VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

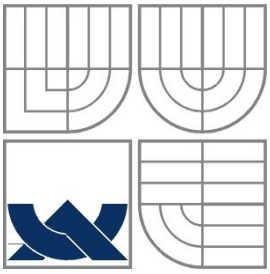
DOLOVÁNÍ ASOCIAČNÍCH PRAVIDEL Z DATOVÝCH SKLADŮ

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

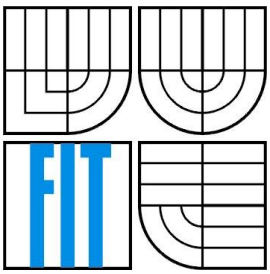
AUTOR PRÁCE
AUTHOR

Bc. LADISLAV HLAVIČKA

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

DOLOVÁNÍ ASOCIAČNÍCH PRAVIDEL Z DATOVÝCH SKLADŮ

ASSOCIATION RULES MINING OVER DATA WAREHOUSES

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. LADISLAV HLAVIČKA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. LUKÁŠ STRYKA

BRNO 2009

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2008/2009

Zadání diplomové práce

Řešitel: **Hlavička Ladislav, Bc.**
Obor: Informační systémy
Téma: **Dolování asociačních pravidel z datových skladů**
Kategorie: Databáze

Pokyny:

1. Seznamte se s problematikou získávání znalostí a datových skladů.
2. Seznamte se s podporou pro vývoj datových skladů v prostředí MS SQL Server nebo Oracle.
3. Navrhněte aplikaci, která bude využívat zvolený algoritmus pro dolování asociačních pravidel z datových skladů.
4. Aplikaci realizujte a její funkčnost ověřte na vhodném vzorku dat.
5. Zhodnoťte dosažené výsledky. Diskutujte další možná rozšíření systému.

Literatura:

- Han, J., Kamber, M.: *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers. 2001.
- Lacko L.: *Business Intelligence v SQL Serveru 2005*. Computer Press a.s. 2005.

Při obhajobě semestrální části diplomového projektu je požadováno:

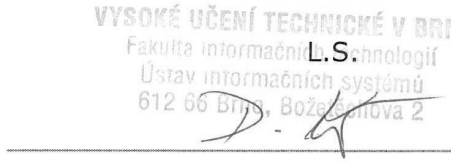
- Body 1 a 2.

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

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci ročníkového a semestrálního projektu (30 až 40% celkového rozsahu technické zprávy).

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

Vedoucí: **Stryka Lukáš, Ing.**, UIFS FIT VUT
Datum zadání: 22. září 2008
Datum odevzdání: 26. května 2009



L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Tato práce se zabývá dolováním asociačních pravidel z datových skladů. V první části bude čtenář obeznámen s pojmy získávání znalostí z databází a dolování dat. Další část práce se zabývá problematikou datových skladů. Dále je popsána samotná asociační analýza, asociační pravidla, jejich typy a dolování. Je představena architektura produktu MS SQL Server a jeho nástroje pro práci s datovými sklady. Zbývající části práce jsou věnovány praktické části. Zahrnují popis a analýzu algoritmu Star-miner, návrh, implementace a testování aplikace.

Abstract

This thesis deals with association rules mining over data warehouses. In the first part the reader will be familiarized with terms like knowledge discovery in databases and data mining. The following part of the work deals with data warehouses. Further the association analysis, the association rules, their types and mining possibilities are described. The architecture of Microsoft SQL Server and its tools for working with data warehouses are presented. The rest of the thesis includes description and analysis of the Star-miner algorithm, design, implementation and testing of the application.

Klíčová slova

Získávání znalostí z databází, dolování dat, datový sklad, ETL, multidimenzionální datový model, frekventovaná množina, asociační pravidlo, MS SQL Server, algoritmus Star-miner

Keywords

Knowledge discovery in databases, data mining, data warehouse, ETL, multidimensional data model, frequent itemset, association rule, MS SQL Server, algorithm Star-miner

Citace

Hlavička Ladislav: Dolování asociačních pravidel z datových skladů, diplomová práce, Brno, FIT VUT v Brně, 2009

Dolování asociačních pravidel z datových skladů

Prohlášení

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

……
Ladislav Hlavička
20.5.2009

Poděkování

Chtěl bych poděkovat panu Ing. Lukášovi Strykovi za jeho pomoc a podporu při řešení této diplomové práce.

Obsah

1 Úvod

V súčasnej dobe sú informačné technológie pre zber a spracovanie údajov nasadené do rôznych odvetví ľudskej činnosti, ako napríklad podnikové oblasti, obchodovanie, výskumné činnosti. Výsledkom toho je zhromaždenie obrovského množstva údajov, ktoré sa ukladajú do databáz. Moderné databázové servery umožňujú nielen bezpečnú a rýchlu prácu s takým množstvom údajov, ale umožňujú aj získať z týchto dát informácie. V databáze sú totiž uložené iba rôzne údaje. Dáta obsahujú iba fakty a napriek tomu, že sa jedná o veľké množstvo dát, niekedy obsahujú iba malé množstvo užitočných informácií. Ďalším problémom môže byť, že užitočné informácie operatívne nie sú k dispozície. Preto je potrebný proces transformácie údajov na informácie a prevod týchto informácií na poznatky prostredníctvom objavovania. Tieto poznatky môžeme potom efektívne využívať napríklad v procese rozhodovania. Dnešné databázové servery obsahujú rozsiahlu podporu pre analytické technológie OLAP, data mining a podporujú vytváranie dátových skladov na ukladanie dát.

V minulosti používané metódy analýzy a podporné nástroje dát založené na štatistike neboli schopné odhaliť požadované znalosti v tak obrovských objemoch dát. Bolo preto potrebné vytvárať nové metódy a algoritmy pre automatizovanú analýzu veľkého množstva dát. Výsledkom bol vznik získavania znalostí z databáz, ktoré v súčasnosti získava stále väčšiu pozornosť v rôznych oblastiach života.

Táto práca sa zaoberá procesom dolovania asociačných pravidiel z dátových skladov. Dolovanie asociačných pravidiel hrá významnú úlohu napríklad v analýze trhu. Získané znalosti je potom možné využívať v procese rozhodovania. V druhej kapitole nájdeme stručný úvod do problematiky získavania znalostí z databáz a podrobnejší popis procesu dolovania dát. Dozvieme sa, aké časti obsahuje architektúra systému pre dolovanie dát a ktoré sú najčastejšie typy dolovacích úloh.

Ďalšia kapitola sa zaoberá dátovými skladmi. Je tu definícia dátového skladu a sú vysvetlené kľúčové vlastnosti dátového skladu. Nájdeme tu popis metód, ktoré môžu byť použité pri budovaní dátového skladu. Pri budovaní dátového skladu hrá veľmi dôležitú úlohu etapa prípravy údajov. Jedná sa o procesy extrakcia, transformácia a presun dát. Okrem týchto etáp je vysvetlený aj pojem multidimenzionálneho dátového modelu, na ktorom je založený celý dátový sklad.

Vo štvrtej kapitole nájdeme asociačnú analýzu. Je tu vysvetlené, čo sú to frekventované množiny a asociačné pravidlá a sú vymenované hlavné typy asociačných pravidiel.

Piata kapitola je venovaná databázovému systému MS SQL Server. Obsahuje popis architektúry celého systému a popis podpory pre vývoj dátových skladov v prostredí tohto systému.

Témou šiestej kapitoly je popis algoritmu Star-miner. Tento algoritmus slúži na dolovanie asociačných pravidiel z dátových skladov a bude implementovaný v navrhnutej aplikácii. Algoritmus Star-miner sa skladá z niekoľko fáz, ktoré sú podrobne popísané.

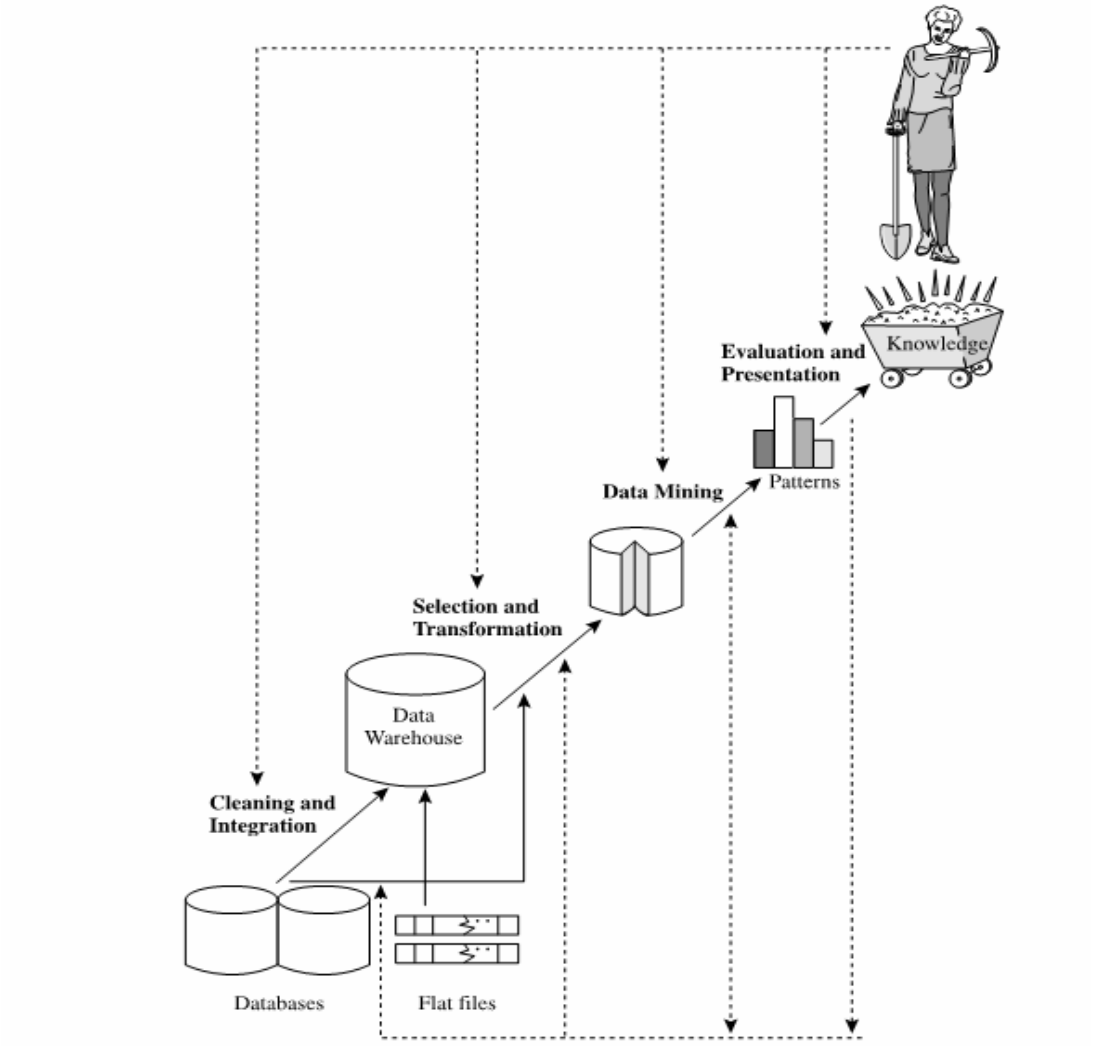
Od siedmej kapitoly nasleduje popis praktickej časti práce. Táto kapitola sa zaoberá analýzou a návrhom databázy a celej aplikácie. Popis samotnej implementácie obsahuje nasledujúca kapitola. Overením funkčnosti algoritmu a testovaním aplikácie sa zaoberá deviata kapitola.

2 Dolovanie dát

Táto kapitola sa zaoberá procesom získavania znalostí z databáz a samotným dolovaním dát. Kapitola obsahuje stručný popis jednotlivých častí procesu získavania znalostí z databáz, ďalej je vysvetlené, ako spolu súvisia pojmy dolovanie dát (data mining) a získavanie znalostí.

2.1 Získavanie znalostí z databáz

Získavanie znalostí z databáz môžeme definovať ako extrakciu zaujímavých modelov dát a vzorov z veľkých objemov dát. Pod slovom „zaujímavých“ rozumieme vzory netriviálne, skryté, predtým neznáme a ktoré sú potenciálne užitočné. Netriviálnosť znamená, že nebudeme hľadať informácie, ktoré by sme mohli získať jednoduchým SQL dotazom, ale je potrebné používať nejaký zložitejší postup. Skrytosť znamená, že máme nájsť také modely a vzory, ktoré na prvý pohľad nevidíme a musíme ich nájsť nejakým netriviálnym spôsobom. A nakoniec pod potenciálnou užitočnosťou rozumieme fakt, že získané znalosti môžeme použiť pre nejaké rozhodnutie [5].



Obr. 2.1: Proces získavania znalostí z databáz [1]

Proces získavania znalostí z databáz obsahuje nasledujúce kroky, ktoré sa môžu iteratívne opakovať [1]:

1. Čistenie dát – odstránenie šumu a nekonzistentných dát.
2. Integrácia dát – v prípade, keď dáta pochádzajú z viacerých dátových zdrojov, je potrebné tieto zdroje nejakým spôsobom spojiť. Dáta viacerých zdrojov sú obvykle uložené do dátového skladu.
3. Výber dát – je potrebné vybrať iba tie dáta, ktoré sú relevantné pre danú analytickú úlohu.
4. Transformácia dát – transformovanie dát do formy, ktorá je vhodná pre dolovanie dát. Môže prebehnúť pomocou agregácie alebo sumarizácie.
5. Dolovanie dát – najdôležitejšia časť, jadro procesu získavania znalostí. Jeho úlohou je extrahovanie vzorov a modelov dát.
6. Hodnotenie vzorov a modelov – identifikovanie skutočne zaujímavých vzorov na základe miery užitočnosti.
7. Prezentácia znalostí – prezentovanie dosahovaného výsledku dolovania užívateľovi pomocou vizualizácie a reprezentácie znalostí.

Prvé 4 kroky sú rôzne formy predspracovania dát, kde dáta sú pripravované na dolovanie dát. V kroku dolovania dát môže dôjsť k interakcii s užívateľom alebo s databázou znalostí. Získané zaujímavé vzory sú potom prezentované užívateľovi a môžu byť uložené ako nové znalosti do databázy znalostí. Ako vidíme, dolovanie dát je iba jedným krokom celého procesu, ale zároveň najdôležitejším, ktorý odhaľuje skryté vzory pre rozhodovanie.

2.2 Data mining

Termín data mining môžeme voľne preložiť ako spôsob získavania, dolovania, odkrývania dát, informácií pre podporu rozhodovania z existujúcich dátových zdrojov. Data mining nám všeobecne môže pomôcť pri identifikácii problémov a identifikácii existujúcich alebo pravdepodobných vzájomných vzťahov medzi jednotlivými entitatmi. Data mining je momentálne jednoznačne najrýchlejšie rastúci segment Business Intelligence a v súčasnosti podobne ako OLAP analýzy majú túto technológiu implementovanú všetky významné komerčne dodávané databázové servery.

Data mining je principiálne založený na heuristických algoritmoch, neurónových sieťach a iných pokročilých softwarových technológiách a metódach umelej inteligencie. Pomáha sledovať a analyzovať trendy a predvídať udalosti. Môže sa využívať v bankovníctve pri analýze a predikcii úverového rizika, predikcii rizika pri vydávaní kreditných kariet, u operátorov telekomunikačných sietí, vo zdravotníctve pre analýzu laboratórnych vzoriek, atď.

Na základe uvedených faktov môžeme uviesť stručnú a jednoduchú definíciu data miningu: *data mining je proces analýzy dát z rôznych perspektív a ich premena na užitočné informácie. Z matematického a štatistického hľadiska ide o hľadanie korelácií, teda vzájomných vzťahov alebo vzorov v údajoch* [4].

Ako reálny príklad na data mining by sme mohli uviesť spoločnosti, ktoré vydávajú kreditné karty. Na základe transakcií, nákupu a iných operácií, ktoré vykonáva majiteľ kreditnej karty, je možno získať veľké množstvo údajov o správaní klienta, ale aj o jeho pohyboch v geografických dimenziách. Postupne by sme tak zrejme mohli zistiť rôzne súvislosti a závislosti a vypracovať akýsi model chovania majiteľa kreditnej karty.

Data mining je často považovaný za získavanie znalostí z databáz, označovaný skratkou KDD (knowledge discovery in databases). Tento pojem vznikol v oblasti umelej inteligencie. Avšak ako sme už v predchádzajúcej podkapitole uviedli, dolovanie dát je iba jedným krokom procesu

získavania znalostí. Oba pojmy sú ale v súčasnosti používané ako synonymá. Proces získavania znalostí vyžaduje niekoľko etáp: výber dát, predpracovanie dát, transformovanie týchto dát, keď je to potrebné, vykonanie dolovania dát pre extrahovanie rôznych vzorov a vzťahov a nakoniec interpretovanie nájdených štruktúr.

Dolovanie dát je analýza súboru dát z dôvodu hľadania doposiaľ neznámych vzťahov a zhrnutie rôznych dát, ktoré sú pochopiteľné a užitočné pre užívateľa. Vzťahy získané pomocou data miningu sú často nazývané ako modely či vzory. Dolovanie dát typicky zvláda dáta, ktoré vznikajú zhromaždením dát na základe rôznych kritérií. To znamená, že objekty data miningu nehrajú úlohu v stratégii zhromaždenia dát. Táto vlastnosť odlišuje dolovanie dát od ostatných štatistických metód, v ktorých sú dáta zhromaždené na základe výkonných, účinných stratégií, ktoré slúžia na odpovedanie špecifických dotazov. Z tohto dôvodu je dolovanie dát často nazývané ako sekundárna dátová analýza [2].

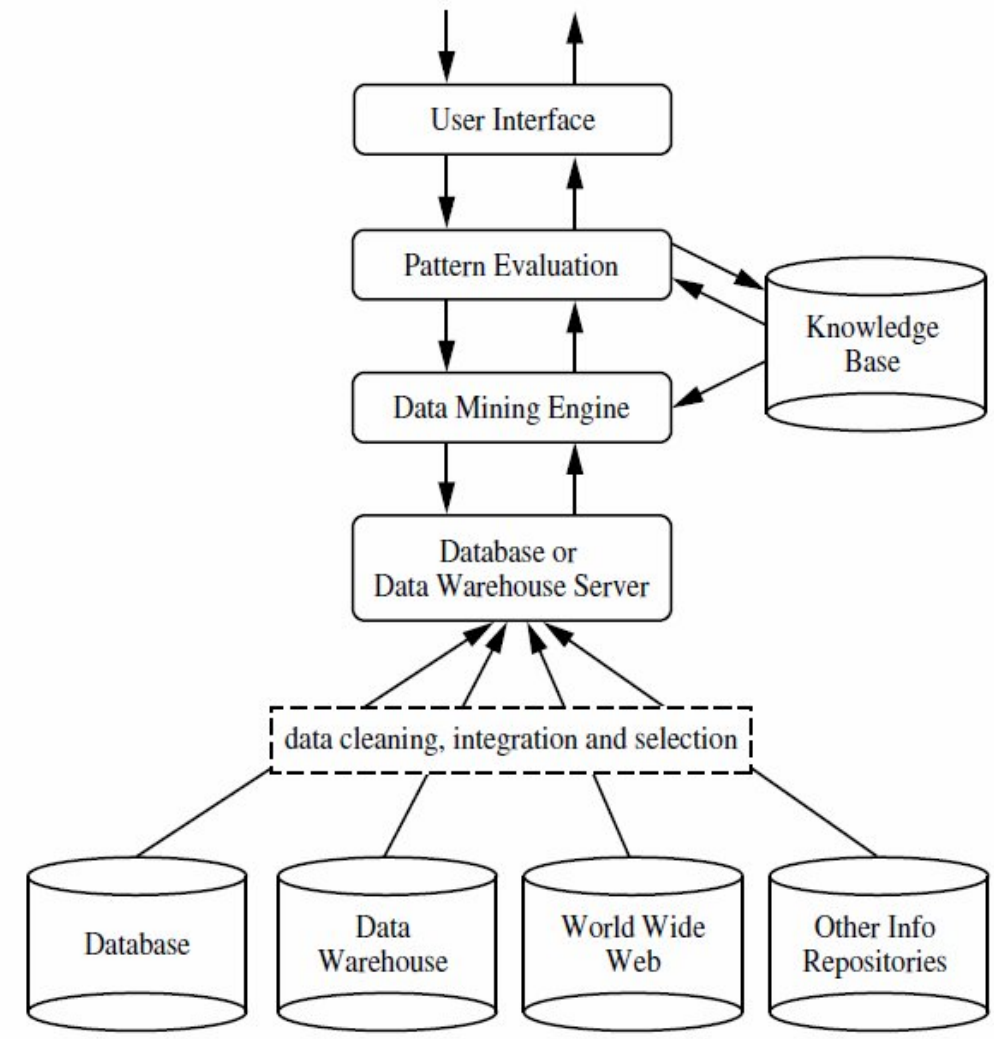
Proces získania vzťahov zo súboru dát vyžaduje niekoľko dôležitých krokov:

- stanovenie charakteru a štruktúry vyjadrenia,
- rozhodnutie, ako kvantifikovať a porovnať odlišné reprezentácie dát,
- výber algoritmického procesu na optimalizáciu,
- rozhodnutie, aké princípy manažmentu dát sú požadované pre implementáciu účinného algoritmu.

Pri dolovaní dát v úvodnej fáze obvykle na základe určitých indícií a nekomplexných poznatkov iba predpokladáme, že sa v definovanej vzorke dát požadované informácie nachádzajú. Na základe štatistického pohľadu sme vyslovili hypotézu. Hypotézu musíme následne na vybranej vzorke pomocou prieskumu overiť a na základe výsledkov prieskumu odmietnuť alebo prijať. Výsledkom tohto procesu by mali byť informácie pre podporu rozhodovania, pričom pri ich využívaní by sa mal dosiahnuť merateľný ekonomický efekt.

Mohli by sme si myslieť, že data mining je univerzálna metóda. Samozrejme to ale nie je tak. Niekedy môžeme na základe viac-menej náhodne vybraných vstupov získať cenné informácie, inde môže byť výsledok data miningu triviálny, v praxi nevyužiteľný, ba dokonca žiadny. Kde sa v dátach žiadne informácie neskrývajú, samozrejme ich ani nemôžeme vydolovať. Ani reálne dáta nemusia byť kvalitné, teda kvalitne pripravené. Vo väčšine prípadov bude nutné upraviť hlavne vstupy, napríklad ich zoskupením do intervalu, prípadne znížiť počet diskrétnych hodnôt. Data mining musíme chápať ako prostriedok pre získavanie informácií pre podporu rozhodovania. Úlohu data miningu teda chápeme ako prostriedok pre poskytnutie kvalitných vstupov do procesu rozhodovania. Nezastupiteľnou úlohou manažmentu je potom vypracovanie a definícia vízií, koncepcií a cieľov a taktiež vyslovovanie hypotéz. Data mining potom môže poslúžiť ako nástroj pre overenie jednotlivých hypotéz [12], [13].

Data mining sa prakticky nedá plne automatizovať, pretože musíme jednak vytvoriť príslušné modely, prispôsobovať ich konkrétnym podmienkam a požiadavkám manažmentu a podobne. Preto teda nemôžeme definovať konkrétne postupy, ale je možné vytvoriť určitú metodiku [3].



Obr. 2.2: Architektúra systému pre dolovanie dát [1]

Architektúra typického systému pre dolovanie dát obsahuje nasledujúce hlavné komponenty:

- **Databáza, dátový sklad, World Wide Web alebo iné dátové skladište:** jedna alebo súbor databáz, dátové sklady, tabuľkový procesor alebo ďalšie typy skladíšť informácií. Čistenie a integrácia dát môžu byť prevedené na dátach.
- **Databázový server alebo server dátového skladu:** sú zodpovedné za získanie relevantných dát, založené na žiadostiach užívateľa pre dolovanie dát.
- **Databáza znalostí:** slúži na hľadanie a ohodnotenie zaujímavosti výsledných získaných modelov, vzorov. Tieto znalosti môžu obsahovať koncepčnú hierarchiu, ktorá je použitá pre usporiadanie atribútov alebo hodnôt atribútov do rôznych abstrakčných úrovní.
- **Stroj data miningu:** je nevyhnutný pre dolovanie dát a v ideálnom prípade sa skladá zo súboru funkčných modelov pre úlohy ako charakteristika, asociačná a korelačná analýza, klasifikácia, predikcia, zhluková analýza a evolučná analýza.
- **Modul hodnotenia vzťahov:** tento komponent typicky zabezpečuje rôzne opatrenia a interaguje s ostatnými modulmi systému, aby sa zamerali na hľadanie zaujímavých vzťahov. Môže používať zaujímavé prahy pre vyfiltrovanie modelov. Táto časť systému môže byť začlenená do modulu pre dolovanie, čo závisí od toho, aká metóda je použitá pre dolovanie.

- **Užívateľské rozhranie:** tento modul zabezpečuje komunikáciu medzi systémom a užívateľom, umožňuje užívateľovi komunikovať so systémom pomocou zadávaní rôznych dotazov, poskytuje informácie, ktoré pomáhajú zamerať sa na hľadanie modelov. Okrem toho tento komponent umožňuje užívateľovi prechádzať databázu, schému dátového skladu, štruktúru dát, či zobrazovať modely v rôznych formách [1], [5].

Z pohľadu dátového skladu, dolovanie dát môžeme chápať ako pokročilú etapu on-line analytického spracovania (OLAP). Avšak dolovanie dát ďaleko presahuje možnosti analytického spracovania, pretože sú tu spojené pokročilé techniky pre dátovú analýzu.

Nie každý nástroj deklarovaný ako nástroj pre dolovanie dát je schopný skutočného dolovania. Systémy pre analýzu dát, ktoré nezvládajú veľké objemy, by mali byť označované skôr ako systémy strojového učenia, nástroje pre štatistickú analýzu dát, prípadne experimentálne prototypové systémy.

Tradičná analýza nie je okrem veľkých objemov dát často použiteľná aj z iných dôvodov. Napríklad je to vysoká dimenzionalita dát. Ďalším dôvodom môže byť zložitosť dát. Okrem dolovania v dátach s jednoduchou štruktúrou, aké sú spravidla uložené v relačných databázach, sa dostáva stále viac do popredia aj potreba dolovania v zložito štruktúrovaných alebo inak zložitých dátach. Príkladom môžu byť prúdy dát, časové postupnosti, grafové štruktúry, priestorové a časovo-priestorové dáta, dáta na webe a ďalšie.

Dolovanie dát integruje v sebe techniky viacerých disciplín, ako je databázová technológia a technológia dátových skladov, štatistika, strojové učenie, vysoko náročné výpočty, rozpoznávanie, neurónové siete, vyhľadávanie informácií, spracovanie signálov apod. Pre nás budú dôležité techniky dolovania, ktoré sú škálovateľné a efektívne. Za škálovateľný považujeme algoritmus, ktorého časová zložitosť rastie približne lineárne s rastúcim objemom dát za predpokladu daných systémových zdrojov, ako je vnútorná pamäť a diskový priestor [5].

2.3 Typy dolovacích úloh

Táto podkapitola sa zaoberá charakterizovaním základných dolovacích úloh. Ide vlastne o to, aký druh modelu dát sa snažíme získať. Teda ide o úlohu riešenú v kroku dolovania dát. Jednotlivé typy dolovacích úloh sú nasledujúce [2]:

1. **Výskumná dátová analýza (exploratory data analysis – EDA):** ako názov naznačuje, cieľom je jednoducho preskúmať dáta bez jasnej predstavy o tom, čo vlastne hľadáme. Typicky sem patria interaktívne a vizuálne metódy a ďalšie efektívne grafické zobrazovacie metódy pre relatívne malé súbory dát s menším počtom dimenzií. So zvyšovaním počtu dimenzií rastie náročnosť zobrazenia mraku bodov v p-rozmernom priestore. Pre hodnoty p vyššie ako 3 alebo 4, môžu byť projekčné techniky, ktoré produkujú informatívne nízko-rozmerné projekcie z dát, veľmi užitočné.
2. **Deskriptívne modelovanie:** cieľom deskriptívneho modelovania je všeobecne popísať, charakterizovať vlastnosti analyzovaných dát. Príklady takejto charakteristiky zahrňujú modely pravdepodobnosti rozdelenia dát, rozdelenie p-rozmerného priestoru do skupín (zhluková analýza a segmentácia) a modely popisujúce vzťahy medzi premennými (modelovanie závislostí).
3. **Prediktívne modelovanie:** jedná sa o klasifikáciu a regresiu. Cieľom je vytvoriť model, ktorý umožňuje predpovedať hodnotu danej premennej na základe známych hodnôt ostatných premenných. Pri klasifikácii je predpovedaná premenná kategorická, kým pri regresii kvantitatívna. Pojem predikcia je použitý vo všeobecnom slova zmysle. Príkladom by mohla byť predpoveď ceny daného tovaru na trhu v budúcnosti, predpoveď víťaza v určitej

disciplíne, či určenie diagnostiky pacienta. Na riešenie problémov prediktívneho modelovania bol vyvinutý veľký počet metód v oblasti štatistiky a strojového učenia a práca v tejto oblasti viedla k výraznému teoretickému pokroku, na základe ktorého sa dajú lepšie pochopiť aj zložité problémy dedukcie. Kľúčovým rozdielom medzi deskriptívnym a prediktívnym modelovaním je to, že predikcia je zameraná na jednu jedinečnú premennú či hodnotu danej premennej, kým v deskriptívnom modelovaní neexistuje centrálna premenná, na základe ktorej by mohol byť vytvorený výsledný model.

4. **Hľadanie vzorov a pravidiel:** prvé tri typy dolovacích úloh sa zaoberali vytvorením modelov. Existujú aj ďalšie aplikácie pre dolovanie dát, ktoré sa zaoberajú zistením vzorov. Príkladom môže byť hľadanie kombinácií položiek, ktoré sa vyskytujú často v transakčnej databáze (napríklad tovary, ktoré sa často kupujú spoločne). Dolovanie dát sa často zameriava na túto problematiku a vyžaduje použitie algoritmických techník založených na asociačných pravidlách.
5. **Vyhľadávanie podľa obsahu:** užívateľ v tomto prípade chce podľa určitého vzoru nájsť podobný vzor v súbore dát. Táto úloha je často použitá pre súbory dát, ktoré obsahujú texty alebo obrázky. V prípade textu môže byť vzorom množina kľúčových slov a cieľom užívateľa je nájsť relevantný dokument v množine možných relevantných dokumentov (napr. webové stránky). V prípade obrázkov užívateľ môže mať vzorový obrázok alebo popis obrázku, na základe čoho chce nájsť podobný obrázok vo veľkej množine obrázkov. Definícia podobnosti je v oboch prípadoch kritická a závisí na zvolenej stratégie vyhľadávania.

Hoci každá z vymenovaných úloh sa jednoznačne líši od ostatných, všetky používajú spoločné komponenty. Príkladom môže byť podobnosť či vzdialenosť medzi dvoma vektormi. Spoločné môžu byť napríklad aj vyhodnocovacie funkcie (score functions) použité na odhadnutie toho, ako model alebo vzor zodpovedá dátam, hoci jednotlivé funkcie môžu byť celkom odlišné medzi jednotlivými kategóriami úloh. Je taktiež zrejmé, že rôzne štruktúry modelov a vzorov sú potrebné pre rôzne úlohy, taktiež ako rôzne štruktúry sú potrebné pre rôzne typy dát [2].

3 Dátové sklady

Témou tejto kapitoly sú dátové sklady. Na začiatku sú popísané kľúčové vlastnosti dátových skladov a metódy budovania dátových skladov. Pri budovaní dátových skladov hrá dôležitú úlohu príprava údajov. Patria sem etapy ako extrakcia, transformácia a prenos dát. Stručný popis týchto procesov a charakteristiku multidimenzionálneho dátového modelu nájdeme taktiež v tejto kapitole.

3.1 Dátový sklad

Dátový sklad podľa W. H. Inmona je definovaný takto: „Dátový sklad je subjektovo orientovaná, integrovaná, časovo variantná a stála kolekcia dát, ktorá poskytuje podporu v procese strategického rozhodovania manažmentu.“ Túto definíciu poznáme aj v nasledujúcej forme: „Dátový sklad je podnikovo štruktúrovaný depozitár subjektovo orientovaných, integrovaných, časovo premenlivých, historických dát použitých na získavanie informácií a podporu rozhodovania. V dátovom sklade sú uložená atomické a sumárne dáta“ [3].

Kľúčové vlastnosti vymenované v definícii sú práve tie vlastnosti, ktoré odlišujú dátové sklady od ostatných skladísk dát. Význam týchto pojmov je nasledujúci [3]:

- **Subjektová orientácia:** údaje sa do dátového skladu zapisujú skôr podľa predmetu záujmu než podľa aplikácie, v ktorej boli vytvorené. Dátový sklad je organizovaný podľa hlavných subjektov, ktorými môžu byť napríklad zákazníci, zamestnanci, produkty, dodávatelia, apod. Nezameriavajú sa na každodenné operácie, skôr na modelovanie a analýzu dát. Zameriavajú sa na konkrétny subjekt odstránením dát, ktoré nie je možné používať pri rozhodovaní. Oproti tomu orientácia na aplikáciu znamená, že údaje sú v systéme uložené podľa jednotlivých aplikácií, napríklad údaje aplikácie pre odbyt, údaje aplikácie pre fakturáciu, údaje aplikácie pre personalistiku.
- **Integrovanosť:** znamená, že dátový sklad je vytvorený spojením niekoľkých heterogénnych zdrojov dát, napríklad relačných databáz. Dátový sklad musí byť jednotný a integrovaný. To znamená, že údaje týkajúce sa konkrétneho predmetu sa do dátového skladu ukladajú iba raz. Preto musíme zaviesť jednotnú terminológiu, jednotné a konzistentné jednotky veličín. Nie je to jednoduchá úloha, pretože údaje prichádzajú do dátového skladu z nekonzistentného a neintegrovaného operačného prostredia. Využíva sa tu preto čistenie a integrácia dát, aby sa zjednotili názvy atribútov, merné jednotky, atď. V prípade, že údaje nie sú konzistentné a dôveryhodné, dátový sklad stratí význam.
- **Časová variabilita:** dáta sú uložené, aby poskytovali informácie z historickej perspektívy (napr. za posledných 5-10 rokov). Dáta sa ukladajú do dátového skladu ako série snímok, z ktorých každá reprezentuje určitý časový úsek. Čas teda predstavuje v dátovom sklade kľúčovú veličinu. Každá kľúčová štruktúra v dátovom sklade obsahuje časový element, a to buď implicitne alebo explicitne. Na rozdiel od operačného databázového prostredia, kde sa údaje ukladajú za kratšie časové obdobie (najčastejšie dni, maximálne mesiace), v dátovom sklade sú údaje za dlhšie časové obdobie (typicky niekoľko rokov).
- **Nemennosť (stálosť):** údaje v dátovom sklade sa obvykle nemenia ani neodstraňujú, iba sa v pravidelných intervaloch pridávajú nové hodnoty. Preto je manipulácia s údajmi omnoho jednoduchšia v dátových skladoch než v transakčných databázach. Dátový sklad je fyzicky oddelený od dát zo vstupnej databázy, a tak nevyžaduje transakčné spracovanie, zotavenie

a ďalšie mechanizmy. Väčšina metód pre optimalizáciu a normalizáciu údajov a transakčný prístup k údajom je v dátovom sklade nepotrebná.

Dátový sklad je teda súbor technológií pre efektívne skladovanie údajov tak, aby tieto údaje po ich premene na informácie slúžili podpore rozhodovania. V dátovom sklade môžeme vykonávať rôzne analýzy pre potreby rozhodovania manažérov, obchodníkov. Nástroje pre budovanie a prevádzku dátových skladov však predstavujú veľkú počiatočnú investíciu do hardwaru a softwaru, takže dátové sklady využívajú väčšinou banky, poisťovne, veľké obchodné reťazce apod.

3.2 Dátové trhy

Ako to už bolo spomínané, dátové sklady sú z hľadiska investície veľmi náročné. Preto pre niektoré menšie firemné organizačné zložky boli vytvorené podmnožiny dátového skladu, tzv. dátové trhy. Dátové trhy teda môžu vznikať spôsobom, keď je najprv vytvorený centrálny integrovaný dátový sklad a potom sa z neho vytvorí niekoľko dátových trhov. Toto riešenie má menšie nároky na prevádzku a údržbu. Druhou možnosťou je, že najprv sú pre jednotlivé organizačné zložky vytvorené dátové trhy a na základe týchto dátových trhov je vybudovaný dátový sklad. Dátové trhy teda môžu existovať ako subsystémy dátových skladov alebo aj samostatne ako jednoduché dátové sklady.

Dátové trhy sú budované pre podporu činnosti jednotlivých podnikových útvarov. Prax ukazuje, že zriedka existuje jednotná definícia pojmov používaných v podniku, čo vedie k rozdielnosti údajov získaných z jednotlivých zdrojov dát. Z tohto priamo vyplýva, že dáta nie sú integrované, užívatelia definujú iný formát dát ukladaných do jednotlivých dátových trhov apod. Aby bola zamedzená redundancia dát, nejednotný výklad pojmov v rámci podniku a aby sa zredukoval počet rozhraní, prišiel Bill Inmon so svojou koncepciou. Medzi jednotlivé nezávislé dátové trhy vložil databázu, ktorú nazývame dátový sklad. Tým došlo k zníženiu počtu rozhraní a k definícii pojmov tvoriacich znalostnú podnikovú bázu [6], [3].

3.3 Budovanie dátového skladu

V predchádzajúcej podkapitole sme si uviedli vzťah medzi vytvorením dátového trhu a budovaním dátového skladu. Pri budovaní dátového skladu okrem organizačnej štruktúry firmy musíme brať do úvahy aj možné problémy, ktoré sa môžu vyskytnúť v priebehu tohto procesu. Je preto dôležité vybrať najvhodnejšiu metódu pre budovanie dátového skladu. Najznámejšie a najčastejšie používané metódy sú metóda „veľkého tresku“ a prírastková metóda.

3.3.1 Metóda „veľkého tresku“

Veľkou výhodou tejto metódy je, že celý projekt môžeme kompletne vypracovať ešte pred začiatkom jeho realizácie. Metódu môžeme rozdeliť na tri dôležité etapy:

- analýza požiadaviek podniku,
- vytvorenie podnikového dátového skladu,
- vytvorenie prístupu buď priamo, alebo cez dátové trhy.

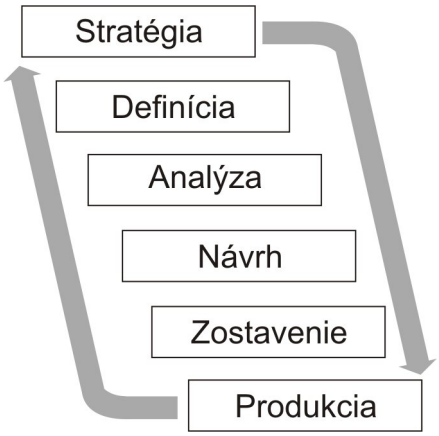
Nevýhodou tejto metódy môže byť fakt, že budovanie dátového skladu je dynamický proces, pri ktorom sa môžu zmeniť technológie alebo požiadavky užívateľov. Okrem toho trvá pomerne dlho, kým takto vybudovaný dátový sklad začne fungovať a prinášať zisk [3].

3.3.2 Prírastková metóda

Na rozdiel od predchádzajúcej metódy, kde budovanie dátového skladu prebehlo v jednom celku, táto metóda predpokladá budovanie skladu po etapách. Do procesu postupne prichádzajú jednotlivé prírastkové riešenia. Jedná sa o čiastočné riešenie rôznych predmetových oblastí. Vytvoria sa tak rôzne subsystémy. Po dôslednom otestovaní jednotlivých častí do systému môžeme pridať ďalšiu predmetovú oblasť, a tak získame novú funkcionalitu.

Prírastková metóda je iteratívny proces, ktorý udržuje neustálu spojitosť medzi dátovým skladom a potrebami užívateľov. Hlavné výhody tejto metódy sú zachovávanie kontinuity budovaného projektu s požiadavkami užívateľov, umožňovanie implementácie škálovateľnej architektúry a zabezpečenie rýchlejšieho zisku pomocou subsystémov.

Prírastková metóda existuje v dvoch formách: prírastková metóda smerom zhora dolu a prírastková metóda smerom zdola nahor. U metódy smerom zhora dolu je vytvorený konceptuálny model dátového skladu na základe požiadaviek užívateľov. Následne je vytvorený konceptuálny model jednotlivých predmetových oblastí. Táto metóda poskytuje pomerne rýchlu implementáciu jednotlivých dátových trhov. Prírastková metóda smerom zhora dolu nie je až taká náročná na analýzu ako metóda „veľkého tresku“ a poskytuje pomerne rýchlu návratnosť investícií [3].



Obr. 3.1: Schéma prírastkovej metódy [3]

Na základe schémy prírastkovej metódy môžeme konštatovať, že jedna iterácia tejto metódy sa skladá z nasledujúcich krokov:

- **Stratégia:** úlohou tejto fázy je definovať ciele. Okrem cieľa podnikania je potrebné definovať účel riešenia dátového skladu. Tieto ciele môžu byť krátkodobé alebo dlhodobé. Dlhodobý cieľ projektu dátového skladu počíta nielen s jeho vybudovaním, ale definuje aj stratégiu správy dátového skladu. V tejto fáze sa definuje aj základ architektúry dátového skladu.
- **Definícia:** patrí sem definícia rozsahu a cieľa prírastkového vývoja. Vytvorí sa počiatočný prírastok, konceptuálne modely, zdokumentujú sa zdroje dát. V tejto fáze sa navrhuje architektúra dátového skladu a architektúra technických prostriedkov.
- **Analýza:** cieľom je zamerať sa na informácie o užívateľoch, získavanie dát a požiadaviek na prístup k dátam na obchodnú analýzu. Sú vytvorené relačné a multimediálne modely pre dátový sklad a metadáta. V tomto kroku sú riešené problémy kvality dát, sú stanovené požiadavky na metadáta a dokončí sa výber nástrojov pre všetky komponenty dátového skladu.

- **Návrh:** cieľom je transformovanie požiadaviek získaných počas analýzy do detailných podmienok návrhu a dokončiť inštaláciu technickej architektúry.
- **Zostavenie:** vytvorenie a otestovanie navrhnutej databázovej štruktúry, modulov pre získavanie dát, správu dátového skladu, metadát a prístup k dátam.
- **Produkcia:** patrí sem inštalovanie dátového skladu, jeho používanie a údržba.

Ako už bolo spomínané, prírastková metóda je iteratívny proces, takže uvedené kroky sa môžu iteratívne opakovať.

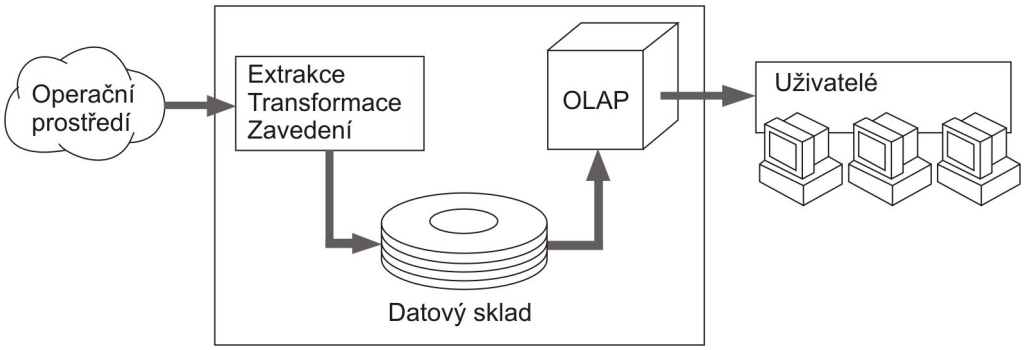
3.4 Príprava údajov

Údaje pre dátový sklad väčšinou pochádzajú z rôznych nehomogénnych zdrojov. Jedná sa o dáta zo súborových databáz, údaje z databáz spravovaných niektorým databázovým serverom alebo údaje vyexportované nejakou databázovou platformou. Príprava a zavedenie údajov je veľmi dôležitou súčasťou riešenia dátového skladu. Údaje z operačného prostredia je potrebné pred zavedením do dátového skladu vyextrahovať, vyčistiť, upraviť.

Tieto kroky patria do etapy ETL (Extraction, Transformation, Loading). Celý proces ETL je komplexný a časovo pomerne náročný a je veľmi dôležitou súčasťou každého projektu dátového skladu. Význam jednotlivých etáp procesu ETL je nasledujúci:

- **Extrakcia:** výber dát pomocou rôznych metód.
- **Transformácia:** overenie, čistenie, integrovanie a časové označenie dát.
- **Loading:** presun dát do dátového skladu.

Údaje sa teda nielen prenášajú do dátového skladu, ale aj spracovávajú, napríklad indexujú, sumarizujú, zisťujú sa prípadné zmeny štruktúr dátových údajov potrebných pre dátový sklad, udržujú sa metadáta.



Obr. 3.2: Dátový sklad [3]

Počiatočným naplnením dátového skladu údajmi úloha ETL nekončí. Dátový sklad sa v pravidelných intervaloch plní aktualizovanými dátami. Dôležité je, aby údaje zavedené do dátového skladu boli kvalitné a presné. Okrem toho je potrebné zaistiť aj dostupnosť, aby jednotliví užívatelia dátového skladu mohli používať dátový sklad účinne a efektívne [3], [10].

3.4.1 Extrakcia

Úlohou extrakcie je získať údaje z dátových zdrojov pomocou rôznych metód. Údaje môžu byť umiestnené v rôznych operačných prostrediach, hardwarových platformách, operačných systémoch, databázových systémoch, podnikových systémoch. Hovoríme teda o interných údajoch.

Okrem interných údajov niekedy potrebujeme pracovať aj s externými údajmi, ktoré môžeme získať analýzou konkurenčného prostredia, zakúpením údajov o zákazníkoch alebo stiahnutím údajov voľne prístupných z Internetu.

Pre túto etapu sú k dispozícii rôzne nástroje, postupy, technológie. Môžeme vytvárať vlastné aplikácie vo vyšších programovacích jazykoch alebo v procedurálnych nadstavbách jazyka SQL. Pri správne navrhnutej etape ETL máme k dispozícii metadáta pre všetky fázy tejto etapy.

3.4.2 Transformácia

Ešte pred samotnou transformáciu prebehne čistenie dát. Cieľom čistenia dát je odstránenie chýbajúcich hodnôt, odstránenie šumu, riešenie redundancie a odstránenie nekonzistencie dát. Čistenie údajov môže byť niekedy veľmi náročné a nákladné. Niekedy sa ani neoplatí čistiť dáta s vysokými nákladmi. Napriek tomu, že systémy OLTP obsahujú väčšinou kvalitné dáta, tieto údaje nemusia byť zárukou kvalitného dátového skladu.

Etapa transformácie zahrňuje aj integráciu dát. Pri dolovaní dát je často potrebné pracovať s dátami pochádzajúcimi z viacerých zdrojov. V takom prípade je potrebné integrovanie dát. Výsledkom integrovania je jeden koherentný zdroj. Pri integrácii je potrebné riešiť problémy ako napríklad konflikty schémy, konflikty hodnôt, konflikty identifikácie alebo redundancia.

Transformácia je súbor úloh, ktoré vedú k zvýšeniu kvality údajov. Cieľom transformácie je previesť dáta do podoby vhodnej pre dolovanie. Transformácia môže zahrňovať operácie ako agregácia, generalizácia, normalizácia, konštrukcia atribútov. Najčastejšie problémy pri transformácii sú [3]:

- **Ľudský faktor:** pravopisné chyby, preklepy.
- **Nejednoznačnosť údajov:** napr. údaje o pohlaví zákazníka môžu byť uložené rôznym spôsobom.
- **Chýbajúce hodnoty:** tieto hodnoty buď doplníme alebo ignorujeme.
- **Duplicitné záznamy:** redundancia údajov.
- **Konvencia názvu pojmov a objektov:** pojmy popisujúce tie isté údaje majú rôzne názvy.
- **Rôzne peňažné meny**
- **Formáty čísel a textových reťazcov:** pre uloženie čísla sú použité rôzne dátové typy (napr. rodné číslo môžeme uložiť ako číselnú hodnotu alebo ako textový reťazec).
- **Referenčná integrita:** organizačná štruktúra alebo údaje sa dynamicky menia a tieto zmeny môžu skresliť údaje a tak ovplyvniť kvalitu týchto údajov.
- **Chýbajúci dátum:** čas plní v dátovom sklade významnú úlohu, ale často tento údaj vo vstupných údajoch chýba.

Transformáciu dát je možné vykonať sériovo alebo paralelne so zavedením údajov. V prvom prípade sa transformácia vykoná pred zavedením dát do dátového skladu. U paralelnej metódy sa tento proces vykonáva súbežne so zavedením.

3.4.3 Prenos dát

Jedná sa o prenos údajov z pamäte zdrojových dát do dátového skladu. Prenos spočíva v presune údajov a ich uložení do tabuliek dátového skladu. Keď sa dá, prenos dát by mal prebehnúť automatizovane. Pri prvotnom naplnení dátového skladu sa môže jednať o obrovské množstvo údajov. Keď je dátový sklad dopĺňaný v pravidelných časových obdobiach, prenáša sa už menší objem dát.

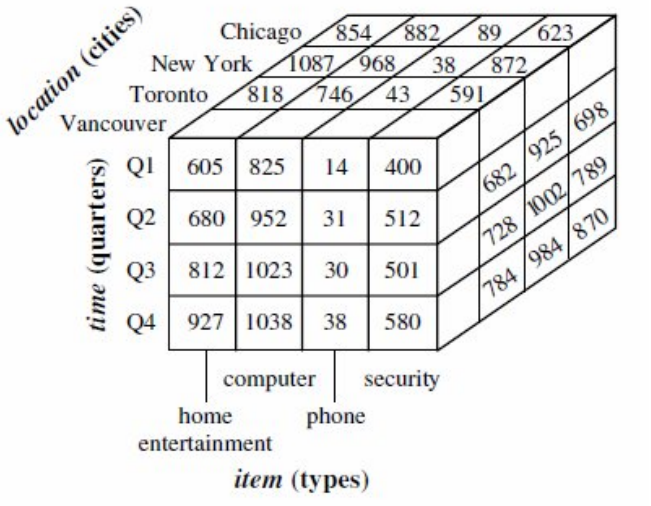
Po zavedení údajov prebieha ich indexovanie, aby prístup k ním bol optimalizovaný. Pre jednoznačnú identifikáciu údajov sa používajú aj umelo vytvorené kľúče, pomocou ktorých je zaistená jednoznačnosť každého riadku v tabuľke. Dáta dátového skladu sú totiž často kombináciou mnohých transformovaných záznamov, ktoré nemajú žiadne prirodzené kľúče, ktoré by sa dali používať pre jednoznačnú identifikáciu.

3.5 Multidimenzionálny dátový model

Dátový sklad je založený na multidimenzionálnom dátovom modele. Tento dátový model môžeme zobraziť ako viacrozmernú dátovú kocku. Táto kocka je vlastne ekvivalent tabuľky v relačnej databáze. Každá kocka má niekoľko dimenzií. Najlepšie si môžeme predstaviť klasickú trojdimenzionálnu kocku, ale v reálnych multidimenzionálnych databázach je počet dimenzií spravidla väčší. V dátových skladoch je kocka *n*-rozmerná. Jednotlivé záznamy sa v multidimenzionálnych kockách nachádzajú na priesečníkoch dimenzií. S rastúcim počtom rozmerov multidimenzionálnej databázy veľmi rýchlo rastú aj požiadavky na úložnú kapacitu. Avšak nie na všetkých priesečníkoch dimenzií sa vždy nachádzajú údaje. Takúto kocku nazývame aj riedkou kockou. V praxi sa pri multidimenzionálnych databázach používajú rôzne technológie na kompresiu objemu použitého diskového priestoru.

Dátová kocka umožňuje, aby dáta boli modelované a vnímané ako viacrozmerné. Každá dátová kocka je vytvorená na základe dvoch druhov údajov. Sú to:

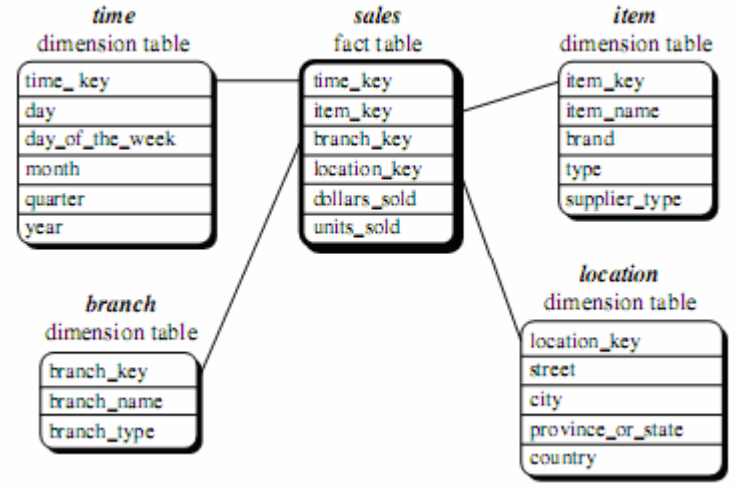
- **Fakty:** sú to vlastne numerické merné jednotky. Predstavujú množstvo, na základe ktorého analyzujeme vzťahy medzi dimenziami. Tabuľka faktov je najväčšia tabuľka v databáze a obsahuje veľký objem dát. Prvotné fakty sa môžu kombinovať alebo vypočítať pomocou iných faktov a vytvoriť tak merné jednotky. Merné jednotky sa môžu uložiť v tabuľke faktov. Príklady faktov môžu byť celková cena alebo počet položiek nákupu.
- **Dimenzie:** obsahujú logicky alebo organizačne hierarchicky usporiadané údaje. Dimenzie sú vlastne entity alebo pohľady. Môžeme ich chápať ako textové popisy obchodovania. Tabuľky dimenzií sú obyčajne menšie než tabuľky faktov a dáta v nich sa nemenia tak často. Dimenzie všeobecne obsahujú relatívne stabilné dáta. Veľmi často sa používajú časové, produktové a geografické dimenzie [3], [4].



Obr. 3.3: 3-D pohľad na dátovú kocku [1]

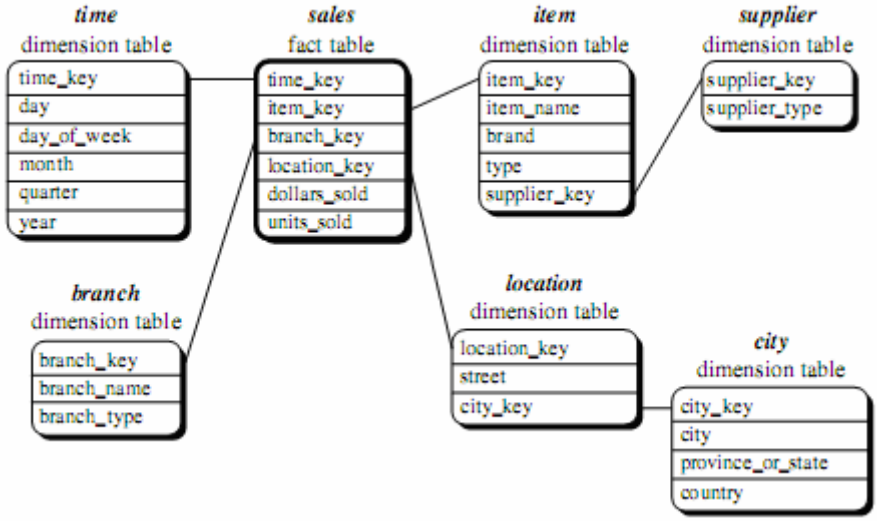
Dátovú kocku vytvárame na základe dimenzionálneho modelu, ktorý má určité topologické usporiadanie, tzv. schému. Pri dátových skladoch sa najčastejšie používa schéma hviezda a schéma snehovej vločky.

Schéma hviezda je najbežnejšou schémou. Skladá sa z tabuľky faktov, ktorá obsahuje cudzie kľúče, ktoré sa vzťahujú k primárnym kľúčom v tabuľkách dimenzií. Tabuľka faktov je teda centrálnou tabuľkou, ktorá obsahuje veľké množstvo dát bez redundancie. Okrem tejto tabuľky schéma hviezda obsahuje množinu menších tabuliek dimenzií. Každá dimenzia je reprezentovaná jednou tabuľkou s niekoľkými atribútmi. Schéma hviezda nemá normalizované dimenzie ani relačné prepojenia medzi tabuľkami dimenzií, preto je veľmi ľahko pochopiteľná. V dôsledku nenormalizovaných dimenzií môže vzniknúť redundancia a vytvorenie takéhoto modelu môže byť veľmi pomalé. Schéma hviezda je znázornená na Obr. 3.4.



Obr. 3.4: Schéma hviezda [1]

Schéma snehovej vločky je variantom schémy hviezda, kde niektoré tabuľky dimenzií sú normalizované. Obsahuje teda dimenzie zložené z mnoho relačne zviazaných tabuliek. Tento model umožňuje rýchlejšie zavedenie údajov do normalizovaných tabuliek.



Obr. 3.5: Schéma snehovej vločky [1]

Táto schéma pri vykonávaní dotazu vyžaduje viac spojení, a preto má podstatne nižší dotazovací výkon. Z tohto dôvodu sa schéma snehovej vločky príliš nepoužíva. Výhodou ale môže byť zníženie redundancie. Pri rozsiahlych tabuľkách dimenzií to môže znamenať veľkú úsporu miesta. Schéma snehovej vločky je znázornená na Obr. 3.5.

4 Asociačná analýza

Pri získavaní asociačných pravidiel hľadáme zaujímavé asociácie alebo korelácie nad veľkými množinami dátových položiek. Nájdenie zaujímavých asociácií nad obchodnými transakčnými záznamami môže pomôcť v procese obchodného rozhodovania, ako je návrh katalógov, akčných ponúk alebo rozmiestnenie produktov v obchode.

Ako najčastejší príklad na získavanie asociačných pravidiel môžeme spomenúť analýzu nákupného košíka. Zisťujeme tu, aké produkty si zákazníci najčastejšie kupujú spoločne. Je teda analyzované správanie zákazníka, hľadá sa asociácia medzi produktmi, ktoré zákazníci umiestnia do nákupného košíka. Na základe takýchto asociácií predajcovia získajú informácie o tom, ktoré produkty kupujú zákazníci častejšie spoločne. Pomocou týchto informácií môžu rozvíjať svoje marketingové stratégie a tým zvýšiť predaj jednotlivých položiek. Príkladom môže byť rozmiestnenie jednotlivých produktov v obchode. Tým, že výrobky, ktoré sa často kupujú spoločne, sú umiestnené blízko k sebe, môžu zvýšiť pravdepodobnosť kúpy oboch produktov. Druhou možnosťou je, že tieto produkty umiestnia na dve rôzne miesta. Tým môžu dosiahnuť, že zákazník kúpi aj iné výrobky, ktoré nájde po ceste medzi týmito produktmi. Získané závislosti sa dajú vyžívať aj pri navrhovaní letákov, akčných ponúk.

Asociačná analýza okrem obchodného manažmentu našla uplatnenie aj v iných zaujímavých oblastiach.

4.1 Frekventované vzory

Ešte predtým, než podrobnejšie rozoberieme tematiku asociačných pravidiel, definujme pojmy frekventované vzory, frekventované množiny, frekventované podpostupnosti. Frekventované vzory sú vzory, ktoré sa často vyskytujú v dátach. U analýzy nákupného košíka, kde sa súbor rôznych položiek (ako napríklad mlieko a chlieb) často objaví v transakciách, hovoríme o frekventovaných množinách. V prípade frekventovaných podpostupností (napríklad, že zákazníci najprv kupujú počítač, potom digitálny fotoaparát a následne pamäťovú kartu), hovoríme o frekvenčných sekvenčných vzoroch. Ešte môžeme spomenúť iné podštruktúry, ako napríklad podgrafy, podstromy. Keď sa v dátach často vyskytujú podštruktúry, hovoríme o frekventovaných štruktúrovaných vzoroch.

Dolovanie frekventovaných vzorov hrá významnú úlohu pri dolovaní asociácií, korelácií a ďalších zaujímavých vzťahov medzi údajmi. Okrem toho môže pomôcť v klasifikácii, zhlukovaní a ďalších úlohách data miningu [5].

4.2 Asociačné pravidlá

Ako sme to už uviedli v predchádzajúcej podkapitole, typickou úlohou pre získavanie asociačných pravidiel je tzv. analýza nákupného košíka. Cieľom je nájdenie častých vzorov, tj. produktov, ktoré zákazníci často kupujú spoločne, resp. nájdenie tzv. asociačných pravidiel. Tieto pravidlá vyjadrujú určitý záver vyplývajúci z analýzy jednotlivých nákupov, napríklad keď si zákazník kúpi nový počítač, kupuje často aj operačný systém a antivírový program.

Keď si vezmeme všetky položky, ktoré obchod ponúka, každú z nich môžeme reprezentovať booleovskou premennou. Táto hodnota reprezentuje prítomnosť alebo neprítomnosť položky

v košíku. Potom môže byť každý košík reprezentovaný bitovým vektorom. Tieto vektory je možné analyzovať a tak získavať asociačné pravidlá.

Všeobecný zápis asociačného pravidla pre booleovské hodnoty $A_1,...,A_p$ vypadá takto:

$$\left((A_{i_1}=1)\wedge...\wedge(A_{i_k}=1)\right)\Rightarrow A_{i_{k+1}}=1,$$

kde $1 \leq i_j \leq p$ pre všetky j. Asociačné pravidlo môžeme stručnejšie napísať ako $\left(A_{i_1}\wedge...\wedge A_{i_k}\right)\Rightarrow A_{i\,k+1}$. Vzor

$$(A_{i_1}=1)\wedge...\wedge(A_{i_k}=1)$$

nazývame množinou [2].

Príklad asociačného pravidla môže byť nasledovný:

$$počítač \Rightarrow antivírový\_software\ [s = 2\%,\ c = 60\%],$$

kde s a c značí podporu (*support*) a spoľahlivosť (*confidence*), čo sú metriky zaujímavosti pravidla. Podpora 2% značí, že v 2% transakcií boli tieto dve položky spoločne. Spoľahlivosť 60% znamená, že 60% zákazníkov, ktorí si kúpili počítač, si taktiež kúpili antivírový software. Asociačné pravidlo je považované za zaujímavé, keď je splnená podmienka minimálnej podpory a minimálnej spoľahlivosti.

Objektívnou mierou asociačného pravidla v tvare $X \Rightarrow Y$, kde X a Y sú množiny položiek, je podpora pravidla. Udáva, ako často sa dané pravidlo vyskytuje v databáze transakcií, ktorú analyzujeme. Udáva sa v relatívnej (v %) alebo absolútnej (počet transakcií) forme. V druhom prípade musíme poznať aj celkový počet analyzovaných transakcií. Vyjadruje teda pravdepodobnosť $P(X \cup Y)$, kde $X \cup Y$ značí, že transakcia obsahuje ako položky množiny X, tak Y. Podpora vyjadruje, ako často sa vyskytujú položky množín X a Y v transakciách spoločne, ale nehovorí o tom, či neexistuje mnoho takých transakcií, ktoré obsahujú položky množiny X, ale neobsahujú položky množiny Y. Preto sa používa ďalšia objektívna miera pre vyjadrenie zaujímavosti asociačného pravidla. Jedná sa o spoľahlivosť. Spoľahlivosť ohodnocuje stupeň istoty platnosti daného asociačného pravidla. Je definovaná ako podmienená pravdepodobnosť $P(Y|X)$. Pri analýze nákupného košíka určuje, s akou pravdepodobnosťou si zákazník, ktorý kúpil položky X, kúpi aj položky Y [5], [11].

Formálna definícia podpory a spoľahlivosti pravidla $X \Rightarrow Y$ vypadá takto:

$$podpora(X \Rightarrow Y) = P(X \cup Y) \tag{4.1}$$

$$spoľahlivosť(X \Rightarrow Y) = P(Y|X) \tag{4.2}$$

Proces dolovania asociačných pravidiel obsahuje nasledujúce kroky:

1. Nájdenie frekventovaných množín – nájdenie množín položiek, ktoré splňujú minimálnu podporu. Tento krok je časovo náročnejší.
2. Generovanie silných asociačných pravidiel – silné asociačné pravidlo je také pravidlo, ktoré spĺňa podmienku minimálnej podpory a minimálnej spoľahlivosti.

Pretože druhý krok je časovo menej náročný, celková výkonnosť dolovania asociačných pravidiel závisí od prvého kroku.

4.2.1 Typy asociačných pravidiel

Prvým krokom dolovania asociačných pravidiel je hľadanie frekventovaných množín. Tento proces môžeme klasifikovať na základe rôznych kritérií, ale hlavným kritériom je druh pravidla. Asociačné pravidlá totiž môžeme rozdeliť do rôznych skupín podľa druhu. Môžeme ich teda klasifikovať [5]:

- **podľa typu hodnôt v pravidlách**
  - *booleovské asociačné pravidlá:* zaujíma nás iba prítomnosť alebo neprítomnosť danej položky, a preto nadobúdajú booleovské hodnoty. Napríklad:

    $počítač \Rightarrow antivírový\_software$
  - *kvantitatívne asociačné pravidlá*: v tomto prípade pravidlo popisuje asociácie medzi kvantitatívnymi položkami alebo atribútmi. V týchto pravidlách sú hodnoty kvantitatívnych položiek alebo atribútov rozdelené na intervaly. Príkladom môže byť

    $vek(X, \text{“30...39”}) \wedge príjem(X, \text{“42000...48000”}) \Rightarrow kúpi(X, \text{“LCD TV”})$

    Atribúty *vek* a *príjem* sú diskretizované.
- **podľa počtu dimenzií v pravidlách**
  - *jednodimenzionálne asociačné pravidlá*: booleovské asociačné pravidlo obsahuje iba jeden predikát. Napríklad:

    $kúpi(X, \text{“digitálny fotoaparát“}) \Rightarrow kúpi(X, \text{“tlačiareň“})$

    Ako vidíme, toto pravidlo obsahuje iba dimenziu (predikát) *kúpi*.
  - *multidimenzionálne asociačné pravidlá*: pravidlá obsahujú viac dimenzií alebo predikátov. Príkladom môže byť:

    $vek(X, \text{“20...29”}) \wedge zamestnanie(X, \text{“študent”}) \Rightarrow kúpi(X, \text{“notebook”})$
  - *medzidimenzionálne asociačné pravidlá*: asociačné pravidlá, v ktorých sa predikáty neopakujú
  - *hybridne dimenzionálne pravidlá:* asociačné pravidlá, v ktorých sa predikáty opakujú

    $vek(X, \text{“20...29”}) \wedge kúpi(X, \text{“notebook”}) \Rightarrow kúpi(X, \text{“tlačiareň”})$
- **podľa úrovní abstrakcie v pravidlách**
  - *jednoúrovňové asociačné pravidlá*: jedná sa o najjednoduchšie booleovské asociačné pravidlá. Napríklad:

    $vek(X, \text{“30...39”}) \Rightarrow kúpi(X, \text{“ TV”})$
  - *viacúrovňové asociačné pravidlá*: existujú metódy, ktoré sú schopné získať pravidlá nad rôznymi úrovňami abstrakcie.

    $vek(X, \text{“30...39”}) \Rightarrow kúpi(X, \text{“LCD TV”})$
- **podľa ďalších rozšírení asociačných pravidiel**
  Dolovanie asociačných pravidiel môže byť rozšírené o ďalšie metódy, ako napríklad korelačná analýza, maximálne vzory alebo uzavreté množiny.

4.2.2 Dolovanie asociačných pravidiel

Ako sme videli, existujú rôzne typy asociačných pravidiel. Najjednoduchším typom asociačných pravidiel sú jednoúrovňové booleovské asociačné pravidlá. V tomto prípade sa najčastejšie používa algoritmus Apriori. Jedná sa o algoritmus pre získavanie frekventovaných množín. V každej iterácii sú získané frekventované k-prvkové množiny použité pre generovanie $(k+1)$-prvkových množín.

Využíva sa tzv. Apriori vlastnosť, podľa ktorej každá podmnožina frekventovanej množiny musí byť taktiež frekventovaná. Tento algoritmus sa skladá z dvoch krokov: spojovací a vylučovací krok. Existujú rôzne varianty tohto algoritmu, ktoré zvyšujú efektivitu základného algoritmu. Pomocou tohto algoritmu získame iba frekventované množiny. Z týchto množín je potrebné vygenerovať asociačné pravidlá. Generovanie sa uskutočňuje s využitím rovnice pre výpočet spoľahlivosti [5]:

$$conf\,(A \Rightarrow B) = P(B|A) = \frac{s(A \cup B)}{s(A)} \qquad (4.3)$$

Na základe tejto rovnice sa postupuje pri generovaní asociačných pravidiel v nasledujúcich krokoch:

- pre každú frekventovanú množinu *l*, generuj všetky jej vlastné podmnožiny,
- pre každú podmnožinu *s*, vygeneruj pravidlo $s \Rightarrow (l - s)$ a podľa rovnice vypočítaj jeho spoľahlivosť. Keď je jeho spoľahlivosť vyššia než minimálna, tak je pravidlo silné.

Pretože pravidlá sú generované z frekventovaných množín, je pre nich splnená podmienka minimálnej podpory.

Pri algoritme Apriori môže byť problémom, že je potrebné neustále prechádzať databázu. Okrem toho pri väčších databázach môže byť počet generovaných kandidátov veľmi veľký. Tento problém môže vyriešiť používanie metódy FP-stromu.

V našom prípade ale budú zaujímavejšie multidimenzionálne asociačné pravidlá, pretože budeme pracovať s multidimenzionálnou databázou. Jedná sa o pravidlá v tvaru

$$vek(\text{X}, \text{“20...29”}) \wedge zamestnanie(\text{X}, \text{“študent”}) \Rightarrow kúpi(\text{X}, \text{“notebook”})$$

Podľa terminológie používanej v multidimenzionálnych databázach každý predikát môžeme nazvať dimenziou. Ako sme to už spomenuli, atribúty relačných databáz môžu byť kategorické alebo kvantitatívne.

Získať multidimenzionálne asociačné pravidlá môžeme s využitím statickej diskretizácie kvantitatívnych atribútov. Podstatou tejto metódy je, že kvantitatívne atribúty sú diskretizované ešte pred začiatkom procesu dolovania. Numerické hodnoty sú nahradené intervalmi. Kategorické atribúty je taktiež možné generalizovať na vyššie konceptuálne úrovne. Modifikovaný algoritmus Aprioiri je schopný pracovať s množinami predikátov. Takto upravené dáta môžu byť uložené taktiež v dátovej kocke.

Ďalšou možnosťou je získavanie kvantitatívnych asociačných pravidiel. Jedná sa o pravidlá, v ktorých diskretizácia prebieha dynamicky v priebehu procesu dolovania tak, aby boli splnené určité kritériá. Môže byť použitý napríklad systém ARCS (Association Rule Clustering System).

Tieto metódy používali počiatočnú diskretizáciu, neskôr boli intervaly spojené. Metóda získavania asociačných pravidiel založených na vzdialenosti používa taktiež diskretizáciu, ale je založená na vzdialenosti – spája do intervalu najbližšie prvky. Metóda do hĺbky môže spojiť veľmi vzdialené hodnoty. Metóda s rovnakými intervalmi vytvorí intervaly, pre ktoré nie sú dáta. Metódy založené na vzdialenosti sú teda najviac zmysluplné. Typicky sa pre tieto metódy využíva zhlukovanie hodnôt kvantitatívneho atribútu.

Pre získavanie asociačných pravidiel samozrejme existuje mnoho metód. Mohli by sme ešte spomenúť napríklad metódu, ktorá využíva štatistické analýzy pre získavanie asociačných pravidiel alebo metódu, ktorá využíva evolučný algoritmus pre diskretizáciu [5].

5 MS SQL Server

Systém pre dolovanie asociačných pravidiel z dátových skladov bude implementovaný pomocou relačného databázového systému Microsoft SQL Server. V tejto kapitole nájdeme stručný popis architektúry tohto databázového systému a popis podpory pre vývoj dátových skladov v prostredí MS SQL Server.

5.1 Prehľad systému SQL Server 2005

Databázový systém MS SQL Server poskytuje pre správu a analýzu dát zvýšené zabezpečenie, škálovateľnosť a dostupnosť podnikových dát a analytických aplikácií a súčasne uľahčuje ich vytváranie, nasadenie a správu. Systém predstavuje integrované riešenie pre správu a analýzu dát a uľahčuje organizáciám všetkých veľkostí nasledujúce operácie:

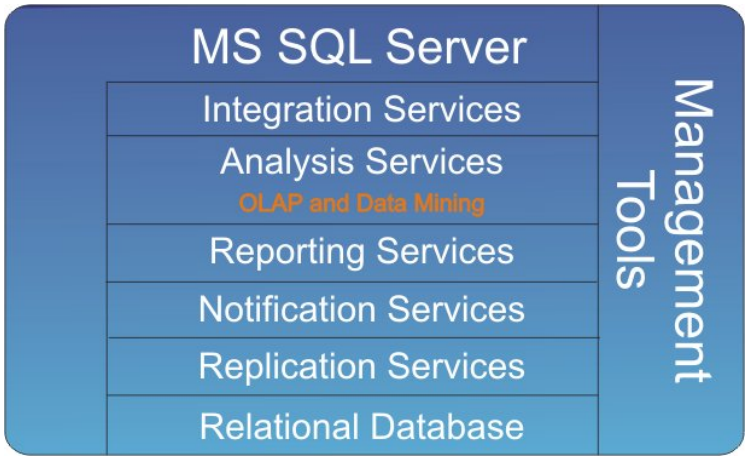
- vytvárať, nasadiť a spravovať podnikové aplikácie, ktoré sú lepšie zabezpečené, škálovateľnejšie a spoľahlivejšie,
- maximalizovať produktivitu informačných technológií zjednodušením vývoja a podpory databázových aplikácií,
- zdieľať dáta medzi viacerými platformami, aplikáciami a zariadeniami a uľahčiť tak prepojenie interných a externých systémov.

Okrem toho umožňuje riadiť náklady tak, aby nedošlo k zníženiu výkonu, dostupnosti, škálovateľnosti či zabezpečenia [8].

5.2 Dátová platforma serveru SQL Server

SQL Server je mnohostranné, integrované a komplexné riešenie pre dáta a poskytuje lepšie zabezpečené, spoľahlivejšie a produktívnejšie platformy pre podnikové dáta a aplikácie business intelligence. Obr. 5.1 znázorňuje rozloženie dátovej platformy serveru SQL Server 2005.
Dátová platforma serveru SQL Server zahrňuje nasledujúce nástroje [8]:

- **Relačná databáza:** zabezpečenejší, spoľahlivejší, škálovateľnejší a vysoko dostupný relačný databázový stroj so zvýšeným výkonom a podporou štruktúrovaných a neštruktúrovaných dát (XML).
- **Služba Replication Services:** replikácia dát pre aplikácie spracovávajúce distribuované či mobilné dáta, vysoká dostupnosť systému.
- **Služba Notification Services:** pokročilá funkcia zaslania upozornenia pre vývoj a nasadenie škálovateľných aplikácií.
- **Služba Integration Services:** funkcie extrakcie, transformácie a načítania dát (ETL) pre dátové sklady a integráciu dát v celom podniku.
- **Služba Analysis Services:** funkcie OLAP pre rýchlu a pokročilú analýzu veľkých a zložitých dátových sád s využitím viacdimenzionálnych skladíšť.
- **Služba Reporting Services:** komplexné riešenie pre vytváranie, správu a zasielanie tradičných papierových a interaktívnych webových zostáv.



Obr. 5.1: Rozloženie dátovej platformy serveru SQL Server 2005 [4]

- **Nástroje pre správu:** SQL Server zahŕňa integrované nástroje pre pokročilú správu a ladenie databáz a umožňuje úzku integráciu s nástrojmi. Štandardné protokoly pre prístup k dátam výrazne skracujú dobu potrebnú k integrácii dát serveru SQL Server s existujúcimi systémami.
- **Nástroje pre vývojárov:** SQL Server poskytuje integrované nástroje pre databázový stroj, extrakciu, transformáciu a načítanie dát (ETL), dolovanie dát, funkcie OLAP a vytváranie zostáv, ktoré sú integrované so sadou Microsoft Visual Studio a poskytujú komplexné funkcie pre vývoj aplikácií.

Pretože témou tejto práce je dolovanie asociačných pravidiel z dátových skladov, pre nás budú dôležité tie služby tejto platformy, ktoré úzko súvisia s dátovými skladmi. Sú to hlavne integračné a analytické služby.

5.3 Integračné služby

Služba SQL Server Integration Services (SSIS) nahradila službu transformácie dát Data Transformation Services (bola súčasťou SQL Server 2000) a poskytuje funkcie a výkon potrebné k vytvoreniu aplikácií pre integráciu dát na úrovni podniku. Vytvorenie sledu procesov pre dáta v celej organizácii môže byť jedným z najobťažnejších úloh. Služba Integration Services uľahčuje vytváranie výkonných aplikácií pre integráciu dát.

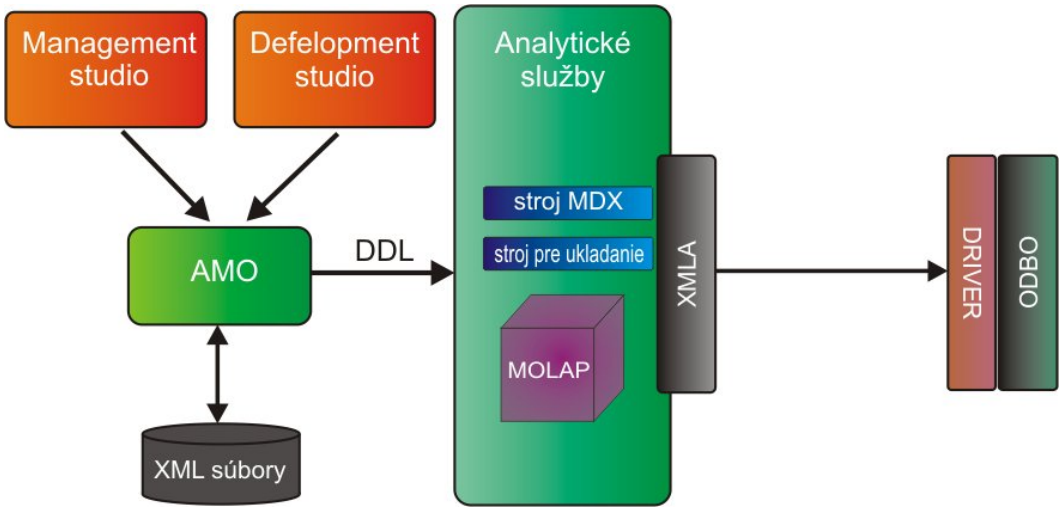
Okrem zavedenia dát z produkčných databáz do dátových skladov nájdu integračné služby uplatnenie aj pri zbere dát z rôznorodých systémov a reorganizácie štruktúry dát, ktoré vyplývajú napríklad zo zmeny organizačnej štruktúry alebo zo zmeny, prípadne reštrukturalizácie predmetu podnikania.

Hlavnými blokmi architektúry integračných služieb sú Data Transformation Pipeline (DTP) a Data Transformation Runtime (DTR). DTP prepojuje zdrojový a cieľový dátový adaptér. Na nižšej vrstve sa vykonávajú komplexné úlohy, z ktorých sa skladá proces extrakcie a transformácie. Základným prvkom tejto vrstvy je Tasks. Sú to vlastne samostatne vykonateľné, jednoduché úlohy podieľajúce sa na komplexnom procese transformácie [4].

5.4 Analytické služby

Služba SQL Server 2005 Analysis Services poskytuje integrované zobrazenie obchodných dát pre účely ako vytváranie zostáv, analýza OLAP, prehľady kľúčových ukazovateľov výkonu (KPI) a dolovanie dát.

Zdrojové súbory, ktoré obsahujú definície objektov OLAP, sú uložené do súboru vo forme dokumentu XML. Prístup k nim je zabezpečený pomocou MS Analysis Management Objects (AMO). Pre správu databáz, analytických služieb a analytických objektov slúži MS SQL Server Management Studio. Nástroj MS Visual Studio 2005 slúži na vytváranie objektov Business Intelligence.



Obr. 5.2: Schéma architektúry analytických služieb MS SQL Serveru 2005 [4]

Knižnica analytických objektov AMO posiela požiadavky na blok analytických služieb. Pre komunikáciu s analytickými službami je použitý štandard XMLA. XMLA je jediným systémovým rozhraním. Existujúce aplikácie môžu využívať pripojenie rozhrania ODBO (OLE DB for OLAP). Na strane klienta je totiž ovládač, ktorý transformuje volanie rozhrania ODBO na volanie natívneho rozhrania XMLA [4].

5.5 XMLA – XML for Analysis

XMLA je nový štandard, ktorý umožňuje komunikáciu medzi klientskými aplikáciami a multidimenznionálnymi alebo OLAP dátovými zdrojmi. Komunikácia je zabezpečená pomocou posielaním správ a je použitý nejaký webový štandard – HTML, SOAP alebo XML. Dotazovacím jazykom je jazyk MDX, ktorý je v súčasnosti najčastejšie používaným dotazovacím jazykom pre OLAP.

Technicky povedané, XMLA je špecifikácia pre sadu rozhraní XML správ, ktoré používajú Simple Object Access Protocol (SOAP) pre definíciu vzájomného prístupu dát medzi klientskou aplikáciou a poskytovateľom analytických dát, ktorý je dostupný cez Internet. Pomocou štandardného API, XMLA poskytuje otvorený prístup k multidimenzionálnym dátam z rôznych zdrojov dát – z akejkoľvek klientskej platformy na akúkoľvek serverovú platformu – prostredníctvom webových služieb, ktoré sú podporované viacerými dodávateľmi [7].

Vzhľadom na to, že XMLA je architektúra založená na webovom rozhraní a používa štandardné internetové protokoly, výsledkom je nezávislá technológia, ktorá je schopná pracovať

s akýmkoľvek nástrojom, aplikačným programovacím jazykom, hardvérom, platformou alebo zariadením.

Čo sa týka základných metód, jedná sa o rozhranie, ktoré používa iba dve metódy:

- **Discover:** práca s metadátami.
- **Execute:** vykonanie príkazov a prípadné vrátenie dát.

Rozhranie XMLA je nezávislé na platforme a operačnom systéme. Hlavnou nevýhodou je relatívna prácnosť pri zostavovaní príkazov a hlavne pri interpretácii odpovedí.

5.5.1 Metóda Discover

Táto metóda slúži na získavanie informácií, ako napríklad zoznam dostupných databáz alebo podrobnosti o konkrétnom objekte, z inštancie MS SQL Server Analysis Services. Údaje získané pomocou metódy Discover sú závislé na zadaných parametroch [4].

Základný syntaktický predpis pre metódu Discover je nasledujúci:

```
Discover
(
[in] RequestType As EnumString,
[in] Restrictions As Restrictions,
[in] Properties As Properties,
[out] Result As Rowset
)
```

Význam jednotlivých parametrov je:

[in] RequestType – typ požiadavky

[in] Restrictions – obmedzenie pre dáta a informácie, ktoré požadujeme ako výstup

[in] Properties – parametre pre metódu Discover

[out] Result – výstup vo forme Rowsetu

5.5.2 Metóda Execute

Vykonáva príkazy, ktoré sú odoslané inštancii SQL Server Analysis Services. Príkaz zahŕňa žiadosti týkajúce sa prenosu dát, ako sú napríklad načítanie a aktualizácia dát na serveri.

Základný syntaktický predpis pre metódu Discover je nasledujúci:

```
Execute
(
[in] Command As Command,
[in] Properties As Properties,
[out] Result As Resultset
)
```

Význam jednotlivých parametrov je:

[in] Command – príkaz

[in] Properties – parametre pre metódu Execute

[out] Result – výstup vo forme Rowsetu

6 Algoritmus Star-miner

Pre získavanie asociačných pravidiel z dátových skladov existuje niekoľko algoritmov. Z týchto sme vybrali algoritmus Star-miner. Ako to aj z názvu vyplýva, algoritmus pracuje s databázovou schémou hviezda. Schéma hviezda je obľúbená schéma relačných databáz, ktoré reprezentujú multidimenzionálne dáta v dátových skladoch. V nasledujúcich podkapitolách bude podrobnejšie popísaný algoritmus Star-miner, ktorý je veľmi efektívny a slúži pre dolovanie asociačných pravidiel z pospájaných tabuliek schémy hviezda. Uvedený algoritmus je navrhnutý tak, aby bol implementovateľný priamo nad relačnou databázou pomocou SQL dotazov, a využíva unikátne vlastnosti schémy hviezda. Algoritmus Star-miner je oveľa výkonnejší a efektívnejší ako ostatné algoritmy, ktoré slúžia pre dolovanie asociačných pravidiel z jednoducho pospájaných tabuliek [9].

6.1 Teoretické predpoklady

Existujú rôzne algoritmy pre dolovanie asociačných pravidiel. Asociačné pravidlá boli najprv používané pri analýze nákupného košíka, ale neskôr sa začali používať aj na rôzne iné druhy údajov. Všetky tieto algoritmy predpokladajú, že dáta sú uložené v jednej tabuľke. Dáta uložené v dátových skladoch sú rozložené do viacerých tabuliek, pričom najbežnejšie používanou schémou je už spomínaná schéma hviezda.

Ako to už bolo spomínané v predchádzajúcich kapitolách, táto schéma je určená pre uloženie multidimenzionálnych dát. Skladá sa z centrálnej tabuľky faktov (je to vlastne tabuľka vzťahov) a z viacerých tabuliek (tabuľky entít), ktoré reprezentujú jednotlivé dimenzie. Tabuľka faktov vytvára vzťahy medzi tabuľkami dimenzií, ktoré obsahujú atribúty jednotlivých entít. Tabuľka faktov obsahuje cudzie kľúče do tabuliek dimenzií. Typické algoritmy pre dolovanie asociačných pravidiel vyžadujú spojenie tabuľky faktov s tabuľkami dimenzií ešte pred získavaním frekventovaných množín. Napriek tomu, že réžia na vytvorenie spojenia nie je významná v porovnaní s cenou dolovacieho kroku, nevýhodou tohto postupu je, že výsledok spojenia je príliš veľký. Počet atribútov je rovný súčtu atribútov jednotlivých tabuliek a cena dolovania je primeraná počtu atribútov. Dáta uložené vo výsledku sú nadbytočné, pretože každý rad tabuľky dimenzií sa v konečnej spojenej tabuľke vyskytuje toľkokrát, koľkokrát sa nachádza jeho kľúč (id transakcie) v tabuľke faktov. A preto sú množiny započítané viackrát aj napriek tomu, že sú iba v niekoľkých radoch tabuľky dimenzií.

Dolovanie frekventovaných množín zo schémy hviezda je rozdelené na dve fázy. V prvej fáze sú získané frekventované množiny z každej tabuľky dimenzií. Po získaní frekventovaných množín s ohľadom na konečný zlúčený výsledok je spočítaný počet výskytov kľúčov tabuliek dimenzií v tabuľke faktov. Tento krok je prevedený jednoducho, pokiaľ je kardinalita tabuliek dimenzií v porovnaní s tabuľkou faktov malá (čo je vlastnosť schémy hviezda). V druhej fáze sú získané frekventované množiny použité pre efektívne dolovanie frekventovaných množín naprieč tabuľkami. Toto je založené na fakte, že všetky podmnožiny frekventovaných množín by mali byť frekventované.

6.2 Popis algoritmu

V popise algoritmu je použitá notácia pre jednotlivé SQL operácie, ktorá je vysvetlená v Tab. 6.1.

Notation	**Operation**
$R1 \cup R2$	UNION
$\sigma_{<\text{selection condition}>}(R)$	SELECT
$\pi_{<\text{attribute list}>}(R)$	PROJECTION
$R1 \bowtie_{<\text{join condition}>} R2$	JOIN
$<\text{grouping attributes}> \Im <\text{function list}>$	AGGREGATE FUNCTION
$\rho_P(R)$	RENAME (table name)
$\rho_{(b1,b2,...,bn)}(R)$	RENAME (attributes)

Tab. 6.1: Použitá notácia

Nech A, B, C reprezentujú tabuľky schémy hviezda a FT je tabuľka faktov, bez straty na všeobecnosti. Nech a, b, a c sú kľúče (ID) tabuliek dimenzií A, B a C v danom poradí.

A

a	i1	i2	i3
a0	0	4	7
a1	2	5	6
a2	1	3	6
a3	0	4	8
a4	1	3	7

FT

a	b	c
a2	b3	c2
a3	b2	c3
a0	b3	c3
a2	b2	c3
a0	b2	c3
a3	b0	c1
a1	b2	c4
a4	b2	c3
a4	b0	c2
a4	b0	c3

B

b	i1	i2	i3
b0	9	12	15
b1	10	12	16
b2	11	13	15
b3	11	14	17
b4	10	12	15

C

c	i1	i2	i3
c0	18	23	25
c1	19	21	25
c2	19	23	24
c3	20	22	25
c4	20	22	26

Obr. 6.1: Schéma hviezda

Potom frekventované množiny nad tabuľkou T ($T=FT \bowtie A \bowtie B \bowtie C$ on FT.a=A.a AND FT.b=B.b AND FT.c=C.c) budú dolované. Ďalej predpokladáme, že atribúty v tabuľkách dimenzií sú unikátne. V rámci každej množiny sú prvky zoradené v poradí tabuliek dimenzií, čo znamená, že prvky z tabuľky A sú nasledované prvkami z B a C.

6.2.1 Fáza 1: Dolovanie frekventovaných množín nad tabuľkou dimenzie

Nasledujúce dotazy sú použité na dolovanie frekventovaných množín, ktorých prvky patria do tabuľky dimenzií A, za predpokladu, že tabuľka A obsahuje *n* atribútov okrem ID transakcie.

1. priechod

$$AI(a,i1) \leftarrow ((\ldots(\pi_{a,i1}(A) \cup \pi_{a,i2}(A)) \cup \ldots) \cup \pi_{a,in}(A))$$

$$T \leftarrow \pi_{i1,cnt}(AI \bowtie_{a=a} \rho_{(a,cnt)}({}_a\Im_{COUNT\ a}(FT)))$$

$$FA_1(i1,cnt) \leftarrow \sigma_{cnt>=:minsup}(\rho_{(i1,cnt)}({}_{i1}\Im_{SUM\ cnt}(T)))$$

$RA_1(a,i1) \leftarrow \pi_{a,i1}(AI \bowtie_{i1=i1} FA_1)$

V prvom priechode sú získané 1-prvkové frekventované množiny. Je vytvorená dvojstĺpcová tabuľka AI, kde jeden stĺpec je tvorený identifikátorom transakcie a druhý položkou. Rady sú vytvorené spojením každého nekľúčového atribútu tabuľky A s jeho kľúčom alebo identifikátorom transakcie. Počet výskytov hodnoty každého kľúča tabuľky A v tabuľke faktov FT je pripojený k tabuľke AI na získanie počtu výskytov každej položky tabuľky A v spojenom výsledku. Jednoprvkové frekventované množiny sú získané pomocou výberu len tých položiek, ktoré spĺňajú podmienku minimálnej podpory. Pre uloženie týchto frekventovaných množín je vytvorená tabuľka RA1.

2. priechod

$T_1 \leftarrow \rho_{(a,i1,i2)}(\pi_{P.a,P.i1,Q.i1}(\rho_P(RA_1) \bowtie_{a=a \text{ AND } i1<i1} \rho_Q(RA_1)))$

$T_2 \leftarrow \rho_{(a,cnt)}({}_a\Im_{COUNT\ a}(FT))$

$FA_2(i1,i2,cnt) \leftarrow \sigma_{cnt>=:minsup}(\rho_{(i1,i2,cnt)}({}_{i1,i2}\Im_{SUM\ cnt}(\pi_{i1,i2,cnt}(T_1 \bowtie_{a=a} T_2))))$

$T \leftarrow \rho_{(a,i1,i2)}(\pi_{P.a,P.i1,Q.i1}(\rho_P(RA_1) \bowtie_{a=a \text{ AND } i1<i1} \rho_Q(RA_1)))$

$RA_2(a,i1,i2) \leftarrow \pi_{a,i1,i2}(T \bowtie_{i1=i1 \text{ AND } i2=i2} FA_2))$

V druhom priechode je spojená tabuľka RA_1 sama so sebou na získanie kandidátnych 2-prvkových množín. Identifikátory transakcií výslednej tabuľky sú nahradené počtom ich výskytov v tabuľke FT a následne sú získané 2-prvkové frekventované množiny pomocou výberu len tých množín, ktoré spĺňajú podmienku minimálnej podpory. Podobným spôsobom sú získané frekventované *k*-prvkové množiny.

***k*-ty priechod:**

$T_1 \leftarrow \rho_{(a,i1,i2,\ldots,ik)}(\pi_{P.a,P.i1,P.i2,\ldots,P.ik-1,Q.ik-1}(\rho_P(RA_{k-1}) \bowtie_{a=a \text{ AND } i1=i1 \text{ AND } \ldots \text{ ik-2=ik-2 AND ik-1<ik-1}} \rho_Q(RA_{k-1})))$

$T_2 \leftarrow \rho_{(a,cnt)}({}_a\Im_{COUNT\ a}(FT))$

$FA_{k(i1,i2,\ldots,ik,cnt)} \leftarrow \sigma_{cnt>=:minsup}(\rho_{(i1,i2,\ldots,ik,cnt)}({}_{i1,i2,\ldots,ik}\Im_{SUM\ cnt}(\pi_{i1,i2\ldots,ik,cnt}(T_1 \bowtie_{a=a} T_2))))$

$T \leftarrow \rho_{(a,i1,i2,\ldots,ik)}(\pi_{P.a,P.i1,P.i2,\ldots,P.ik-1,Q.ik-1}(\rho_P(RA_{k-1}) \bowtie_{a=a \text{ AND } i1=i1 \text{ AND } \ldots \text{ ik-2=ik-2 AND ik-1<ik-1}} \rho_Q(RA_{k-1})))$

$RA_k(a,i1,i2,\ldots,ik) \leftarrow \pi_{a,i1,i2,\ldots,ik}(T \bowtie_{i1=i1 \text{ AND } \ldots \text{ ik=ik}} FA_k)$

Cieľom pre vytvorenie tabuľky RA_k, ktorá obsahuje všetky *k*-prvkové frekventované množiny je jednoducho nájsť (*k+1*)-prvkové frekventované množiny a množiny naprieč tabuľkami. Zvyčajne sú tabuľky dimenzií menšie ako tabuľky faktov, preto réžia na vytvorenie tabuľky RA_k je minimálna, čo sa týka vyžadovaného priestoru a času.

Táto procedúra sa opakuje so všetkými tabuľkami dimenzií. Na konci tohto kroku sú nájdené všetky frekventované množiny, ktorých prvky patria do jednej tabuľky dimenzií (FA_1, RA_1, FA_2, RA_2,.... FB_1, RB_1,....).

6.2.2 Fáza 2: Dolovanie frekventovaných množín nad tabuľkami dimenzií

Dolovanie frekventovaných množín nad tabuľkami dimenzií sa skladá z nasledujúcich krokov:

6.2.2.1 Dolovanie frekventovaných množín naprieč ľubovoľnými dvomi dimenziami

Nasledujúce dotazy sú použité na dolovanie frekventovaných množín, ktorých prvky patria do tabuliek dimenzií A a B. A_mB_n udáva množinu kandidátnych množín s *m* prvkami množiny A a *n* prvkami množiny B. FA_mB_n udáva množinu frekventovaných množín s *m* prvkami množiny A a *n* prvkami množiny B.

1. priechod:

$T_1 \leftarrow \rho_{(a,b,cnt)}({}_{a,b}\Im_{COUNT\ a,b}(FT))$

$T_2 \leftarrow \rho_{(i1,b,cnt)}({}_{i1,b}\Im_{SUM\ cnt}(\pi_{i1,b,cnt}(RA_1 \bowtie_{a=a} T_1)))$

$T_3 \leftarrow \rho_{(i1,i2,cnt)}(\pi_{P.i1,Q.i1,P.cnt}(\rho_P(T_2) \bowtie_{b=b} \rho_Q(RB_1)))$

$FA_1B_1(i1,i2,cnt) \leftarrow \sigma_{cnt>=:minsup}(\rho_{(i1,i2,cnt)}({}_{i1,i2}\Im_{SUM\ cnt}(T_3)))$

V 1. priechode sú získané také 2-prvkové frekventované množiny, ktorých prvá položka je z tabuľky A a druhá z tabuľky B. Sú získané rôzne kombinácie identifikátorov transakcií tabuľky A a tabuľky B a spojená s tabuľkou RA_1 podľa identifikátora transakcie tabuľky A. Výsledná tabuľka sa skladá z asociácií medzi položkami tabuľky A a identifikátorom transakcie tabuľky B, ktoré sú zoskupené a spojené s tabuľkou RB_1 podľa identifikátoru transakcie tabuľky B. Každá *n*-tica výslednej tabuľky sa skladá z položky tabuľky A, z položky tabuľky B a čísla vyjadrujúceho, koľkokrát sa vyskytujú spoločne v spojenom výsledku, z ktorého sú frekventované množiny získané.

2. priechod

$T \leftarrow \rho_{(i1,i2,i3)}(\pi_{P.i1,P.i2,Q.i2}(\rho_P(FA_1B_1) \bowtie_{i1=i1\ AND\ i2<i2} \rho_Q(FA_1B_1)))$

$A_1B_2(i1,i2,i3) \leftarrow \pi_{i1,i2,i3}(T \bowtie_{i2=i1\ AND\ i3=i2} FB_2)$

$T_1 \leftarrow \rho_{(a,b,cnt)}({}_{a,b}\Im_{COUNT\ a,b}(FT))$

$T_2 \leftarrow \rho_{(i1,b,cnt)}({}_{i1,b}\Im_{SUM\ cnt}(\pi_{i1,b,cnt}(\pi_{a,i1}(RA_1 \bowtie_{i1=i1} \pi_{i1}(A_1B_2)) \bowtie_{a=a} T_1)))$

$T_3 \leftarrow \pi_{b,i1,i2}(RB_2 \bowtie_{i1=i2\ AND\ i2=i3} \pi_{i2,i3}(A_1B_2))$

$T_4 \leftarrow \rho_{(i1,i2,i3,cnt)}(\pi_{P.i1,Q.i1,Q.i2,P.cnt}(\rho_P(T_2) \bowtie_{b=b} \rho_Q(T_3)))$

$FA_1B_2(i1,i2,i3,cnt) \leftarrow \sigma_{cnt>=:minsup}(\rho_{(i1,i2,i3,cnt)}({}_{i1,i2,i3}\Im_{SUM\ cnt}(T_4)))$

V druhom priechode sú získané 3-prvkové frekventované množiny, ktorých prvky patria aj do tabuľky A, aj do tabuľky B (FA_1B_2, FA_2B_1). Kandidátne množiny (A_1B_2) pre tabuľku FA_1B_2 sú generované spojením dvoch množín s rovnakým prefixom v FA_1B_1 a potom obmedzením tabuľkou FB_2. K nájdeniu frekventovaných množín sú najprv všetky prvky tabuľky A, ktoré sú prítomné v kandidátnych množinách, získané spojením RA_1 s A_1B_2 prvkom tabuľky A. Táto tabuľka je spojená so získanou tabuľkou, ktorá obsahuje rôzne kombinácie identifikátorov transakcií A a B v FT (T_1), cez identifikátor transakcií tabuľky A (a), a výsledkom je tabuľka T_2. Všetky 2-prvkové množiny v tabuľke B, ktoré sa nachádzajú v kandidátnych množinách, sú získané spojením RB_2 s A_1B_2 cez prvky tabuľky B, a výsledok je tabuľka T_3. Tieto odvodené tabuľky T_2 a T_3 sú spojené cez identifikátor transakcií tabuľky B (b) a výsledok pozostáva z množín, ktoré majú jeden prvok z A a dva prvky z B, a počtu, koľkokrát sa nachádzajú spolu v spojenom výsledku.

$T \leftarrow \rho_{(i1,i2,i3)}(\pi_{P.i1,P.i2,Q.i2}(\rho_P(FA_2) \bowtie_{i1=i1 \text{ AND } i2<i2} \rho Q(FA_1B_1)))$

$A_2B_1(i1,i2,i3) \leftarrow \pi_{i1,i2,i3}(T \bowtie_{i2=i1 \text{ AND } i3=i2} FA_1B_1)$

$T_1 \leftarrow \rho_{(a,b,cnt)}({}_{a,b}\Im_{COUNT\ a,b}(FT))$

$T_2 \leftarrow \pi_{a,i1,i2}(RA_2 \bowtie_{i1=i1 \text{ AND } i2=i2} \pi_{i1,i2}(A_2B_1))$

$T_3 \leftarrow \rho_{(i1,a,cnt)}({}_{i1,a}\Im_{SUM\ cnt}(\pi_{i1,a,cnt}(\pi_{b,i1}(RB_1 \bowtie_{i1=i3} \pi_{i3}(A_2B_1)) \bowtie_{b=b} T1)))$

$T_4 \leftarrow \rho_{(i1,i2,i3,cnt)}(\pi_{P.i1,P.i2,Q.i1,Q.cnt}(\rho_P(T_2) \bowtie_{a=a} \rho_Q(T_3)))$

$FA_2B_1(i1,i2,i3,cnt) \leftarrow \sigma_{cnt>=:minsup}(\rho_{(i1,i2,i3,cnt)}({}_{i1,i2,i3}\Im_{SUM\ cnt}(T_4)))$

Kandidátne množiny (A_2B_1) pre FA_2B_1 sú generované spojením dvoch množín s rovnakým prefixom v FA_2 a FA_1B_1 a potom obmedzením tabuľkou FA_1B_1. Rozdiel medzi generovaním FA_1B_2 a FA_2B_1 je, že odvodená tabuľka s menším počtom prvkov je spojená s FT. Dôvodom je, že výsledná odvodená tabuľka pozostáva z asociácií medzi množinami jednej tabuľky a identifikátorom transakcií druhej tabuľky, takže sa dá zmenšiť viac, keď má menej prvkov. Preto sú na generovanie FA_2B_1 všetky prvky B, ktoré sú prítomné v kandidátnych množinách, spojené s FT cez identifikátor transakcie B (b). Táto tabuľka (T_3) je spojená s odvodenou tabuľkou, ktorá pozostáva zo všetkých prvkov dvojprvkových množín tabuľky A (T_2) cez identifikátor transakcií A (a).

***k*-ty priechod:**

V *k*-tom priechode sú získané (*k+1*)-prvkové množiny, ktorých prvky patria aj do A, aj do B (FA_1B_k, FA_2B_{k-1}, … , $FA_{k-1}B_2$, FA_kB_1). Kandidátne (*k+1*)-prvkové množiny sú generované spojením dvoch *k*-prvkových frekventovaných množín s rovnakým prefixom, a potom obmedzením použitím (*k-1*)-krát vhodných tabuliek. Obmedzením sa vykonáva kontrola, či sú všetky *k*-prvkové podmnožiny (*k+1*)-prvkovej kandidátnej množiny frekventované. Nasledujúca procedúra je použitá za predpokladu, že (*k+1*)-prvkové kandidátne množiny sú generované pre FA_mB_n, kde $(m+n)=k+1$.

```
for j := 1 to j := k
 if j = 1
    if n != 1
      T1←ρ(i1,i2, ..., ik+1)(πP.i1,P.i2, ... ,P.ik,Q.ik(ρP(FAmBn-1) ⋈i1=i1 AND
... AND ik-1=ik-1 AND ik<ikρQ(FAmBn-1)))
    else
      T1←ρ(i1,i2, ..., ik+1)(πP.i1,P.i2, ... ,P.ik,Q.ik(ρP(FAm) ⋈ i1=i1 AND ...
                                                AND ik-1=ik-1 AND ik<ikρQ(FAm-1Bn)))
 else if j ≤ m+1
    if j = 2                                  // FA0Bn => FBn
      Tj←πi1,i2 ... ,ik+1(Tj-1 ⋈ i2=i1 AND ... ik=ik-1 AND ik+1=ikFAm-1Bn)
    else
      Tj←πi1,i2 ... ,ik+1(Tj-1 ⋈ i1=i1 AND ij-2=ij-2 AND ij=ij-1 ... ik=ik-1 AND
                                                ik+1=ikFAm-1Bn)
else
    Tj←πi1,i2 ... ,ik+1(Tj-1 ⋈ i1=i1 AND ij-2=ij-2 AND ij=ij-1 ... ik=ik-1 AND ik+1=ikFAmBn-1)
```

(*k+1*)-prvkové frekventované množiny sú získané podobným spôsobom, ako bolo vysvetlené v druhom priechode. Všetky prvky množín z tabuľky A, ktoré sa nachádzajú v kandidátnych množinách, sú získané spojením RA_m s A_mB_n cez prvky tabuľky A. Všetky prvky množín z tabuľky B, ktoré sa nachádzajú v kandidátnych množinách, sú získané spojením RB_n s A_mB_n cez prvky tabuľky B. Odvodená tabuľka s menším počtom prvkov vo frekventovanej množine je spojená

cez identifikátor transakcií s projekciou FT. Výsledná odvodená tabuľka je potom spojená s predchádzajúcou odvodenou tabuľkou, a následne sú získané frekventované množiny.

$T_1 \leftarrow \rho_{(a,b,cnt)}({}_{a,b}\Im_{COUNT\ a,b}(FT))$

if $\leq$ mn

$T_2 \leftarrow \rho_{(i1,..,im,b,cnt)}({}_{i1,..,im,b}\Im_{SUM\ cnt}(\pi_{i1,..,im,b,cnt}(\pi_{a,i1,..,im}(RA_m \bowtie_{i1=i1\ AND\ \ldots\ im=im} \pi_{i1,..,im}(A_mB_n)) \bowtie_{a=a} T_1)))$

$T_3 \leftarrow \pi_{b,i1,\ldots,in}(RB_n \bowtie_{i1=im+1\ AND\ i2=im+2\ \ldots\ AND\ in=im+n} \pi_{im+1,\ldots,im+n}(A_mB_n))$

$T_4 \leftarrow \rho_{(i1,\ldots,im+n,cnt)}(\pi_{P.i1,P.i2,\ \ldots\ ,P.im,Q.i1,Q.i2,\ldots,Q.in,P.cnt}(\rho_P(T_2) \bowtie_{b=b} \rho_Q(T_3)))$

else

$T_2 \leftarrow \pi_{a,i1,i2}(RA_m \bowtie_{i1=i1\ AND\ i2=i2\ \ldots\ AND\ im=im} \pi_{i1,i2}(A_mB_n))$

$T_3 \leftarrow \rho_{(i1,\ldots,in,a,cnt)}({}_{i1,\ldots,in,a}\Im_{SUM\ cnt}(\pi_{i1,\ldots,in,a,cnt}(\pi_{b,i1,\ldots in}(RB_n \bowtie_{i1=im+1\ AND\ \ldots\ in=im+n} \pi_{im+1,\ldots,im+n}(A_mB_n)) \bowtie_{b=b} T_1)))$

$T_4 \leftarrow \rho_{(i1,\ldots,im+n,cnt)}(\pi_{P.i1,P.i2,\ \ldots\ ,P.im,Q.i1,Q.i2,\ \ldots,Q.in,Q.cnt}(\rho_P(T_2) \bowtie_{a=a} \rho_Q(T_3)))$

$FA_mB_n(i1,i2,\ldots,im+n,cnt) \leftarrow \sigma_{cnt>=:minsup}(\rho_{(i1,i2,\ldots,im+n,cnt)}({}_{i1,i2,\ldots,im+n}\Im_{SUM\ cnt}(T_4)))$

Použité dotazy sú navrhnuté tak, aby veľkosti tabuliek zahrnutých v operáciách spojenia boli maximálne redukované. V prvom optimalizačnom kroku sú odvodené spojením tabuliek R s kandidátnymi množinami ($(RA_m \bowtie \pi_{i1,..,im}(AmBn))$, $(RB_n \bowtie \pi_{im+1,..,im+n}(A_mB_n))$) len tie množiny, ktoré sú prítomné v kandidátnych množinách. Veľkosť FT je tiež redukovaná výberom len rôznych kombinácií (vrátane ich počtov) identifikátorov transakcií s príslušnými tabuľkami (${}_{a,b}\Im_{COUNT\ a,b}(FT)$). Všetky prvky množín jednej tabuľky (s menším počtom prvkov v kandidátnych množinách) sú spojené s projekciou tabuľky FT cez jej identifikátor transakcií ($(RA_m \bowtie A_mB_n) \bowtie_{a=a} \pi_{a,b}(FT)$). Výsledná odvodená tabuľka je zoskupená s ohľadom na prvky jednej tabuľky a identifikátorom transakcií druhej tabuľky ($\pi_{i1,..,im,b}((RA_m \bowtie A_mB_n) \bowtie_{a=a} \pi_{a,b}(FT))$). Tento krok redukuje veľkosť odvodenej tabuľky a optimalizuje dolovanie frekventovaných množín, obzvlášť v prvých priechodoch. Napríklad v prvom priechode po tom, čo je RA_1 spojená s FT ($RA_1 \bowtie_{a=a} \pi_{a,b}(FT)$), je výsledná odvodená tabuľka zoskupená jedným prvkom z A a identifikátorom transakcií z B. Táto tabuľka je spojená s RB_1 a sú získané 2-prvkové frekventované množiny.

V neskorších priechodoch nie sú veľkosti odvodených tabuliek produkovaných zoskupením z množín z jednej tabuľky a identifikátoru transakcií druhej tabuľky veľmi redukované. K tomuto dochádza najmä keď je počet prvkov tabuľky s menším počtom prvkov prítomný v kandidátnych množinách viac ako raz (napr. A_2B_2, A_2B_3, A_3B_3, …). Avšak veľkosti odvodených tabuliek sa redukujú zmenšovaním sa počtu generovaných kandidátov. Preto prevedenie vyššie uvedených dotazov použitých k nájdeniu frekventovaných množín z dvoch tabuliek dimenzií nezávisí výhradne od počtu transakcií v FT, kým prevedenie algoritmov, ktoré dolujú z jednoduchých tabuliek, závisí na počte generovaných kandidátov a počte transakcií v FT.

Táto procedúra sa opakuje pre všetky dvojice tabuliek dimenzií (AC, BC). Na konci tohto kroku sú získané všetky frekventované množiny súvisiace s ľubovoľnými dvomi tabuľkami dimenzií.

6.2.2.2 Dolovanie frekventovaných množín nad viac ako dvomi dimenziami

Nasledujúce dotazy sú použité na dolovanie frekventovaných množín, ktorých prvky patria do tabuliek dimenzií A, B a C.

1.priechod:

Kandidátne množiny ($A_1B_1C_1$) pre $FA_1B_1C_1$ sú generované spojením dvoch množín s rovnakým prefixom v FA_1B_1 a FA_1C_1 a potom obmedzením tabuľkou FB_1C_1.

Frekventované množiny sú nájdené najprv odvodením všetkých prvkov z A a B, ktoré sa nachádzajú v kandidátnych množinách a potom spojením tabuliek RA_1, RB_1, $A_1B_1C_1$ (kandidátne množiny) a FT (rôzne kombinácie a a b). Odvodená tabuľka je spojená s FT (kombinácie a, b a c) pre získanie asociácií medzi množinami z A a B a identifikátormi transakcií z C. Frekventované množiny sú nájdené spojením so všetkými prvkami z C (RC_1), ktoré sa nachádzajú v kandidátnych množinách.

$T \leftarrow \rho_{(i1,i2,i3)}(\pi_{P.i1,P.i2,Q.i2}(\rho_P(FA_1B_1) \bowtie_{i1=i1 \text{ AND } i2<i2} \rho_Q(FA_1C_1)))$

$A_1B_1C_1(i1,i2,i3) \leftarrow \pi_{i1,i2,i3}(T \bowtie_{i2=i1 \text{ AND } i3=i2} FB_1C_1)$

$T_1 \leftarrow \pi_{a,i1,i2}(RA_1 \bowtie_{i1=i1} \pi_{i1,i2}(A_1B_1C_1))$

$T_2 \leftarrow \pi_{b,i1,a}(RB_1 \bowtie_{b=b} \pi_{a,b}(FT))$

$T_3 \leftarrow \pi_{a,i1,i2,b}(T_1 \bowtie_{a=a \text{ AND } i2=i1} T2)$

$T_4 \leftarrow \pi_{i1,i2,c,cnt}(T_3 \bowtie_{a=a \text{ AND } b=b} \rho_{(a,b,c,cnt)}({}_{a,b,c}\Im_{\text{COUNT } a,b,c}(FT)))$

$T_5 \leftarrow \rho_{(i1,i2,c,cnt)}({}_{i1,i2,c}\Im_{\text{SUM } cnt}(T_4))$

$T_6 \leftarrow \rho_{(i1,i2,i3,cnt)}(\pi_{P.i1,P.i2,Q.i1,P.cnt}(\rho_P(T_5) \bowtie_{c=c} \rho_Q(\pi_{c,i1}(RC_1 \bowtie_{i1=i3} \pi_{i3}(A_1B_1C_1)))))$

$FA_1B_1C_1(i1,i2,i3,cnt) \leftarrow \sigma_{cnt>=:minsup}(\rho_{(i1,i2,i3,cnt)}({}_{i1,i2,i3}\Im_{\text{SUM } cnt}(T_6)))$

2. priechod:

Sú nájdené 4-prvkové frekventované množiny, ktorých prvky partia do A, B a C. Kandidátne množiny pre $FA_2B_1C_1$, $FA_1B_2C_1$, $FA_1B_1C_2$ sú generované nasledujúcimi dotazmi. Frekventované množiny sú získané podobným spôsobom ako v prvom priechode použitím príslušných tabuliek. Napríklad na generovanie $FA_2B_1C_1$ sú použité RA_2 a $A_2B_1C_1$ namiesto RA_1 a $A_1B_1C_1$.

$T \leftarrow \rho_{(i1,i2,i3,i4)}(\pi_{P.i1,P.i2,P.i3,Q.i3}(\rho_P(FA_2B_1) \bowtie_{i1=i1 \text{ AND } i2=i2 \text{ AND } i3<i3} \rho_Q(FA_2C_1)))$

$A_2B_1C_1(i1,i2,i3,i4) \leftarrow \pi_{i1,i2,i3,i4}(\pi_{i1,i2,i3,i4}(T \bowtie_{i2=i1 \text{ AND } i3=i2 \text{ AND } i4=i3} FA_1B_1C_1) \bowtie_{i1=i1 \text{ AND } i3=i2 \text{ AND } i4=i3} FA_1B_1C_1)$

$T \leftarrow \rho_{(i1,i2,i3,i4)}(\pi_{P.i1,P.i2,P.i3,Q.i3}(\rho_P(FA_1B_2) \bowtie_{i1=i1 \text{ AND } i2=i2 \text{ AND } i3<i3} \rho_Q(FA_1B_1C_1)))$

$A_1B_2C_1(i1,i2,i3,i4) \leftarrow \pi_{i1,i2,i3,i4}(\pi_{i1,i2,i3,i4}(T \bowtie_{i2=i1 \text{ AND } i3=i2 \text{ AND } i4=i3} FB_2C_1) \bowtie_{i1=i1 \text{ AND } i3=i2 \text{ AND } i4=i3} FA_1B_1C_1)$

$T \leftarrow \rho_{(i1,i2,i3,i4)}(\pi_{P.i1,P.i2,P.i3,Q.i3}(\rho_P(FA_1B_1C_1) \bowtie_{i1=i1 \text{ AND } i2=i2 \text{ AND } i3<i3} \rho_Q(FA_1B_1C_1)))$

$A_1B_1C_2(i1,i2,i3,i4) \leftarrow \pi_{i1,i2,i3,i4}(\pi_{i1,i2,i3,i4}(T \bowtie_{i2=i1 \text{ AND } i3=i2 \text{ AND } i4=i3} FB_1C_2) \bowtie_{i1=i1 \text{ AND } i3=i2 \text{ AND } i4=i3} FA_1C_2)$

V nasledujúcich priechodoch sú generované všetky kombinácie prvkov. Na konci tohto kroku sú získané všetky frekventované množiny nad tabuľkami A, B a C. Ak schéma obsahuje štyri tabuľky dimenzií, najprv sú nájdené všetky množiny, ktorých prvky patria do ABC, ABD, ACD, BCD, a potom sú nájdené všetky množiny, ktorých prvky patria do ABCD.

Dolovanie frekventovaných množín naprieč viac ako dvomi tabuľkami dimenzií je odlišné od dolovania naprieč dvomi tabuľkami dimenzií. Predpokladá sa, že sú nájdené 3-prvkové frekventované množiny nad tabuľkami A, B a C. Všetky výskyty 2-prvkových množín tabuliek

A a B, ktoré sú v kandidátnych množinách, sú odvodené spojením tabuliek RA_1, RB_1, $A_1B_1C_1$ a FT $((RA_1 \ _{i1=i1}\pi_{i1,i2}(A_1B_1C_1))\bowtie_{a=a \ AND \ i2=i1}(RB_1\bowtie_{b=b}\pi_{a,b}(FT)))$. Odvodená tabuľka je spojená s tabuľkou FT pre získanie asociácií medzi položkami z A a B a identifikátormi transakcií z C. Veľkosť výslednej odvodenej tabuľky nie je po zostručnení príliš zmenšená, ako to bolo v prípade počiatočných priechodov dolovania frekventovaných množín nad dvomi tabuľkami dimenzií. Vo väčšine prípadov je však počet kandidátnych množín menší, a teda veľkosť odvodených tabuliek, ktoré majú byť zlúčené, je obmedzená. Frekventované množiny sú získané spojením odvodených tabuliek so všetkými výskytmi položiek z C. Takže výkonnosť nájdenia 3-prvkových frekventovaných množín závisí na počte nájdených frekventovaných množín medzi dvoma dimenziami. V prípade veľkého počtu tabuliek dimenzií a veľmi nízkej minimálnej podpory je počet nájdených 2-prvkových frekventovaných množín zo všetkých dvojíc tabuliek dimenzií veľmi veľký, a preto by bola výkonnosť dotazov degradovaná.

V neskorších priechodoch je počet generovaných kandidátov oveľa menší a frekventované množiny sú nájdené jednoducho. Podobne sú nájdené frekventované množiny, ktorých prvky patria do viacerých tabuliek dimenzií. So zvyšujúcim sa počtom tabuliek dimenzií by sa počet generovaných kandidátnych množín zmenšoval.

Jedna malá nevýhoda navrhnutého algoritmu spočíva v tom, že je generovaných viac tabuliek, pretože pre každú kombináciu prvkov v predchádzajúcich priechodoch je vytvorená tabuľka, kým nie sú nájdené ďalšie frekventované množiny. Na minimalizáciu tejto réžie sú vylúčené v nasledujúcich priechodoch kombinácie s aspoň jednou podmnožinou, ktorá nie je frekventovanou množinou. Napríklad, ak sa počet frekventovaných množín pre kombinácie $A_pB_qC_r$ rovná 0, potom v nasledujúcich priechodoch nie sú kombinácie $A_sB_tC_u$, kde $p \leq s$, $q \leq t$ a $r \leq u$, uvažované pri generovaní kandidátnych množín [9].

7 Analýza a návrh

Témou tejto kapitoly je analýza problému dolovania asociačných pravidiel pomocou algoritmu Star-miner a stručná analýza tohto algoritmu. Ďalej obsahuje návrh vlastnej aplikácie pre získanie asociačných pravidiel z dátových skladov a návrh dátového skladu, v ktorom budú uložené testovacie dáta pre dolovanie.

7.1 Analýza algoritmu

Popisom algoritmu Star-miner sme sa zaoberali v predchádzajúcej kapitole. Vieme, že sa jedná o veľmi efektívny a výkonný algoritmus, ktorý slúži na dolovanie asociačných pravidiel z dátových skladov a pracuje s databázou, ktorá je navrhnutá na základe databázovej schémy hviezda. Na začiatku analýzy uvedeného algoritmu boli najväčšie problémy s tým, že algoritmus je popísaný pomocou formálneho zápisu, ktorý na prvý pohľad vypadá dosť zložito. Samozrejme po pochopení použitej notácie pre jednotlivé SQL operácie už to bolo oveľa jednoduchšie.

Algoritmus je rozdelený na niekoľko krokov, v ktorých sú získané frekventované množiny. Pracujeme s multidimenzionálnou databázou, takže výsledkom budú multidimenzionálne asociačné pravidlá, ktoré majú nasledujúci tvar:

$$\textit{vek}(X, \text{“20...29”}) \wedge \textit{zamestnanie}(X, \text{“študent”}) \Rightarrow \textit{kúpi}(X, \text{“notebook”})$$

Ako vidíme, toto pravidlo má tri predikáty, ktoré reprezentujú jednotlivé dimenzie. Algoritmus Star-miner umožňuje získať okrem frekventovaných množín, ktoré sú získané z viacerých dimenzií, aj frekventované množiny z jednej dimenzie vďaka tomu, že pracuje v niekoľkých fázach. Takže môžeme vydolovať nielen multidimenzionálne asociačné pravidlá, ale aj jednodimenzionálne či medzidimenzonálne asociačné pravidlá.

Väčšina algoritmov pre dolovanie asociačných pravidiel predpokladá, že dáta sú uložené v systémových súboroch a vyžaduje špeciálne dátové štruktúry. Okrem toho tieto algoritmy vyžadujú načítať dáta z dátového skladu a uložiť ich. Algoritmus Star-miner je navrhnutý tak, aby sa dal naimplementovať na strane databázového systému pomocou SQL dotazov, a preto je oveľa efektívnejší a rýchlejší. Menšou nevýhodou je, že v každom kroku algoritmu je potrebné vytvoriť pomocné tabuľky, ktorých môže byť pri väčších databázach veľa a môžu byť dosť veľké. Ich počet závisí od počtu dimenzií a počtu prvkov frekventovaných množín. Niektoré pomocné tabuľky nie sú v ďalších krokoch algoritmu potrebné, preto sa môžu priebežne vymazať alebo sa vymažú na konci všetky naraz.

7.2 Návrh aplikácie

Cieľom tejto práce je navrhnúť aplikáciu, ktorá bude využívať zvolený algoritmus pre dolovanie asociačných pravidiel z dátových skladov. Jadrom aplikácie bude teda algoritmus Star-miner, pomocou ktorého budú získané *k*-prvkové frekventované množiny, ktoré majú spĺňať podmienku minimálnej podpory. Z týchto množín budú potom získané asociačné pravidlá na základe minimálnej spoľahlivosti. Hodnotu minimálnej podpory a minimálnej spoľahlivosti si môže zvoliť užívateľ aplikácie. Okrem týchto hodnôt bude potrebné nastaviť aj ďalšie parametre, ako napríklad počet dimenzií, počet prvkov frekventovaných množín, atď.

Vstupom aplikácie budú dáta uložené v dátovom sklade, výstupom bude tabuľka asociačných pravidiel, ktoré sme získali na základe zadaných parametrov. Pre overenie funkčnosti by postačila aj jednoduchá konzolová aplikácia, ktorá by jednoducho vypísala získané asociačné pravidlá. Ale manipulácia so zadávaním parametrov by bola oveľa obtiažnejšia, pretože by sme museli zadať okrem hodnoty minimálnej podpory a spoľahlivosti aj počet dimenzií, názvy jednotlivých dimenzií, atď. Pri 1-2 parametroch by to ešte bolo prijateľné, ale pri väčšom počte parametrov pre bežného užívateľa to už môže byť komplikované. Preto bude rozumnejšie vytvoriť aplikáciu s grafickým užívateľským rozhraním, ktoré bude jednoducho ovládateľné nielen pre skúsených užívateľov. Aby práca s aplikáciu bola ešte jednoduchšia, budú použité ovládacie prvky formulárov Windows (combobox, radiobutton, trackbar, atď.), pomocou ktorých sa dajú veľmi jednoducho nastaviť vstupné parametre pre dolovanie. Získané asociačné pravidlá budú potom vypísané do tabuľky.

Parametre, ktoré môžeme nastaviť, budú nasledujúce:

- minimálna podpora,
- minimálna spoľahlivosť,
- počet dimenzií,
- počet prvkov frekventovaných množín,
- dimenzie, z ktorých chceme dolovať,
- rozloženie prvkov vo frekventovanej množine (znamená, z ktorej dimenzie koľko prvkov bude vo výsledku).

Asociačné pravidlá, ktoré spĺňajú podmienku minimálnej podpory a minimálnej spoľahlivosti, budú výsledkom dolovania a budú zobrazené na obrazovke. Ale aby sme mohli výsledky aj uchovávať pre prípadné ďalšie spracovanie, bude dobré implementovať funkciu pre uloženie týchto pravidiel do súboru. Najrozumnejšie by bolo vyexportovať získané asociačné pravidlá buď do textového súboru, alebo do súboru XML.

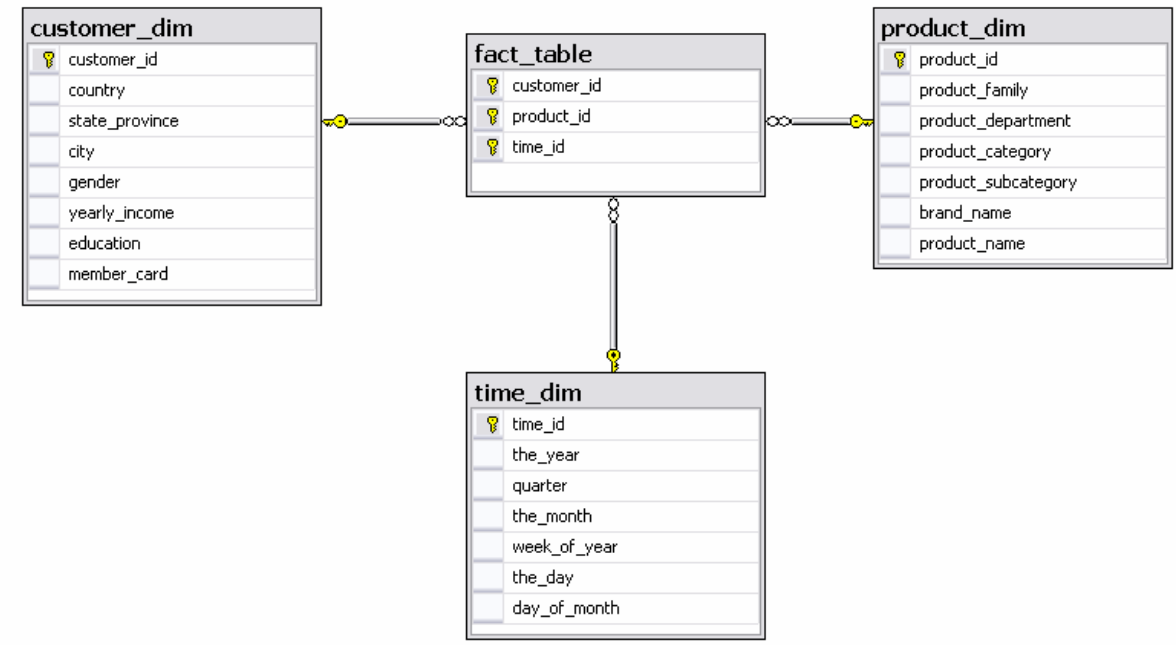
Pomocou algoritmu Star-miner sa dajú získať *k*-prvkové frekventované množiny, kde hodnota premennej „*k*“ závisí na počte atribútov jednotlivých dimenzií. Pri dolovaní z jednej dimenzie táto hodnota môže nadobúdať hodnoty z intervalu <1,n>, kde *n* je počet atribútov v danej dimenzii. Pri dolovaní z viacerých dimenzií závisí hodnota tejto premennej na počte dimenzií a na počte atribútov týchto dimenzií. V našej aplikácii bude počet prvkov frekventovaných množín limitovaný na štyri. Dôvodom je, že počet vlastných podmnožín tých množín, ktoré majú viac prvkov ako štyri, by bol dosť veľký, a preto by generovanie týchto podmnožín bolo už obtiažne.

7.3 Návrh databázy

Pre overenie funkčnosti navrhnutej aplikácie bude potrebné vytvoriť dátový sklad, ktorý bude obsahovať multidimenzionálne dáta. Databázu máme navrhnúť na základe vlastností schémy hviezda a naplniť vhodnými dátami pre dolovanie. Ako to už bolo v predchádzajúcich kapitolách vysvetlené, schéma hviezda sa skladá z tabuľky faktov a niekoľkých tabuliek dimenzií. Táto schéma je zobrazená na Obr. 6.1, pomocou ktorej sme popísali algoritmus Star-miner. Pre overenie funkčnosti algoritmu by postačila aj takáto malá databáza, ale zaujímavejšie asociačné pravidlá môžeme získať z takej databázy, ktorá obsahuje oveľa väčší počet transakcií. Samozrejme aj jednotlivé dimenzie by mali mať väčší počet atribútov. Dôležité je, aby jednotlivé atribúty v tabuľkách dimenzií boli unikátne.

Návrhári algoritmu Star-miner testovali naimplementovaný algoritmus pomocou synteticky generovanej databázy. Okrem tabuľky faktu mali tri tabuľky dimenzií, v každej bolo 1000 transakcií. Počet atribútov v každej dimenzii bol 10 a hľadali najviac 4-prvkové frekventované množiny.

Najväčšia tabuľka, čo je samozrejme tabuľka faktov, obsahovala stotisíc transakcií. Tabuľky dimenzií s podobnými parametrami môžeme vytvoriť aj my, ale pre demonštráciu funkčnosti algoritmu bude postačujúce naplniť tabuľku faktov iba pár tisíc transakciami.



Obr. 7.1: Schéma databázy

Aby sme nemuseli vygenerovať dáta do databázy, vychádzali sme z databázy Foodmart2000. Jedná sa o cvičnú databázu, ktorá obsahuje dáta s niekoľko stotisíc transakciami. Táto databáza má viac ako 20 tabuliek, z ktorých niektoré sú tabuľky faktov. Z týchto nám bude stačiť vybrať pár tabuliek dimenzií a jednu tabuľku faktov.

Schéma navrhnutej databázy je na Obr. 7.1. Ako vidíme, centrálnou tabuľkou je tabuľka fact_table, čiže tabuľka faktov. Primárnym kľúčom tejto tabuľky je zložený kľúč, ktorý sa skladá z cudzích kľúčov tabuliek dimenzií. Tabuľky dimenzií sú nasledujúce:

- **customer_dim:** obsahuje dáta o zákazníkoch,
- **product_dim:** obsahuje informácie o jednotlivých produktoch,
- **time_dim:** v ktorej sú dáta o času kúpy produktov.

Cvičná databáza Foodmart2000 obsahuje okrem týchto dát ešte aj informácie o obchodoch, zamestnancoch, ale tie nebudú pre nás dôležité. Pre vytvorenie dimenzie customer_dim bola použitá tabuľka customer. Ďalšou tabuľkou je product_dim, ktorú sme vytvorili spojením dvoch tabuliek – product a product_class. Samozrejme tieto tabuľky majú veľa atribútov, z ktorých si vyberieme iba niekoľko. Poslednou tabuľkou dimenzií je time_dim, ktorá bola vytvorená na základe databázovej tabuľky time_by_day.

Posledným krokom je vytvoriť tabuľku faktov, ktorá spojí všetky dimenzie a vytvára tak vzťahy medzi nimi. Z databázy Foodmart2000 sme použili tabuľku faktov sales_fact_1998.

Atribúty v tabuľkách dimenzií sú usporiadané do hierarchie, ktorá je nasledujúca:

dimenzia zákazníkov:

▪ country – štát
▪ ▪ state_province – provincia
▪ ▪ ▪ city – mesto
▪ ▪ ▪ ▪ gender – pohlavie
▪ ▪ ▪ ▪ ▪ yearly_income – ročný príjem zákazníka
▪ ▪ ▪ ▪ ▪ ▪ education – vzdelanie
▪ ▪ ▪ ▪ ▪ ▪ ▪ member_card – členská karta

dimenzia produktov:

▪ product_family – druh produktu
▪ ▪ product_department – oddelenie
▪ ▪ ▪ product_category – kategória produktu
▪ ▪ ▪ ▪ product_subcategory – podkategória produktu
▪ ▪ ▪ ▪ ▪ brand_name – výrobca
▪ ▪ ▪ ▪ ▪ ▪ product_name – názov produktu

dimenzia času:

▪ the_year – rok
▪ ▪ quarter – štvrťrok
▪ ▪ ▪ the_month – mesiac
▪ ▪ ▪ ▪ week_of_year – týždeň v roku
▪ ▪ ▪ ▪ ▪ the_day – deň v týždni
▪ ▪ ▪ ▪ ▪ ▪ day_of_month – deň v mesiaci

Po vytvorení všetkých tabuliek je potrebné naplniť databázu dátami. Počet položiek a nekľúčových atribútov v jednotlivých tabuľkách dimenzií a v tabuľke faktov je:

- **customer_dim:** 7 atribútov, 3000 položiek,
- **product_dim:** 6 atribútov, 1560 položiek,
- **time_dim:** 6 atribútov, 730 položiek,
- **fact_table:** 5000 transakcií.

Pre dolovanie zaujímavých asociačných pravidiel takýto počet transakcií určite bude postačujúci.

8 Implementácia

Pre implementáciu navrhnutej aplikácie bolo potrebné vybrať programovací jazyk, ktorý podporuje vytvorenie jednoduchého grafického užívateľského rozhrania a komunikáciu s nejakým databázovým serverom, na ktorom budú umiestnené dáta pre dolovanie. Ako najrozumnejším a najlepším výberom pre realizáciu týchto úloh bol programovací jazyk C#. C# je objektovo orientovaný programovací jazyk, vyvinutý firmou Microsoft zároveň s platformou .NET Framework. Je založený na jazykoch C++ a Java a je nepriamym potomkom jazyka C, z ktorého čerpá syntax. C# je možné využívať k tvorbe databázových programov a formulárových aplikácií vo Windows, takže bol ideálnou alternatívou pre vytvorenie našej aplikácie. Ako databázový systém bol použitý relačný databázový systém Microsoft SQL Server 2005.

8.1 Vytvorenie databázy

Aby sme mohli aplikáciu otestovať, je potrebná databáza, v ktorej sú uložené testovacie dáta. Preto prvým krokom implementácie navrhnutej aplikácie bolo vytvorenie databázy, teda dátového skladu, pomocou databázového systému MSSQL Server. V predchádzajúcej kapitole bola podrobne popísaná schéma našej databázy, ktorá bola navrhnutá na základe databázovej schémy hviezda a cvičnej databázy Foodmart2000. Po vytvorení prázdnej databázy bolo potrebné nastaviť práva na strane servera, pretože podľa implicitných nastavení pre užívateľov nie sú povolené operácie ako vytvorenie novej tabuľky, pohľadu, či zmazanie tabuliek. Pri implementácii algoritmu tieto operácie budeme potrebovať, pretože je potrebné vytvoriť buď pomocné tabuľky alebo pohľady, ktoré budú obsahovať dočasné dáta, a potom ich postupne alebo naraz vymazať.

Cvičná databáza Foodmart2000 je dostupná ako Access Database (.mdb), na ktorú sa dá bez problémov pripojiť pomocou MSSQL Server. Po pripojení bolo potrebné iba vybrať pomocou projekcie tie atribúty jednotlivých tabuliek, ktoré sme určili pri návrhu databázy. Na vytvorenie databázy a jej naplnenie dátami bol vytvorený sql skript.

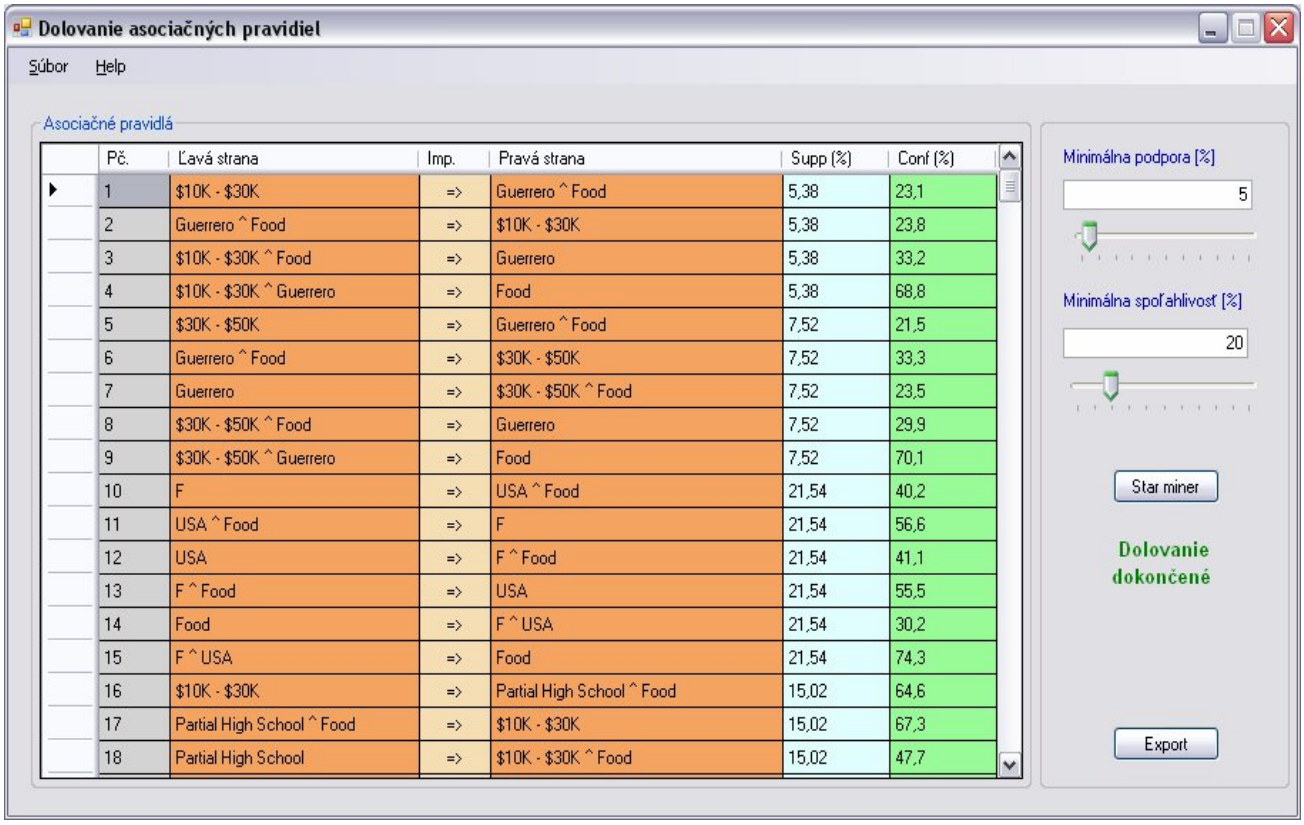
8.2 Implementácia aplikácie

Aplikácia sa skladá z piatich tried: Program.cs, ApplicationWindow.cs, PropertiesDialog.cs, StarMiner.cs a ExportXML.cs. Trieda Program.cs slúži pre spustenie aplikácie – obsahuje funkciu Main(), v ktorej je vytvorená nová inštancia triedy ApplicationWindow, ktorá vytvorí grafické užívateľské rozhranie aplikácie, teda vstupné okno. Toto okno obstaráva komunikáciu s užívateľom. Slúži na zadávanie niekoľkých parametrov pre dolovací algoritmus a výsledné asociačné pravidlá sú taktiež zobrazené v tomto okne. Ďalšia trieda, PropertiesDialog.cs vytvorí malé dialógové okno, v ktorom sa zadávajú ďalšie parametre algoritmu – počet dimenzií, počet prvkov frekventovaných množín. Po zadaní týchto parametrov sa vytvorí inštancia triedy StarMiner.cs, v ktorej je implementovaný celý algoritmus Star-miner. Táto trieda tvorí jadro celej aplikácie. Komunikuje s databázou, vytvorí pomocné tabuľky počas behu algoritmu a získa frekventované množiny na základe vstupných parametrov. Z týchto množín potom vytvára asociačné pravidlá a do hlavného okna vráti iba tie, ktoré spĺňajú podmienku minimálnej spoľahlivosti. Trieda ExportXML.cs slúži pre vyexportovanie získaných asociačných pravidiel do XML súboru.

8.2.1 Trieda ApplicationWindow.cs

Jedná sa o okno aplikácie, ktoré tvorí grafické užívateľské rozhranie. Okno je rozdelené na dve časti. Na pravej strane sú umiestnené ovládacie prvky pre nastavenie rôznych parametrov a tlačidlá na spustenie dolovacieho procesu či export výsledkov. Hlavnými parametrami pri hľadaní asociačných pravidiel sú minimálna podpora a minimálna spoľahlivosť. Hodnoty týchto premenných je možné zadať do textových polí alebo ich zmeniť pomocou trackbarov. Tieto hodnoty sú zadané v percentách a môžu nadobúdať hodnoty z intervalu <1,100>. Počet nájdených asociačných pravidiel veľmi závisí na týchto hodnotách. Hlavne pri väčších databázach zvýšením minimálnej podpory počet výsledných asociačných pravidiel výrazne klesá.

Dolovací algoritmus Star-miner sa spustí po kliknutí na tlačidlo Star miner. Pred spustením tohto procesu je potrebné nastaviť ďalšie vstupné parametre v dialógovom okne Nastavenia.



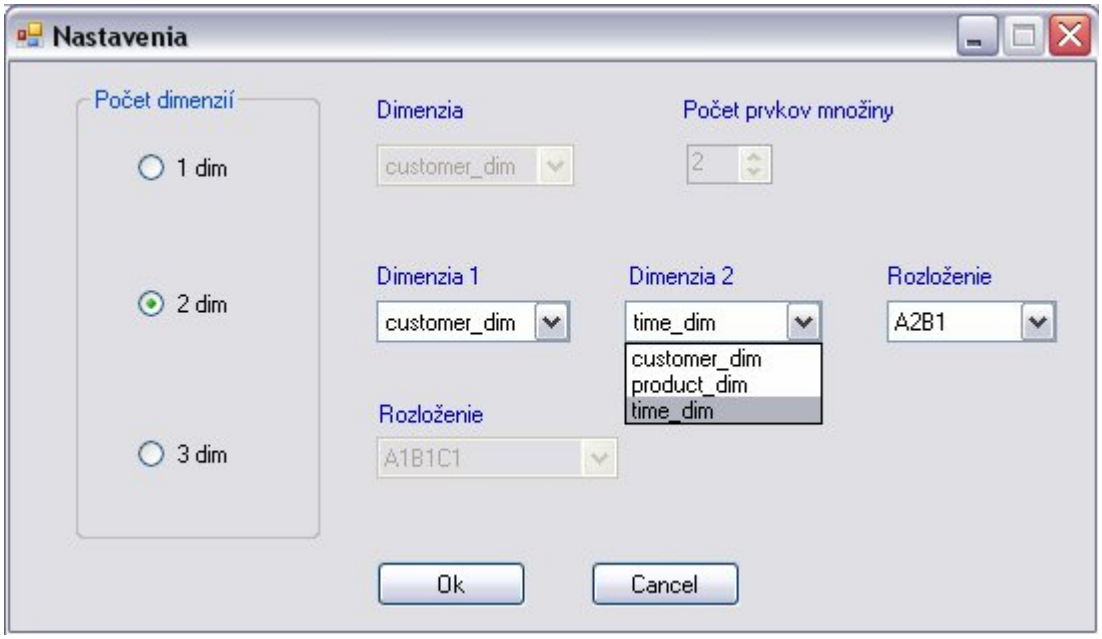
Obr. 8.1: Hlavné okno aplikácie

V druhej časti okna je umiestnený jeden DataGridView komponent, ktorý bude slúžiť ako tabuľka pre získané asociačné pravidlá. Je vypísané poradové číslo asociačného pravidla, ľavá a pravá strana pravidla a vypočítané hodnoty jeho podpory a spoľahlivosti. Výhodou tohto komponentu je, že jeho obsah môžeme zoradiť podľa ľubovoľného stĺpca výslednej tabuľky.

Beh algoritmu môže trvať aj niekoľko minút. Doba výpočtu závisí od vstupných parametrov a od veľkosti databázy. Počas dolovacieho procesu vidíme iba prázdne okno a výpis, ktorý upozorňuje užívateľa, že práve prebieha výpočet. Po ukončení behu algoritmu sú zobrazené asociačné pravidlá a upozornenie, že dolovanie je ukončené.

8.2.2 Trieda PropertiesDialog.cs

Je to vlastne dialógové okno, ktoré slúži na zadávanie ďalších dôležitých parametrov dolovacieho procesu. Ako to bolo popísané pri analýze algoritmu, môžeme dolovať z jednej dimenzie, získať asociačné pravidlá z dvoch dimenzií, alebo použiť všetky tri dimenzie. Výber počtu dimenzií je realizovaný pomocou RadioButtonov. Podľa vybraného počtu dimenzií môžeme ešte pomocou ďalších ovládacích prvkov vybrať konkrétnu dimenziu, zadať počet prvkov frekventovaných množín alebo rozloženie prvkov v získaných frekventovaných množinách.



Obr. 8.2: Dialógové okno Nastavenia

Napríklad pri dolovaní z troch dimenzií bude rozloženie A2B1C1 znamenať, že budeme hľadať 4-prvkové frekventované množiny a v získaných frekventovaných množinách budú prvé dva prvky z prvej dimenzie, ďalší prvok z druhej dimenzie a štvrtý prvok z poslednej. V našom prípade je poradie jednotlivých dimenzií nasledujúce: customer_dim, product_dim, time_dim. Na Obr. 8.1 je zobrazené dialógové okno Nastavenia. V prípade, keď chceme získať asociačné pravidlá z dvoch dimenzií, musíme vybrať konkrétne dimenzie. Samozrejme, keď zadáme rovnaké dimenzie, algoritmus nie je spustený a užívateľ je upozornený pomocou chybového hlásenia, že vybral tie isté dimenzie.

8.2.3 Trieda StarMiner.cs

V tejto triede je implementovaný celý algoritmus Star-miner, ktorý je podrobne popísaný v šiestej kapitole. Asociačné pravidlá budú uložené do komponentu DataTable, obsah ktorého bude predaný do hlavného okna a zobrazený ako výsledok dolovacieho kroku. Na začiatku je táto tabuľka výsledných asociačných pravidiel inicializovaná a sú vytvorené všetky jej stĺpce. Aby bola aplikácia univerzálnejšia a dala sa používať na rôzne databázy, údaje o jednotlivých dimenziách sú uložené v premenných. Napríklad atribúty každej dimenzie s niekoľkými ďalšími pomocnými údajmi sú uložené v dvojrozmernom poli:

```
string[][] dimensions = new string[][] {
      new string[] { "customer_dim", "AI", "a", "customer_id", "country",
      "state_province", "city",  "gender", "yearly_income", "education",
      "member_card" },
      .
      .
      .
};
```

Prvá hodnota označuje názov dimenzie, druhou hodnotou je názov tabuľky, ktorá je vytvorená ako prvá na začiatku prvej fázy. Je to dvojstĺpcová tabuľka, kde jeden stĺpec je tvorený identifikátorom transakcie a druhý položkou. Rady sú vytvorené spojením každého nekľúčového atribútu tabuľky s jeho kľúčom alebo identifikátorom transakcie. Tretia hodnota slúži iba na premenovanie kľúčového atribútu (id transakcie) danej dimenzie. Tieto hodnoty musia byť unikátne, aby sa s nimi dalo pracovať bez problémov. Ostatné hodnoty v poli sú atribúty dimenzie spolu s kľúčovým atribútom. Takto sa dajú jednoducho nadefinovať nové dimenzie, z ktorých chceme získať asociačné pravidlá. Samozrejme, musíme zadať aj názov tabuľky faktov.

Celý algoritmus pracuje pomocou SQL dotazov, takže dolovací proces vyžaduje častú komunikáciu s databázou. Algoritmus je navrhnutý tak, že počas jednotlivých fáz sú vytvorené pomocné tabuľky, ktoré sú postupne pospájané s ďalšími tabuľkami.

Na komunikáciu s databázou slúžia knižnice ADO.NET, ktoré sa dajú používať s dvomi koncepčne jedinečnými štýlmi: pripojeným alebo odpojeným. Keď budeme používať pripojenú vrstvu, bude sa báza kódu explicitne pripojovať k podkladovému úložišťu dát a odpojovať sa od neho. Pri tejto možnosti typicky komunikujeme s úložišťom dát pomocou objektov pripojenia, príkazov a čitateľov dát. Takto môžeme získavať záznamy z úložišťa prístupom, ktorý je určený iba na čítanie dát.

Oproti tomu odpojená vrstva dovoľuje získať množinu dátových tabuliek (objekty DataTable obsiahnuté vnútri nejakej sady dát, Dataset), ktorá funguje ako kópia externých dát u klienta. Keď pomocou asociovaného objektu dátového adaptéru získame sadu dát (DataSet), pripojenie sa automaticky otvára a zatvára. Po tom, ako klient získa sadu dát, môže sa po nej prechádzať a manipulovať s jej obsahom bez toho, aby mu narástli náklady spojené so sieťovou prevádzkou. Keď bude chcieť klient odoslať prevedené zmeny späť do úložišťa dát, opäť použije dátový adaptér (v súčinnosti so sadou príkazov SQL), potom aktualizuje zdroj dát a spojenie sa okamžite uzavrie.

Pomocou dátových sád by sa dal implementovať aj náš algoritmus. Na začiatku by bolo potrebné načítať potrebné tabuľky dimenzií a tabuľku faktov a potom pracovať s DataSetmi. Ale pretože algoritmus vyžaduje spojiť vytvorené databázové tabuľky podľa zložitých podmienok, a okrem toho používať rôzne agregačné funkcie, oveľa efektívnejšie bude pracovať s tabuľkami na strane databázového servera. Tieto operácie sú v databázovom systéme optimalizované, a preto je ich vykonávanie oveľa rýchlejšie.

Pri implementácii bolo použité prvé riešenie, teda pripojená vrstva. Typy pripojenia .NET sú dané na základe naformátovaného pripojovacieho reťazca (ConnectionString), ktorý obsahuje dvojice názov/hodnota oddelené bodkočiarkami. Tieto informácie špecifikujú názov stroja, ku ktorému sa chceme pripojiť, požadované nastavenia týkajúce sa bezpečnosti, názov databázy na stroji a ďalšie informácie. Programátorská práca s pripojovacími reťazcami môže byť dosť obtiažna, pretože sú to veľmi dlhé reťazové literály s nie vždy jasným obsahom, v ktorých sa veľmi ľahko robia rôzne chyby. Poskytovatelia dát ADO.NET spoločnosti Microsoft pod .NET 2.0 už poskytujú tzv. objekty tvorcov pripojovacích reťazcov (connection string builder objects), ktoré umožňujú zriadiť dvojice

názov/hodnota pomocou silne typovaných vlastností. Vytvorenie tohto reťazca je demonštrované na nasledujúcom príklade:

```
SqlConnectionStringBuilder cnStr = new SqlConnectionStringBuilder();
cnStr.UserID = "user";
cnStr.Password = "aaabbb";
cnStr.InitialCatalog = "test";
cnStr.DataSource = "(local)";
cnStr.ConnectTimeout = 0;
SqlConnection cn = new SqlConnection();
cn.ConnectionString = cnStr.ConnectionString;
```

Pripojovacie údaje sú uložené v konfiguračnom súbore conn.txt. Tento súbor sa nachádza na mieste, odkiaľ sa spúšťa aplikácia, a obsahuje štyri údaje:

- **UserID:** meno užívateľa, ktorý má prístup k databáze,
- **Password:** heslo užívateľa,
- **InitialCatalog:** databáza,
- **DataSource**.

Každý údaj musí byť na novom riadku v uvedenom poradí. Tieto údaje sú načítané do lineárneho zoznamu a po úspešnom načítaní sú na základe nich nastavené jednotlivé atribúty pripojovacieho reťazca. Keď sa súbor nenájde alebo obsahuje menej riadkov, je vypísané chybové hlásenie.

Po úspešnom načítaní parametrov pripojenia je vytvorené nové pripojenie a ostane aktívne, kým nie je dokončený celý dolovací proces. Keď konfiguračný súbor obsahuje nesprávne dáta, vytvorenie spojenia je neúspešné a je vypísané chybové hlásenie.

Keď sú zadané údaje správne a podarilo sa vytvoriť spojenie s databázou, v ktorej sú uložené dáta na dolovanie, je spustený dolovací algoritmus Star-miner. Na začiatku je inicializovaná tabuľka asociačných pravidiel. Podľa zadaných vstupných parametrov v dialógom okne Nastavenia je rozvetvený algoritmus a sú spustené príslušné metódy.

Dolovanie asociačných pravidiel pomocou tohto algoritmu je rozdelené na 3 fázy. Prvá fáza je nevyhnutná pre vykonanie druhej fázy, ktorá je zas potrebná na spustenie tretej fázy celého algoritmu, pretože využíva získané frekventované množiny z dvoch dimenzií. V prvej fáze môžeme získať 2, 3 a 4-prvkové frekventované množiny z vybranej dimenzie. Ako to už bolo niekoľkokrát spomínané, v priebehu behu algoritmu potrebujeme vytvoriť pomocné tabuľky. Pri implementácii sme najprv namiesto vytvorenia nových tabuliek vytvorili iba pohľady na dáta. Výhodou tohto postupu bolo, že pohľady v databáze nezaberajú skoro žiadne miesto, čo môže byť veľká výhoda, keď pracujeme s veľkým množstvom dát. Avšak pri testovaní funkčnosti algoritmu bolo už počas implementácie zistené, že tento princíp je síce funkčný, ale veľmi spomaľuje výpočet frekventovaných množín. Výrazné spomalenie nastalo hlavne v prípade, keď sme museli pospájať niekoľko pohľadov na základe zložitejších podmienok. Preto sme namiesto vytvorenia pohľadov zvolili druhú možnosť, čiže vytvorenie nových pomocných databázových tabuliek.

Algoritmus je založený na vytvorení týchto tabuliek a operáciách s nimi. Na vytvorenie pomocných tabuliek slúži metóda `CreateTable()`, ktorá na základe vstupných parametrov zostrojí SQL dotaz. Vytvorí sa nový čitateľ dát (SqlDataReader) a je vykonaný dotaz. Pri zložitejších dotazoch môže vytvorenie nových tabuliek trvať dosť dlho. Samozrejme to záleží na vstupných parametroch, ale hlavne na veľkosti databázy. Pri testovaní sa vyskytli problémy vtedy, keď výpočet trval viac ako 15 sekúnd. V takom prípade bolo vykonanie dotazu zrušené a vygenerovala sa chyba.

Dôvodom bolo to, že podľa pôvodných nastavení je doba, ktorú má používaný príkaz čakať, kým sa pokus ukončí, nastavená na túto hodnotu. Preto bolo potrebné pri vytvorení príkazu nastaviť parameter CommandTimeout na nulu, čo znamená, že túto dobu nastavíme na nekonečno. Po tomto nastavení už vykonanie zložitejších dotazov prebehlo bez problému.

Prvá fáza algoritmu je rozdelená na tri časti. Prvé dve sú v každom prípade vykonané, pretože na konci druhého priechodu získame 2-prvkové frekventované množiny, ktoré budeme potrebovať aj pri ostatných fázach. Tretia časť, čiže *k*-ty priechod je vykonaný iba vtedy, keď hľadáme 3-prvkové alebo 4-prvkové frekventované množiny. Pretože pre nás budú zaujímavé iba tie asociačné pravidlá, ktoré spĺňajú podmienku minimálnej podpory, bolo potrebné na základe zadanej hodnoty vypočítať, akú minimálnu hodnotu musí mať počet výskytov jednotlivých množín. Počet transakcií je rovný počtu riadkov v tabuľke faktov. Tento počet získame pomocou metódy `getItemSum()`, ktorá vykoná jednoduchý dotaz. Z počtu transakcií potom vypočítame, akú minimálnu hodnotu musí mať počet výskytov jednotlivých množín, aby bola podmienka minimálnej podpory splnená.

Ostatné dve fázy sú implementované podobným spôsobom. Metóda `Phase2pass1()` je volaná vždy. Podľa vybraného rozloženia je potom volaná buď metóda `Phase2pass2part1()` alebo `Phase2pass2part2()`. Počas implementácie tejto fázy vznikli menšie problémy s pochopením algoritmu. Z formálneho zápisu nebolo vždy na prvý pohľad jednoznačné, ako presne algoritmus funguje. Pri spojení viacerých tabuliek algoritmus začal generovať tabuľky s veľkou redundanciu, čím sme dostali nesprávny výsledok. Hľadanie chyby zabralo dosť času, ale nakoniec sa podarilo nájsť a opraviť chybné dotazy. Po oprave už algoritmus na prvý pohľad generoval správne výsledky, ale pri získaní asociačných pravidiel z frekventovaných množín sme zistili, že v niektorých prípadoch sa ešte stále vyskytujú chyby. To bolo vidieť až vtedy, keď sme začali počítať hodnotu spoľahlivosti. Ako aj minimálna podpora, aj hodnota spoľahlivosti musí nadobúdať hodnoty z intervalu <1,100>. Pri niektorých asociačných pravidlách sme ale dostali hodnoty vyššie ako 100%, čo samozrejme nie je možné. Po dôkladnejšom testovaní sa podarilo nájsť ďalšiu malú chybu, ktorá sa už dala jednoducho odstrániť. Ako sa ukázalo, po tejto oprave už algoritmus fungoval správne a dostali sme správne asociačné pravidlá so správnou hodnotou spoľahlivosti. Našťastie na väčšie chyby sme pri implementácii nenarazili.

Nasledujúci príklad ukazuje, že niekedy bolo potrebné spojiť dohromady niekoľko tabuliek a takto vznikali už trošku dlhšie a zložitejšie dotazy.

```
SELECT Y.i1 AS i1, Y.a AS a, sum(cnt) AS cnt INTO T322B0 FROM  (SELECT X.i1 AS i1, T.a AS a, T.cnt AS cnt FROM(SELECT distinct R.b AS b, R.i1 AS i1 FROM RB1 AS R JOIN A2B1 AS A ON R.i1 = A.i3) AS X JOIN T122B0 AS T ON X.b = T.b) AS Y group by Y.i1, Y.a
```

Pri písaní takých dotazov sa dá veľmi ľahko pomýliť a nájdenie chyby môže byť náročné a môže trvať dosť dlho. Ako vidíme, názvy tabuliek sú trošku nezvyčajné. Je to preto, lebo sú pomenované na základe toho, v ktorej fáze sa práve nachádzame a s ktorou dimenziou pracujeme.

Tretia fáza algoritmu sa skladá zo štyroch častí, ale vždy je vykonaná iba jedna z nich. Príslušné metódy sú volané na základe toho, aké rozloženie bolo vybrané pri spustení algoritmu. V tomto kroku pracujeme už s troma dimenziami a podľa vybraného rozloženia hľadáme buď 3-prvkové, alebo 4-prvkové frekventované množiny. Z časového hľadiska je táto fáza najnáročnejšia.

Po získaní frekventovaných množín ostáva už iba posledný krok – vytvorenie asociačných pravidiel z týchto množín. Vychádzame z výpočtu spoľahlivosti, ktorý je definovaný nasledujúcim vzorcom:

$$conf(A \Rightarrow B) = P(B|A) = \frac{s(A \cup B)}{s(A)} \qquad (8.1)$$

Na základe tejto rovnice sa postupuje pri generovaní asociačných pravidiel. Najprv je potrebné pre každú frekventovanú množinu vygenerovať všetky jej vlastné podmnožiny. Potom je vygenerované nové pravidlo, pre ktoré je potrebné podľa uvedenej rovnice vypočítať jeho spoľahlivosť. Pravidlo bude silné, keď spoľahlivosť bude vyššia než minimálna spoľahlivosť. Hodnotu minimálnej spoľahlivosti môže určiť užívateľ pred spustením dolovacieho algoritmu v hlavnom okne aplikácie. Hodnota minimálnej spoľahlivosti môže nadobúdať hodnoty z intervalu <1,100> a je daná v percentách. Generovanie asociačných pravidiel je znázornené na nasledujúcom príklade.

Nech abc je frekventovaná množina. Najprv generujeme všetky jej vlastné podmnožiny, ktoré budú: a, b, c, ab, ac, bc. Na základe týchto množín vytvoríme asociačné pravidlá, ktoré budú nasledujúce:

- a $\Rightarrow$ bc
- b $\Rightarrow$ ac
- c $\Rightarrow$ ab
- bc $\Rightarrow$ a
- ac $\Rightarrow$ b
- ab $\Rightarrow$ c

Pre každé pravidlo je potrebné vypočítať hodnotu spoľahlivosti. Berme napr. pravidlo a $\Rightarrow$ bc. Ako to vidíme z rovnice (8.1), je potrebné vypočítať podiel hodnoty podpory $a \cup bc$ a hodnoty podpory ľavej strany pravidla. Hodnotu podpory $a \cup bc$ udáva podiel počtu výskytov týchto prvkov a počtu transakcií v tabuľke faktov. Počet výskytov nájdeme v tabuľke trojprvkových frekventovaných množín. Ďalej je potrebné vypočítať podporu ľavej strany pravidla. V tomto prípade sa jedná o prvok „a“. V tabuľke 1-prvkových frekventovaných množín môžeme nájsť počet, koľkokrát sa tento prvok vyskytuje v transakciách a pomocou počtu transakcií vypočítať hodnotu podpory.

Na získanie asociačných pravidiel z frekventovaných množín je napísaných niekoľko metód. Princíp je vždy rovnaký, ale počet vlastných podmnožín závisí na počte prvkov frekventovaných množín. Počet týchto podmnožín je 2^n - 2, kde n je počet prvkov množiny. Kým pri 3-prvkových frekventovaných množinách je počet vlastných podmnožín 6, pri 4-prvkových 14, pri 5-prvkových ich je už 30. Generovanie asociačných pravidiel by už bolo v tomto prípade dosť náročné, preto sme obmedzili maximálny počet prvkov na 4, a tak užívateľ pri nastavení parametrov nemôže zadať väčšie hodnoty. Trojprvkové frekventované množiny sú generované v každej fáze a pre vytvorenie asociačných pravidiel je volaná metóda `asocPravidla3prvky()`. Premenné, ktoré obsahujú názvy jednotlivých tabuliek, sú nastavené podľa toho, v ktorej fáze sa nachádzame a z ktorých dimenzií dolujeme. Aby sme nemuseli prechádzať databázu viackrát, tabuľka frekventovaných množín je načítaná do DataSetu. Po vytvorení asociačných pravidiel je hodnota spoľahlivosti porovnaná s hodnotou minimálnej spoľahlivosti. Pomocou metódy `addRow()` sú potom do výslednej tabuľky pridané iba tie pravidlá, ktoré spĺňajú podmienku minimálnej spoľahlivosti zadanej užívateľom. Na základe tohto princípu sú získané aj asociačné pravidlá zo 4-prvkových frekventovaných množín. Ako to už bolo spomínané, v tomto prípade môžeme vytvoriť až 14 rôznych asociačných pravidiel. Počet pravidiel bude závisieť od hodnoty vypočítanej spoľahlivosti.

8.2.4 Trieda ExportXML.cs

Trieda ExportXML.cs slúži na archivovanie získaných asociačných pravidiel. Jedná sa o export tabuľky asociačných pravidiel do XML súboru. Ako vieme, XML je formát súboru obsahujúci dáta. Sila XML je najmä v jeho hierarchickej štruktúre a pomerne jednoduchom spôsobe kódovania. Umožňuje popisovať – označovať ľubovoľné dáta a prenášať ich medzi rôznymi aplikáciami a platformami. Všetky dokumenty XML spĺňajúce špecifikáciu XML musia dodržiavať isté pravidlá. Sú to napríklad:

- každý element XML musí mať začiatočnú aj koncovú značku,
- dokument XML musí obsahovať jediný pár značiek (skladajúci sa zo začiatočnej a koncovej značky) – koreňový element dokumentu, v ktorom sú všetky ostatné elementy vložené. To zaisťuje hierarchickú štruktúru dokumentu XML [14].

Ak dokument XML spĺňa tieto pravidlá, potom je správne štruktúrovaný. Pri zápise do súboru teda musíme dodržať uvedené pravidlá. Jednou z možností na export by mohlo byť vytvorenie cyklu a postupný zápis všetkých elementov do súboru. Oveľa jednoduchším spôsobom je ale použitie základnej metódy `WriteXML()` objektu DataSet. Táto metóda do zadaného súboru zapíše obsah celého DataSetu vo formáte XML. Stačí nám iba naplniť tento komponent dátami a zavolať uvedenú metódu.

Tabuľka výsledných asociačných pravidiel obsahuje šesť stĺpcov, z ktorých nám bude stačiť vyexportovať iba štyri. Stĺpec poradového čísla pravidla a stĺpec obsahujúci znak implikácie pre nás nie sú dôležité. Ako koreňový element bude `<asociacnepravidla>`, ostatné elementy budú:

- `<lavastrana>`: ľavá strana pravidla,
- `<pravastrana>`: pravá strana pravidla,
- `<support>`: hodnota podpory,
- `<confidence>`: hodnota spoľahlivosti.

Nasledujúci príklad ukazuje, ako vypadá výstupný súbor, do ktorého sú vyexportované asociačné pravidlá:

```
<asociacnepravidla>
  <pravidlo>
    <lavastrana>$10K - $30K</lavastrana>
    <pravastrana>Guerrero ^ Food</pravastrana>
    <support>5,38</support>
    <confidence>23,1</confidence>
  </pravidlo>
  <pravidlo>
    <lavastrana>$10K - $30K</lavastrana>
    <pravastrana>Partial High School ^ Food</pravastrana>
    <support>15,02</support>
    <confidence>64,6</confidence>
  </pravidlo>
  <pravidlo>
    <lavastrana>Bronze ^ Non-Consumable</lavastrana>
    <pravastrana>USA</pravastrana>
    <support>5,42</support>
    <confidence>54,5</confidence>
  </pravidlo>
</asociacnepravidla>
```

Pri exportovaní je vytvorené dialógové okno `SaveFileDialog`, pomocou ktorého užívateľ môže vybrať názov a umiestnenie súboru. V prípade, že výsledkom dolovania je prázdna tabuľka, čiže podľa zadaných parametrov sa nepodarilo vydolovať žiadne asociačné pravidlo, do vybraného súboru bude zapísaný iba koreňový element <asociacnepravidla />.

8.3 Ukážka práce algoritmu

V tejto podkapitole bude pomocou tabuliek demonštrované, ako funguje algoritmus Star-miner, ako sa tvoria jednotlivé pomocné tabuľky. V príklade budú uvedené jednotlivé kroky prvého priechodu druhej fázy algoritmu, kde sú získané dvojprvkové frekventované množiny z dvoch dimenzií a niekoľko krokov z druhého priechodu, v ktorom sa hľadajú 3-prvkové frekventované množiny.

Použitou databázou bude databáza, ktorá bola uvedená pri popise algoritmu. Pozostáva z tabuľky faktov FT a troch dimenzií – A, B a C.

FT		
a	b	c
a2	b3	c2
a3	b2	c3
a0	b3	c3
a2	b2	c3
a0	b2	c3
a3	b0	c1
a1	b2	c4
a4	b2	c3
a4	b0	c2
a4	b0	c3

a)

FT								
a			b			c		
$i1_a$	$i2_a$	$i3_a$	$i1_b$	$i2_b$	$i3_b$	$i1_c$	$i2_c$	$i3_c$
1	3	6	11	14	17	19	23	25
0	4	8	11	13	15	20	22	24
0	4	7	11	14	17	20	22	24
1	3	6	11	13	15	20	22	24
0	4	7	11	13	15	20	22	24
0	4	8	9	12	15	19	21	25
2	5	6	11	13	15	20	22	26
1	3	7	11	13	15	20	22	24
1	3	7	9	12	15	19	23	25
1	3	7	9	12	15	20	22	24

s

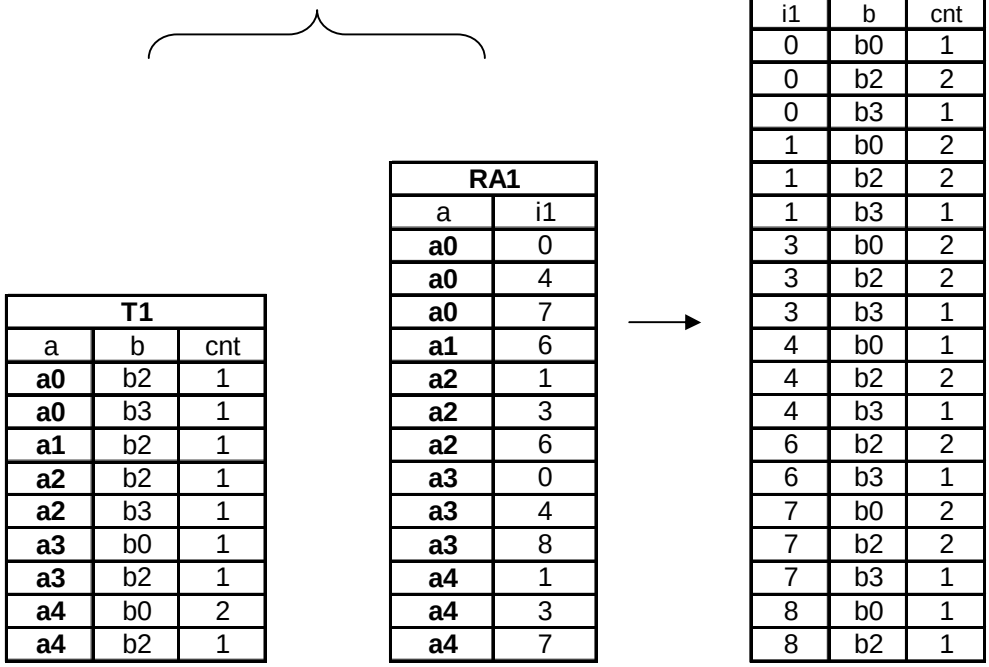
b)

Obr. 8.3: Tabuľka faktov

Na Obr. 8.3 a) je znázornená tabuľka faktov. Obsahuje iba tri atribúty, ktorými sú cudzie kľúče jednotlivých dimenzií. V databáze je tabuľka faktov uložená v uvedenej podobe. Na Obr. 8.3 b) sú už na základe kľúčových atribútov jednotlivých dimenzií uvedené všetky nekľúčové atribúty každej tabuľky dimenzie. Podľa tejto tabuľky sa dá potom jednoducho zistiť, či algoritmus funguje správne a generuje správne výsledky.

Prvý priechod druhej fázy slúži na získanie 2-prvkových frekventovaných množín. Vyberieme si dimenzie A a B. Získané frekventované množiny teda budú mať dva prvky, kde prvá položka bude z tabuľky A a druhá z tabuľky B. Prvým krokom je získanie rôznych kombinácií identifikátorov transakcií tabuľky A a tabuľky B a určiť ich počet výskytov v tabuľke faktov. Takto vznikne tabuľka T1, ktorá je znázornená na Obr. 8.4 a).

Tabuľka T2, ktorá je na Obr. 8.4 c), sa vytvorí spojením dvoch tabuliek – tabuľky RA1 s tabuľkou T1. Tabuľka RA1 bola vytvorená ešte v priebehu prvej fázy a obsahuje všetky inštancie 1-prvkových frekventovaných množín tabuľky A, spolu s identifikátormi transakcií, a je znázornená na Obr. 8.4 b). Tieto dve tabuľky sú teda spojené podľa identifikátora transakcie tabuľky A.



T1		
a	b	cnt
a0	b2	1
a0	b3	1
a1	b2	1
a2	b2	1
a2	b3	1
a3	b0	1
a3	b2	1
a4	b0	2
a4	b2	1

RA1	
a	i1
a0	0
a0	4
a0	7
a1	6
a2	1
a2	3
a2	6
a3	0
a3	4
a3	8
a4	1
a4	3
a4	7

T2		
i1	b	cnt
0	b0	1
0	b2	2
0	b3	1
1	b0	2
1	b2	2
1	b3	1
3	b0	2
3	b2	2
3	b3	1
4	b0	1
4	b2	2
4	b3	1
6	b2	2
6	b3	1
7	b0	2
7	b2	2
7	b3	1
8	b0	1
8	b2	1

a) b) c)

Obr. 8.4: Tabuľky T1, RA1 a T2

V ďalšom kroku je vytvorená tabuľka T3, ktorá vznikne spojením tabuľky T2 s tabuľkou RB1 podľa identifikátora transakcie tabuľky B. Tabuľka RB1 obsahuje 1-prvkové frekventované množiny dimenzie B a identifikátory transakcií. Spomínané tabuľky sú znázornené na Obr. 8.5 a) a Obr. 8.5 b).

RB1	
b	i1
b0	9
b2	11
b3	11
b0	12
b1	12
b2	13
b3	14
b4	12
b0	15
b2	15
b3	17
b4	15

a)

T3								
i1	i2	cnt	i1	i2	cnt	i1	i2	cnt
0	9	1	3	11	2	6	13	2
0	11	2	3	11	1	6	14	1
0	11	1	3	12	2	6	15	2
0	12	1	3	13	2	6	17	1
0	13	2	3	14	1	7	9	2
0	14	1	3	15	2	7	11	2
0	15	1	3	15	2	7	11	1
0	15	2	3	17	1	7	12	2
0	17	1	4	9	1	7	13	2
1	9	2	4	11	2	7	14	1
1	11	2	4	11	1	7	15	2
1	11	1	4	12	1	7	15	2
1	12	2	4	13	2	7	17	1
1	13	2	4	14	1	8	9	1
1	14	1	4	15	1	8	11	1
1	15	2	4	15	2	8	12	1
1	15	2	4	17	1	8	13	1
1	17	1	6	11	2	8	15	1
3	9	2	6	11	1	8	15	1

b)

Obr. 8.5: Tabuľky RB1 a T3

2-prvkové frekventované množiny sú uložené do tabuľky FA1B1. Táto tabuľka sa skladá z položky tabuľky A, z položky tabuľky B a čísla cnt, ktoré vyjadruje, koľkokrát sa tieto položky vyskytujú spoločne v spojenom výsledku. Do tejto tabuľky sú uložené iba tie *n*-tice, ktoré spĺňajú podmienku minimálnej podpory.

FA1B1					
i1	i2	cnt	i1	i2	cnt
0	11	3	4	11	3
0	13	2	4	13	2
0	15	3	4	15	3
1	9	2	6	11	3
1	11	3	6	13	2
1	12	2	6	15	2
1	13	2	7	9	2
1	15	4	7	11	3
3	9	2	7	12	2
3	11	3	7	13	2
3	12	2	7	15	4
3	13	2	8	15	2
3	15	4			

Obr. 8.6: Tabuľka FA1B1

Ako vidíme, minimálny počet výskytov jednotlivých dvojíc v tabuľke FA1B1 je 2. Je to kvôli tomu, že hodnota minimálnej podpory bola pred spustením dolovacieho algoritmu nastavená na 20%. Počet transakcií v tabuľke faktov je 10, takže do výslednej tabuľky budú uložené iba tie kombinácie položiek, ktoré sa vo všetkých transakciách vyskytujú minimálne dvakrát.

T2		
i1	b	cnt
0	b0	1
1	b0	2
3	b0	2
4	b0	1
7	b0	2
0	b2	2
1	b2	2
3	b2	2
4	b2	2
6	b2	2
7	b2	2
0	b3	1
1	b3	1
3	b3	1
4	b3	1
6	b3	1
7	b3	1

a)

FA1B2							
i1	i2	i3	cnt	i1	i2	i3	cnt
0	11	13	2	3	13	15	2
0	11	15	2	4	11	13	2
0	13	15	2	4	11	15	2
1	9	12	2	4	13	15	2
1	9	15	2	6	11	13	2
1	11	13	2	6	11	15	2
1	11	15	2	6	13	15	2
1	12	15	2	7	9	12	2
1	13	15	2	7	9	15	2
3	9	12	2	7	11	13	2
3	9	15	2	7	11	15	2
3	11	13	2	7	12	15	2
3	11	15	2	7	13	15	2
3	12	15	2				

b)

Obr. 8.7: Tabuľky T2 a FA1B2

Tabuľky T2 a FA1B1 na Obr. 8.7 a) a Obr. b) sú už vytvorené počas druhého priechodu druhej fázy. Táto etapa sa skladá z viacerých krokov, v ktorých sú okrem týchto vytvorené aj ďalšie pomocné tabuľky. Avšak väčšina z nich už obsahuje dosť veľa položiek aj pri takej malej databáze, akú sme použili na demonštráciu fungovania algoritmu, takže tieto neznázorníme.

Na začiatku druhého priechodu druhej fázy sú vygenerované 3-prvkové kandidátne množiny (jedná sa o tabuľku A1B1). V tejto tabuľke sú uložené všetky kombinácie položiek tabuliek A a B, z ktorých sú potom vybrané iba tie množiny, ktoré budú frekventované. To, ktorá množina bude frekventovaná, je zistené pomocou ďalších krokov. V našom príklade je znázornená už iba výsledná tabuľka FA1B2, ktorá obsahuje iba tie 3-prvkové množiny, ktoré sú frekventované. Vo väčšine prípadov obsahujú kandidátne množiny oveľa viac prvkov ako tabuľka frekventovaných množín a vyberajú sa z nich iba niektoré. V našom prípade bola tabuľka kandidátnych množín A1B1 totožná s tabuľkou 3-prvkových frekventovaných množín FA1B2, čo znamená, že všetky kandidátne množiny sú frekventované. Každá kombinácia položiek sa vyskytovala presne dvakrát, takže pri zadaní väčšej hodnoty minimálnej podpory by sme nezískali žiadnu frekventovanú množinu.

Počet frekventovaných množín pri 20% minimálnej podpore bol 27. Jedná sa o 3-prvkové frekventované množiny, takže z každej množiny sa dá vygenerovať maximálne 6 asociačných pravidiel. Minimálna spoľahlivosť bola nastavená taktiež na 20%. Získali sme tak 162 asociačných pravidiel, čo znamená, že všetky vygenerované pravidlá mali väčšiu hodnotu spoľahlivosti, ako nami zadaná hodnota. Pri zvýšení hodnoty minimálnej spoľahlivosti počet pravidiel klesol na 132.

9 Testovanie aplikácie

Aplikácia bola implementovaná v programovacom jazyku C# s platformou .NET framework, vo vývojovom prostredí Microsoft Visual Studio 2005. Testovanie programu bolo preto prevedené v operačnom systému MS Windows. Boli prevedené rôzne typy testov, výsledky ktorých budú uvedené v tejto kapitole a znázornené pomocou grafov.

9.1 Overenie funkčnosti

Overenie správnej funkčnosti aplikácie môže byť dosť náročné hlavne pri väčších databázach. Pri veľkom počte transakcií sa nedá jednoducho kontrolovať, či sú generované výsledky správne alebo nie. Preto bola použitá najprv malá ukážková databáza, ktorá bola uvedená pri popise algoritmu v kapitole 6. Jednotlivé tabuľky dimenzií obsahujú iba niekoľko atribútov a v tabuľke faktov je iba desať transakcií. Pri takom malom počte transakcií pri zvolení rôznych hodnôt minimálnej podpory a minimálnej spoľahlivosti sa dá jednoducho zistiť, či sme dostali správne asociačné pravidlá.

Pri implementácii sa vyskytli menšie problémy, ktoré viedli k tomu, že algoritmus začal generovať tabuľky s redundantnými hodnotami. Ale ako to bolo popísané v kapitole o implementácii, všetky chyby sa podarilo odstrániť a ako to aj testy dokázali, aplikácia funguje správne. Jednotlivé výsledky, ktoré sme dostali pri dolovaní z ukážkovej databázy, boli kontrolované a overené.

Okrem ukážkovej databázy bola aplikácia testovaná aj pomocou databázy, ktorú sme navrhli my. Pretože táto databáza obsahuje niekoľko tisíc transakcií, overenie správnosti funkčnosti algoritmu je obtiažne. Pri overení funkčnosti sme použili nasledujúci postup: z vygenerovaných asociačných pravidiel sme náhodne vybrali niekoľko. Potom sme pomocou SQL dotazov zistili počet tých transakcií, v ktorých sa vyskytujú spoločne tie položky, ktoré máme vo vybraných asociačných pravidlách. Z tohto počtu a počtu transakcií v tabuľke faktov sa dá vypočítať podpora daného asociačného pravidla. Pretože vypočítané hodnoty podpory asociačných pravidiel boli totožné s hodnotami, ktoré sme získali pomocou aplikácie, môžeme konštatovať, že algoritmus je implementovaný správne.

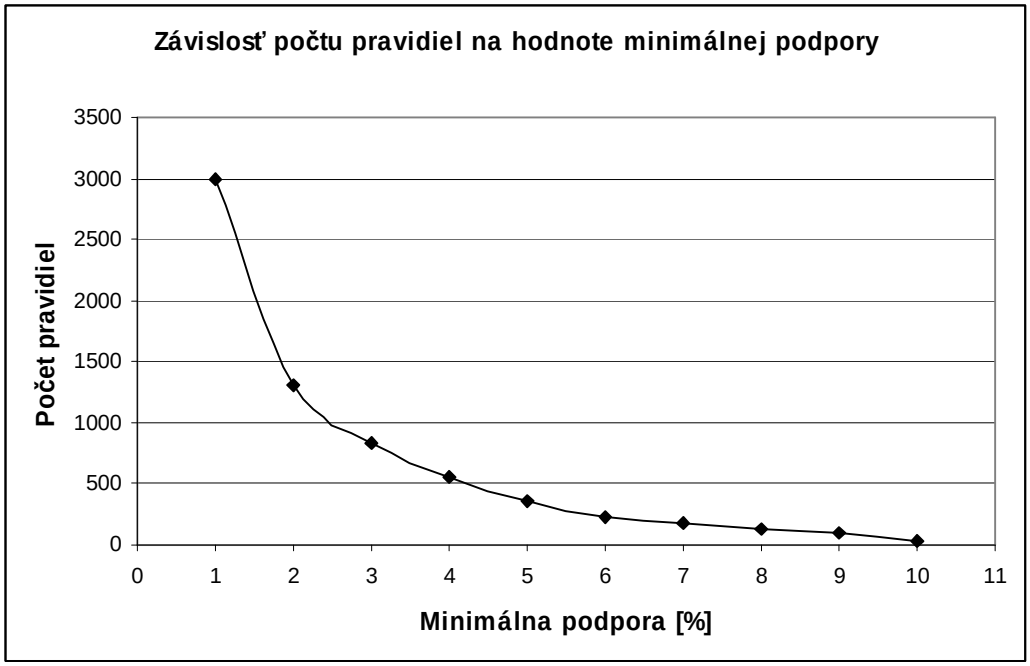
9.2 Testy počtu asociačných pravidiel

Počet asociačných pravidiel závisí na dvoch hodnotách: minimálna podpora a minimálna spoľahlivosť. Podpora vlastne udáva, ako často sa vyskytujú položky množín v transakciách spoločne. Spoľahlivosť určuje, aká bude pravdepodobnosť výskytu položiek na pravej strane pravidla, keď sa v transakcii vyskytuje aj položka na ľavej strane pravidla.

Cieľom pri týchto testoch bolo určiť závislosť počtu asociačných pravidiel na hodnotách minimálnej podpory a minimálnej spoľahlivosti. Použili sme našu testovaciu databázu, ktorá obsahuje 5000 transakcií. Najprv sme si zvolili dolovanie z troch dimenzií a hľadali sme 3-prvkové frekventované množiny. Hodnotu minimálnej spoľahlivosti sme nastavili na 10%. Pretože sa jedná o veľký počet transakcií, ktoré sú na seba nezávislé, pravdepodobne bude potrebné zadať malé hodnoty pre minimálnu podporu.

Podpora [%]	1	2	3	4	5	6	7	8	9	10
Počet	2993	1306	826	564	359	230	185	126	90	30

Tab. 9.1: Závislosť počtu pravidiel na hodnote minimálnej podpory



Graf 9.1: Závislosť počtu pravidiel na hodnote minimálnej podpory

Tab. 9.1 obsahuje zistené hodnoty počas testovania a vyjadruje závislosť počtu pravidiel na hodnote minimálnej podpory. Hodnotu minimálnej hodnoty sme nastavovali od 1% do 10%. Ako vidíme, pri väčších hodnotách minimálnej podpory počet získaných pravidiel výrazne klesá. Najmenšia hodnota, na ktorú môžeme nastaviť minimálnu podporu, je 1%. Pri tejto hodnote nám algoritmus našiel 2993 asociačných pravidiel, čo môžeme považovať za dosť veľký počet. Pri zvýšení minimálnej podpory na 2% sme dostali už iba 1306 pravidiel, čo je o 56% menej ako v predchádzajúcom kroku. Ako z tabuľky vidíme, pri 10%-nej minimálnej podpore počet vygenerovaných pravidiel klesol na 30. Po zvýšení tejto hodnoty na 13% už algoritmus žiadne asociačné pravidlo nenašiel. Zistené hodnoty sú znázornené na Grafe 9.1.

Aplikácia zobrazuje iba asociačné pravidlá, tabuľky získaných frekventovaných množiny sú pre užívateľa programu neviditeľné a po skončení behu algoritmu sú vymazané. Pri testovaní len pre zaujímavosť bola aplikácia modifikovaná tak, aby nevymazala pomocné tabuľky, čo sme využili na analýzu frekventovaných množín. Minimálnu podporu sme nastavili na 1% a pre minimálnu spoľahlivosť sme zvolili hodnotu 10%. Pri takomto nastavení v tretej fáze algoritmu bolo vygenerovaných 3498 3-prvkových kandidátnych množín. Z týchto je iba 798 frekventovaných. Pretože sa jedná o 3-prvkové množiny, z každej množiny sa dá vygenerovať 6 asociačných pravidiel. Takto by sme mohli získať maximálne 4788 asociačných pravidiel. Pri generovaní pravidiel z frekventovaných množín počítame ale aj s hodnotou minimálnej spoľahlivosti. To znamená, že asociačné pravidlá, ktoré nám aplikácia vypíše, spĺňajú obidve podmienky, a preto sú to silné asociačné pravidlá. Ako nám Tab. 9.1 ukazuje, získali sme 2993 silných asociačných pravidiel, čo je 61% z celkového počtu asociačných pravidiel.

Keď ďalej analyzujeme získané frekventované množiny, môžeme si všimnúť pár zaujímavostí. Počet frekventovaných množín je 798 a v 407 prípadoch frekventovaná množina obsahuje položku

Food. Je to aj logické, pretože ľudia najviac kupujú nejaké jedlo. V našom prípade položka Food je z dimenzie product_dim a patrí k atribútu product_family, čiže označuje druh produktu. Ak si pozrieme hierarchické usporiadanie jednotlivých atribútov v tejto dimenzii, vidíme, že atribút product_family je na jeho vrchole. Čím väčšiu minimálnu podporu zadáme, tým väčšia bude pravdepodobnosť, že nájdené frekventované množiny budú obsahovať iba také položky, ktoré patria k atribútom z vrcholu hierarchie.

Pri ostatných položkách, ktoré patria k dimenziám customer_dim a time_dim, už také výrazné rozdiely nie sú. Hoci počet transakcií v prvom štvrťroku (Q1) je dvojnásobný oproti štvrtému (Q4), medzi prvým, druhým a tretím štvrťrokom sú rozdiely nezanedbateľné. Najviac zákazníkov bolo z USA, hlavne zo štátnej provincie WA, a z Mexika. Počet zákazníkov mužského a ženského pohlavia bol skoro rovnaký. Samozrejme tieto štatistické údaje sú odvodené len zo získaných frekventovaných množín a nie z celej databázy.

Keď analyzujeme výsledné asociačné pravidlá a zoradíme ich podľa hodnoty podpory, vidíme, že najväčšiu podporu mali tie pravidlá, ktoré boli vygenerované z frekventovanej množiny {F, Food, Q1}. Prvá položka tejto množiny je z dimenzie customer_dim, patrí k atribútu gender a označuje, že zákazník je žena. Druhá položka patrí do dimenzie product_dim, teda kúpený tovar patrí do kategórie Food. Treťou položkou je Q1, čo znamená, že transakcia bola uskutočnená v prvom štvrťroku. Je teda z dimenzie time_dim. Všetky asociačné pravidlá vygenerované z tejto množiny sú silné. Dve z nich sú nasledujúce:

$$F \text{ Ù } Q1 \Rightarrow Food \quad [s = 12{,}08\%, c = 74{,}3\%]$$

$$Q1 \Rightarrow F \text{ Ù } Food \quad [s = 12{,}08\%, c = 40{,}5\%]$$

Z tejto množiny sa teda dá vygenerovať 6 silných asociačných pravidiel. Každé pravidlo má rovnakú hodnotu podpory (s = 12,08%), avšak hodnoty spoľahlivosti sa menia od 16,9% do 74,3%. Na základe hodnoty spoľahlivosti a celkového počtu transakcií sa dá vypočítať, že počet výskytov uvedenej frekventovanej množiny je 604.

Medzi generovanými asociačnými pravidlami nájdeme 57 takých, ktoré majú hodnotu podpory 1%. To znamená, že počet výskytov frekventovaných množín, z ktorých boli získané, je 50. Príkladom môže byť množina {Mexico, Snack Foods, Tuesday}. Bolo z nej vygenerované pravidlo

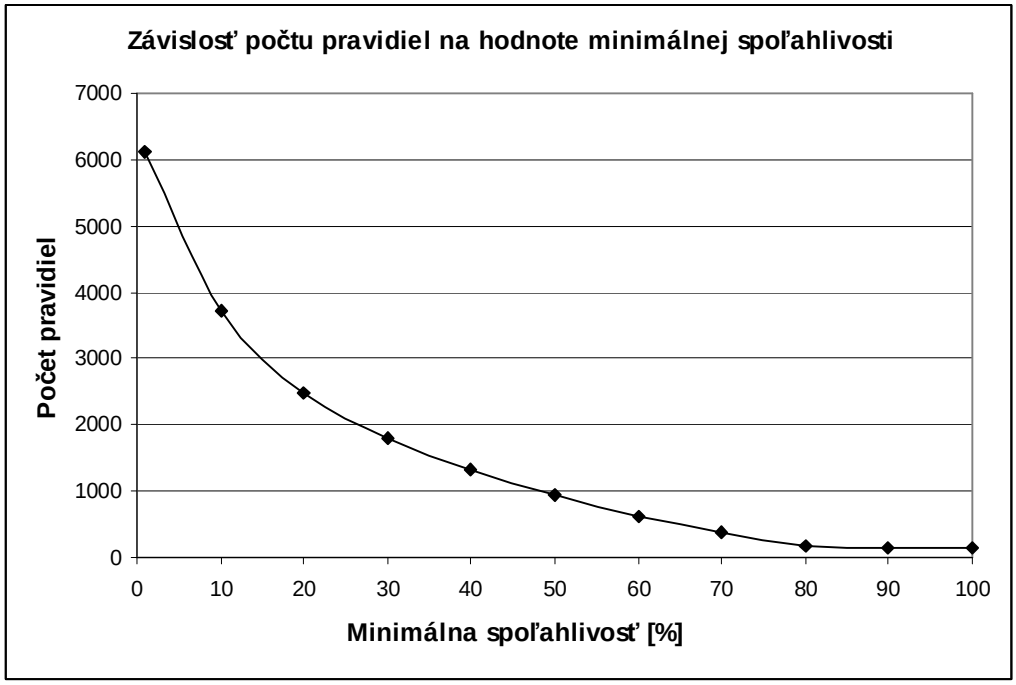
$$Mexico \text{ Ù } Tuesday \Rightarrow Snack\ Foods \quad [s = 1\%, c = 12\%].$$

Z tohto asociačného pravidla vidíme, že ak je zákazník z Mexika a tovar kúpil v utorok, patrí daný tovar do kategórie Snack Foods.

Počet výsledných asociačných pravidiel závisí teda aj na hodnote minimálnej spoľahlivosti. V predchádzajúcom príklade sme mali túto hodnotu pevne nastavenú a postupne sme menili hodnotu minimálnej spoľahlivosti. V nasledujúcom teste bude hodnota minimálnej podpory nastavená na 1% a budeme nastavovať rôzne hodnoty minimálnej spoľahlivosti.

Minimálna spoľahlivosť [%]	1	10	20	30	40	50	60	70	80	90	100
Počet	6102	3707	2485	1803	1325	946	616	388	191	154	134

Tab. 9.2: Závislosť počtu pravidiel na hodnote minimálnej spoľahlivosti



Graf 9.2: Závislosť počtu pravidiel na hodnote minimálnej spoľahlivosti

Zistené hodnoty sú zaznamenané v Tab. 9.2 a graficky znázornené na Grafe 9.2. Ako vidíme, dostali sme podobný graf ako v predchádzajúcom teste. Čím väčšiu minimálnu spoľahlivosť zadáme, tým menej asociačných pravidiel získame.

Dolovali sme z dvoch tabuliek dimenzií (customer_dim a product_dim) a rozloženie sme zvolili A2B1. Hľadali sme teda 3-prvkové asociačné pravidlá, v ktorých prvé dve položky sú z dimenzie customer_dim a tretia z dimenzie product_dim. Pri 1%-ej minimálnej spoľahlivosti algoritmus vygeneroval 6102 silných asociačných pravidiel. Pri postupnom zvyšovaní hodnoty minimálnej spoľahlivosti síce počet asociačných pravidiel klesal, ale nie tak výrazne ako v predchádzajúcom teste pri zmene minimálnej podpory.

9.3 Testy doby výpočtu algoritmu

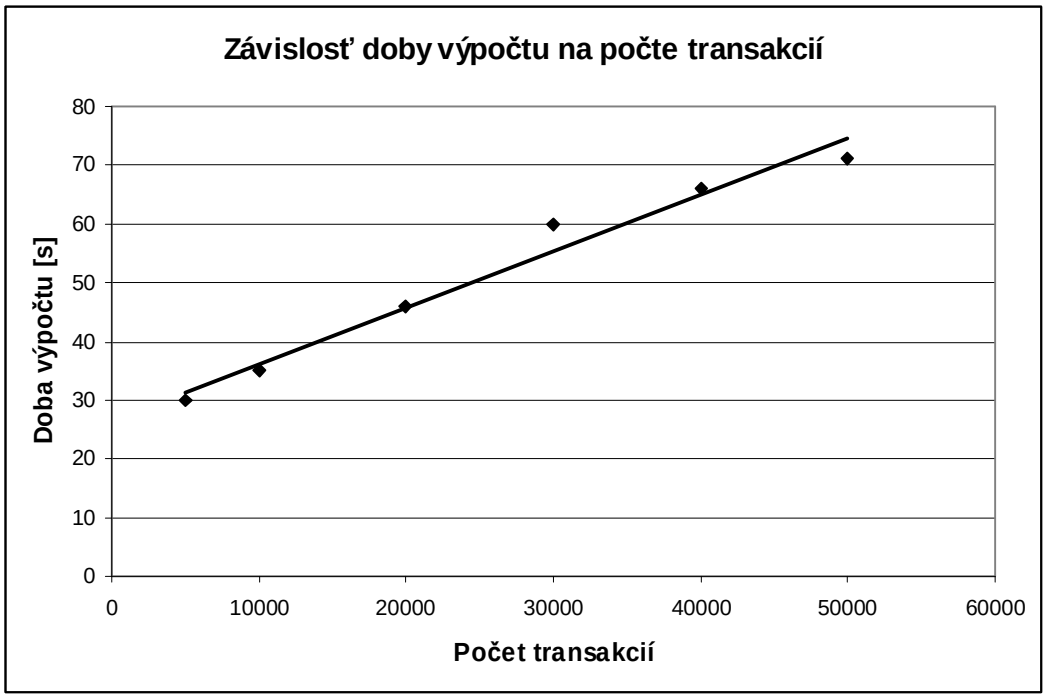
Ďalším testom bol test doby výpočtu algoritmu. Cieľom bolo zistiť, aká je závislosť medzi počtom transakcií a dobou výpočtu. Postupne sme vygenerovali rôzne ukážkové databázy podľa navrhnutého modelu a naplnili sme ich dátami z cvičnej databázy Foodmart2000. Dáta v tabuľkách dimenzií ostali nemenné, tabuľka faktov však obsahovala v každom pokuse iný počet transakcií.

Program bol pre každý test spustený s rovnakou hodnotou minimálnej podpory aj minimálnej spoľahlivosti. Obe hodnoty sme nastavili na 1%. Pri takomto nastavení algoritmus nájde najviac frekventovaných množín a vygeneruje najväčší počet asociačných pravidiel. Tým sme zaručili, aby beh algoritmu trval čo najdlhšie a tým doba výpočtu bola ľahšie merateľná. Nastavenie počtu prvkov frekventovaných množín ostalo nezmenené. Hľadali sme teda 3-prvkové frekventované množiny zo všetkých dimenzií a rozloženie sme zvolili A1B1C1.

Zistené hodnoty sú zaznamenávané v Tab. 9.3 a graficky znázornené na Grafe 9.3. Výsledky testov boli také, aké sme očakávali. Podľa Grafu 9.3 vidíme, že závislosť doby výpočtu algoritmu na počte transakcií je lineárna. V grafe sa objavujú iba malé odchýlky od lineárnej regresnej krivky.

Počet transakcií	5000	10000	20000	30000	40000	50000
Doba výpočtu [s]	30	35	46	60	66	71
Počet pravidiel	4788	4265	4008	3930	3780	3726

Tab. 9.3: Závislosť doby výpočtu na počte transakcií



Graf 9.3: Závislosť doby výpočtu na počte transakcií

Postupným zvyšovaním počtu transakcií v databáze sa doba behu algoritmu zvyšuje. Kým pri 5000 transakciách trval výpočet 30s, pri 50000 transakciách bolo už na beh algoritmu potrebných 71 sekúnd. Počet nájdených frekventovaných množín sa pritom výrazne nezmenil. Dôvodom je, že pri väčšej databáze sa zvyšuje aj minimálny počet výskytov množín na to, aby boli dané množiny frekventované. Napríklad pri databáze o veľkosti 5000 transakcií a minimálnej podpore 1% je hranica, pri ktorej je už množina frekventovaná, 50 a pri databáze o veľkosti 50000 transakcií je táto hranica pri rovnakej minimálnej podpore 500 výskytov.

10 Záver

V dnešnej dobe sa vďaka dostupnosti obrovského množstva dát v elektronickej podobe dostala do popredia potreba premeny dát na užitočné informácie a znalosti, ktoré je možné využívať v rôznych oblastiach, ako je analýza trhu alebo proces rozhodovania. Pre riešenie týchto problémov sa začali vyvíjať technológie na vytváranie dátových skladov a aplikácií rôznych dotazovacích a analytických nástrojov OLAP. Vznikla nová počítačová veda, ktorá sa nazýva získavanie znalostí z databáz alebo v súčasnosti možno ešte častejšie dolovanie dát. Jedná sa o extrakciu modelov dát, ktoré predstavujú nejakú znalosť a ktoré sú obsadené v dátach uložených v rozsiahlych databázach alebo dátových skladoch.

Cieľom teoretickej časti práce bolo poskytnúť základy o tematike dolovania asociačných pravidiel z dátových skladov. Patrí sem vysvetľovanie základných pojmov, vysvetľovanie toho, čo vlastne rozumieme pod pojmom data mining alebo ktoré sú základné typy dolovacích úloh. Dozvedeli sme sa, že jedným z často používaných zdrojov pre dolovanie dát je dátový sklad a že jeho dátovým modelom je multidimenzionálny model. Boli uvedené aj základné metódy na budovanie dátového skladu. Dáta ale nie sú vždy v takej podobe, aby sa dali využívať bez akýchkoľvek úprav pre dolovanie. Preto je dôležité ešte pred presunom dát do dátového skladu predspracovanie týchto dát – jedná sa o etapu ETL.

Ďalšia časť tejto práce bola venovaná asociačnej analýze. Jej význam bol vysvetlený pomocou analýzy nákupného košíka. Tento príklad je najčastejšie používaným príkladom na získavanie asociačných pravidiel. Boli vysvetlené pojmy ako frekventovaný vzor, frekventovaná množina a asociačné pravidlo. U asociačných pravidlách sme uviedli, aké typy asociačných pravidiel poznáme.

V rámci praktickej časti práce bolo cieľom navrhnúť aplikáciu, ktorá bude využívať zvolený algoritmus pre dolovanie dát z dátových skladov. Vybrali sme algoritmus Star-miner, ktorý slúži na získavanie frekventovaných množín z dátových skladov, ktoré sú navrhnuté na základe databázovej schémy hviezda. Tento algoritmus je implementovateľný pomocou SQL dotazov a je oveľa efektívnejší ako ostatné algoritmy slúžiace na dolovanie dát. Popisu algoritmu bola venovaná šiesta kapitola, v ktorej sú vysvetlené jednotlivé časti algoritmu, ich význam a princíp funkčnosti.

Na základe získaných teoretických znalostí bol analyzovaný problém dolovania dát pomocou algoritmu Star-miner a navrhnutá najprv ukážková databáza, potom vlastná aplikácia využívajúca tento algoritmus. Aby sme nemuseli vygenerovať syntetickú databázu, na naplnenie jednotlivých tabuliek dimenzií a tabuľky faktov dátového skladu sme použili dáta z cvičnej databázy Foodmart2000.

Aplikácia bola implementovaná v programovacom jazyku C#, databázová časť pomocou relačného databázového serveru Microsoft SQL Server 2005. Overenie jej funkčnosti bolo prevedené najprv pomocou databázy, ktorá bola použitá nielen pre testovanie aplikácie, ale aj pre vysvetľovanie fungovania algoritmu. Ukážka práce algoritmu je takisto súčasťou práce. Pretože táto databáza je veľmi malá a obsahuje iba niekoľko transakcií, jednoducho sa dala overiť správnosť funkčnosti algoritmu. Testy prebehli úspešne a bolo dokázané, že algoritmus je správne naimplementovaný a generuje správne výsledky.

Okrem testov na dokázanie funkčnosti algoritmu boli prevedené aj ďalšie testy, pre ktoré už bola použitá navrhnutá ukážková databáza s reálnymi transakciami. Pomocou testov zameraných na odhalenie závislostí medzi počtom získaných pravidiel a hodnotami minimálnej podpory či minimálnej spoľahlivosti sme chceli zistiť, ako závisí počet výsledných asociačných pravidiel

na týchto hodnotách. V oboch prípadoch bolo dokázané, že zvýšením týchto hodnôt počet nájdených frekventovaných množín klesá, a tým aj počet vygenerovaných asociačných pravidiel.

Pomocou aplikácie sme otestovali aj výkonnosť algoritmu. Jednalo sa o testy doby výpočtu algoritmu. Zistili sme aj, že doba výpočtu lineárne stúpa so zvyšujúcim sa počtom transakcií v databáze. Počet nájdených asociačných pravidiel sa výrazne nezmenil.

Ako možné rozšírenie aplikácie v budúcnosti by sme mohli spomenúť implementovanie ďalších algoritmov slúžiacich na dolovanie asociačných pravidiel z dátových skladov. Takto by sa dalo jednoducho porovnať efektivitu a výkonnosť jednotlivých algoritmov. Ďalšou možnosťou rozšírenia by mohlo byť vytvorenie interaktívneho rozhrania, ktoré by slúžilo pre prezentáciu výsledkov priebehu algoritmu. Pomocou tohto rozhrania by sa dalo nastaviť, aké medzivýsledky či ďalšie informácie o behu algoritmu chceme zobraziť. Užívateľovi aplikácie by takto boli poskytnuté aj ďalšie dôležité údaje, ktoré môžu byť pre spracovanie dosiahnutých výsledkov užitočné.

Literatúra

[1] Han, J., Kamber, M.: *Data Mining : Concepts and Techniques.* Morgan Kaufmann Publishers, San Francisco, 2006. 770s. ISBN 1-55860-901-6.

[2] Hand, D, Mannila, H, Smyth, P.: *Principles of Data Mining.* MIT Press, Massachusetts, 2001. 546s. ISBN 0-262-08290-X.

[3] Lacko, L.: *Datové sklady, analýza OLAP a dolování dat*. Computer Press, Brno, 2003. 486s. ISBN 80-7226-969-0.

[4] Lacko, L.: *Business Intelligence v SQL Serveru 2005* : *reportovací, analytické a další datové služby.* Computer Press, Brno, 2006. 391s. ISBN 80-251-1110-5.

[5] Zendulka, J. a kol.: *Získávání znalostí z databází. : Studijní opora*. FIT VUT v Brně, Brno, 2006. 160s.

[6] Šanda, A.: *Přístupy k budování dátového skladu.* [online]. 1999 [cit. 2008-12-30]. Dostupný z WWW: <http://archiv.computerworld.cz/cwarchiv.nsf/clanky/9CFBDEF3424C6479C12569B00054537D?OpenDocument>.

[7] *XML for Analysis* [online]. 2008 [cit. 2009-05-18]. Dostupný z WWW: <http://www.xmlforanalysis.com/index.htm>.

[8] *Přehled produktu SQL Server 2005* [online]. 2005 [cit. 2009-05-18]. Dostupný z WWW: <http://www.microsoft.com/cze/windowsserversystem/sql/prodinfo/overview/default.mspx>.

[9] Chung, S. M., Mangamuri, M.: Mining Association Rules from the Star Schema on a Parallel NCR Teradata Database System. *Proceedings of the International Conference on Information Technology : Coding and Computing* [online]. 2005, vol. 1 [cit. 2009-05-18], s. 206-212. Dostupný z WWW: <http://ieeexplore.ieee.org/stampPDF/getPDF.jsp?tp=&arnumber=01428463&isnumber=30835>. ISSN 0-7695-2315-3.

[10] Pyle, D.: *Data preparation for data mining*. Morgan Kaufmann Publishers, 1999. 539s. ISBN 1-55860-529-0.

[11] Guillet F., Howard J. H.: *Quality Measures in Data Mining*. Springer, 2007. 313s. ISBN-10 3-540-44911-6.

[12] Ponniah, P.: *Data Warehousing Fundamentals.* John Wiley & Sons, Inc., New York, 2001. 516s. ISBN 0-471-41254-6.

[13] Kimball, R., Ross, M.: *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling*. 2. edition. John Wiley & Sons, Inc., New York, 2002. 436s. ISBN 0-471-20024-7.

[14] *Extensible Markup Language (XML)* [online]. 2009 [cit. 2009-05-18]. Dostupný z WWW: <http://www.w3.org/XML/>.

Zoznam príloh

Príloha 1. CD