, v aktuálnej verzií systému to nie je implementované.

Modul projekty predstavuje projekty, ktoré sú priradené k účtom. Projekt sa vzťahuje na určitý produkt. Môžu byť ku nemu priradené faktúry vystavované v priebehu práce. Môže mať viacerých riešiteľov. Modul obsahuje nástroj pre správu skladu súborov potrebných k riešeniu projektu.

Modul úlohy spravuje úlohy, ktoré sú priradené užívateľom systému. Každá úloha môže mať viacerých riešiteľov a vzťahuje sa k účtu, kontaktu alebo projektu. Definuje popis, prioritu úlohy a aktuálny stav riešenia. Dôležitou položkou pre štatistiky je odhad hodín, ktorý bude strávený na úlohe. K úlohe sa úzko viaže modul pre zápis práce, kde sa zapíše reálny odpracovaný čas na úlohe. V následných štatistikách sa potom uvádza odhadovaný čas a reálny čas, čo v konečnom dôsledku definuje zisk alebo stratu.

Modul udalosti je možné popísať ako plánovač udalostí, ktoré sa viažu na účet, kon-



Obrázek 6.8: Diagram prípadov použitia aplikácie CRMminer.

takt alebo projekt. Definuje začiatok, koniec a miesto udalosti. Modul záznamy hovorov sa vzťahuje na účet alebo kontakt, v aktuálnej verzii systému nie je využívaný.

Modul správa užívateľov slúži na správu užívateľov systému. Štruktúra modulu je definovaná použitou knižnicou pre zabezpečenie pre rámec Spring, viď [23]. Každý užívateľ ma prístupové údaje a môže zastávať jednu alebo viac rolí v administrácií systému. Podľa príslušnej role má potom oprávnenia vstupu k jednotlivým modulom.

Modul pre monitoring a analýzu slúži na zobrazovanie analýz dát v rámci systému. Ako hlavný zdroj dát používa OLAP server Mondrian. Ten používa niekoľko dátových skladov a dáta zobrazuje pomocou nástroja pre vizualizáciu viacrozmerných dátových kociek *JPivot*. Štatistiky pochádzajú z rôznych oblastí, vzorová verzia systému zobrazuje štatistiky pre projekty a predaje.

Kapitola 7

Popis modulu pre podporu rozhodovania

Navrhovaný CRM systém je zameraný pre použitie v menších spoločnostiach s malým a stredným počtom zamestnancov, v ktorých je kladený dôraz na efektivitu práce. Chybné strategické rozhodnutia môžu mať negatívny dopad pre ďalšie pôsobenie firmy. Modul pre podporu rozhodovania by mal užívateľom jednoduchým spôsobom umožniť získať prehľad z oblasti jednotlivých modulov aplikácie ako sú napríklad prehľady predaja za jednotlivé časové obdobia, prehľad záujmu o jednotlivé kategórie produktov alebo konkrétne produkty a podobne.

Aplikácia je modulárna a jej konečná podoba môže byť pre jednotlivých zákazníkov odlišná. Preto je potrebné, aby modul pre podporu rozhodovania umožňoval jeho prispôsobenie potrebám zákazníkov a získaval informácie z rôznych dátových zdrojov. Zdrojom dát je vo väčšine prípadov relačná databáza, pri napojení aplikácie na externé zdroje však musí modul umožniť spracovať dáta aj z databáz, ktoré nie je priamo súčasťou CRM systému. Ako príklad je možné uviesť napojenie CRM systému na elektronický obchod pre získavanie štatistík predaja.

Modul pre podporu rozhodovania je možné rozdeliť na dve časti, viď kapitolu 5. Prvou je modul pre analýzu dát, ktorý je reprezentovaný serverom pre OLAP analýzy nad relačnou databázou v spojení s klasickým dotazovaním nad touto databázou. Informácie získané v tejto časti sú užívateľovi prezentované pomocou rozličných typov grafov a pomocou kontingenčnej tabuľky pre zobrazenie dátovej kocky v dvojrozmernom priestore. Druhou časťou je modul pre získavanie znalostí. Tento modul slúži na získavania nových, zaujímavých informácií, ktoré z dát nie sú normálnym spôsobom viditeľné. Využitie bude nachádzať v oblastiach klasifikácie, pri kategorizácií dát a v oblasti prediktívnych úloh, keď budú predpovedané hodnoty atribútov na základe histórie.

Táto kapitola sa zaoberá integráciou open-source nástrojov pre analýzu dát od skupiny *Pentaho Analysis* v prostredi webovej aplikácie CRMminer za účelom vytvorenia modulu pre podporu rozhodovania v rámci vytvoreného CRM systému. V prvej časti tejto kapitoly je popísaný analytický modul reprezentovaný OLAP serverom *Mondrian*. Je popísaná jeho architektúra a spôsob akým bol začlenený do aplikácie. Nasleduje popis návrhu dátového skladu, príklady dotazov a popisuje spôsob napojenia na ďalšie moduly zobrazujúce výsledky dotazov. Druhá časť popisuje použitie nástroja pre získavanie znalostí *WEKA*. Zaoberá sa postupom, akým bol tento modul upravený pre potreby integrácie do aplikácie a poukazuje na možnosti jeho využitia.

7.1 Mondrian - analytický server

Mondrian je OLAP server implementovaný v jazyku Java. Spracováva dotazy napísané v *MDX* (*Multi-Dimensional eXpressions*, viď sekciu 2.2.6). Zdrojom dát je relačná databáza a výsledky dotazov sú prezentované vo viacrozmernej forme prostredníctvom Java API. Umožňuje podnikovým analytikom spracovať množstvo informácii v reálnom čase. Jeho tvorcom je skupina *Pentaho*, ktorá sa zaoberá tvorbou voľne dostupných riešení pre „business intelligence“. Podporuje takmer všetky aktuálne používané relačné databázové systémy [20].

Architektúru serveru Mondrian tvoria štyri vrstvy a to prezentačná vrstva, vrstva dimenzií, vrstva hviezd a databázová vrstva. Sú zobrazené na obrázku 7.1.

Prezentačná vrstva predstavuje samotný výstup, ktorý uvidí užívateľ na svojom monitore. Existuje mnoho spôsobov zobrazovania multidimenzionálnych dát. Jedná sa napríklad o takzvané pivotové tabuľky (*pivot tables*, viď sekciu 2.2.5), ktoré dovoľujú určitú interakciu s užívateľom v podobe jednoduchého zadávania nových dotazov. Ďalším spôsobom je napríklad zobrazenie pomocou rôznych typov grafov. Hlavnou úlohou prezentačnej vrstvy je vytváranie multidimenzionálnych dotazov, tieto sú spracované analytickým serverom. Následne sa stará o zobrazovanie výsledkov týchto dotazov. Všetky klientské riešenia majú spoločnú gramatiku, v ktorej odosielajú MDX dotazy OLAP serveru a v nej dostávajú odpovede. Server podporuje použitie tenkého klienta, tučného klienta alebo je možné server inštalovať ako webovú službu. Posledný uvedený spôsob predstavuje veľkú výhodu, pretože umožňuje spoluprácu servera s technológiami, ktoré sú postavené na rôznych platformách.

Druhou vrstvou je vrstva dimenzií. Jej úlohou je parsovanie, validácia, a spúšťanie MDX dotazov. Dotazy sú vyhodnocované v niekoľkých fázach. V rámci vyššej efektivity sa posielajú požiadavky do agregačnej vrstvy po dávkach a transformácia dotazov umožňuje použitie existujúcich dotazov skôr, ako sa dotazy budú spracovávať pre každú požiadavku od úplného začiatku. Model dimenzií a jeho mapovanie na relačný model je popísaný pomocou metadát reprezentovaných XML súborom.

Treťou vrstvou je vrstva hviezd. Je zodpovedná za udržiavanie *cache* (*vyrovnávacej pamäti*) agregovaných záznamov. Agregovanými záznamami rozumieme predpočítaný súbor merateľných hodnôt (buniek), ktoré sú zoskupené podľa určitej množiny dimenzií. Dimenzionálna vrstva pošle požiadavku na množinu buniek. Ak požadované bunky nie sú v cache, alebo ich nie je možné získať zobecnením uložených záznamov v cache, manažér agregácií pošle požiadavku do databázovej vrstvy.

Poslednou, štvrtou vrstvou je databázová vrstva reprezentovaná relačným databázovým systémom. Jej úlohou je poskytovať agregované hodnoty buniek tabuliek a členov tabulek dimenzií. Základnou stratégiou Mondrian je prenechať čo najväčšie množstvo operácií na relačnú databázu. Agregované dotazy sú získavané pomocou `group by` sql dotazov, viď [20]. Podporuje využívanie materializovaných pohľadov, ak používaná databáza umožňuje ich vytváranie. Výhodou tohto prístupu, keď Mondrian nepotrebuje vlastné úložište dát, je jeho veľmi jednoduchá integrácia s inými aplikáciami prostým inštalovaním knižnice Mondrian. Pretože nepoužíva žiadne nadbytočné dáta pre správu, je proces spracovania dát jednoduchší a Mondrian sa tak stáva ideálným nástrojom pre aplikáciu OLAP nad databázou v reálnom čase.

Pre klientské aplikácie poskytuje Mondrian aplikačné rozhranie, ktoré je podobné rozhraniu JDBC. Rozdiel je v tom, že dotazy sú písané v jazyku MDX namiesto jazyka SQL. Fragment zdrojového kódu, ktorý je uvedený nižšie, ukazuje spôsob pripojenia ku Mondrianu, následné zadanie dotazu použitím MDX a výpis výsledku dotazu na štandardný

výstup.

Príklad 7.1: Ukážka pripojenia k serveru Mondrian prostredníctvom API rozhrania [20].

```
import mondrian.olap.*;
import java.io.PrintWriter;

Connection connection = DriverManager.getConnection(
    "Provider=mondrian;" +
    "Jdbc=jdbc:odbc:MondrianFoodMart;" +
    "Catalog=/WEB-INF/FoodMart.xml;",
    null,
    false);
Query query = connection.parseQuery(
    "SELECT {[Measures].[Unit Sales], [Measures].[Store Sales]} " +
    " on columns, {[Product].children} on rows " +
    "FROM [Sales] " +
    "WHERE ([Time].[1997].[Q1], [Store].[CA].[San Francisco])");
Result result = connection.execute(query);
result.print(new PrintWriter(System.out));
```

Prístup na OLAP server cez webové služby je umožnený pomocou štandardu XMLA (*XML for Analysis*). Aplikáciám, ktoré nepoužívajú Java platformu, tak môžu posielať dotazy na Mondrian a následne spracovať ich výsledok, pre podrobnejšie informácie o servery Mondrian viď [20].

7.1.1 Návrh a implementácia dátového skladu

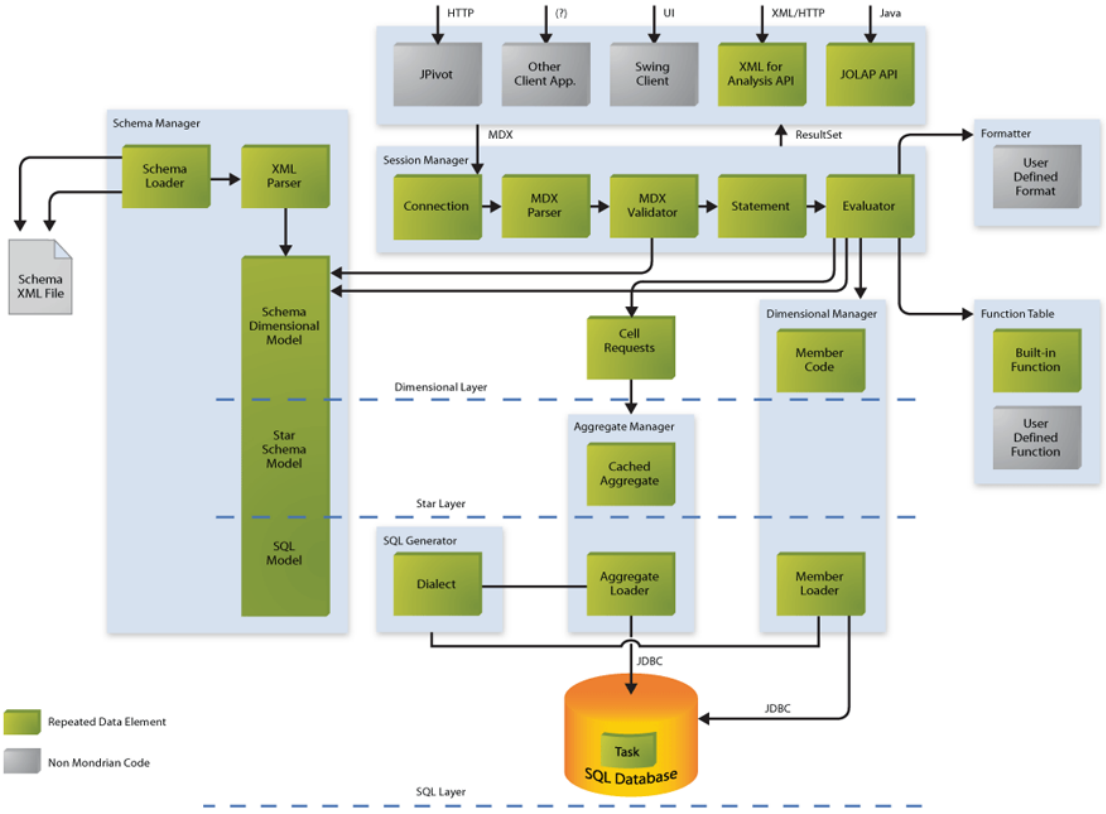
Dátový sklad, umožňujúci efektívnu analýzu a dotazovanie, je v Mondrian tvorený logickým a fyzickým modelom. Logický model predstavuje schému vo formáte XML. Táto schéma definuje multi-dimenzionálnu databázu. Pozostáva z kociek, hierarchií, prvkov a nastavenia mapovaní modelu na fyzický model. Štruktúra logického modelu je potom používaná na vytváranie MDX dotazov nad databázou. Fyzický model, tvorený relačnou databázou, je zdrojom dát, ktoré sú prezentované prostredníctvom logického modelu. V aplikácií je implementovaný pomocou schémy typu hviezda. Táto je tvorená súborom tabuliek faktov a tabuliek dimenzií v relačnej databáze [20].

V nasledujúcich odsekoch sa nachádza podrobnejší popis návrhu fyzického a logického modelu.

Návrh fyzického modelu

Relačná databáza aplikácie je normalizovaná, atribúty obsahujú atomické (ďalej nedeliteľné) hodnoty a nie je preto vhodná pre spracovanie pomocou OLAP servera. Riešením je vytvorenie dátového skladu, ktorý by obsahoval potrebné dáta vo vhodnej forme (atomické aj agregované hodnoty). Požiadavkou je, aby OLAP server pracoval vždy s aktuálnymi dátami. V praxi to znamená, že keď sa urobí zmena v operačnej časti aplikácie, analytická časť ju musí ihneď zohľadniť.

Ponúkajú sa dve vhodné riešenia, ktoré by tejto požiadavke vyhoveli. Prvým riešením je vytvorenie oddeleného dátového skladu, kde by sa prevádzali dáta z operačnej databázy

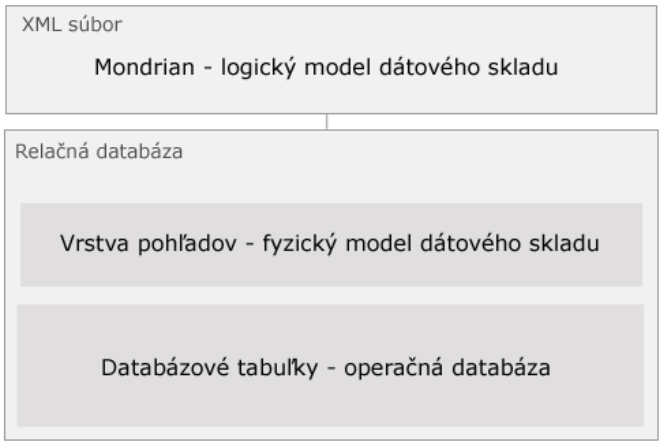


Obrázek 7.1: Architektúra serveru Mondrian, prevzaté z [20].

pomocou takzvaných dátových púmp (ETL nástroje, viď kapitolu 2.2.1). Zmeny v operačnej databáze by sa tak vhodným spôsobom synchronizovali s dátovým skladom. Toto riešenie má svoj význam pri veľkých objemoch dát, kde je potrebné, aby bol dátový sklad umiestnený na oddelenom servery a nezaťažoval tak operačnú databázu.

Druhým riešením je vytvorenie vrstvy pohľadov nad relačnou databázou. *Pohľad* je v teórii databáz virtuálna, či logická databázová tabuľa vytvorená ako výsledok množiny predspracovaných dotazov. Na rozdiel od ostatných tabuliek v relačných databázach pohľad nie je súčasťou fyzickej schémy databázy. Pohľad odkazuje na dáta z jednej alebo viacerých databázových tabuliek alebo na iné pohľady. Umožňuje filtrovať potrebné stĺpce a riadky s tabuľky, vytvárať agregácie nad dátami alebo združovať dáta z viacerých tabuliek a zobrazovať ich ako celok, viac na [2]. Pomocou pohľadov tak vieme transformovať dáta z operačnej databázy do štruktúry, ktorá bude vhodná na napojenie na logický model dátového skladu. Tento fyzický model môžeme nazvať ako virtuálny fyzický model nad relačnou databázou.

Aplikácia CRMminer, ako už bolo uvedené, je určená pre malé a stredné firmy, kde sa nepredpokladajú veľké objemy dát. Preto sa primárne pri vytváraní fyzického modelu dátového skladu použije druhý spôsob, t.j. vytvorenie vrstvy pohľadov nad databázou, zobrazený na obrázku 7.2. V konfigurácií analytického modulu je však možné jednoducho nastaviť ako zdroj dát externý databázový server s oddeleným dátový skladom a v prípade potreby prejsť na tento spôsob riešenia dátového skladu.



Obrázek 7.2: Spôsob vytvorenia dátového skladu v aplikácií CRMminer.

Návrh logického modelu

Najdôležitejšími prvkami pri návrhu logického modelu sú kocky, fakty a dimenzie, pre podrobnejšie vysvetlenie viď kapitolu 2.2.

- *kocka* – je kolekciou dimenzií a faktov so zameraním na určitú oblasť.
- *fakty* – sú jednotky, ktoré nás zaujímajú pri analýzach, napríklad počet predaných položiek, náklady na výrobu a podobne.
- *dimenzie* – je atribút alebo skupina atribútov, podľa ktorých môžeme rozdelovať fakty na menšie celky (kategórie). Napríklad predaj produktov podľa pohlavia zákazníka, viac na [20].

Nižšie uvedený kód zobrazuje jednoduchý logický model dátového skladu. Obsahuje jednu tabuľku faktov, ktorá obsahuje údaje o predaji produktov spoločnosti a jednu tabuľku dimenzie času, ktorá umožňuje vytvárať pohľady na predaj na troch úrovniach hiearchie a to mesiac, rok, kvartál. V ukážke kódu je vidieť spôsob, akým sa definuje dátová kocka, jej dimenzie a jednotlivé fakty. Z praktického hľadiska nemá uvedený model veľkú informačnú hodnotu. Slúži len pre základnú ilustráciu. Pre detailnejšie vysvetlenie nastavenia atribútov jednotlivých XML elementov pozri na [20].

Problémov pri spracovaní dát z operačnej databázy sú výčtové typy alebo rôzne druhy číselníkov. Hodnoty týchto typov sú v databáze uložené len ako číselná hodnota a táto nemá pri zobrazovaní v kontingenčnej tabuľke takmer žiadnu informačnú hodnotu. Niektoré atribúty je možné transformovať do vhodnej podoby pri vytváraní vrstvy pohľadov nad databázou, niektoré je nutné upraviť programovo. Mondrian obsahuje nástroje pre úpravu formátu atribútov používaním takzvaných *callback metód*. Callback metódy sa definujú pri vytváraní logického modelu. Jedná sa o volania Java tried, ktoré implementujú rozhranie definované v Mondrian API, viac na [20].

Príklad 7.2: Ukážka logického modelu dátového skladu.

```
<Schema>
  <Cube name="Predaj">
    <Table name="zaznamy_o_predaji"/>
    <Dimension name="Cas" foreignKey="cas_id">
      <Hierarchy hasAll="false" primaryKey="cas_id">
        <Table name="cas_predaja"/>
        <Level name="Rok" column="rok" uniqueMembers="true"/>
        <Level name="Kvartal" column="kvartal" uniqueMembers="false"/>
        <Level name="Mesiac" column="mesiac" uniqueMembers="false"/>
      </Hierarchy>
    </Dimension>
    <Measure name="Trzby" column="sales" aggregator="sum" />
    <Measure name="Naklady" column="cost" aggregator="sum" />
  </Cube>
</Schema>
```

7.1.2 Zobrazovanie výsledkov dotazov a napojenie na moduly CRM

Výsledky analytického spracovania dát sa v aplikácií CRMminer zobrazujú dvomi spôsobmi. Prvým spôsobom je použitie knižnice JPivot pre zobrazovanie viacdimenzionálnych dát vo forme kontingenčnej tabuľky. Druhý spôsob je zobrazenie pomocou knižnice grafov. V nasledujúcich odstavcoch sú popísané obidva spôsoby zobrazenia s ukážkami použitia v aplikácií.

JPivot

JPivot je knižnica obsahujúca sadu JSP značiek určená primárne pre webové aplikácie postavené na Java platforme, ktoré slúžia na vykresľovanie multi-dimenzionálnych dát ktorých zdrojom je OLAP server, vo forme kontingenčnej tabuľky. Umožňuje používateľom vykonávať typické OLAP operácie, ako je natáčanie kocky, prevádzanie rezov, zobrazovanie dát na rôznej úrovni hierarchií a iné. Súčasťou distribúcie JPivot je knižnica *WCF* (*Web Component Framework*, viac na [25]) značiek, ktoré podporujú vytváranie komponentov užívateľského rozhrania pre webové aplikácie pomocou technológií XML a XSLT.

Zdrojom dát pre JPivot je analytický server Mondrian. V konfigurácií značky pre vykreslenie tabuľky je potrebné zadať adresu na spojenie s databázovým serverom, na ktorom sa nachádza fyzický model dátového skladu, cestu k XML súboru s logickou reprezentáciou dátového skladu a nakoniec samotný MDX dotaz, ktorého výsledok bude JPivot zobrazovať. JPivot umožňuje aj vzdialené načítavanie dát pomocou webových služieb, kedy sa môže OLAP server nachádzať na oddelenom servery. Novšie verzie serveru Mondrian obsahujú vlastnú verziu knižnicu JPivot, preto je pri inštalácií z dôvodu kompatibility a odstránenia problémov pri kompilácií projektu, vhodné použiť túto upravenú verziu [25].

Súčasťou knižnice JPivot je skupina nástrojov užívateľského rozhrania, ktoré umožňujú pracovať s výsledkami dotazov a prispôsobovať ich zobrazenie potrebám užívateľa. Jedná sa o filtrovanie, zoraďovanie, grafické zvýrazňovanie určitých položiek a podobne. Mondrian distribuuje JPivot s knižnicou *Jfreechart* rozširujúca JPivot o zobrazovanie aktuálnych dát pomocou rôznych typov grafov. Typ grafu, jeho farebnosť a ďalšie nastavenia je možné

nadefinovať pomocou konfigurátora.

			Measures	
Account	Project	User	estimateOfHours	realHours
+All Accounts	−All Projects	+All Users	337	54
	−cms system	+All Users	335	54
	+pagepack basic	−All Users	335	54
		−admin	335	54
		Jan Růžička	1	1
		Lepka Peter		
		Lukáš Matejka	10	2
		Michal Filus	324	51
		Tomáš Kymlička		
		+admin,user		

Obrázek 7.3: Ukážka JPivot.

Na obrázku 7.3 sa nachádza ukážka použitia knižnice JPivot v aplikácií CRM miner. Na základe tabuľky faktov o úlohách, ktorá obsahuje informácie o počte odhadovaných a skutočných hodín strávených na úlohách naviazaných k projektom, je možné jednoducho zistiť, ktorý projekt je stratový a na ňom strávený čas prekračuje plánovaný rozpočet a naopak, ktorý projekt bol odhadnutý správne. Zobrazená tabuľka umožňuje vykonávať OLAP operácie z pohľadu troch dimenzií a to dimenzie projektov, dimenzie účtov a dimenzie užívateľov.

FussionChart

FussionChart je voľne dostupná knižnica umožňujúca vytvárať animované a interaktívne grafy pre webové aplikácie. Knižnica je vytvorená technológiou *Macromedia Flash MX* a obsahuje podporu pre použitie v mnohých skriptovacích jazykoch ako PHP, ASP, .NET, JSP, JavaScript a iné. Obsahuje všetky základné typy grafov, niektoré typy sú podporované vo viacerých verziách [9].

Inštalácia knižnice spočíva v nakopírovaní všetkých potrebných súborov do koreňového adresára webovej aplikácie. FussionChart obsahuje podporu pre použitie s java aplikáciami v podobe JSP súboru, ktorý sa stará o spracovanie vstupných parametrov a následné vykreslenie grafu. V šablónach aplikácie je potom potrebné tento súbor načítať, nadefinovať parametre a vykreslenie grafu prebehne automaticky.

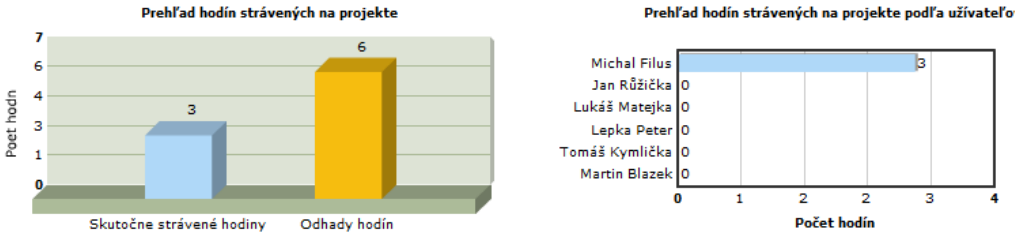
Vstupné dáta pre knižnicu sú reprezentované XML súborom. Jeho základná štruktúra je pre všetky grafy rovnaká. Odlišnosti potom vznikajú vzhľadom na rôzny spôsob zobrazovania dát podľa typu grafu. Tento súbor je možné nadefinovať ako URI parameter. To znamená ako adresu, na ktorej sa XML súbor nachádza. Môže to byť webová adresa alebo fyzická adresa súboru na disku. Druhou možnosťou je nadefinovať XML súbor ako textový reťazec a práve táto možnosť je použitá aplikácií. Vytvárať kompletný XML súbor pri každom použití grafu by bolo zdĺhavé a pri možnej zmene konfigurácie grafu, alebo jeho typu by bolo potrebné na všetkých miestach v kóde, kde je použitý, upraviť XML reťazec. Z toho dôvodu je výhodné vytvorenie generátora XML súborov, ktorý by obsahoval položky grafu a všetky potrebné nastavenia. V CRMminer sa o to stará trieda XMLgenerátor. Pre každú položku je možné nadefinovať jej názov, hodnotu a farbu pod ktorou sa bude zobrazovať.

Ďalšie nastavenia spočívajú v názve grafu a názvoch jeho osí. Z dôvodu odlišnej štruktúry XML súboru pre použité typy grafov existuje niekoľko verzií triedy na jeho vygenerovanie. Rozmery grafu je možné nadefinovať priamo v šablóne podľa možností konkrétnej stránky. Podrobnejší popis nastavní grafu sa nachádza v [9]. Ukážka použitia grafu v JSP šablóne je zobrazená v nasledujúcom kóde.

Príklad 7.3: Ukážka použitia grafov v JSP šablóne.

```
<jsp:include page="../Includes/FusionChartsHTMLRenderer.jsp" flush="true">
    <jsp:param name="chartSWF" value="FusionCharts/FCF_Column3D.swf" />
    <jsp:param name="strURL" value="" />
    <jsp:param name="strXML" value="XML_retazec_na_vstupe" />
    <jsp:param name="chartId" value="Nazov grafu" />
    <jsp:param name="chartWidth" value="600" />
    <jsp:param name="chartHeight" value="300" />
    <jsp:param name="debugMode" value="false" />
</jsp:include>
```

Zdrojom dát pre generátor XML súborov v aplikácií je relačná databáza, výsledok dotazovania nad dátovým skladom, výsledok dolovania dát alebo ich kombinácia. Generovanie XML prebieha na aplikačnej vrstve, ktorú tvoria manažéri dát. Ich základnou vlastnosťou je odtienenie zdroja dát vzhľadom na ich ďalšie použitie a preto ich pôvod nemá žiadny vplyv na ich spracovanie a zobrazenie v grafe. Na obrázku 7.4 sa nachádza ukážka zobrazenia grafov pre modul účet. Prvý graf predstavuje počet odhadovaných a skutočných hodín ktoré bol strávené na projektov zákazníka a druhý graf ukazuje počty hodín ktoré na projektoch strávili užívatelia systému.



Obrázek 7.4: Ukážka FussionChart.

Riešenie problému so zápisom dát do vyrovnávacej pamäti

Pri zobrazovaní dát spôsobuje problém vyrovnávacia pamäť (anglicky *cache*). Mondrian štandardne používa cache pre zvýšenie výkonu. Pri prvom spustení dotazu bude Mondrian načítavať dáta pomocou viacerých SQL dotazov, následne však bude používať už načítané dáta z tejto pamäte. Informácie v databáze CRM systému sa však menia veľmi často, ako príklad môže byť zápis práce pre jednotlivé úlohy, čo je činnosť ktorú používatelia vykonávajú niekoľkokrát za deň. Tieto zmeny sa musia ihneď preniesť aj do analýz, preto je použitie cache v tomto prípade neprípustné. Problém s cache je potrebné riešiť na dvoch úrovniach.

Prvou úrovňou je použitie servera priamo v objektoch pre prístup k databáze pomocou rozhrania Mondrian API (programové rozhranie pre prácu so serverom). Na tejto úrovni slúži pre prácu s cache balíček tried `CacheControl`. Kontrola pamäti sa vzťahuje vždy ku konkrétnej dátovej kocke, s ktorou sa aktuálne pracuje. Z hľadiska výkonu je to výhodné riešenie, pretože nie je potrebné zakaždým načítavať do pamäti všetky použité dátové kocky. Programátor si zvolí len tú, v ktorej chce obnoviť zobrazované dáta.

Druhá úroveň, na ktorej sa rieši problém neaktuálnych dát je volanie servera Mondrian priamo z knižnice JPivot. V tomto prípade sa to rieši špeciálnou triedou, ktorá sa nastavuje ako vstupný parameter do značky na vykreslenie kontingenčnej tabuľky. Táto trieda implementuje dve metódy. Jedna informuje o zmene v hierarchii zobrazenia a druhá o zmene v agregovaných hodnotách.

Výstupom obidvoch metód je booleovská hodnota, ktorá hovorí o tom či sa daný typ dát zmenil alebo nie. V aplikácií tieto metódy vracajú štandardne kladnú hodnotu, dáta sú tak zakaždým načítavané pri inicializácií tabuľky, viac o správe pamäti na [20].

Úpravy OLAP modulu podľa požiadaviek manažérov

Požiadavky manažérov sa môžu v priebehu používania CRM systému v reálnom prostredí často meniť. Je potrebné, aby systém umožňoval splniť tieto požiadavky rýchlo a efektívne, to znamená bez zásahu do logiky aplikácie. Úpravy analytického modulu CRM systému by mohli byť nasledovné.

V prípade, že manažér požaduje len jednoduchú upravu v rámci existujúceho dátového skladu, ktorá spočíva v úprave dimenzií a iných požiadaviek týkajúcich sa prezentácie dátového skladu, knižnica JPivot (pozri 7.1.2) obsahuje nástroje, ktoré jednoduchým a prehľadným spôsobom umožňujú nakonfigurovať dátovú kocku. Dotazy nad multimediálnou databázou sú uložené v oddelených súboroch a načítavané pomocou parametra v URL adrese. Je preto jednoduché vytvoriť nový preddefinovaný dotaz presne podľa požiadaviek manažéra, pridať ho do adresára pre dotazy a v aplikácií na nový dotaz jednoducho odkázať.

Ďalšia možnosť úprav sa týka štruktúry dátového skladu v rámci jednej relačnej databázy, to znamená aj jedného konfiguračného súboru. V tomto prípade sa môže jednať o vytvorenie novej dimenzie do dátovej kocky alebo vytvorenie úplne novej dátovej kocky. Riešením je doplnenie vrstvy pohľadov nad databázou a následné doplnenie konfiguračného súboru. Na zmeny v dátovom sklade aplikácia reaguje automaticky.

Najzložitejším zásahom by sa mohla zdať požiadavka na vytvorenie dátového skladu z nového zdroja dát. Zdrojom dát môže byť napríklad dodaná externá databáza. Aj v tomto prípade je riešenie jednoduché. Použitá knižnica JPivot (pozri 7.1.2) umožňuje zadať pri nastavovaní parametrov MDX dotazov cestu k súboru, v ktorom sa nachádza XML súbor s logickým modelom dátového skladu. Ďalším parametrom je cesta k použitej relačnej databáze s fyzickým modelom dátového skladu. Pridanie nového dátového skladu teda spočíva vo vytvorení fyzického a logického modelu, nastavenia parametrov v preddefinovaných súboroch a nakopírovanie týchto súborov do adresára aplikácie. Dátový sklad je následne pripravený k použitiu. Knižnica JPivot umožňuje nastaviť ako zdroj dát webové služby, OLAP server tak môže byť nainštalovaný aj mimo aplikáciu. V tomto prípade je potrebné vytvoriť len preddefinované dotazy, fyzický a logický model je uložený externe.

7.2 WEKA

Weka je kolekciou algoritmov strojového učenia pre úlohy získavania znalostí. Algoritmy môžu byť aplikované priamo na dátový súbor, alebo môžu byť používané z kódu Java aplikácie. Weka obsahuje nástroje pre predspracovanie dát, klasifikáciu, regresiu, zhlukovanie, asociačné pravidlá a vizualizácie. Je to tiež vhodný nástroj pre vývoj nových metód strojového učenia [14].

Zdrojový kód Weka je otvorený, voľne dostupný. Weka je poskytovaná v rôznych prostrediach, ako konzolová aplikácia pre priame vykonávanie Weka príkazov, prieskumníka pre skúmanie dát a Weka experimentátora pre vykonávanie experimentov a štatistických testov medzi rôznymi algoritmami [14].

7.2.1 Napojenie na aplikáciu CRMminer

Na webových stránkach venovaných projektu je k dispozícií niekoľko verzií Weky, ktoré je možné stiahnuť. Weka je prioritne distribuovaná ako aplikácia s vlastným užívateľským rozhraním. Ďalej existuje verzia, ktorá je spúšťaná z príkazového riadku. Nastavenia pre jednotlivé úlohy sú v tomto prípade nadefinované v podobe parametrov. Rôzne verzie Weky vyžadujú inú verziu prostredia Javy na hostiteľskom počítači. V prípade použitia inej verzie môžu nastať problémy s výsledným prekladom aplikácie.

Pre potreby aplikácie CRMminer bola použitá verzia *3.4.18*, ktorá je spúšťaná cez príkazový riadok ako konzolová aplikácia. Jedná sa o stabilnú verziu, je zdokumentovaná aj v manuálovej knihe na webových stránkach [14]. Inštalácia spočívala vo vytvorení modulov na aplikačnej vrstve aplikácie, to znamená na rovnakej úrovni na akej pracuje analytický server Mondrian. Weka obsahuje rôzne metódy klasifikácie a predikcie, ich zakomponovanie do aplikácie zahrňuje vytvorenie tried, ktoré implementujú spoločné rozhranie definujúce formát výstupu dát. Výsledky dolovania znalostí sú potom ďalej spracovávané na aplikačnej vrstve pomocou manažérov, ktoré získané informácie upravia do vhodnej podoby, poprípade skombinujú s dátami z relačnej databázy alebo OLAP serveru Mondrian.

7.2.2 Zdroj dát pre dolovanie

Zdrojom dát pre dolovanie môže byť dátový súbor alebo databáza. V prípade dátového súboru sa jedna o takzvané *ARFF* (*Attribute-Relation File Format*) súbory. Hlavička súboru ARFF obsahuje názov relácie a zoznam atribútov. Pri každom z atribútov je uvedený jeho dátový typ. Posledný atribút je označený návestím *class*. Tento atribút je cieľový, jeho hodnoty budú tvoriť triedy pre klasifikáciu. Ak nie je cieľový atribút v hlavičke uvedený, je potrebné ho nastavit pomocou WEKA API pri vytváraní modelu. Pod hlavičkou sú uvedené samotné dáta pre dolovanie. Jeden záznam predstavuje jeden riadok, položky atribútov sú oddelené čiarkami. Nasledujúci blok obsahuje ukážku obsahu súboru ARFF.

Databáza je ďalším možným zdrojom dát pre získavanie znalostí. V prípade databázy je konfigurácia o niečo zložitejšia. Je potrebné stiahnuť konfiguračný súbor pre konkrétny typ použitej databázy. V tomto súbore je potrebné nastaviť ovládač pre komunikáciu s databázou a cestu pre spojenie s databázovým serverom. Mapovanie atribútov a ich dátových typov prebieha automaticky pri importe dát.

Novšie verzie weky (3.5.5 +) umožňujú prevádzať konverziu dát z iných súborových formátov, ako napríklad CSV, do formátu ARFF.

Príklad 7.4: Ukážku formátu súboru ARFF.

```
@RELATION ukazka

@ATTRIBUTE atribut1 REAL
@ATTRIBUTE atribut2 REAL
@ATTRIBUTE atribut3 REAL
@ATTRIBUTE atribut4 REAL
@ATTRIBUTE class  {class1,class2,class3}

@DATA
5.1,3.5,1.4,0.2,class1
4.9,3.0,1.4,0.2,class1
...
```

7.2.3 Ukážka postup pri dolovaní

Najpoužívanejšie súčasti WEKA API rozhrania pri procese získavania znalostí pomocou WEKA sú nasledovné [14]:

- *Inštancie* – predstavujú vstupné dáta pre dolovanie.
- *Filtre* – slúžia pri predspracovaní dát.
- *Klasifikátory a zhlukovanie* – modely, postavené nad predspracovanými dátami.
- *Vyhodnotenie* – vyhodnotenie výsledkov, porovnanie modelov.
- *Výber atribútov* – odstránenie nerelevantných a zbytočných atribútov zo vstupných dát.

Inštancie sú načítavané z ARFF súboru alebo z databázy, viď odsek 7.2.2. Vstupné dáta sú upravené do podobny vhodnej pre použitie v ďalších moduloch.

Filtre pre predspracovanie dát majú dve rozličné nastavenia. Prvým nastavením je filtrovanie s dohľadom alebo bez dohľadu. V tomto nastavení ide o to, či má pri úprave dát vziať do úvahy cieľový atribút alebo nie. Druhým nastavením je filtrovanie na základe inštancií alebo na základe atribútov. V tomto prípade sú odstránené atribúty alebo celé inštancie na základe nejakých vlastností.

WEKA obsahuje niekoľko modelov pre klasikátory a zhlukovanie. Pri vytváraní modelu stači zadať cestu, v ktorom balíku sa klasifikátor nachádza. Následne na základe vyfiltrovaných dát bude klasifikátor vytvorený. Klasifikátor je možné úložiť do súboru pomocou serializácie. Vytvorený klasifikátor je možné zobraziť na štandardný výstup, v prípade rozhodovacieho stromu je výstup nasledovný:

Na uvedenom výstupe je zobrazený výsledný model rozhodovacieho stromu, ktorý bol vytvorený na základe vstupných dát. Tieto testovacie dáta obsahujú databázu rastlín, úlohou bolo na základe ich rozmerov správne rozhodnúť o akú rastlinu sa jedná. V strome je viditeľné, že atribút s najväčšou rozlišovacou schopnosťou bola šírka listu (v strome „petal width“).

Príklad 7.5: Ukážka modelu rozhodovacieho stromu, spracované pomocou [14].

```
petalwidth <= 0.6: Iris-setosa (50.0)
petalwidth > 0.6
|   petalwidth <= 1.7
|   |   petallength <= 4.9: Iris-versicolor (48.0/1.0)
|   |   petallength > 4.9
|   |   |   petalwidth <= 1.5: Iris-virginica (3.0)
|   |   |   petalwidth > 1.5: Iris-versicolor (3.0/1.0)
|   petalwidth > 1.7: Iris-virginica (46.0/1.0)
```

Príklad 7.6: Výsledok vyhodnotenia modelu pomocou krížovej validácie, spracované pomocou [14].

```
  a  b  c   <-- classified as
 49  1  0 |  a = Iris-setosa
  0 46  4 |  b = Iris-versicolor
  0  3 47 |  c = Iris-virginica
```

Vyhodnotenie modelu prebieha pomocou krížovej validácie. Výsledky je možné zobraziť v rôznom formáte ako textový výstup, v prípade spomenutého modelu pre klasifikáciu rastlín sú výsledky nasledovné:

Druhým spôsobom zhodnotenia modelu je použitie trénovacej množiny dát pre vytvorenie modelu a testovacej množiny pre jeho zhodnotenie. Vytvorený model podrobíme testom na testovacích dátach a výsledky podobným spôsobom vypíšeme ako textový výstup.

Samotná klasifikácia inštancií pri reálnom nasadení v aplikácií prebieha nasledovne. Vytvorený klasifikátor je načítaný zo súboru. Objekt, ktorý chceme zaraďovať do tried, je potrebné upraviť do tvaru inštancie, to znamená jeho atribúty vhodne upraviť do takého formátu, na akých dátach bol klasifikátor vytváraný. Výstupom metódy pre klasifikáciu je číselná hodnota, ktorá reprezentuje triedu, kde daný príklad zaradíme. V prípade, že cieľový atribút tvoria kategorické (textové) atribúty, ako v hore uvedenom príklade, tieto budú reprezentované číslami od nuly vyššie. V uvedenom príklade to budú hodnoty `@ATTRIBUTE class {0.0,1.0,2.0}`. Prezentovanie získaných výsledkov je už potom závislé od konkrétnej úlohy, viac na [14].

7.3 Demonštrácia funkčnosti modulu pre podporu rozhodovania

Aplikácia CRMminer bola skúšobne nasadená v spoločnosti Actimmy a.s., ktorá je dcérskou spoločnosťou GC system a.s. Spoločnosť Actimmy sa zaoberá predajom hardvérových komponentov do počítačov a vývojom informačných systémov s primárnym zameraním na webové aplikácie. Mnoho projektov, na ktorých spoločnosť pracovala a ktoré má rozpracované alebo sa stará o ich podporu, sú aplikácie robené na mieru. Pred zahájením projektu sa vypracuje projektový plán zahrňujúci aj časový odhad vývoja projektu, od ktorého závisí odhad ceny. Na týchto aplikáciach zvyčajne pracuje viacero ľudí. Pred vypracovaním plánu je vhodné vedieť, ako prebiehali v minulosti práce na projekte s podobným zameraním a ak

už so zákazníkom, ktorý si aplikáciu objednáva, spoločnosť pracovala, je vhodné si prejsť záznamy práce a pozrieť sa na prehľady z rôznych aspektov. Práve v tejto oblasti bude aplikovaný modul pre podporu rozhodovania.

Stav pre nasadením aplikácie CRMminer z hľadiska ukladania informácií o zákazníkoch a projektoch bol nasledovný:

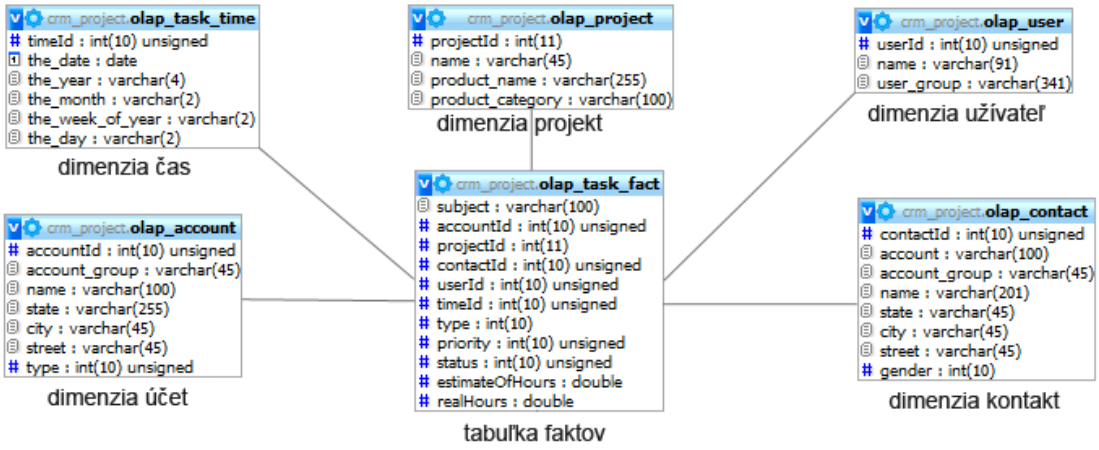
- jediné prehľady, ktoré existovali o práce na projekte boli uložené v textových dokumentoch pre jednotlivé projekty,
- e-mailová komunikácia nebola centralizovaná,
- neexistoval jednotný dátový sklad, kde by slúžil na sumarizácie a porovnávanie.

Z možných modulov CRM systému boli pre Actimmy použité nasledovné moduly, pre presnejší popis modulov viď sekciu 6.5:

- Účty – správa účtov firiem a osôb, z ktorými ma Actimmy vzťah, primárne sa jedná o zákazníkov spoločnosti. Účet je v hierarchií umiestnený najvyššie.
- Kontakty – kontakty, ktoré sa vzťahujú ku účtu.
- Projekty – modul pre správu projektov, sklad dokumentov k projektu, používaný primárne pre aplikácie na zákazku za účelom získania prehľadov.
- Úlohy – modul pre zadávania úloh, tieto sa môžu viazať na účty, kontakty alebo projekty.
- Udalosti – kalendár udalostí pre účty, kontakty, projekty.
- Správa užívateľov – správa užívateľov systému.
- Monitoring a analýzy – analytický modul pre prehľady.

Modul pre podporu rozhodovania sa skladá z OLAP analýz a modulu pre získavanie znalostí. V spoločnosti Actimmy je v skúšobnej prevádzke použitý prvý z modulov. Jednou z úloh je získanie prehľadov o odhadoch a skutočných nákladoch v rámci jednotlivých účtov, čo zahrňuje odhady pre kontakty a projekty. Dátový sklad, ktorý je zdrojom dát pre túto úlohu, je na obrázku 7.5. Obsahuje jednu tabuľku faktov (`olap_task_fact`), v ktorej sú záznamy o odhadnutých a skutočne strávených hodinách na projekte. Úloha sa môže viazať na projekt (tabuľka `olap_project`), účet (tabuľka `olap_account`) alebo priamy kontakt (tabuľka `olap_contact`). Každá úloha má svojho zadávateľa, tabuľka (`olap_user`). Všetky tieto spomenuté moduly tvoria tabuľky dimenzií. Časová dimenzia je tvorená tabuľkou (`olap_task_time`). Tabuľka faktov aj tabuľky dimenzií sú v relačnej databáze tvorené vrstvou pohľadov. Takto vytvorená dátová kocka umožňuje vytváranie pohľadov na dáta v rámci uvedených dimenzií a ich kombinácií. V praxi to prináša ucelený prehľad na vynaložené úsilie, ktoré je vkladané spoločnosťou do spolupráce so zákazníkom. Ako príklad využitia takto získaných znalostí by mohol byť prípad, keď je z analýz zrejmé, že vo všetkých projektoch pre určitého zákazníka, na ktorých sa spoločnosť podielala, bol prekročený rozpočet projektu. Takýto zákazník je zjavne rizikový a spolupráca s ním je problematická. Preto je potrebné pri ďalšej spolupráci počítať s väčšou rezervou na rozpočet projektov.

Na obrázku 7.6 sa nachádza ukážka zobrazenia výsledkov OLAP dotazovania v aplikácií CRM miner pomocou kontingenčnej tabuľky.



Obrázek 7.5: Ukážka dátovej kocky pre analýzu úloh z dátového skladu CRMminer.

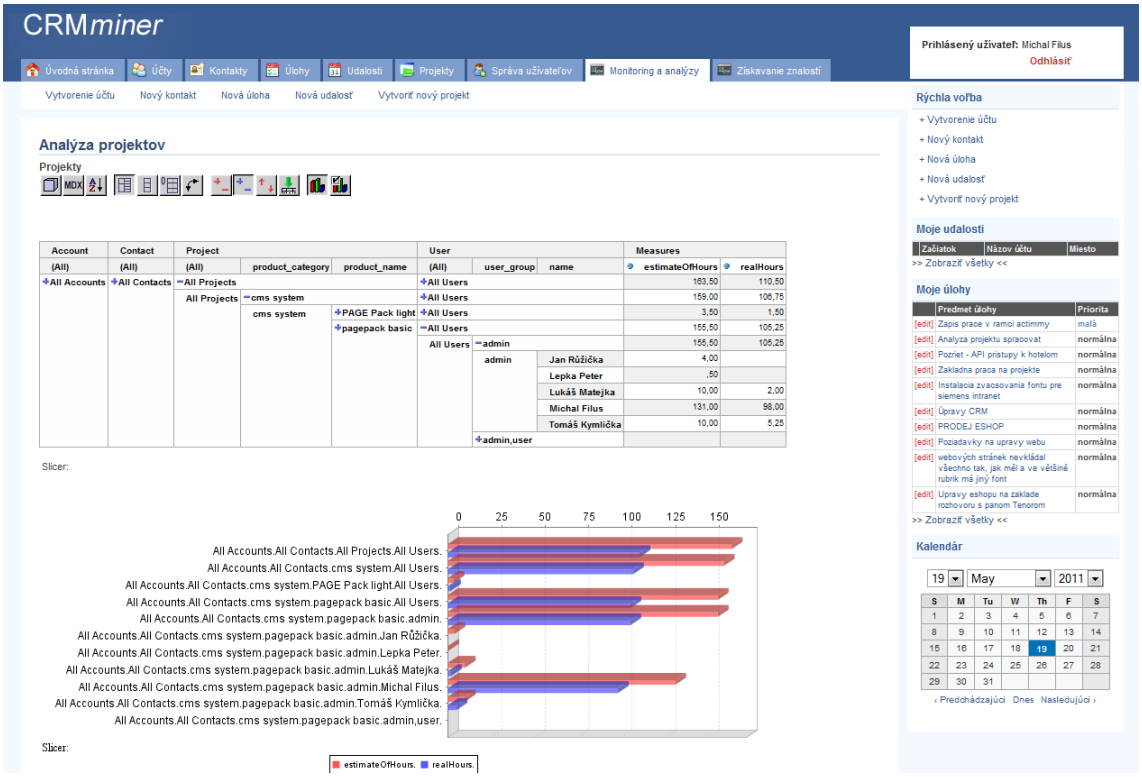
Pre overenie správnej integrácie modulu pre získavanie znalostí WEKA bol do aplikácie pridaný jednoduchý modul pre predikciu dát pomocou metódy rozhodovacích stromov. Vstup modulu je tvorený ARFF súborom s trénovacími dátami. Po uložení sa súbor nahrá do adresára aplikácie, následne sa načíta a spracuje pomocou uvedených funkcií WEKA API. Výsledný strom a zhodnotenie modulu sú následne zobrazované na webovej stránke.

Nasadenie modulu pre aplikáciu získavania znalostí v reálnom prostredí sa plánuje v ďalšej fáze vývoja projektu. V čase písania tejto diplomovej práce nebolo k dispozícii dostatočné množstvo dát pre získanie relevantných výsledkov. Modul získavania znalostí bude zaisťovať rozšírenie do systémov elektronického obchodu o možnosti segmentácie zákazníkov a následnú optimalizáciu eshopu. Operačná časť sa tak presunie do administrácie obchodu a analytickú časť bude zaisťovať CRM systém.

7.4 Zhodnotenie vlastností modulu

Analytický modul v aplikácií CRMminer pozostáva so štandarných SQL dotazov nad relačnou databázou, kde sú využívané agregačné funkcie a výsledky týchto dotazov sú zobrazované v grafoch. Použitím rámca Hibernate sa dosiahla nezávislosť aplikácie na použitom relačnom databázovom systéme. Pri zmene databázy tak nie je potrebné dotazy upravovať.

Hlavnou časťou analytického modulu je použitie relačného OLAP servera Mondrian, ktorý prioritne používa databázu aplikácie ako zdroj svojich dát a pracuje s vrstvou pohľadou nad touto databázou. Neexistuje žiadne obmedzenie v napojení na relačnú databázu, ktorá je zdrojom fyzického modelu dátového skladu. Ak je databáza podporovaná serverom Mondrian, stačí nadefinovať súbor s logickým modelom skladu. Pri zmene typu databázy je potrebné vytvoriť vrstvu pohľadov nad databázou, ktoré reflektujú logický model. V tej chvíli je aplikácia schopná spracovávať a zobrazovať dáta z tohto databázového zdroja. Aplikácia umožňuje zobrazovať výsledky na preddefinované dotazy pomocou knižnice pre vykresľovanie grafov. Tento spôsob slúži na prehľadné a rýchle zobrazenie informácií z oblasti jednotlivých modulov. V prípade, že je potrebné s dátami ďalej pracovať a získavať tak podrobnejšie informácie je k dispozícii knižnica na vykresľovanie pomocou kontingenčnej tabuľky JPivot, ktorá obsahuje nástroje na manipuláciu s dátami.



Obrázek 7.6: Ukážka použitia modulu pre podporu rozhodovania v CRMminer.

V prípadoch, že nie je umožnený priamy prístup do databázy, je aplikácia schopná získavať dáta pomocou webových služieb a tieto zobrazovať do grafov a aj v spomínaných kontingenčných tabuľkách.

Ďalší vývoj modulu a udržovanie jeho aktuálnych súčastí je vďaka dobrému návrhu architektúry jednoduché. Neexistuje žiadna priama závislosť medzi jednotlivými vrstvami. Novo pridané moduly len musia implementovať dané rozhrania. Rámec Spring umožňuje ich dynamické načítavanie a pomocou vzoru inverzie kontroly odstraňuje uvedené závislosti medzi modulmi.

Z pohľadu zákazníka je analytický modul navrhnutý tak, aby existovala jednoduchá možnosť ako ho rozširovať a upravovať. Celá aplikácia bola navrhovaná s tým, že požiadavky konečných zákazníkov môžu byť rôznorodé a aplikácia musí byť ľahko prispôsobiteľná. Vrstvená architektúra a modularita umožňuje tieto požiadavky splniť.

7.5 Návrh na možné rozšírenia modulu

V aktuálnej verzii systému pracujú modul pre OLAP a modul pre získavanie znalostí nezávisle na sebe. Z hľadiska prípravy dát to znamená, že tento proces je potrebné robiť osobitne pre Mondrian a osobitne pre WEKA. Zlepšenie efektivity a modulárnosti by prinieslo napojenie WEKA na výstup Mondriana. Vstupné dáta pre proces získavania znalostí by sa tak získavali z dátového skladu, ktorý obsahuje dáta v správnom formáte. Toto rozšírenie by zasahovalo do oblasti OLAM systémov, ktoré sú kombináciou OLAP a získavania znalostí, viac v kapitole 3.6.

Ďalšie možnosti rozšírenia modulu sa vzťahujú hlavne k oblasti získavania znalostí a k jeho uplatneniu pri konkrétnej implementácií CRM systému u zákazníka. Konkrétne sa jedná o využitie modulu pri segmentácií zákazníkov elektronických obchodov. V aktuálnej dobe prebieha zber dát u niekoľkých klientov. Vytvára sa dátový sklad, kde sú ukladané všetky dostupné infomácie o pohybe zákazníkov na webových stránkach obchodu. Pohyb je zaznamenaný od úvodnej stránky až po prípadné vytvorenie objednávky. Okrem pohybu sa do databázy zaznamenávajú napríklad aj reakcie na rôzne reklamné kampane. Takto je možné získať o zákazníkoch množstvo užitočných informácií, tieto sa následne budú spracovávať pomocou nástroja WEKA. Výsledné znalosti budú potom zhodnocované napríklad pomocou zasielania reklamných emailov s ponukami na produkty. Ponuka bude prispôsobená na konkrétneho zákazníka na základe jeho správneho zaradenia do určitej skupiny.

Súčasti modulu pre získavanie znalostí implementujú jednotné rozhranie. V prípade, že by z WEKA z nejakých dôvodov nebola vhodná pre použitie v danej dolovacej úlohe, je možné jednoducho integrovať a rozšíriť tak modul o iný typ nástroja pre získavanie znalostí, ktorý bude implementovať nadefinované rozhranie. Prezentácia získaných výsledkov je potom zhodná s WEKA.

Ďalším užitočným rozšírením by bolo napojenie modulu na rôzne systémy účtovníctva za účelom podpory predaja, alebo na systémy, ktoré sú určené pre správu softvérových projektov.

Kapitola 8

Záver

Diplomová práca sa zaoberá problematikou aplikácie podpory rozhodovania v CRM systémoch. Prvá časť práce obsahuje stručné oboznámenie s problematikou analýzy dát, procesom získavania znalostí. Vysleťuje pojem CRM systémov a aplikovanie podpory rozhodovania v týchto systémoch.

V druhej časti práce je na základe získaných teoretických znalostí popísaný návrh CRM systému so zameraním na návrh modulu pre podporu rozhodovania v CRM aplikácii. Systém ako celok bol vytváraný v súlade s modernými trendami pre vývoj webových aplikácií. Pri vývoji bol kladený veľký dôraz na modulárnosť a celkovú univerzálnosť aplikácie. Pri implementácií boli v maximálnej možnej miere použíté voľne dostupné technológie. Samotný modul pre rozhodovanie je tvorený pomocou nástrojov Mondrian a WEKA. Pri každom z nich je uvedený spôsob integrácie do aplikácie a akým spôsobom sa prezentujú získané výsledky koncovým používateľom systému.

Testovanie CRM systému bolo riešené reálnym nasadením v spoločnosti Actimmy a.s., kde sa modul pre podporu rozhodovania využíva pri analýzach spojených s vývojom projektov na mieru. Kokrétne použitie súvisí s odhadom rozpočtu pre jednotlivé projekty a správu úloh, ktoré sa ku nim viažu. Pomocou OLAP servera Mondrian je potom možné nad vytvoreným dátovým skladom získavať pohľady na stav vývoja projektu z rôznych aspektov.

Časť modulu, ktorá sa zaoberá technikami získavania znalostí, sa nepodarilo v praxi otestovať z dôvodu nedostatku relevantných dát. WEKA je časom overený nástroj, ktorý pomocou API rozhrania jasne definuje proces dolovania dát. Pri integrácií do aplikácie tieto postupy ostali zachované. Preto je otázka funkčnosti modulu zameraná skôr na vhodnú dolovaciu úlohu a kvalitu vstupných dát, ako na overenie funkčnosti konkrétnych metód. Pre základnú demonštráciu funkčnosti WEKA bol v CRM systéme vytvorený modul pre vytvorenie rozhodovacieho stromu na základe vstupných dát zo súboru.

Počas vývoja CRM systému boli jednotlivé moduly konzultované s vedením a zamestnancami spoločnosti. Všetky pripomienky, ktoré som počas konzultácií obdržal, boli do systému implementované.

Ďalší vývoj CRM systému ako celku bude spočívať v napojení na systémy pre elektronický obchod. V tejto oblasti je plánované použitie techniky získavania znalostí pre segmentáciu zákazníkov. Získané výsledky budú následne použité napríklad pre zasielanie reklamných emailov na určité skupiny produktov.

Vytváranie tejto diplomovej práce malo pre mňa veľký osobný prínos. Prehĺbil som si svoje znalosti z oblasti podpory rozhodovania. V praxi som mohol aplikovať vedomosti z oblasti návrhu informačných systémov, ktoré som získal počas štúdia a riešiť problémy

pri integrácií rôznych systémov do jedného celku. Najviac si cením znalosti, ktoré som získal pri vývoji aplikácií pod platformou Java EE. Rád by som sa jej aj v budúcnosti venoval.

Literatura

[1] *The building of Customer Relationship Management system based on OLAP*, New York: IEEE, máj 2010, ISBN 978-1-4244-7653-4.

[2] *SQL Views*. [online], [citované 18.4.2011].
URL http://www.sql-tutorial.com/sql-views-sql-tutorial/

[3] Apache: *Apache Maven Project*. [online], [citované 6.4.2011].
URL http://maven.apache.org/

[4] BERKA, P.: *Dobývání znalostí z databází*. Praha: Academia, 2003, ISBN 80-200-1062-9.

[5] CHLEBOVSKÝ, V.: *CRM řízení vztahů se zákazníky*. Brno: Computer Press, 2005, ISBN 80-251-0798-1.

[6] EDELSTEIN, H.: *Building profitable customer relationships with data mining*. online, [citované 17.12.2010].
URL http://www.pse.pt/Documentos/profitable%20relationships.pdf

[7] HALL, M.: *Java servlety a stránky JSP*. Praha: Neocortex, 2003, ISBN 80-86330-06-0.

[8] HAN, J.; KAMBER, M.: *Data Mining: Concepts and Techniques*. San Francisco: Morgan Kaufmann Publishers, 2006, ISBN 1-55860-901-6.

[9] InfoSoft Global: *FusionCharts Free*. [online], [citované 18.4.2011].
URL http://www.fusioncharts.com/free/

[10] JBoss Community: *Hibernate Reference Documentation*. [online], [citované 13.4.2011].
URL http://docs.jboss.org/ejb3/app-server/Hibernate3/reference/en/html_single/

[11] KASPŘÍKOVÁ, N.: *Data mining pro řízení vztahů se zákazníky*. [online], [citované 17.12.2010].
URL http://data.tulipany.cz/dataminingCRM.php

[12] KOSTÍK, L.; SALOKY, T.: Niektoré z problémov pri získavaní dát pomocou rozhodovacích stromov. *AT&P journal*, ročník 10, č. 2, november 2006.
URL http://www.atpjournal.sk/buxus/docs//casopisy/atp_plus/plus_2006_2/plus55_57.pdf

[13] LACKO, L.: *Databáze, datové sklady, analýza OLAP a dolování dat s příklady v Microsoft SQL Serveru a Oracle*. Brno: Computer Press, 2003, ISBN 80-7226-969-0.

[14] Machine Learning Group at University of Waikato: *Weka 3: Data Mining Software in Java*. [online], [citované 15.4.2011].
URL http://www.cs.waikato.ac.nz/ml/weka/

[15] MoroSystems: *Using Sitemesh in projects based on MVC frameworks*. [online], [citované 10.3.2011].
URL http://vsadnajavu.cz/2006-11/java-j2ee/using-sitemesh-in-projects-based-on-mvc-frameworks/

[16] MoroSystems: *Moderní JEE^{TM} technologie a nástroje*. [online], [citované 11.4.2011].
URL http://morosystems.cz/java/

[17] OpenSymphony: *SiteMesh - SiteMesh Overview*. [online], [citované 12.4.2011].
URL http://www.opensymphony.com/sitemesh/

[18] Oracle: *Annotations*. [online], [citované 13.4.2011].
URL http://download.oracle.com/javase/tutorial/java/javaOO/annotations.html

[19] Oracle: *Getting Started With Java Data Objects (JDO)*. [online], [citované 13.4.2011].
URL http://docs.jboss.org/ejb3/app-server/Hibernate3/reference/en/html_single/

[20] Pentaho Analysis: *Mondrian Documentation*. [online], [citované 17.4.2011].
URL http://mondrian.pentaho.com/documentation/

[21] SAXENA, S.: *CRM Analytics KEY to ROI on CRM*. [online], [citované 18.12.2010].
URL http://www.realmarket.com/required/supportscape1.pdf

[22] SpringSource: *Spring Framework Reference Documentation*. [online], [citované 20.3.2011].
URL http://static.springsource.org/spring/docs/3.0.5.RELEASE/reference

[23] SpringSource: *Spring Security Reference Documentation*. [online], [citované 20.3.2011].
URL http://static.springsource.org/spring-security/site/docs/3.1.x/reference/springsecurity-single.html

[24] SRIVASTAVA, J.: *Data Mining for Customer Relationship Management (CRM)*. [online], [citované 19.12.2010].
URL http://www.dtc.umn.edu/ddmc/resources/crm.pdf

[25] Tonbeller: *JPivot*. [online], [citované 17.4.2011].
URL http://jpivot.sourceforge.net/index.html

[26] TORGGLER, M.: *The Functionality and Usage of CRM Systems*. [online], [citované 15.12.2010].
URL http://www.waset.org/journals/ijhss/v4/v4-3-22.pdf

[27] ZENDULKA, J.; BARTÍK, V.; LUKÁŠ, R.; aj.: *Získávání znalostí z databází*, 2009/10, prednášky na VUT FIT v Brně.
URL http://www.fit.vutbr.cz/

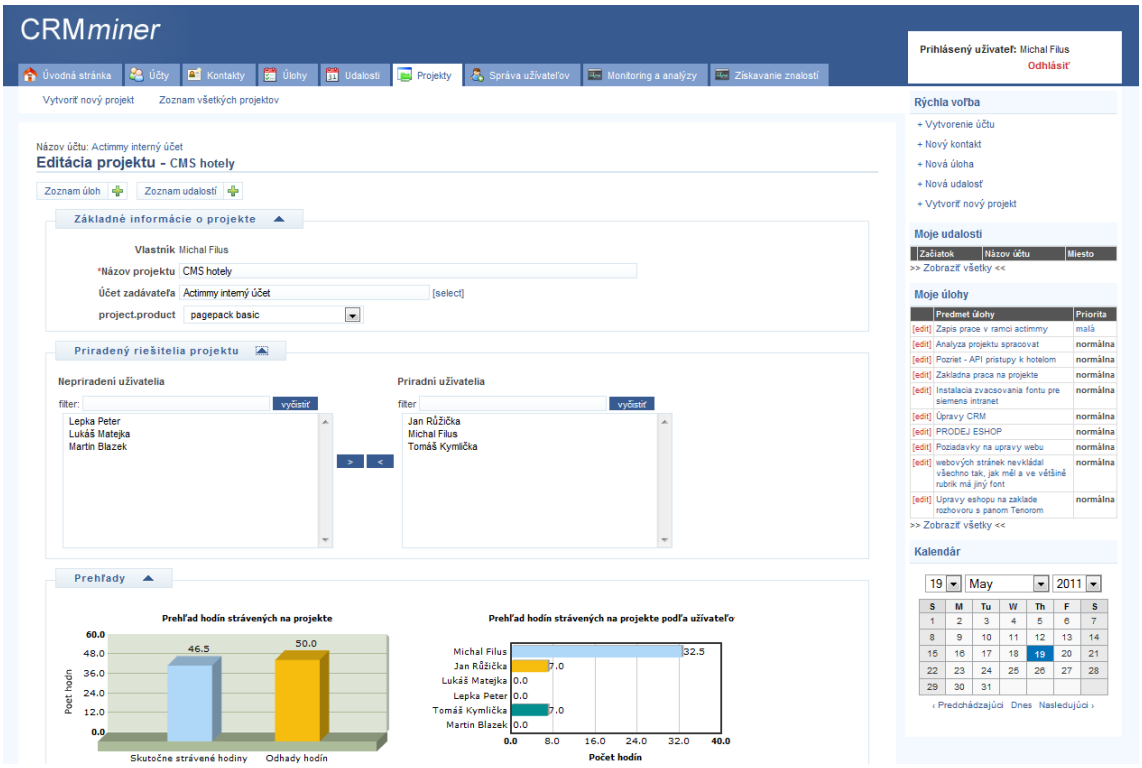
Příloha A

Obsah CD

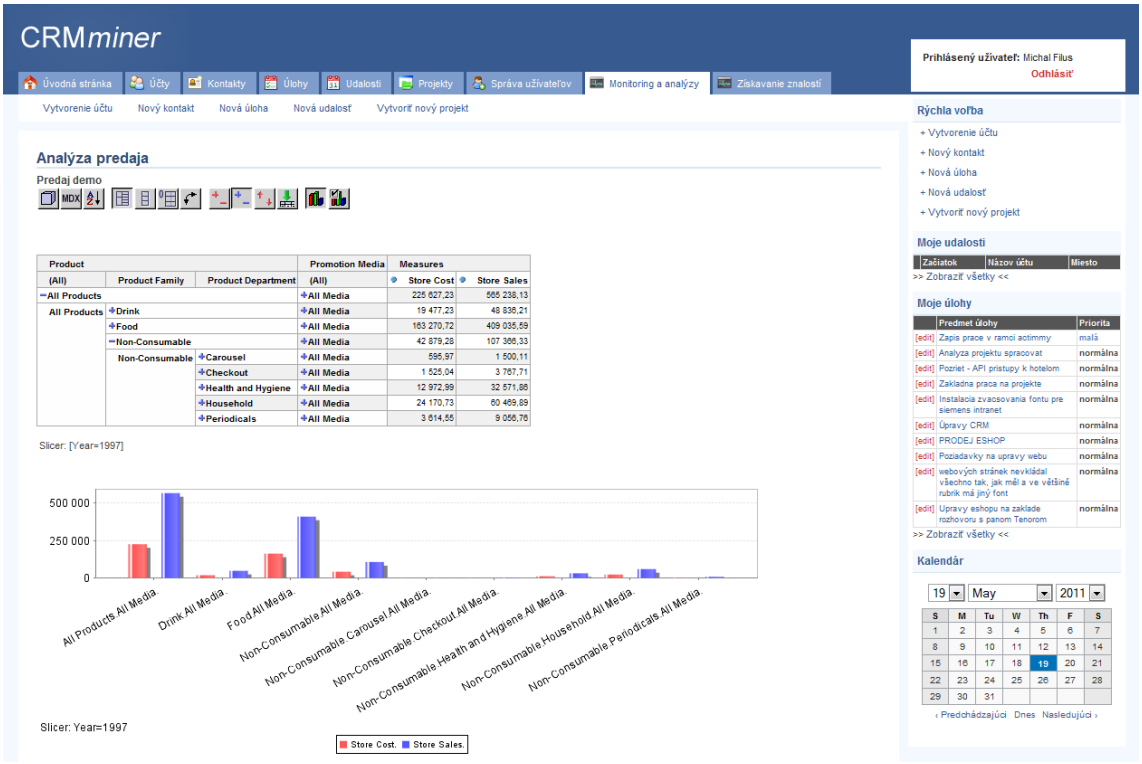
- Zdrojové súbory, knižnice, a iné súbory potrebné pre preklad aplikácie CRMminer.
- Skript pre vytvorenie databázy.
- Programová dokumentácia.
- Popis konfigurácie a inštalácie.
- Tento dokument v elektronickej podobe.

Příloha B

Ukážky systému CRMminer



Obrázek B.1: Ukážka použitia knižnice FussionChart pre zobrazenie prehľadov o úlohách na projekte.



Obrázek B.2: Ukážka použitia knižnice JPivot v kombinácií s knižnicou JFreeChart pre zobrazenie dát.