VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

INTEGRACE BUSINESS INTELLIGENCE NÁSTROJŮ DO IS

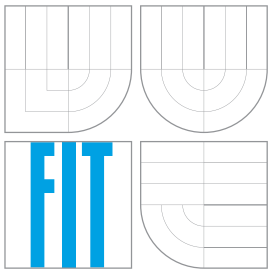
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. JOSEF NOVÁK

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

INTEGRACE BUSINESS INTELLIGENCE NÁSTROJŮ DO IS

INTEGRATION OF BUSINESS INTELLIGENCE TOOLS INTO IS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. JOSEF NOVÁK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. LUKÁŠ STRYKA

BRNO 2009

Zadání práce

1. Seznamte se s problematikou získávání znalostí z datových skladů a tvorby informačních systémů.
2. Seznamte se s podporou pro vývoj datových skladů v prostředí MS SQL Server nebo Oracle.
3. Navrhněte modulární informační systém. Začleňte do systému modul pro dolování asociačních pravidel z datového skladu využívající zvolený algoritmus.
4. Aplikaci realizujte a její funkčnost ověřte na vhodném vzorku dat.
5. Zhodnoťte dosažené výsledky. Diskutujte další možná rozšíření systému.

Licenční smlouva

Licenční smlouva je uložena v archívu Fakulty informačních technologií Vysokého učení technického v Brně.

Abstrakt

Tato diplomová práce se zabývá problematikou integrace prostředků Business Intelligence do informačních systémů. V prvních kapitolách jsou představeny oblasti Business Intelligence, datových skladů, OLAP analýzy a získávání znalostí z databází, zejména se zaměřením na dolování asociačních pravidel. V kapitolách věnujícím se praktické části je popsán návrh a implementace výsledné aplikace spolu s použitými technologiemi jako je např. Microsoft SQL Server 2005.

Abstract

This Master's Thesis deals with the integration of Business Intelligence tools into an information system. There are concepts of BI, data warehouses, the OLAP analysis introduced as well as the knowledge discovery from databases, especially the association rule mining. In the chapters focused on practical part of the thesis, the design and implementation of resultant application are depicted. There are also the applied technologies like i.e. Microsoft SQL Server 2005 described.

Klíčová slova

Business Intelligence, OLAP, Datové sklady, Asociační pravidla, MS SQL Server 2005, Java, Olap4j, Apriori

Keywords

Business Intelligence, OLAP, Data Warehouse, Association rules, MS SQL Server 2005, Java, Olap4j, Apriori

Citace

Josef Novák: Integrace Business Intelligence nástrojů do IS, diplomová práce, Brno, FIT VUT v Brně, 2009

Integrace Business Intelligence nástrojů do IS

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Lukáše Stryky

.....
Josef Novák
26. května 2009

Poděkování

Tímto bych rád poděkoval Ing. Lukáši Strykovi za vedení práce a ochotné rady.

Obsah

Kapitola 1

Úvod

Pro dnešní dobu je typické nasazování informačních technologií do všech oblastí, na které si jen člověk vzpomene. Jsou to nejrůznější obchodní aplikace, podnikové a administrativní systémy, pomocí nichž jsou shromažďována obrovská množství dat z nejrůznějších oborů. Vlastnictví takového objemu dat však ještě neznamená, že jsou tato data okamžitě využitelná. Nelze totiž ztotožnit pojmy data a informace. Kvůli tomu přichází ke slovu proces *Business Intelligence*(zkráceně *BI*), který je zaměřen na přeměnu dat na poznatky, které jsou pro koncové uživatele důležité a využitelné pro jejich rozhodování.

Podle informací získaných v BI procesech mohou obchodní manažeři řídit své marketingové kampaně a zvýšit tak efektivitu prodeje. Pro obchodníka může mít velkou vypovídající hodnotu například analýza nákupního košíku nebo statistika prodeje jednotlivých druhů zboží. V bankovnictví je důležité vědět o důvěryhodnosti klienta při žádosti o úvěr. V oblasti pojišťovnictví je nutné zajímat se o pravděpodobnost, zda klient bude provádět pojistné podvody. Business Intelligence proces je tedy velmi nápomocný při aktivitách, u nichž je předpokládán globální pohled na uložená data, například při plánování rozpočtu. Z toho vyplývá, že oblast Business Intelligence se dostává ve firmách do popředí, a tak bych rád v této práci popsal důležité věci, jenž se dané problematiky týkají.

V úvodní kapitole teoretické části 2 je popsáno, čemu se proces Business Intelligence věnuje. Následující část 3 se zabývá procesem získávání znalostí z databází, přičemž jsou zde rovněž rozebrány jednotlivé fáze tohoto procesu. Další kapitola 4 se věnuje problematice tvorby datových skladů. Zde jsou popsány důvody k jejich využívání a metody jejich budování. Analýza OLAP, typy OLAP serverů, multidimenzionální datový model a OLAP operace jsou popsány v části 5. Téma 6 se zabývá asociačními pravidly, základními pojmy z této problematiky a dále pak dolovacím algoritmem Apriori. V poslední kapitole teoretické části 7 bude představen MS SQL Server 2005 a jeho nejdůležitější části související s problematikou Business Intelligence. Celá teoretická část vychází a navazuje na předchozí text ze Semestrálního projektu(SEP).

Jelikož je cílem této práce vytvořit informační systém a do něj začlenit modul pro dolování asociačních pravidel z datového skladu, tak v praktické části 8 jsou zmíněny nejdůležitější technologie, jenž jsou při tvorbě aplikace využity. Je zde uveden stručný postup při tvorbě datové kostky v MS SQL Serveru 2005 a jsou zde vysvětleny další pojmy užitečné pro pochopení problematiky. Dále jsou zmíněny obtíže, které při vývoji a zprovozňování technologií potřebných k provozu aplikace nastaly. Poté je popsána vytvořená aplikace - informační systém a jeho jednotlivé moduly. Podrobněji je uveden popis mo-

dulu pro dolování asociačních pravidel a jednotlivé typy dolovacích úloh, pro něž byl vytvořen. V závěrečné části 9 této práce jsou pak uvedeny testy výsledků výpočtu aplikace nad konkrétními daty a zhodnoceny dosažené výsledky při dolování asociačních pravidel.

Kapitola 2

Business Intelligence

Následující kapitola se věnuje procesu Business Intelligence, definuje pojmy, jenž s problematikou souvisí, popisuje aktéry, kteří v tomto procesu figurují a dále pak uvádí základní vlastnosti úspěšného BI.

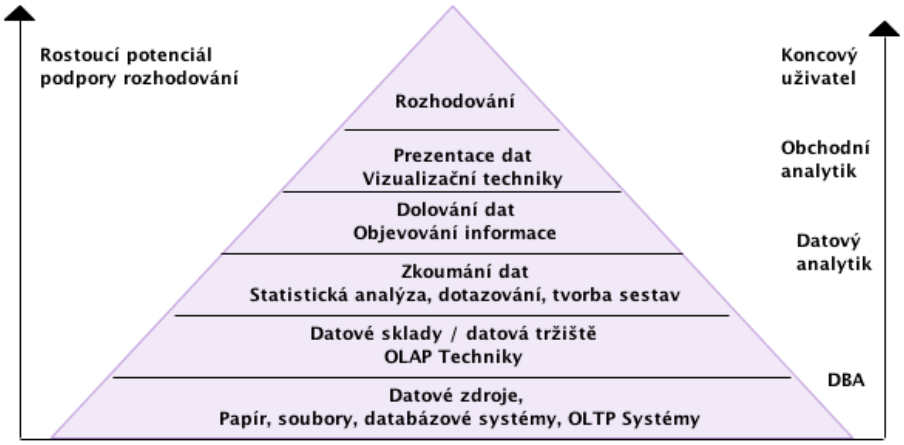
2.1 Definice

Business Intelligence můžeme definovat jako proces transformace dat na informace a převod těchto informací na poznatky prostřednictvím objevování [14].

Transformujeme tedy data pro účely zákazníků na znalosti, které jsou pak využívány pro obchodní rozhodování manažerů a vedoucích pracovníků. Tento proces můžeme demonstrovat na jednoduchém diagramu ve tvaru trojúhelníku. V něm se vyskytují pojmy jako Údaje, Informace, Znalosti a Moudrost. Vše vychází ze spodní řady, kdy dobré údaje jsou předpokladem pro kvalitní informace. K informacím se dostaneme jen v tom případě, pokud je správně interpretujeme. Znalost vychází ze smysluplného zpracování informací do znalostních bází a moudrost jako vrchol je pak správné pochopení a zhodnocení získaných znalostí pro účel rozhodování.

Obrázek 2.1 ukazuje souvislost mezi Business Intelligence a dolováním znalostí. Figurují v něm role jednotlivých subjektů(uživatelů), kteří mají co do činění s jednotlivými fázemi dolování dat. Nejníže v této hierarchii stojí databázový administrátor. Na této úrovni se jedná o aplikace typu centrálního podnikového informačního systému, popřípadě o systém podpory řízení se zákazníky. Data jsou z uvedené úrovně převáděna do datových skladů a multidimenzionálních databází. Přichází na řadu datový analytik, který se stará o analýzy dat a OLAP techniky při vstupu dat do tohoto procesu. Porozumění těmto vyextrahovaným informacím, znalostem pak náleží business analytikovi se znalostí podnikání. A na vrcholu hierarchie stojí manažer, pro kterého jsme celý tento proces dělali a který si pak přečte podklady od business analytika.

Když bychom chtěli popsat nejdůležitější části celého BI systému, tak se jedná o datový sklad, nástroje určené pro datamining, OLAP server, klientské aplikace a uživatelské nástroje, reportingový server a vývojové nástroje [25]. Celý Business Intelligence systém obvykle nebývá realizován jako jeden celek, ale bývá budován ve formě více menších projektů. Tímto manažeři a lidé zodpovědní ve firmě mohou dostat dílčí výsledky už během procesu budování.



Obrázek 2.1: Vztah dolování dat a Business Intelligence [27]

Konečně, celý tento proces je dělán pro zákazníka. To, jak je celý proces úspěšný, záleží na jeho spokojenosti a na tom, jaké úspěchy mu BI přináší při jeho obchodních aktivitách. Proces BI může zákazníkovi pomoci například ve zvýšení účinnosti marketingových kampaní, segmentaci prodejních kanálů, cenotvorbě, lepším vztahům se zákazníky, zvýšením efektivity prodeje, minimalizací rizik nebo optimalizací zásobovacích činností. Pokud bychom chtěli tedy přesně vyjádřit vlastnosti úspěšného BI systému, pak tento systém musí být přijat a používán uživateli a mít jejich důvěru, musí umožňovat snadný přístup k informacím organizace - rychlý přístup ke srozumitelným a intuitivním datům, prezentuje informace konzistentně a věrohodně, je bezpečným místem, které chrání citlivá informační aktiva organizace, je adaptivní a schopný pružně a rychle reagovat na změny [13].

Je nutno zmínit i někdy opomíjený fakt, že velký podíl na procesu BI má kvalita dat, jenž do něj vstupují. Mnohdy se ve firmách pracuje s předpokladem, že zpracovávaná data jsou správná. Získané výsledky jsou však tak spolehlivé, jak spolehlivá jsou vstupní data, protože ani to nejkvalitnější BI řešení nebude fungovat, pokud mu kvalitní a správná data nedáme.

Co se týká koncových uživatelů, předpokládáme, že mají dostatečné odborné znalosti, jako je ovládání pracovních stanic, kdy musí být schopni OLAP nástroje pod grafickým uživatelským rozhraním používat k získání přístupu k datům. Dále jsou kladeny požadavky na jejich obchodní znalost, kdy odpovědi, které datové sklady poskytují, jsou pouze tak kvalitní, jako jsou kvalitní položené otázky[10]. Koncoví uživatelé nebudou schopni formulovat správné otázky nebo dotazy bez dostatečného pochopení vlastního obchodního prostředí. A nakonec bychom měli zmínit i fakt, že koncoví uživatelé musí rozumět datům dostupným ve skladu a musí být schopni spojit tato data se svými obchodními znalostmi.

Kapitola 3

Proces získávání znalostí

V této části si vysvětlíme proces získávání znalostí se zaměřením na fáze týkající se předzpracování dat.

3.1 Proces získávání znalostí

Proces získávání znalostí je extrakce (neboli dolování) zajímavých (netriviálních, skrytých, dříve neznámých a potenciálně užitečných) modelů dat a vzorů z velkých objemů dat. Tyto modely a vzory reprezentují znalosti získané z dat [27].

Vysvětleme si nyní blíže danou definici. Pojem *netriviálnosti* vyjadřuje, že informaci nejde získat například SQL dotazem z databáze. *Skrytost* - informace nelze na první pohled mezi daty objevit, zde opět musí nastoupit speciální techniky, stejně, jako je tomu u netriviálnosti.
Potenciální užitečnost získaných znalostí - vyjadřuje fakt, že tuto znalost využijeme pro rozhodování. Zde se může jednat např. o rozhodnutí jiného rozložení zboží v supermarketu, nebo rozhodnutí banky poskytnout půjčku.

Konkrétní aplikace dobývání znalostí můžeme nalézt v celé řadě oblastí: segmentace a klasifikace klientů banky(např. rozpoznání problémových nebo naopak vysoce bonitních zákazníků), predikce vývoje kurzů akcií, predikce spotřeby elektrické energie, analýza důvodů změny poskytovatele služeb(mobilní telefony, internet), segmentace a klasifikace klientů pojišťovny, určení příčin poruch automobilů, rozporu databáze pacientů v nemocnici, analýza nákupního košíku a další.
Proces získávání znalostí můžeme rozdělit do několika fází: (viz obrázek 3.1) [27]

Čištění dat - odstranění šumu, chybějících dat

Integrace dat - integrace dat z více datových zdrojů

Výběr dat - vybíráme data, která nás zajímají

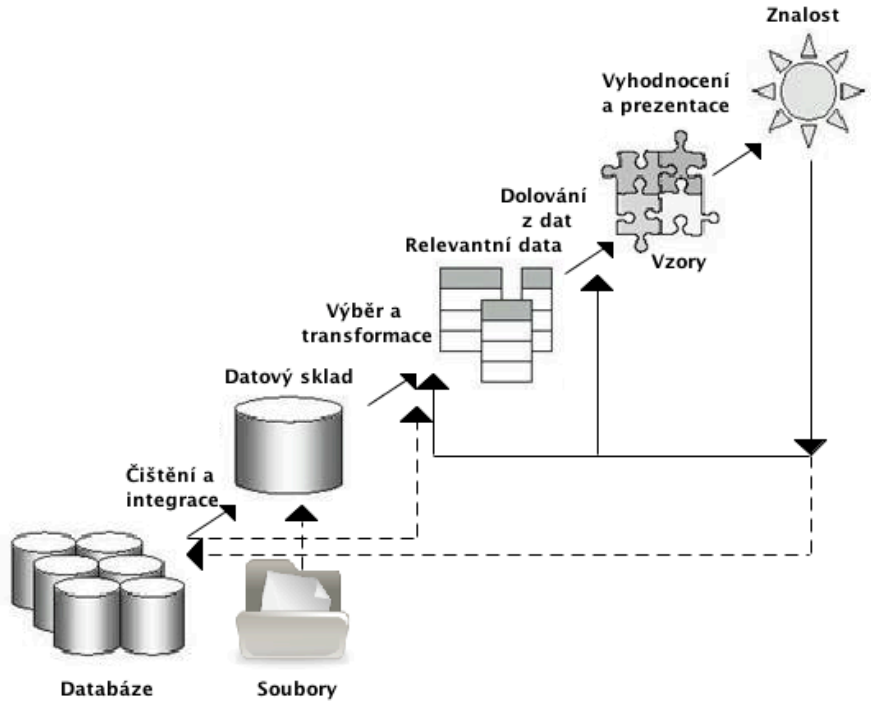
Transformace dat - transformace dat do podoby, která nás zajímá

Dolování dat - pomocí konkrétní metody extrahovat z dat vzory

Hodnocení modelů a vzorů - identifikace zajímavých vzorů

Prezentace znalostí - prezentace získaných výsledků koncovému uživateli

Dále se budeme některým těmto fázím věnovat podrobněji.



Obrázek 3.1: Proces získávání znalostí [27]

3.2 Příprava údajů - předzpracování dat

Údaje pro datové sklady pocházejí z různých nehomogenních zdrojů. Obvykle se jedná o údaje získané z transakčních systémů OLTP, různých souborových databází nebo databází spravovaných některým databázovým serverem. Tato data bývají velice často zašuměná, nekonzistentní, v některých případech mohou hodnoty i chybět. Pod pojmem *zašuměnost* vyjadřujeme to, že atributy obsahují odlehlé nebo nesprávné hodnoty. *Nekonzistentnost* pro nás znamená nekonzistenci v pojmenování, formátech nebo kódování. *Nekompletností* pak rozumíme to, že chybí některé atributy, které pro naši analýzu nutně potřebujeme. Nekompletnost může vzniknout např. při poruše nebo vynechání zařízení pro sběr dat. Pokud bychom pracovali se zašuměnými či jinak znehodnocenými daty, mohli bychom při naší analýze dojít k nesprávným a zavádějícím výsledkům.

Údaje, se kterými budeme chtít pracovat je tedy nutno před uložením do datového skladu upravit, vyčistit, předzpracovat. Teprve po vykonání těchto nezbytných úkonů je můžeme zavést do datového skladu.

3.2.1 Čištění dat

Jedná se o jednu z úloh předzpracování dat. Cílem čištění dat je odstranit chybějící hodnoty, nalézt a odstranit odlehlé hodnoty, které nezapadají do našeho výběru a vyřešit datovou

nekonzistenci. Nekvalitní data mohou ovlivnit i datový algoritmus. Kvůli tomu bychom pak mohli dostávat nespolehlivé výsledky. Odstranění chybějících hodnot můžeme dosáhnout například *ignorováním n-tice, manuálním nahrazením* nebo *automatickou náhradou*, která představuje pravděpodobně nejpoužívanější způsob, neboť využívá informace obsažené v datech. Další možností je náhrada chybějící hodnoty nejpravděpodobnější hodnotou - lze vyjít z regrese, rozhodovacího stromu nebo Bayesovské klasifikace.

3.2.2 Integrace

Jelikož musíme často pracovat s více zdroji dat, je nezbytné tato data vhodně integrovat, tj. vytvořit z nich jeden celistvý zdroj. Mezi problémy *integrace* patří následující(viz [27]): *Konflikty schématu, konflikty hodnot, konflikty identifikace a redundance.* Pod pojmem *konflikt schématu* si lze představit integraci metadat z různých zdrojů do metadat popisujících výsledný zdroj(pole záznamů, které získáme průzkumem, se může lišit od názvu sloupce tabulky). *Konflikty hodnot* - hodnoty odpovídajících se záznamů se liší. *Konflikty identifikace* - stejné objekty existující v reálném světě můžeme identifikovat jiným způsobem Při *redundanci* se jedná nejen o výskyt stejných atributů v různých zdrojích dat, ale i o odvozené atributy, jejichž hodnotu lze odvodit z hodnot jiných atributů - např. věk vs. rok narození.

3.2.3 Transformace

Transformace si dává za úkol transformovat data do takové podoby, jenž bude vhodná pro dolovací úlohy. Proces transformace zahrnuje následující operace :

Vyhlazení - odstranění šumu z dat.

Agregace - má za úkol detailní informace agregovat.

Generalizace - zdrojová data nahradíme koncepty z konceptuální hierarchie.

Normalizace - transformace a namapování numerických hodnot na konkrétní interval.

Konstrukce atributů - vytvoření nových atributů, zkvalitňující a usnadňujících dolování. Jejich hodnoty jsou vytvořeny z atributů jiných.

Při *transformaci údajů* můžeme narazit na následující problémy:

Nejednoznačnost údajů - například údaje o pohlaví zákazníka může být uvedena rozdílným způsobem - female, woman, f, Female. Po provedení transformace máme výsledná data např. jen ve tvaru FEMALE.

Různé peněžní měny - tato problematika může být aktuální např. při přechodu na jinou peněžní měnu. V mezidobí při zavádění nové měny se ceny mohou uvádět v obou měnách najednou.

Formáty čísel a textových řetězců - Při ukládání numerických a řetězcových datových typů, kdy může např. rodné číslo být uloženo buď jako číslo nebo jako textový řetězec.

Problém při chybějícím datu - v datovém skladu má čas velice významnou roli. Časový údaj tedy musí být přítomný v datech ještě před jejich zavedením do datového skladu, popřípadě je ho nutné přidat při zavádění dat.

3.2.4 Redukce

Redukce zmenšuje objem dat, se kterým budeme pracovat při dolování. Musíme ale zachovat datovou integritu a charakter dat, abychom dostali stejné výsledky jako z dat původních. Máme různé strategie redukce[27]:

Agregace datové kostky - používá se pro datové sklady, znamená redukci dat agregováním údajů.

Výběr podmnožiny atributů - tzv. feature selection - vylučujeme nerelevantní a redundantní atributy.

Redukce dimenzionality - zpracovávaná data zakódujeme vhodným způsobem. Tímto zapříčiníme redukci, ale s daty bude možné provádět potřebné operace např. vlnkové operace a analýzu hlavních komponent.

Redukce počtu hodnot - nahrazení dat nějakým modelem. Tyto data jsou pak reprezentována jeho parametry nebo jsou reprezentována v redukované podobě.

Diskretizace a generace konceptuální hierarchie - hodnoty atributů jsou nahrazeny intervaly nebo pojmy z konceptuální hierarchie.

Kapitola 4

Datový sklad

V následující kapitole si vysvětlíme pojem datový sklad, motivaci pro použití a metody jeho budování.

4.1 Definice

Datový sklad je podnikově strukturovaný depozitář subjektově orientovaných, integrovaných, časově proměnlivých, historických dat použitých na získávání informací a podporu rozhodování. V datovém skladu jsou uložena atomická a sumární data. [14]
Uvedená definice shrnuje několik podstatných rysů, které datový sklad musí splňovat.

Subjektová orientace - údaje zaměřeny na předměty zájmu (zákazníky, produkty). Zaměřuje se spíše na data, na jejich analýzu a modelování.

Integrovanost - jelikož údaje do datového skladu získáváme z různých heterogenních zdrojů, různých nekonzistentních a neintegrovaných prostředí, musíme data sjednotit, aby měli stejné názvy atributů, stejné jednotky a ostatní parametry.

Časová variabilita - čas je velmi důležitou veličinou v datových skladech. Data máme v datovém skladu uloženy v závislosti na čase jako časový snímek. V datovém skladu shromažďujeme data týkající se období třeba několika let.

Neměnnost - data v datovém skladě se nesmí mazat ani modifikovat, jen se stále přidávají nová. Vyžadujeme jen operace pro nahrání dat a pro přístup k nim. Kvůli tomu také v datových skladech nepotřebujeme metody pro transakce, normalizaci ani optimalizaci.

4.1.1 Rozdíl mezi klasickou (operační databází) a datovým skladem

Mezi hlavní rozdíly řadíme odlišné využití pro zákazníka. *Datové sklady(OLAP - Online Analytical Processing)* slouží k analýze dat, jejich potenciálu využíváme pro obchodní účely, pro analýzu, kterou potřebují manažeři, obchodní ředitelé a firemní analytici. *Operační databáze, OLTP(Online transaction processing)*, je určena pro zpracovávání dotazů a pro on-line transakcí uživatele. Uživatel po nich požaduje okamžitou reakci, kdežto statistiky z datového skladu se klidně mohou zpracovávat celou noc. Rozdílnost plyne i z datového obsahu, kterými disponují. Datový sklad je svou strukturou určen pro analyzování dat, spravuje velké množství historických dat, v něm uložených, kdežto operační databáze spravuje data v podobě nevhodné k analyzování. Datový sklad má i prostředky pro agregaci

i sumarizaci. OLTP se striktně orientuje na aplikaci, zatímco OLAP je subjektově orientován. Co se týká návrhu databáze, OLAP používá schéma sněhové vločky (viz obrázek 5.2) nebo hvězdy (viz obrázek 5.3). Při přístupu k datům OLTP používá krátké transakce, které jsou atomické, ale přístup OLAP je pouze přes operace čtení.
Rozdíl spočívá i v počtu uživatelů, kteří k datům přistupují, u OLAP je jich řádově stovky, u OLTP řádově tisíce. Rovněž velikosti databáze se liší, u OLAP jsou to typicky stovky MB až stovky GB, u OLAP pak stovky GB až stovky TB. Hlavní prioritou OLAP databází je odezva, propustnost dotazů, u OLTP je to rychlost a propustnost transakcí.

4.2 Metody budování datového skladu

Mezi nejčastější a nejznámější metody budování datového skladu patří *Přírůstková metoda* a *Metoda „Velkého třesku“* [14].

4.2.1 Metoda „velkého třesku“

Tato metoda vychází z myšlenky, která předpokládá, že se datový sklad dá vybudovat v rámci jediného projektu. Ovšem vývoj datového skladu není jednoduchou záležitostí, což můžeme poznat v průběhu projektu, kdy se požadavky od uživatele mohou dramaticky změnit, na trh přijdou nové technologie, které si bude zákazník přát. Obecně můžeme tuto metodu rozdělit do tří etap:

Analýza požadavků podniku

Vytvoření podnikového datového skladu

Vytvoření přístupu buď přímo, nebo přes datové trhy

Pokud můžeme přiznat této metodě nějakou výhodu, pak je to fakt, že celý projekt lze zpracovat ještě před začátkem realizace. S tím však souvisí již zmiňované nevýhody, a to riziko změny požadavků od zákazníka a dlouhá doba, než nám tato metoda začne vydělávat.

4.2.2 Přírůstková metoda

Datový sklad budujeme po jednotlivých etapách. Začíná se budovat po jednotlivých částech, na rozdíl od předchozí metody místo vybudování celého datového skladu postupně přidáváme jednotlivé předmětné oblasti, které nás zajímají. Koncovým uživatelům zpřístupňujeme naše řešení již během procesu budování. Toto řešení je lepší pro zákazníka a management, v tom ohledu, že se v krátké době dostanou k informacím, které je zajímají a tudíž naše řešení začne vydělávat peníze již po krátkém čase od zahájení projektu. Další výhodou je i otestování našeho řešení uživateli, kteří s naší aplikací již během iterativního vývoje měli možnost pracovat.
Přírůstková metoda se používá ve dvou variantách: *směrem „shora dolů“* a *směrem „zdola nahoru“*

Směrem „shora dolů“ - u této metody postupně vytváříme datové trhy jednotlivých předmětných oblastí v rámci struktury datového skladu. Poskytuje poměrně rychlou implementaci a tím i rychlé návratnosti investic.

Směrem „zdola nahoru“ - v rámci této metody budujeme nejdříve datové trhy předmětných oblastí v rámci struktury datového skladu. Před obchodním ziskem mají prioritu údaje. Pokud bereme jako prioritu managerské hlediska, potom u této metody převažují nevýhody.

Přírůstková metoda se skládá z částí:

Strategie - velice důležitá část, při které definujeme cíle datového skladu a to jednak účel a také cíl podnikání.

Definice - definuje se rozsah a cíl přírůstkového vývoje. Navrhuje se architektura datového skladu i architektura technických prostředků.

Analýza - zaměřujeme se na informace o uživatelích, získávání dat a požadavků na přístup k datům.

Návrh - zesumarizování požadavků získaných během fáze analýzy, vytvoření detailního návrhu.

Sestavení - vytvoření a otestování databázové struktury a všech potřebných modulů.

Produkce - instalace datového skladu, jeho nasazení, stejně tak už musíme začít s jeho údržbou a řízením růstu.

Kapitola 5

OLAP

Tato kapitola pojednává o OLAP, jsou zde definovány základní pojmy spojené s tematikou, dále pak vysvětleny pojmy datové kostky a multidimenzionálního datového modelu včetně popisu jednotlivých schémat multidimenzionálních databází. Jsou zde rovněž popsány jednotlivé typy OLAP serverů a zmíněny konkrétní OLAP operace.

5.1 Úvod do OLAP (Online Analytical Processing)

Jeden z možných pohledů na tuto problematiku by se dal zformulovat tak, že OLAP lze považovat za databázovou technologii určenou k hloubkovým analýzám, jenž umožňuje provádět rychlé hloubkové dotazy. Cílem OLAP je analýza dat a podpora rozhodování. Definujme si nyní tato tvrzení formálněji.

5.1.1 Definice

OLAP je volně definovaný řád principů, které poskytují dimenzionální rámec pro podporu rozhodování [14].

Jinou definici nabízí Dr. E. F. Codd (Dvanáctero pravidel OLAP) [14]

Multidimenzionální konceptuální pohled - OLAP by měl poskytovat uživateli multidimenzionální model odpovídající jeho podnikatelským potřebám tak, aby tento model mohl využívat pro analýzu shromážděných údajů

Transparentnost - technologie systému OLAP, podřízená databáze a architektura výpočtů by měly být pro uživatele transparentní, aby mohl naplno využívat svoji produktivitu a odbornost při použití nástrojů front-end a prostředí. Pro platformu Microsoft můžeme jako klientský nástroj použít například program MS Excel z balíku MS Office, který bude napojený na Server OLAP. Důležitá je samozřejmě i heterogennost vstupních dat, kterou zajistíme v procesu ETL.

Dostupnost - systém OLAP by měl přistupovat jen k těm údajům, které jsou potřebné pro analýzu. Systém by měl být navíc schopen přistupovat ke všem údajům potřebným pro analýzu nezávisle na tom, z kterého heterogenního podnikového zdroje tyto údaje pocházejí, jak často jsou obnovovány a podobně. Znovu zdůrazňujeme význam etapy ETL, ve které se údaje očistí, zahustí a přetransformují.

Konzistentní vykazování - i když počet záznamů a tedy i velikost databází postupem času roste, uživatel by neměl pocítit žádné podstatné snížení výkonu.

Architektura klient-server - systém OLAP musí odpovídat principu architektury klient-server s přihlédnutím k maximální ceně a výkonu, flexibilitě a interoperabilitě.

Geometrická dimenzionalita - každá dimenze údajů musí být ekvivalentní ve struktuře i operačních schopnostech.

Dynamické ošetření řídkých matic - systém OLAP by měl byt schopen adaptovat své fyzické schéma na konkrétní analytický model, který optimalizuje ošetření řídkých matic, přičemž dosáhne a udrží požadovanou úroveň výkonu.

Podpora pro více uživatelů - systém OLAP musí být schopen podporovat pracovní skupinu uživatelů pracujících současně na konkrétním modelu

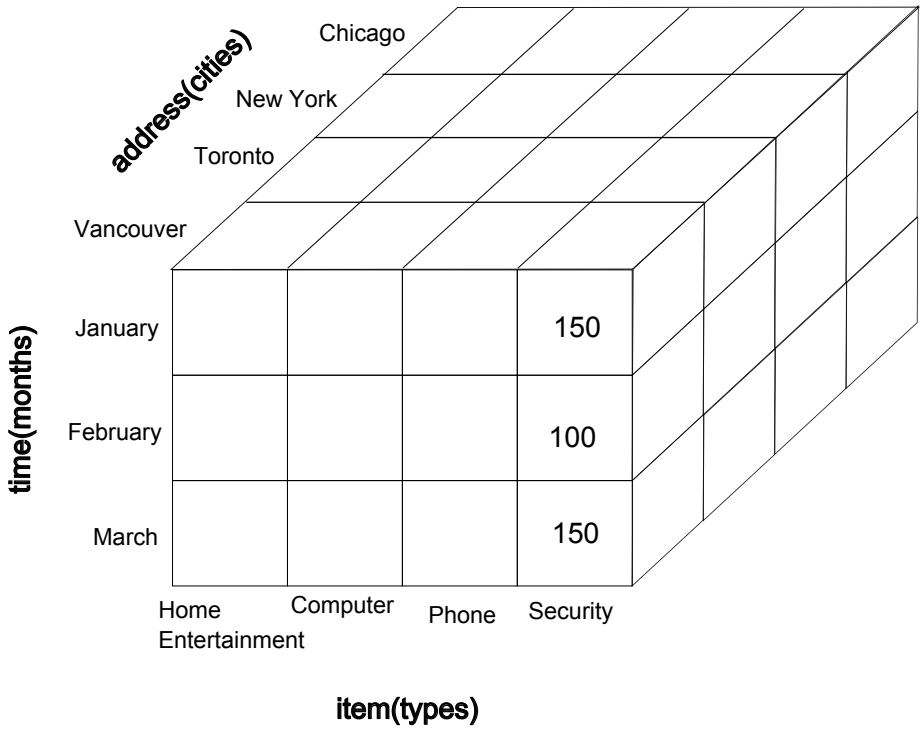
Neomezené křížové dimenzionální operace - systém OLAP musí dokázat rozeznat dimenzionální hierarchie a automaticky vykonat asociované kumulované kalkulace v rámci dimenzí i mezi nimi.

Intuitivní manipulace s údaji - pravidlo definuje konsolidované přeorientování cest na detailní úroveň a zpět(drill down, drill up), Uživatelské rozhraní by mělo umožňovat všechny manipulace způsobem "ukázat a klepnout, případně zachytit a přemístit v buňkách kostky".

Flexibilní vykazování - musí existovat schopnost uspořádat řádky, sloupce a buňky způsobem, který umožní analýzu a intuitivní vizuální prezentaci analytických sestav.

Neomezené dimenze a úrovně agregace - v závislosti na požadavcích podnikání může mít analytický model více dimenzí, přičemž každá z nich může mít vícenásobné hierarchie. Systém OLAP by neměl zavádět(například z technických důvodů), žádné umělé omezení počtu dimenzí nebo úrovní agregace.

Jak už bylo na počátku kapitoly zmíněno, OLAP lze považovat za databázovou technologii určenou k hloubkovým analýzám, které umožňuje provádět rychlé hloubkové dotazy [6]. To vše je možné díky tomu, že i když OLAP kostka 5.2 obsahuje velké množství dat, tak souhrny na středních a vyšších úrovních jsou v mnoha případech již předvypočteny a my pouze vypíšeme výsledek. Stroje OLAP jsou optimalizovány i na takové výpočty, které je nutno provádět za běhu, jelikož se nemusí starat o správu indexů, spojování tabulek apod. Při svém vytváření předpočítávají některé nebo všechny agregace a v průběhu dotazů vypočítávají další agregace ze záznamů, které jsou již předpočítány. Vše výše uvedené nabízí uživatelům skvělou příležitost se lépe seznámit se svými daty, získávají lepší přehled a vše jim napomáhá k tomu dělat úspěšná obchodní rozhodnutí.



Obrázek 5.1: Datová kostka [9]

5.2 Multidimenzionální datový model

Nástroje OLAP jsou založeny na multidimenzionálním datovém modelu. Data jsou zobrazena ve formě *datové kostky(datové krychle)*

Multidimenzionální datová kostka vznikne jako výsledek agregace a analýzy údajů. Zjednodušeně je to ekvivalent databázové tabulky v relační databázi. Využívá se v analýze OLAP pro analýzu velkého objemu dat. Pro výpočet krychlí OLAP je potřebné vykonat velké množství výpočtů a agregací. To vše musí proběhnout v reálném čase. Typický příklad na datovou krychli s třemi dimenzemi čas, produkt, region. Ovšem datová kostka může mít i více dimenzí než jen tři. Například u MS SQL Serveru 2000 je možno vytvořit až 64 dimenzí.

V datové krychli najdeme údaje na průnicích jednotlivých dimenzí. Ovšem ne na všech průnicích dimenzí se vždy nacházejí údaje, takovou krychli pak nazýváme řídkou krychlí.

5.2.1 Fakta a dimenze

Na základě faktů a dimenzí je vytvořena každá multidimenzionální kostka OLAP [14].

Fakta - jsou metriky obchodování. Tabulka faktů obsahuje obrovské množství dat a je i největší tabulkou v databázi. Na jejich základě analyzujeme vztahy mezi dimenzemi.

Dimenze - jsou to entity nebo pohledy. Obsahem dimenzí jsou uspořádané údaje podle jejich logiky nebo podle hierarchie. Když porovnáme tabulky dimenzí a tabulky faktů, potom je tabulka dimenzí obvykle menší. V tabulce dimenzí se data často nemění,

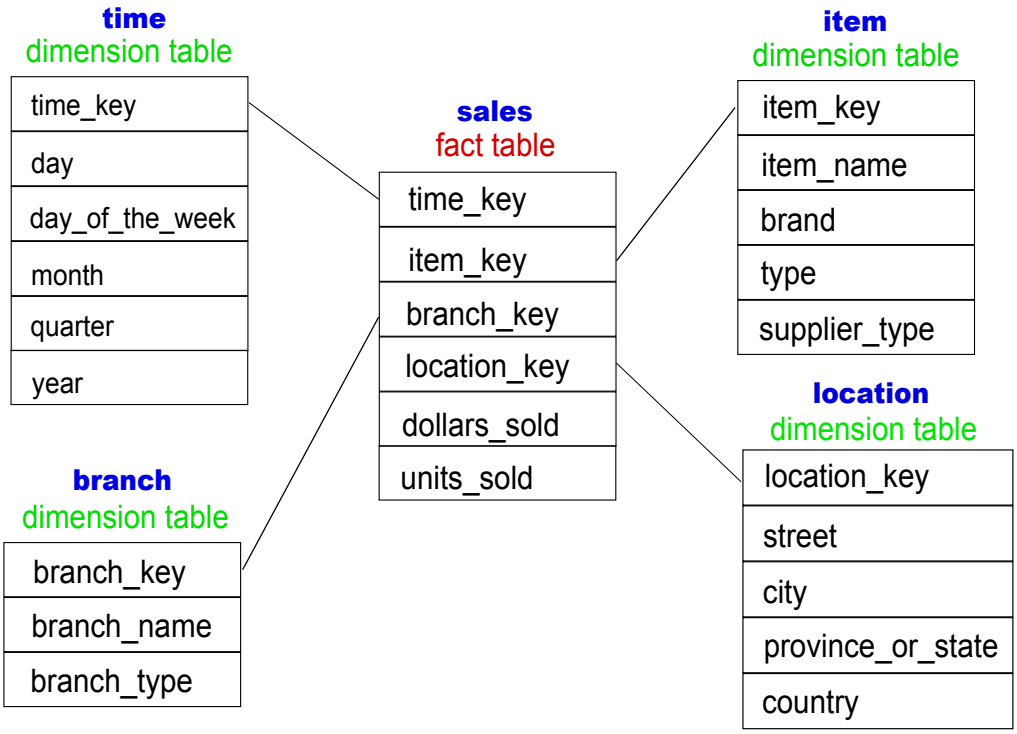
obsahují poměrně stálá data. Tabulky dimenzí obvykle obsahují stromovou strukturu. Jako nejčastější dimenze se používají geografické, produktové a časové.

5.2.2 Schémata multidimenzionálních databází

Multidimenzionální kostku vytváříme na základě topologického uspořádání, tzv. modelu. Používají se nejčastěji schémata *„sněhové vločky“(snowflake schema), hvězdy(star schema)* nebo *souhvězdí*.

Schéma hvězdy(nebo též hvězdicové schéma)

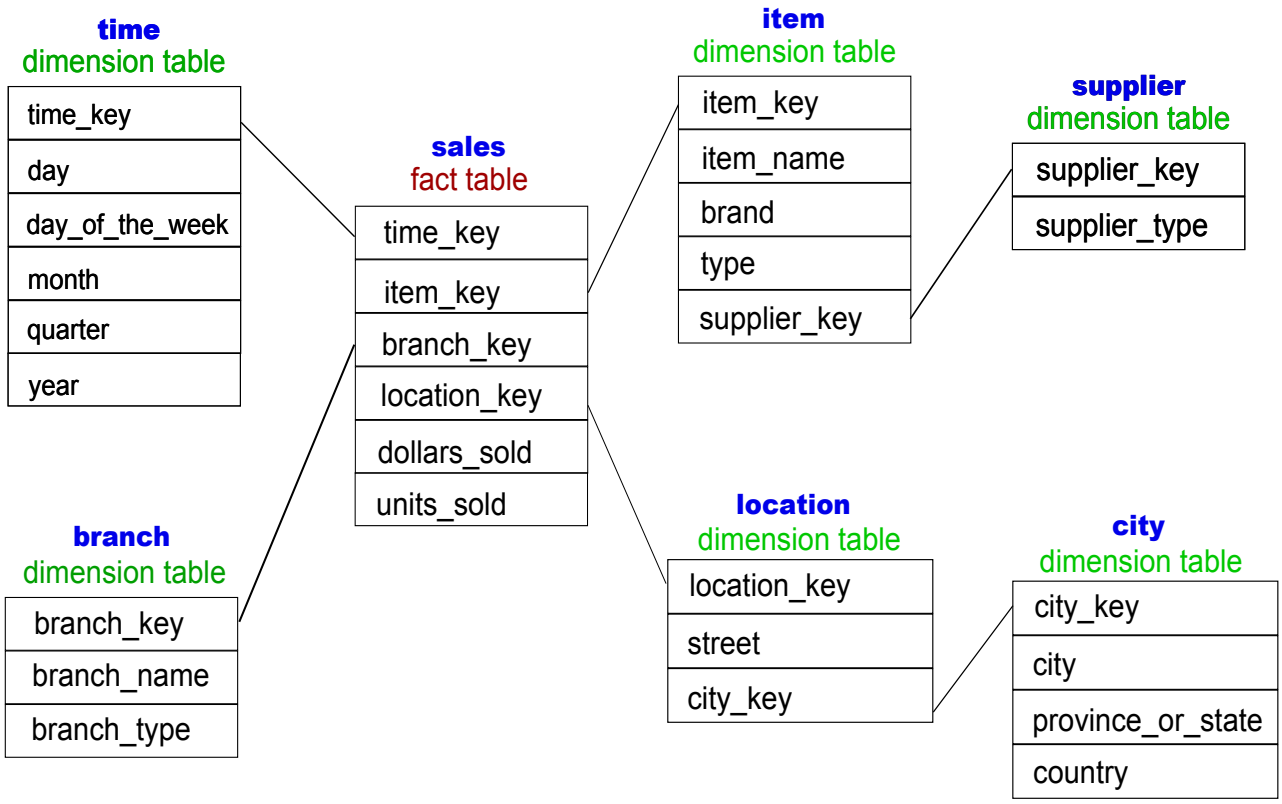
Toto schéma 5.2 se skládá z tabulky faktů, která obsahuje cizí klíče vztahující se k primárním klíčům v tabulkách dimenzí. Z hlediska dotazování poskytuje vysoký výkon. Její dimenze nejsou normalizované. V tabulkách dimenzí taktéž vzniká redundance. Vysoký dotazovací výkon je dán získáním všech údajů najednou, nemusíme tudíž skládat žádné relační tabulky dohromady.



Obrázek 5.2: Schéma hvězdy [9]

Schéma „sněhové vločky“

Může obsahovat více než jednu tabulku faktů. Popisované schéma (viz obrázek 5.3) má některé dimenze složené z více relačně propojených tabulek. Má výrazně nižší dotazovací výkon než hvězdicové schéma, jelikož výsledek skládá z většího množství tabulek. Umožňuje však rychlejší zavedení do normalizovaných tabulek.



Obrázek 5.3: Schéma „sněhové vločky“ [9]

5.2.3 Srovnání multidimenzionálního a relačního modelu

Když porovnáme relační a multidimenzionální model, potom zjistíme, že relační model má výhodu v použitelnosti v transakčních databázích i datových skladech, velkému počtu odborníků, kteří této technologií rozumí a umí ji používat a ve velkém množství vývojových nástrojů pro vývoj a ladění aplikací. Nevýhoda relačního modelu spočívá v potenciálním omezení objemu údajů, ke kterým je možné přistupovat v nějakém rozumném čase a též v absenci analytických nástrojů. Oproti tomu multidimenzionální model poskytuje výhody v komplexním a rychlém přístupu k velkému objemu dat, k velkému množství analýz, k schopnosti pro modelování a prognózy a přístupu jak k multidimenzionálním, tak k relačním datovým strukturám. Nevýhoda tohoto modelu je ve vyšších nárocích na kapacitu úložiště a v možnostech problému při změně dimenzí, bez přizpůsobení časové dimenzi [14].

5.2.4 Typy OLAP Serverů(MOLAP, ROLAP, HOLAP)

Jako úložiště multidimenzionálních dat se se používá několik typů architektur. Mezi nejvyužívanější náleží multidimenzionální, relační a hybridní OLAP servery.

Multidimenzionální OLAP(MOLAP)

Pro typ tohoto zpracování získáváme data jednak z operačních zdrojů nebo z datového skladu. *MOLAP* - ukládá data ve vlastních datových strukturách. Počítají se předběžné výsledky, údaje se ukládají jako předem vypočítaná pole. Ve formě vypočítaných polí zde uchováváme hodnoty dat i indexů. Umožníme klientům přistupovat rychle k údajům z více dimenzí, to pak následně umožňuje velice rychlé analýzy. Mezi nevýhody tohoto přístupu patří redundance údajů, protože jsou uloženy jednak v relační a jednak v multidimenzionální databázi. Úložná kapacita může extrémně narůstat. Jako jednu z největších výhod vidíme velký výkon vzhledem k uživatelským dotazům.

Relační databázový OLAP (ROLAP)

ROLAP(Relační online analytické zpracování dat) používá pro uložení dat relační nebo rozšířené relační databázové systémy. Tyto data se pak předkládají po zpracování uživateli jako multidimenzionální pohled. ROLAP tedy používá pro uložení dat a metadat relační tabulky. Nevzniká problém s redundancí. Server OLAP využívá metadata příslušná k datům na tvorbu SQL dotazů, které používá pro získávání dat vyžadovaných uživatelem.

Hybridní OLAP (HOLAP)

vzniká kombinací ROLAP a MOLAP, využívá předností obou typů úložišť - rychlosti MOLAPu a škálovatelnosti ROLAPu. Údaje ukládáme do relačních databázových systémů a vypočítané agregace vkládáme do multidimenzionálních struktur. U tohoto řešení se do relačních databází vkládají spousty detailních dat a o sumární data se naopak stará multidimenzionální model.

5.2.5 OLAP operace

V datovém skladu máme možnost prezentovat díky OLAP operacím data na různých úrovních abstrakce. Mezi nejdůležitější OLAP operace patří *Drill-down(Roll-down) a Roll-up(Drill-up), Slice and dice, pivot, drill across a drill through* [27] [3].

Drill-down - umožňuje vidět data na detailnější úrovni v nějaké dimenzi, posun z vyšší úrovně sumarizace na nižší, (zobrazujeme detailnější data). Tato operace může být provedena zavedením nové dimenze do datové kostky.

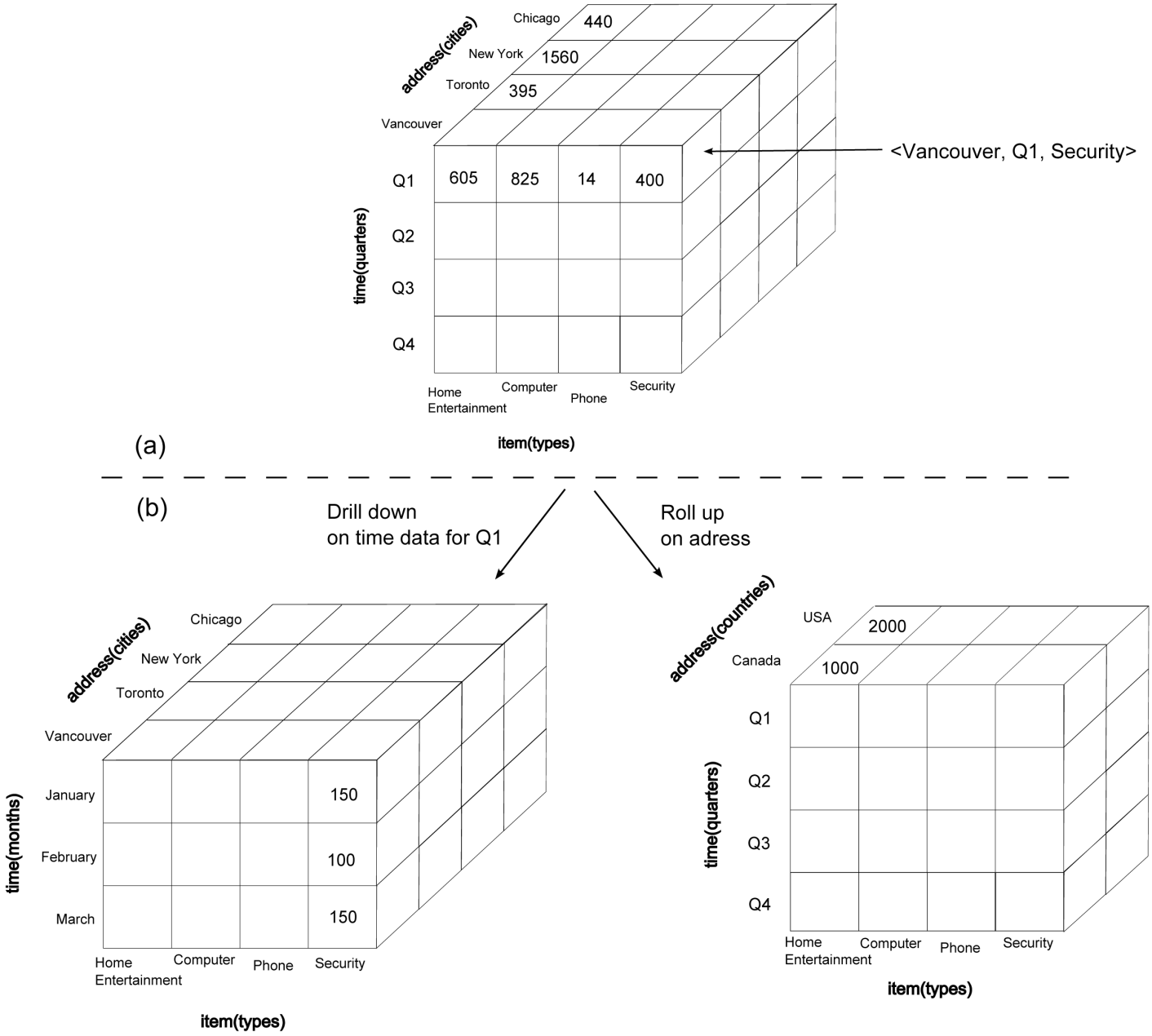
Roll-up - poskytuje abstraktnější pohled, sumarizace dat pomocí posunu v hierarchii o jednu dimenzi výše(redukce dimenze).

Slice and dice - operace *Slice* provede selekci nad jednou dimenzí. Jako výsledek této operace dostaneme kostku. Operace *Dice* - provádí selekci nad více než jednou dimenzí.

Pivot(Rotate) - jedná se o vizualizační operaci, která změní datové osy za účelem jiné prezentace dat kostky, může jít o rotaci os ve 2D nebo 3D kostce nebo zobrazení 3D kostky jako kolekce 2D kostek.

Drill-accross - operace spouští dotazy nad více tabulkami faktů.

Drill-through - umožňuje se zanořit až na úroveň samotných záznamů v tabulce.



Obrázek 5.4: Operace Drill-down a Roll-up [9]

Kapitola 6

Asociační pravidla

V této části se budeme věnovat asociačním pravidlům a jejich využití v praxi. Uvedeme rovněž algoritmus Apriori, který se používá pro získávání frekventovaných množin.

6.1 Úvod

Získávání asociačních pravidel hledá zajímavé asociace nebo korelace nad velkými množinami datových položek. Nalezení zajímavých asociací nad obchodními transakčními záznamy může pomoci v procesu obchodního rozhodování, jako je návrh katalogů, akčních nabídek nebo rozmístění zboží v obchodě [27].

Jedna z možných úloh, ve které budeme aplikovat metodu získávání asociačních pravidel, je analýza nákupního košíku [4]. Je to typická úloha z praxe, kdy obchodníka, respektive manažera obchodu, zajímá, jaké výrobky si zákazníci kupují nejčastěji pohromadě. Při analýze nákupního košíku vycházíme z dat, které shromažďují např. různé nákupní řetězce. Data tvoří jednak charakteristiky zákazníků(pohlaví, vlastnictví domu, příjem a věk), dále pak údaje o jednotlivých nákupech(způsob placení, částka, zakoupený typ zboží). Data jsou již předpracována do podoby relační tabulky(co záznam, to jeden zákazník). Typy zboží jsou pevně dány - uvádí se informace, zda byl výrobek koupen, nebo nebyl. Můžeme hledat souvislosti mezi jednotlivými typy zboží, můžeme se zajímat o to, jestli existují skupiny produktů, které si zákazníci kupují současně(např. párek a pivo, mouka a cukr, špagety a kečup). Můžeme objevit, že velmi často se v nákupním košíku objevuje současně pivo, konzervovaná zelenina, zmražené maso nebo ryby, zelenina a ovoce. Jak je vidět z předchozí ukázky, „nezdravou“ a „zdravou“ stravu si nekupují stejní zákazníci. Další věc, která by nás mohla zajímat, by tedy byla, čím se tito zákazníci, skupiny zákazníků vyznačují. Např. pizzu, pivo a fazole kupují muži s nižším přijmem.

Tyto poznatky mají pro obchodníka velký užitek. Může je využít pro vhodně cílenou reklamu v letácích, položky, které se nejčastěji kupují dohromady může umístit do regálu vedle sebe, popřípadě dát na opačnou stranu obchodu, aby zákazník musel projít kolem dalšího zboží, které by si mohl koupit. Výše uvedené funguje na principu, že každá položka v obchodě je reprezentována booleovskou proměnnou značící skutečnost, zda si ji zákazník koupil, nebo nekoupil. Košík je pak brán jako bitový vektor.

6.1.1 Definice asociačního pravidla

Nechť $I = \{i_1, i_2, i_3, ...\}$ je množina položek a nechť D je množina transakcí, kde každá transakce T je množina položek taková, že T $\subseteq$ I. Ke každé transakci přísluší unikátní

identifikátor TID. Nechť A je množina položek. Říkáme, že transakce T obsahuje A, pokud X $\subseteq$ T. Asociační pravidlo je implikace ve tvaru $A \Rightarrow B$, kde $A \subset T, B \subset T$ a $A \cap B = \emptyset$
V rámci tohoto kontextu je důležité zmínit základní pojmy: [27]

Podpora(support) - udává četnost výskytu daného pravidla v databázi transakcí, kterou analyzujeme. Je to objektivní míra asociačního pravidla ve tvaru $X \Rightarrow Y$, kde X a Y jsou množiny položek. Udává se jako relativní (v procentech) nebo absolutní (počet transakcí). V druhém případě musíme znát pro posouzení i celkový počet analyzovaných transakcí. Vyjadřuje tedy pravděpodobnost. $P(X \cup Y)$ kde $(X \cup Y)$ značí, že transakce obsahuje jak položky množiny X tak Y.
Formálně vyjadřujeme jako $podpora(X \Rightarrow Y) = P(X \cup Y)$

Spolehlivost(confidence) - představuje významnost zastoupení položek na pravé straně pravidla v transakcích, kde se vyskytují položky z levé strany pravidla.
Formálně vyjadřujeme jako $spolehlivost(X \Rightarrow Y) = P(Y|X)$

Frekventovaná množina - množina položek, které má podporu vyšší než minimální hodnota(vyskytuje se často v daném vzorku dat).

Silná pravidla - pravidla s vysokou, (předem určenou) hodnotou a spolehlivostí.

Asociační pravidlo je považováno za *zajímavé*, je-li splněna podmínka minimální podpory a minimální spolehlivosti, tedy hodnoty podpory a spolehlivosti jsou vyšší než nějaká předem stanovená mez.

Asociační pravidla můžeme rozdělit na *jednodimenzionální* a *multidimenzionální pravidla.*

Jednodimenzionální pravidla - obsahují jen jediný predikát, např.

$kupuje(X,'testoviny') \Rightarrow kupuje(X,'syr')$

Vícedimenzionální asociační pravidlo - obsahuje obecně více predikátů, např.

$vek(X,'40...50') \wedge vzdelani(X,'vysokoskoleske') \Rightarrow plat(X, 30 - 40tisic)$

Průběh získávání asociačních pravidel

1) Nalezneme frekventované množiny

2) Budeme generovat silná asociační pravidla z frekventovaných množin. Odstraníme ta pravidla, jejichž spolehlivost nedosahuje předem určené minimální hodnoty

6.1.2 Algoritmus Apriori

Tento algoritmus je využíván pro získávání frekventovaných množin. Využívá předchozí znalosti o dříve získaných frekventovaných množinách. Při každé iteraci získané frekventované k-množiny jsou použity pro generování(k+1)-množin. Využívá se tzv. *Apriori vlastnosti*, tedy že každá podmnožina frekventované množiny musí být také frekventovaná. Vychází to z faktu, že přidání prvku k množině nemůže způsobit, že by její podpora vzrostla.

Algoritmus tedy postupně prochází databázi a počítá podporu pro kandidáty. Při každém kroku generuje množiny kandidátů a zkontroluje se jejich minimální podpora. V dalším

kroku vznikne množina kandidátů o velikosti o jednu větší než v předchozím kroku atd. Procedura Apriori_Gen vytváří všechny možné kandidáty z frekventovaných množin a pak z nich vylučuje ty množiny, jejichž některá podmnožina není frekventovaná [27].

Pseudokód Algoritmu apriori
Vstupem: databáze D, minimální hodnota podpory min_sup
Výstup: L - frekventované množiny v D

```
1)   L_1 = nalezni_frekventované_1-množiny(D);
2)     for(k=2; L_{k-1} ≠ Ø; k++) {
2)       C_k = apriori_Gen (L_{k-1}, min_supp);
4)         for each transakce t ∈ D {
5)           C_t = subset (C_k, t);
6)           for each kandidát c ∈ C_t
7)               c.počet_výskytů++;
8)         }
9)         L_k = { c ∈ C_k | c.počet_výskytů ≥ min_supp };
10)    }
11)    return L=U_k L_k;
```

V prvním kroku jsou vygenerovány frekventované l-množiny. Dále se v jednotlivých iteracích spouští funkce Apriori_Gen, která provádí spojovací a vylučovací fázi, tedy generuje kandidáty. Poté se prochází databáze a pro každého kandidáta se spočítá počet jeho výskytů. Kandidáti, jejichž podpora je vyšší než minimální, jsou uloženi do L_k.

Procedura Apriori_Gen

```
1) for each množina l_1 ∈ L_{k-1}
2)    for each množina l_2 ∈ L_{k-1}
3)    if((l_1[1] = l_2[1]) ∧ (l_1[2] = l_2[2]) ... ∧l_1[k-2] = l_2[k-2]) ∧ (l_1[k-1] = l_2[k-1]) )){
4)      c = l_1 JOIN l_2;
5)      if(má_nefrekventovanou_podmnožinu(c, L_{k-1}))
6)        odstraň c;
7)          else přidej c do C_k;
8)    }
9)  return C_k;
```

Kapitola 7

Business Intelligence v MS SQL Serveru 2005

V následujícím odstavci se budeme věnovat MS SQL Serveru 2005 a několika jeho nejdůležitějším součástem pro BI. V kontextu popíšeme i vylepšení architektury oproti MS SQL Serveru 2000.

7.1 MS SQL Server 2005

MS SQL Server 2005 je databázový server od společnosti Microsoft, navazující na MS SQL Server 2000. Poskytuje komplexní datové řešení pro organizace a vývojáře prostřednictvím velkého množství funkcí, integrovatelnosti a spoluprácí se stávajícími systémy. Poskytuje specialistům známé používané nástroje pro správu, nasazování a využívání podnikových dat a správu analytických aplikací. Rozšiřuje uvedený SQL server 2000 o nové výkonné funkce. IT odborníkům umožňuje tvorbu, nasazení a správu podnikových aplikací, zjednodušení a podporu tvorby databázových aplikací, sdílení dat mezi více aplikačními platformami[20]. Datovou platformu SQL Serveru 2005 můžeme rozdělit na následující nástroje: [21]

Relační databáze - relační databázový stroj.

Služba Replication Services - služba pro replikaci dat pro aplikace zpracovávající mobilní či distribuovaná data.

Služba Notification Services - služba sloužící pro zasílání upozornění pro vývoj a nasazení škálovatelných aplikací. Umožňuje zasílat aktuální informace přizpůsobené individuálním požadavkům.

Služba Integration Services - funkce pro ETL (extrakci, transformaci a načítání dat) pro datové sklady a intergraci dat v celém podniku.

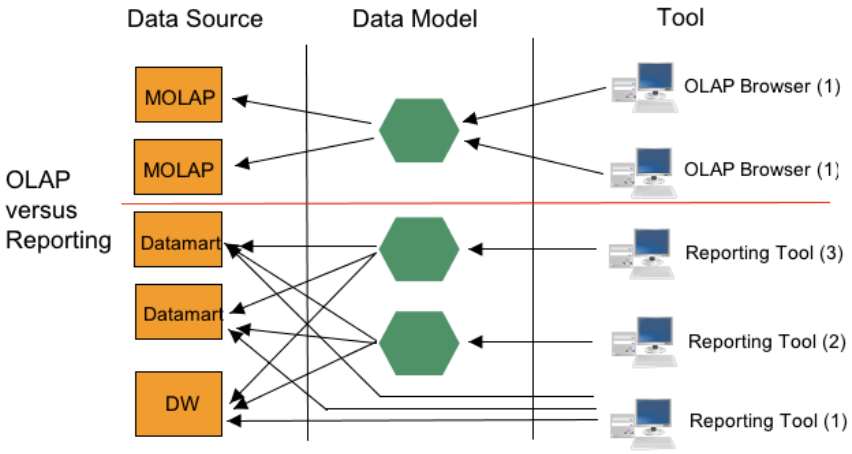
Služba Analysis Services - slouží pro OLAP analýzu velkých datových sad při využití vícedimenzionálních úložišť.

Služba Reporting Services - řešení pro vytváření, správu a zasílání klasických papírových i webových sestav.

Nástroje pro správu - integrované nástroje pro pokročilou správu a ladění databází.

Nástroje pro vývojáře - integrované nástroje pro vývojáře, které jsou určené pro databázový stroj, ETL proces, dolování dat, funkce OLAP a vytváření sestav. Nástroje jsou úzce integrovány se sadou MS Visual Studio a poskytují pokročilé funkce pro vývoj aplikací.

SQL Server 2005 se liší od SQL Serveru 2000 přepracováním architektury. U verze 2000 jsou striktně odděleny vrstvy reportovacích, databázových a analytických služeb(viz obrázek 7.1). Jako jedna z nevýhod se jeví redundance údajů jednotlivých modelů, jelikož stejná data jsou uložena jak v multidimenzionálních databázích, tak i v relačních databázích. SQL Server 2000 postupoval tak, že data nejdříve získal z datového skladu nebo relačních tabulek a vytvořil novou datovou strukturu pro rychlé multidimenzionální dotazování MOLAP. Při přidání několika dimenzí navíc velmi rychle rostly nároky na požadované místo.
SQL Server 2005 využívá *Unified dimensional model*(UDM) (viz obrázek 7.2), kde jsou



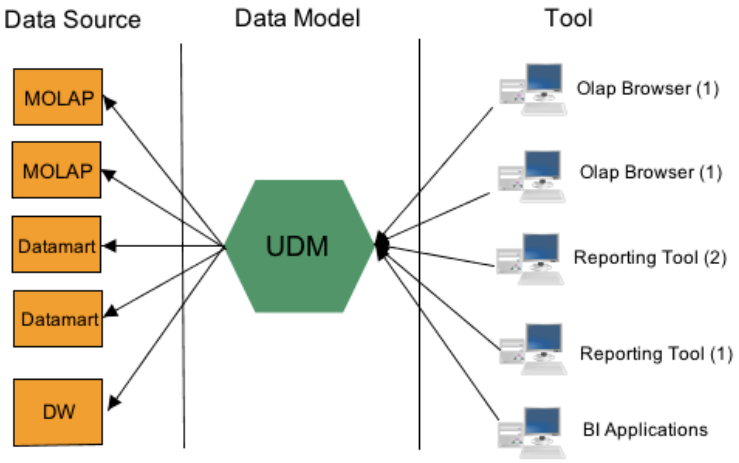
Obrázek 7.1: Oddělení vrstvy OLAP a reportovacích služeb u MS SQL Serveru 2000 [16]

vrstvy Business Intelligence sjednoceny do jednoho pohledu, tudíž je potřeba pouze jeden dimenzionální model pro generování reportů i OLAP kostek [15]. UDM je takové centrální místo, na které se obracejí všechny další služby BI. Verze 2005 používá speciální cache paměť MOLAP, která slouží pro uchovávání agregací u nejčastěji pokládaných dotazů, což zvyšuje výkon serveru[5]. Údaje tedy zůstávají v relačních databázích a vypočítané agregace se ukládají do multidimenzionálních struktur. Při dotazech se údaje vybírají z multidimenzionální cache paměti.

Nás bude, kvůli zaměření práce, z funkcí SQL Serveru 2005 zajímat hlavně implementace Business Intelligence(BI), kterou můžeme rozdělit do čtyř bloků: *Integrace, Analýza dat, Reporty* a *Data mining* [17].

Analytické služby(Analysis services)

MS SQL Server 2005 poskytuje pro analytické služby nástroje MS SQL Server Management Studio, které slouží pro správu databází, analytických objektů a služeb pro analytiky, a Business Intelligence Development studio, které slouží pro vytváření objektů Business Intelligence. BI Development Studio je součástí MS Visual Studia 2005.



Obrázek 7.2: Unified Dimensional Model [16]

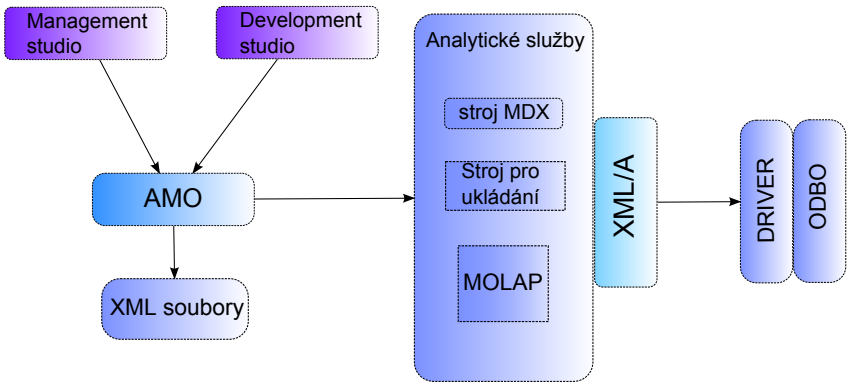
V MS SQL Serveru 2005 je využito nového standardu *XMLA(XML for Analysis)*. Toto rozhraní se používá pro komunikaci s analytickými službami(viz obrázek 7.3). Jedná se o jediné rozhraní, přes které se dá přistupovat ke službám analytického serveru, takže všechny aplikace a klienti musí s MS Analysis services 2005 komunikovat přes XMLA. Je založen na SOAP(Simple Object Access Protocol) a XML(Extensible Markup Language). Definuje dvě metody: Execute a Discover. První z nich se využívá pro vykonání příkazu a vrácení dat. Druhé je pak použito pro práci s metadaty. Výhoda XMLA je nezávislost na platformě a na operačním systému. Naopak mezi nevýhody se řadí relativní pracnost při tvorbě dotazů i při interpretaci odpovědí, které ovšem řeší klientské knihovny.

Integrační služby(Integration Services)

Jako nová služba MS SQL Serveru 2005 nahrazuje službu transformace dat (Data Transformation Services pro SQL Server 2000). Integrační služby zahrnují procesy, které se věnují importu, zavedení dat z produkčních databází do datového skladu MS SQL 2005. Můžeme je využít i v případě sběru dat z různorodých systémů. V architektuře SSIS(SQL Server Integration Services) viz obrázek 7.4, je jedním ze základních prvků DTP(Data Transformation Pipeline), propojující zdrojové a cílové adaptéry. V nižších vrstvách probíhají procesy extrakce a transformace [5].

Reportovací služby(Reporting services)

Tyto služby slouží k generování interaktivních i papírových prezentací provedených analýz, umožňující zaměstnancům efektivní přístup k datům, které potřebují pro svou práci, nebo manažerům, kteří tyto reporty využívají v procesu rozhodování. Architektura reportovacích služeb je uvedena na obrázku 7.5. Základem reportovacích služeb je SQL Server Report Catalog, jehož údaje využívají report servery, určené pro reportovací služby[5] [18]. Ty si do katalogu SQL Serveru ukládají definice reportů, metadata aj. Reportovací server má nad

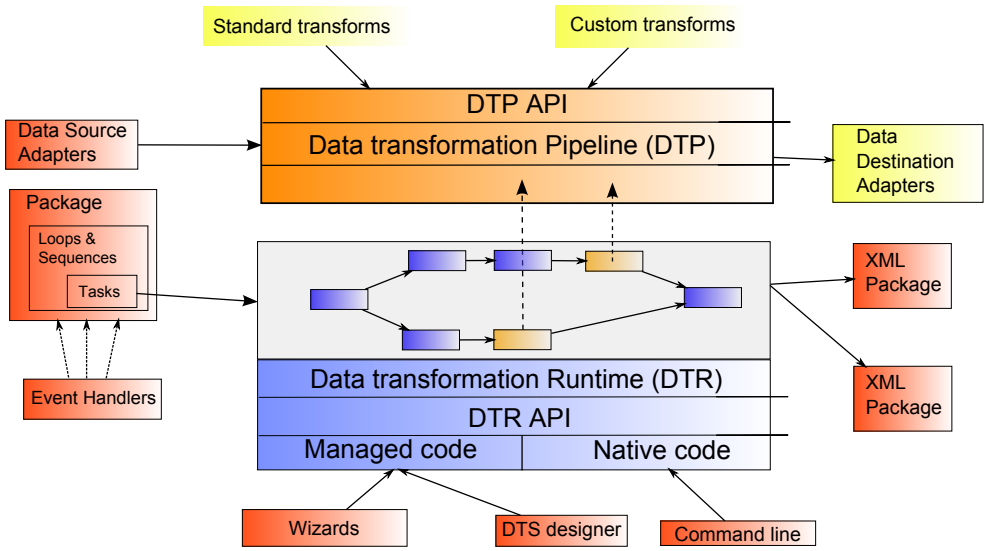


Obrázek 7.3: Schéma architektury analytických služeb MS SQL Serveru 2005 [17]

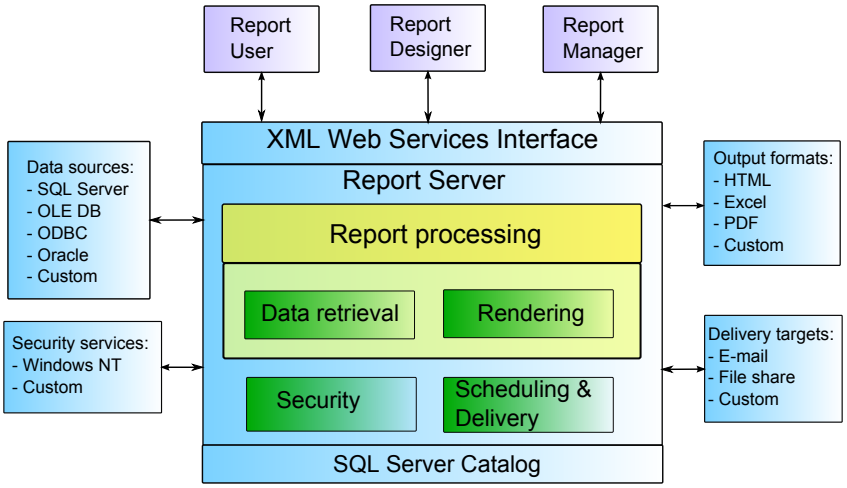
sebou několik aplikačních rozhraní, např. webové služby, WMI(Windows Management Instrumentation) a URL(Uniform Resource Locator). Reportovací služby zahrnují kompletní životní cyklus reportů, tzn. vytvoření reportu vývojářem nebo určeným specialistou, tento report je pak nutné spravovat a buď doručovat nebo zpřístupňovat uživatelům. Všechny tyto činnosti jsou integrovány do MS Management studia. Uživatel si může vygenerovaný report buď zobrazit webovým prohlížečem nebo mu může být doručen emailem. Důležitá je otázka bezpečnosti, jelikož reporty mohou obsahovat citlivá a neveřejná data pro firmu velmi důležitá.
Jedno z možných dělení RS by mohlo být:
Enterprise - prezentující data v podnikové informatice
Embedded - vygenerování reportů je integrální součástí aplikací
B2B(Business to Business) - generování reportů pro obchodní partnery

Data mining

Integrace data miningu s Business Intelligence platformou je výraznou novinkou v SQL Serveru 2005. Při dolování dat tudíž můžeme vyjít z dat získaných a připravených integračními službami, obohacenými o multidimenzionální pohled analytickými službami. MS SQL Server 2005 přináší i nově implementované algoritmy pro data mining. Uvedeme alespoň stručný přehled: *Neuronové sítě, Naive Bayes, analýza časových řad, nevyvážené rozhodovací stromy, vícerozměrné shlukové diagramy, sekvenční shlukování a asociační pravidla.*



Obrázek 7.4: Schéma SQL Server Integration services [5]



Obrázek 7.5: Reportovací služby - architektura a návaznost (viz. [12])

Kapitola 8

Praktická část

V této kapitole bude popsána aplikace realizující dolování asociačních pravidel. Postupně se budemě věnovat vytvoření aplikační databáze v MS SQL Serveru 2005, tvorbě datového skladu, vytvoření datové kostky v SQL Serveru, zavedení dat z aplikační databáze do datového skladu a konečně vlastnímu dolování asociačních pravidel pomocí algoritmu AprioriT. V závěru práce jsou pak zhodnoceny výsledky dolování asociačních pravidel, rychlost jednotlivých algoritmů a rychlost konkrétních dolovacích úloh.

8.1 Uvedení

Cílem práce bylo realizovat integraci modulu pro dolování asociačních pravidel do informačního systému. V business oblasti může mít aplikace pro firmu přínos v tom, že firma už nepotřebuje další drahé komerční nástroje, kdy licence pro jednotlivé produkty mohou představovat nemalou částku. Sloužila by pro jednotlivé manažery nebo obchodníky, kteří si chtějí zjišťovat charakteristiky prodeje, stačí jim tedy jedna webová aplikace, přes kterou se mohou podívat na konkrétní oblast svého zájmu. Pro lokálního manažera představuje výhodu ta skutečnost, že se může zaměřit jen na konkrétní oblast, která ho momentálně zajímá a nemusí si pročítat rozsáhlé souhrnné reporty, které by člověk určený ve velké firmě pro tuto práci připravil. Další výhodu představuje skutečnost, že může oblast svého zájmu sledovat, kdykoliv to budou jeho potřeby vyžadovat, a nemusí čekat na případné doručení reportů. Tímto se zvyšuje operativnost prodeje, neboť podle těchto statistik může už druhý den připravit marketingovou akci třeba na prodej benzínových sekaček a mazacího oleje.

Aplikace by měla umožnit přihlašování jednotlivým uživatelům, přičemž by obchodníci zodpovědní za konkrétní oblasti měli přístup jen k těm datům, které jsou relevantní vůči jejich zájmům. Každý obchodník by už měl mít předpřipravené dotazy také vzhledem k tomu, že se předpokládá alespoň minimální znalost problematiky pokládání dotazů datové kostce. Uživatelům tedy stačí tyto dotazy pouze spustit a není třeba je nijak proškolovat. Aplikace se jim bude pohodlněji využívat a vrchní manažer bude mít přehled o tom, k jakým datům bude mít obchodník přístup. Tím může také zabránit například rivalitě mezi jednotlivými regiony, jelikož obchodník z jednoho kraje bude moci sledovat jen prodej ve svém kraji a nikoliv už ve vedlejším. Toto nastavení už záleží na zodpovědných osobách, které budou o omezeních rozhodovat.

8.2 Použité technologie

Pro připojení k analytickým službám *MS SQL Serveru 2005* je využita knihovna *olap4j* [11], které se budeme věnovat dále. Při tvorbě webové aplikace ve vývojovém prostředí *MyEclipse Enterprise Workbench* v programovacím jazce *Java* byl použit java framework *Stripes* [8] a jako objektově relační mapování na databázi využit *EclipseLink* [1], který umožňuje pohodlné připojení a snadnou práci s uloženými údaji. Je nutné si vysvětlit i pojmy jako *MDX* nebo *XMLA*, těm budou věnovány samostatné odstavce.

8.2.1 Knihovna Olap4j

Knihovna olap4j, která umožňuje realizovat dotazy nad analytickými službami serveru MS SQL 2005 představuje otevřené Java API pro tvorbu OLAP aplikací. Ve své podstatě je olap4j pro multidimenzionální data to stejné jako JDBC pro relační data. Má podobný programovací model, sdílí některé jeho vnitřní třídy a má mnoho podobných výhod. Je možno napsat OLAP aplikaci v Javě pro jeden server (např. Mondrian) a pak jednoduše přejít na jiný (např. MS SQL Server a jeho Microsoft Analysis Services, ke kterým se přistupuje přes XML pro analýzu (XMLA)).

Pokud bychom se chtěli blíže podívat na historii Open Source nástrojů pro OLAP [11], tak nejdříve přišly na svět technologie MDAPI, potom JOLAP API, ovšem ani jedna se nesetkala s velkým úspěchem, kvůli tomu, že žádní výrobci OLAP serverů se nezabývali dodržováním implementací standardů, které byly velké a komplexní. Mezitím Microsoft představil OLE DB(Object Linking and Embedding, Database) pro OLAP, jenž ovšem fungoval jen mezi Windows klientem a serverem a konečně XML/A (XML for Analysis). Tyto standardy už byly daleko úspěšnější, a to z důvodu, že např. OLAP DB bylo vyvíjeno jednou firmou(Microsoft), a také tu byl dotazovací jazyk MDX. Používání jakéhokoli dotazovacího jazyka je jednodušší než využití jakéhokoliv aplikačního rozhraní. Existuje více Open source projektů, které se OLAP zabývají - např. Mondrian, open source OLAP server, JPivot, první klient sloužící pro připojení k Mondrian a nakonec také XML/A.

8.2.2 Jazyk MDX

MDX je zkratka ze slovního spojení *Multidimensional Expresions (multidimenzionální výrazy)* a je pro multidimenzionální databáze určitým ekvivalentem jazyka SQL pro relační databáze[14]. Jeho hlavním cílem je navigace v multidimenzionálních údajích. Pomocí dotazu v MDX vypíšeme vybranou podmnožinu údajů z multidimenzionální struktury(kostka OLAP) do dvourozměrné tabulky, která obsahuje množinu buněk, proto této struktuře říkáme *Cellset*. Stejně jako jazyk SQL ani jazyk MDX není cílem, ale jen prostředkem pro přístup k multidimenzionálním údajům. A to jednak prostřednictvím nějaké klientské konzoly, ale hlavně také pro přístup k multidimenzionálním údajům z různých aplikací. Nejdůležitější pojmy v jazyce MDX jsou:

Prvek (Member) - jedná se o kterýkoliv prvek(objekt) v hierarchii.

N-tice(tupple) - jeden prvek, případně průnik dvou, nebo více prvků.

Množina(Set) - množina n-tic nebo prvků, prakticky je to vlastně množina buněk ve výsledné tabulce.

Dimenze(Osy tabulky buněk) - množina elementů stejného typu, definují se pomocí klauzulí ON ROWS a ON COLUMNS - které představují dvě základní dimenze, ovšem jazyk MDX umožňuje práci až se 128 dimenzemi, které buď můžeme definovat slovně(ROWS, COLUMNS, PAGES, SECTIONS, CHAPTERS) nebo číselně(AXIS<jméno_osy>).

Pro příklad úplná syntaxe příkazu SELECT jazyka MDX vypadá následovně:

```
SELECT [<Specifikace_osy> [, <specifikace_osy>...]]
FROM [<specifikace_kostky>]
[WHERE [<specifikace_řezu>]]
```

8.2.3 Jazyk XMLA

Zkratka *XMLA* značí *XML for Analysis* (XML pro analýzu). Na XMLA je možno nahlédnout třemi způsoby [6]. Zaprvé jako na jednoduché programovatelné API. Jádro služby XMLA obsahuje pouze dvě metody:

Discover - jenž se využívá na získávání metadat ze serveru

Execute - většinou se využívá na spouštění dotazů MDX a získávání výsledků

Tyto metody přijímají a vrací informace ve formátu dokumentů XML, které je možné analyzovat nebo transformovat k dalšímu zpracování nebo zobrazení na obrazovce.

I když je možné XMLA brát jako skutečnou webovou službu, bylo povýšeno na nativní API. V platformě .NET pracuje nad protokolem TCP/IP místo HTTP.

Další možnost pro využití XMLA je nejenom na už zmíněné dotazování, ale i pro vytváření dotazů, které kostky nejen vytváří, ale i mění jejich strukturu nebo na nich provádí sadu úkolů v oblasti zprávy.

Třetí možnost se týká využití ADO MD.NET, s jehož pomocí můžeme přistupovat k obsahu XML navráceného XMLA při zpouštění dotazů MDX.

Jako ukázku [6] si uvedeme načítání metadat a informací o statusu pomocí elementu <*Discover*>. Využijeme k tomu následující příklad:

```
<Discover xmlns="urn:schemas-microsoft-com:xml-analysis">
<RequestType>DISCOVER_SESSIONS</RequestType>
<Restrictions>
<RestrictionList>
</RestrictionList>
<Restrictions>

<Properties>
<PropertList></PropertyList>
</Properties>
</Discover>
```

Element <*Discover*> nám říká, že se jedná o metodu *Discover* webové služby XMLA. Elementy <*RequestType*> a <*Restrictions*> chápeme jako výčtové konstanty a filtrovací

pole GetSchemaDataSet. Element na prvním místě nastavíme na hodnotu DISCOVER_SESSIONS a druhý necháme bez hodnoty. Výsledkem je seznam aktivních relací na serveru. Element <*Properities*> obvykle obsahuje informaci spojení a formátovací informace. XMLA jako takový podporuje dva formáty výsledků. Za prvé v tabulce a za druhé vícerozměrné. Metoda Discover vrací výsledky v tabulce a každý řádek je v samostatném elementu <row> a každý sloupec ve svém samostatném pojmenovaném elementu.

Oproti metodě *Discover* metoda *Execute* používá pro prezentaci vícerozměrný formát. Uvedeme si ukázku této metody a demonstrujeme na následujícím příkladě.

```
<Execute xlmns="urn:schemas-microsoft.com:xml-analysis>
   <Command>
    <Process xmlns="http://schemas.microsoft.com/analysisserveices/2003/engine">
     <Type>ProcessFull</Type>
         <Object>
          <DatabaseID>Miner</DatabaseID>
            <CubeID>Minerdb</CubeID>
         </Object>
    </Process>
    </Command>
 <Properities />
</Execute>
```

Na rozdíl od prvního příkladu jsme zde použili element <*Execute*>, tento dotaz tedy bude provádět akci. Element <*Process*> říká analytickým službám, aby spustily plný proces v krychli Minerdb v databázi Miner.
Jako poslední příklad si uvedeme spouštění dotazů MDX pomocí <*Execute*>. XMLA bylo už od počátku určeno hlavně na spouštění dotazů MDX (a na průzkum metadat) a snadno akceptuje model množin buněk s osami, pozicemi a kolekcemi buněk. Každý z těchto prvků je hezky zabalen ve vícerozměrném návratovém XML. Např.

```
<Execute xmlns="urn:schemas-microsoft-com:xml-analysis">
  <Command>
    <Statement>
      SELECT [Customers].[Customer Name].MEMBERS ON COLUMNS,
      [Time].[Year].Members ON ROWS
      FROM Minerdb
      WHERE
      [Customers].[Company Name].[Santé Gourmet]
    </Statement>
  </Command>
  <Properties>
    <PropertyList>
      <Catalog>Miner</Catalog>
    </PropertyList>
  </Properties>
</Execute>
```

V našem případě se element <Command> skládá pouze z elementu <Statement>, jehož hodnotou je příkaz MDX, který chceme spustit. Pokud tento dotaz spustíme, dostaneme vícerozměrné výsledky. Bude v nich velký dokument XML. Přeskočíme počáteční uzly, a všimneme si např. elementu AxesInfo, kde jsou popsány všechny osy, a dále pak i osa „Slicer“, představující klauzuli WHERE v dotazu. V elementu Axis jsou pak vypsány všechny tuple i členy. Jako příklad si zobrazíme osu Columns, jenž je označena jako Axis0.

```
<Axes>
      <Axis name="Axis0">
        <Tuples>
          <Tuple>
            <Member Hierarchy="[Customers].[Customer Name]">
              <UName>[Customers].[Customer Name].[All]</UName>
              <Caption>All</Caption>
              <LName>[Customers].[Customer Name].[(All)]</LName>
              <LNum>0</LNum>
              <DisplayInfo>65628</DisplayInfo>
            </Member>
          </Tuple>
          <Tuple>
            <Member Hierarchy="[Customers].[Customer Name]">
              <UName>[Customers].[Customer Name].&amp;[Jonas Bergulfsen]</UName>
              <Caption>Jonas Bergulfsen</Caption>
              <LName>[Customers].[Customer Name].[Customer Name]</LName>
              <LNum>1</LNum>
              <DisplayInfo>0</DisplayInfo>
            </Member>
          </Tuple>
          ...
```

8.2.4 Zprovoznění technologií

Pro práci s MS SQL Serverem jsem nejprve jako platformu zvolil Windows XP 64bit SP2. Po nainstalování SQL Serveru jsem se snažil server aktualizovat stažením a nainstalováním jednotlivých opravných balíků pro SQL server. Při aktualizaci posledního balíčku(SP3) ovšem vždy zhavarovala nejenom aktualizace, ale dokonce i operační systém Windows. Toto se muselo vyřešit aktualizací celých Windows XP na SP3, po kterém již aktualizace SQL serveru proběhla úspěšně.
V první řadě je nutné připomenout, že pro pro TCP/IP připojení k SQL serveru z vzdálené aplikace je nutné povolit TCP/IP připojení v SQL Server Configuration manageru. Dále je potřeba v nastavení analytických služeb(Analysis server Properities) nastavit příslušná přístupová práva, aby nás k datům SQL Server vůbec pustil, viz. [26].
Při připojování k analytickým službám pomocí knihovny olap4j je ve specifikaci knihovny v ukázkovém příkladu uvedena jako knihovna k připojení msxisapi.dll, kvůli níž je nutné stáhnout a nainstalovat balíček Microsoft XML For Analysis SDK. To vše jsem provedl, nainstaloval příslušný Internet Information Services (IIS) verze 6, který jsem měl k dispozici na instalačním médiu k MS Windows. I po tomhle všem se ale zprovoznit připojení k Analytickým službám SQL serveru nepodařilo. Po delším experimentování, konzultací s vedoucím práce a hledání všech možných řešení jsem dospěl k závěru, že uvedenou

knihovnu lze použít zřejmě jen pro připojení ke starší verzi SQL Serveru verze 2000. K připojení k SQL Serveru 2005 bylo tedy nutno využít knihovny msmdpump.dll distribuovanou v instalaci MSSQL Serveru 2005. Problém byl zřejmě i v použitém IIS 6, jelikož se nepovedlo rozběhnout ani uvedenou knihovnu na Windows XP, natož se k serveru připojit. Dospěl jsem tedy k závěru, že nejlepší a nejjednodušší řešení bude nainstalovat Win XP SP3 v 32 bitové verzi, IIS verze 5 a dále pak využít knihovnu msmdpump.dll. Při zprovozňování knihovny msmdpump.dll [22] jsem pak narazil na (naštěstí řešitelnou) chybu ve Windows a IIS 5 viz. [24]. Další problém se ukázal při provozu aplikace a to opětovnou chybou java.lang.OutOfMemoryError: Java heap space, kterou je nutné vyřešit zvětšením paměti pro webový server Apache Tomcat. Ve vývojovém prostředí MyEclipse Enterprise WorkBench, který jsem při tvorbě aplikace využíval, je nutné v záložce Preferences najít položku MyEclipse Tomcat, který představuje nakonfigurovaný Tomcat pro toto vývojové prostředí a přidat ve volbě JDK jako dodatečné Java VM argumenty jako např. -Xms1024m -Xmx1024m. Mimo vývojové prostředí Eclipse je nutné nastavit proměnnou CATALINA_OPTS a restartovat Tomcat. Příklad tohoto nastavení (pro různé operační systémy) může být následující:

```
set CATALINA_OPTS="-Xms512m -Xmx512m" (Windows)
export CATALINA_OPTS="-Xms512m -Xmx512m" (ksh/bash)
setenv CATALINA_OPTS "-Xms512m -Xmx512m" (tcsh/csh).
```

Pokud nebyla tato proměnná nastavena, pak např. při odvozování asociačních pravidel při použití následujícího MDX dotazu:

```
SELECT
NONEMPTY([Time].[Year].MEMBERS) ON COLUMNS,
NONEMPTYCROSSJOIN([Customers].[Company Name].MEMBERS,
[Customers].[Customers].MEMBERS,
[Geography].[Country].MEMBERS,
[Products].[Category Name].MEMBERS, [Shops].[Name].MEMBERS) ON ROWS
FROM [Minerdb]
```

už tento výpočet pravidel kvůli nedostatečné paměti nebyl možný. Řešení výše shrnutých problémů zabralo opravdu hodně času i energie a jakmile se naskytne volná chvilka, plánuji o tom vydat nějaký ucelený návod na internetu, který by lidem, zabývajícím se touto tématikou pomohl podobné problémy rychle vyřešit.

8.3 Algoritmus pro dolování asociačních pravidel

Jako algoritmus, který bude dolování asociačních pravidel realizovat, jsem si vybral algoritmus AprioriT(Apriori Total)[7], a to jeho jednotlivé modifikace. AprioriT je algoritmus pro dolování asociačních pravidel, vyvinutý vývojovým týmem LUCS-KDD [2]. Algoritmus byl původně vyvinut jako součást sofistikovanějšího ARM algoritmu - Apriori TFP (Apriori Total From Partial).

V aplikaci jsou k dispozici, jak už bylo uvedeno, 3 typy algoritmu AprioriT: *AprioriT*, *AprioriT sorted* - jedná se o aplikaci algoritmu aprioriT s předzpracovanými vstupními daty, jsou seřazeny v souladu s frekvencí jednotlivých položek - redukuje se doba výpočtu algoritmu.
AprioriT sorted pruned - opět se jedná o aplikaci algoritmu AprioriT s předzpracovanými

vstupními daty, která jsou seřazena v souladu s frekvencí jednotlivých položek a položky reprezentující nepodporované 1-množiny jsou odstraněny - opět se redukuje doba výpočtu algoritmu.

Algoritmus využívá strukturu T-stromu(T-Tree), která slouží pro ukládání informací o frekventovaných množinách. Co odlišuje strukturu T-tree od jiných stromových struktur je fakt, že jednotlivé úrovně v každé podvětvi jsou definovány jako pole, to dovoluje indexování na všech úrovních a následně umožňuje výpočetní výhody. Na podporu tohoto indexování je strom postaven v obráceném pořadí.

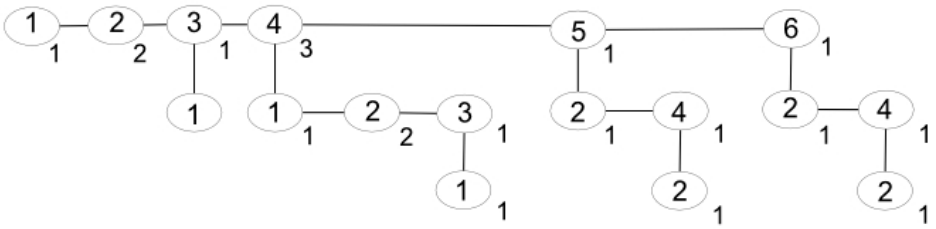
Pokud máme například datový soubor :

{1,3,4}
{2,4,5}
{2,4,6}

a předpokládáme podporu menší jak jedna, dostáváme následující frekventované množiny (podpora je vypočítána v závorkách)

1 (1)	1 3 (1)	4 5 (1)
2 (2)	1 4 (1)	4 6 (1)
3 (1)	2 4 (2)	1 3 4 (1)
4 (3)	2 5 (1)	2 4 5 (1)
5 (1)	2 6 (1)	2 4 6 (1)
6 (1)	3 4 (1)	

Tyto mohou být prezentovány v T-stromu viz. obrázek 8.1. Vnitřní reprezentaci tohoto obráceného T-stromu založeného na poli uzlů T-stromu si můžeme ukázat na obrázku 8.2

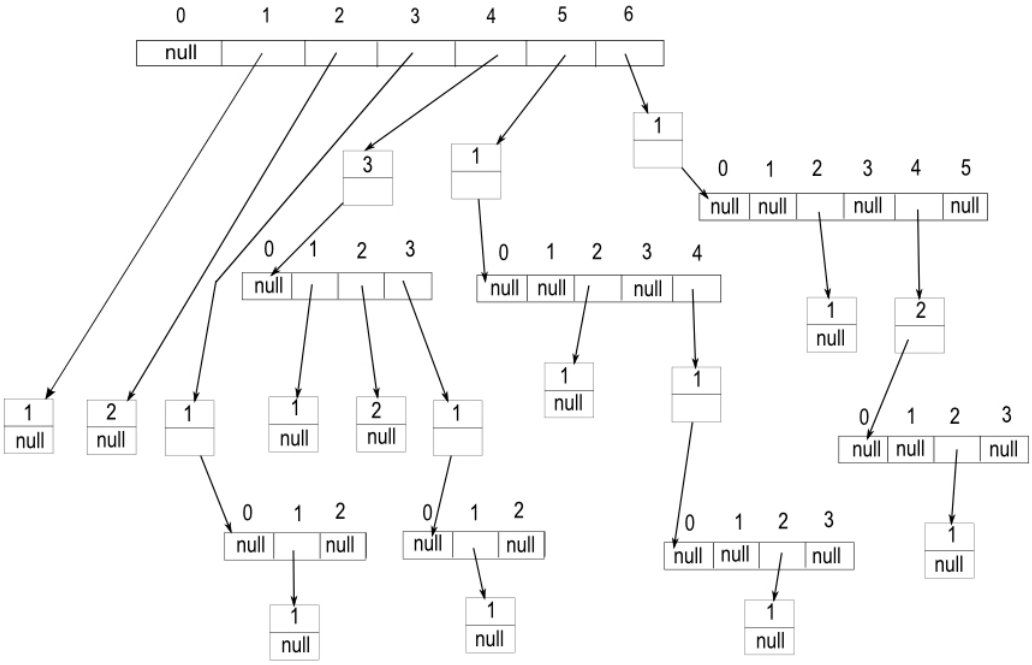


Obrázek 8.1: Reprezentace T-stromu [7]

Výhody při ukládání plynou z toho, že není nutné explicitně uchovávat indexy jednotlivých atributů, tzn. čísla reprezentujících prvky. Samozřejmě tento přístup vyžaduje také ukládání náhražek tam, kde uzly chybí. Výhoda ukládání je tedy částečně závislá na počtu chybějících kombinací obsažených v souboru dat.

8.4 Návrh a vybudování produkční databáze a datového skladu

Data pro relační databázi jsou čerpána z ukázkové databáze Microsoft Northwind, kterou lze stáhnout z [23]. Byla použita jen ta data, jenž jsou relevantní vzhledem k náplni práce a bylo možno z nich dostat relevantní výsledky. Vybudovaný datový sklad může sloužit manažerům jako podklad pro plánování dalšího vývoje firmy, kdy si mohou zjistit např.



Obrázek 8.2: Interní reprezentace T-stromu [7]

které výrobky se prodávají v kterém regionu nejvíce, jaké výrobky se prodávají nejvíce společně a podobně.

Datový sklad v této aplikaci je přizpůsoben datům uloženým v aplikační databázi. Pro naše řešení bylo vybráno schéma hvězdy, kde centrální tabulka faktů obsahuje celkovou cenu prodeje a dále pak obsahuje jednotlivé dimenze - zákaznickou dimenzi, produktovou dimenzi, časovou dimenzi, dimenzi oblasti prodeje, dimenzi prodeje a dimenzi prodejce.

V následujícím odstavci bude popsán relační model datového skladu. Mohli bychom pro uložení tabulek dimenzí a faktů použít novou databázi, ale pro jednoduchost je využita stejná databáze s produkčními tabulkami pro online zpracováni transakcí OLTP. Jelikož jsou tabulky faktů a dimenzí samostatné tabulky, uživatelé mohou nad nimi spouštět operace, aniž by OLTP tabulky byly zatěžovány. V návrhu jsou použity jak klasické tabulky, v našem případě tblFact - tabulka faktů, tblGeography - geografická dimenze, tblTime - časová dimenze, tak i pohledy - vwCustomers - zákaznická dimenze, vwProducts - produktová dimenze, vwSales - dimenze prodeje a vwShops - dimenze obchodů.

Tabulka faktů byla naplněna z produkční databáze následujícím SQL příkazem:

```
SELECT Sale.saleID,
       Product.productID,
   Customer.customerID,
       Sale.shopID,
       tblGeography.geographyKey,
       tblTime.TimeKey,
```

```
       SUM([SaleDetail].unitPrice * [SaleDetail].quantity)  AS [totalSales]

FROM Sale INNER JOIN
    [SaleDetail] ON
    Sale.saleID = [SaleDetail].saleID
    INNER JOIN Customer ON
    Sale.customerID = Customer.customerID
    INNER JOIN Product ON
    [SaleDetail].productID = Product.productID
    INNER JOIN tblTime ON
    [Sale].saleDate = tblTime.saleDate
    INNER JOIN tblGeography ON
    [tblGeography].[postalCode] = Customer.postalCode
    AND [tblGeography].[city] = Customer.city
    AND [tblGeography].[street] = Customer.street
    AND [tblGeography].[country] = Customer.country

GROUP BY Sale.saleID, Customer.customerID, Sale.saleDate,
tblGeography.geographyKey, Product.productID, Sale.shopID, tblTime.TimeKey
```

Dimenze produktů, představující jednotlivé produkty, které byly prodány v konkrétních prodejích, byla naplněna následovně:

```
SELECT Product.productID,
Product.name AS productName,
category.name AS categoryName,
Product.unitsInStock,
Product.discontinued FROM Product
INNER JOIN Category
ON Product.categoryID = category.CategoryID
```

Dimenze zákazníků, obsahující zákazníky, kteří figurují v prodejích, byla naplněna způsobem:

```
SELECT customerID, customerName, companyName, customerPosition, phone, fax
FROM Customer
```

Geografická dimenze, představující místa bydliště jednotlivých zákazníku naplněna jako:

```
SELECT DISTINCT city, postalCode, street, region, country
FROM  Customer
ORDER BY [city], [postalCode]
```

U vytvoření *časové dimenze* můžeme použít dva postupy. První představuje naplnění dat z produkčních tabulek, takže obsahuje jen ta data, která se konkrétních prodejů týká, nebo můžeme použít speciální dimenzi SQL Serveru 2005, která se jmenuje dimenze Serverového času. Tato implementace časové dimenze používá pseudodimenzionální tabulku generovanou na serveru a pole předdefinovaných atributů a hierarchií. V našem případě ale pro názornost a jednoduchost využijeme první možnost, kdy data získáme z tabulky prodeje. Tabulky dimenze času jsou speciálně navrženy, aby obsahovaly klíč a sloupce s odpovídajícím rokem, měsícem nebo jinými hodnotami času.

```
SELECT DISTINCT saleDate, YEAR(saleDate) AS [Year],
{ fn QUARTER(saleDate) } AS Quarter,
MONTH(saleDate) AS [Month]
FROM Sale
ORDER BY [Year], [Quarter], [Month]
```

A nakonec vytvoříme *dimenzi obchodů* a *dimenzi prodeje* následovně:

```
SELECT Shop.shopId, Shop.name,Shop.phone, shop.homePage
FROM Shop


SELECT Sale.saleID, Sale.freight
FROM Sale
```

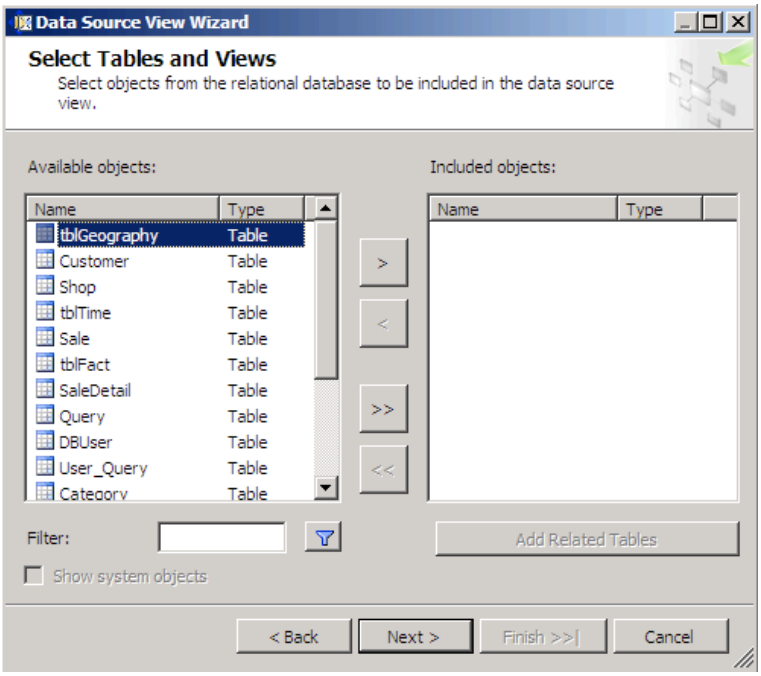
Dimenze prodeje může obsahovat např. datum prodeje, jestli nebyl prodej stornován, datum proběhlé reklamace, pokud by se vyskytla, nebo položku poštovné. Některé dimeze v našem příkladu neobsahují moc relevatních údajů, ale to je jen kvůli nedostatku reálných dat. Mohli jsme si nějaká taková data vymyslet a uměle nagenerovat, ale náplní práce není odvozovat konkrétní výsledky z produkční databáze, ale vytvořit systém, který nad daty analýzu umožňuje.

8.5 Vytvoření datové kostky v MS SQL 2005

V této kapitole stručně shrneme vytvoření datové kostky v prostředí MS SQL Serveru 2005 a SQL Server Business Intelligence Development Studio, které je k dispozici přímo v instalačních médiích SQL serveru. Jako zdroje informací byly použity hlavně [6] a [19]. Nejdříve je samozřejmě nutné v databázi vytvořit všechny potřebné tabulky a pohledy, jenž v datové kostce budeme využívat a to za pomocí Microsoft SQL Server management studia. V dalších krocích již budeme využívat služby již zmíněného SQL Server Business Intelligence Development Studia, pod jehož honosným názvem se skrývá ořezaná verze Microsoft Visual Studia 2005(VS), jelikož si firma Microsoft uvědomila, že všichni analytici a vývojáři nemusí mít VS 2005 k dispozici. Stalo se tak z toho důvodu, aby zákazník dostal společně s SQL Serverem i nejzákladnější verzi vývojového prostředí Visual Studia, ve kterém by mohl používat různé návrháře a vytvářet projekty pomocí komponent BI SQL Serveru(využívajících VS), i když na celý produkt Visual Studia nevlastní licenci.

Pokud už má uživatel jisté zkušenosti s vytvářením datové kostky, je práce s jednotlivými průvodci služeb poměrně rychlá. Dále budou naznačeny jen nejdůležitější věci, které by mohly uživateli v celém procesu pomoci. Podrobnější informace lze nalézt už ve zmíněných [6] nebo v [19]. Nejdříve musíme ve VS vytvořit nový Business Intelligence projekt, konkrétně Analysis Services Project, dále pak přidat datový zdroj, což představuje naše produkční databáze - pomocí průvodce Data Source wizard. Po přidání datového zdroje je nutno specifikovat, které objekty(tabulky z naší databáze, představujících tabulku faktů a množinu dimenzí) chceme použít pro stavbu naší kostky. Musíme tedy specifikovat pohled na datový zdroj, což učiníme opět přes průvodce Data Source View Wizard, který nás přidáním přehlednou formou provede.

Po ukončení průvodce jsme automaticky přepnuti do pohledu na datový zdroj v návrháři. Zde vidíme množinu tabulek a pohledů, které jsme si v předchozích krocích vybrali. V

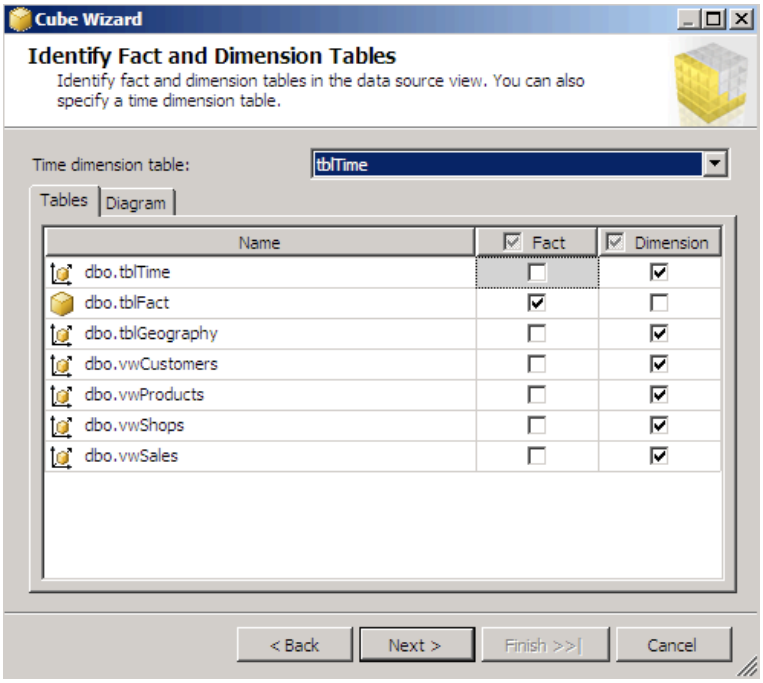


Obrázek 8.3: Dialog průvodce Pohledu na datový zdroj

tomto pohledu musíme specifikovat propojení mezi tabulkami, pokud ještě žádné takové předdefinované vazby neexistují. Vše je uděláno poměrně intuitivně a tyto vazby vytvoříme prostým přetáhnutím kurzoru myši z jedné tabulky do druhé a spojením příslušných sloupců. Po skončení celého procesu upravování vazeb mezi tabulkami již můžeme přejít k samotnému vytváření kostky. A jako na skoro vše, i zde využijeme průvodce Cube wizard. Při samotném procesu tvorby kostky je pro náš případ nejvhodnější využít metodu Build the Cube Using a Data Source - využití připraveného datového zdroje. Analytické služby mohou dokonce prozkoumáním zvoleného schématu našeho pohledu na datový zdroj samy určit, které tabulky představují tabulku faktů a které představují tabulky dimenzí.

V určitých případech je možné i určit, jaké konkrétní hierarchie dimenzí se mohou vytvořit. Ve formuláři Identify Fact And Dimension Tables, sloužícím pro určení, která tabulka přísluší dimenzi nebo tabulce faktů je nutné určit dimenzi času, v našem případě ji budeme určovat z konkrétní tabulky dimenzí tblTime. V příslušném formuláři(Select The Dimension Type) tedy zvolíme Time Dimension, naši tabulku tblTime, v dalším bodě si označíme ve formuláři které sloupce budou odpovídat jakému konkrétnímu časovému nastavení - vybereme dvojice Year/Year, Quarter/Quarter, Month/Month a OrderDate/Date. V nabídce Select Measures budeme vyzváni, abychom vybrali své metriky. Nakonec u nabídky Review New Dimensions si můžeme označením nebo odznačením zrušit nebo povolit používání libovolného atributu nebo ho případně přejmenovat na vhodnější název. Ke konci celého průvodce je možné změnit název námi nadefinované kostka. Celý průvodce končí obligátním tlačítkem Finish a následovným přepnutím do Návrháře krychlí a záložky Cube Structure. Musíme podotknout, že nástroj Návrhář krychlí je hlavní nástroj pro navrhování krychlí, průvodce Cube wizard je jen pomůckou(uživatelským rozhraním), které návrháři pomáhá začít.

Výsledná struktura kostky je na obrázku 8.6 V Návrháři krychlí můžeme specifikovat detailně všechny operace znova, jako je přidávání nové míry(pomocí dialogu New measure),



Obrázek 8.4: Identifikace faktů a dimenzí v průvodci Cube wizard

dále můžeme kontrolovat, přidávat a spravovat dimenze, atributy, hierarchie.

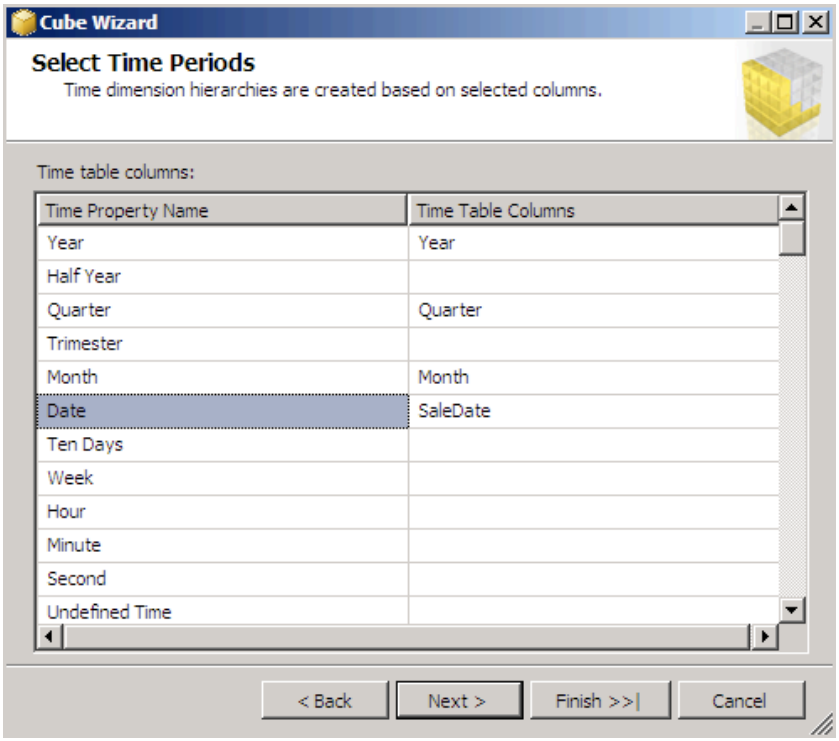
Pokud jsme již dokončili celý návrh kostky, jejich dimezí, atributů a dalších nezbytných vlastností, pak již nebrání nic v cestě k tomu, abychom přistoupili ke zpracování kostky, která vyžaduje kompilaci a umístění na server. Zpracování provedeme volbou Build z položky v hlavní nabídce. Celý postup kompilace a umístění na server může návrhář sledovat v dialogu Deployment Progress, kdy je uživatel po skončení vyzván v Dialogu Process Cube ke zpracování naší kostky.

Po skončení zpracování jsme informováni o případných obtížích při zpracovávacím procesu nebo naopak o úspěšném dokončení. Poté již nic nebrání v tom, abychom mohli nad danou krychlí začít vykonávat dotazy. Pro tyto účely použijeme záložku Browser, jenž obsahuje přehledný prohlížeč. Do tabulky uprostřed již můžeme přetahovat konkrétní míry a osy jednotlivých dimenzí. I když je výsledková sada omezena jen na dvě osy, i tak se jedná o veliké vylepšení vzhledem k nástroji Cube Browser v SQL Serveru 2000. Daný prohlížeč dovoluje zanořování do jednotlivých hierarchií a přidávání filtrů v podobě jednoduchých logických výrazů nebo složitých dotazů MDX.

8.6 Popis programu

Aplikace představuje jednoduchý informační systém s modulem pro dolování asociačních pravidel. Jak už bylo řečeno, informační systém obsahuje jen základní funkčnost, jelikož cílem nebylo vytvořit např. kompletní elektronický obchod, ale demonstrovat začlenění dolovacího modulu do IS. Jelikož je využita knihovna olap4j pro připojení k analytickým službám MS SQL Serveru 2005, neměl by být problém rozběhnout modul nad jakýmkoliv jiným serverem s podporou XML/A.

Celá aplikace představuje IS s daty vztahující se k prodeji v elektronických obchodech, jenž realizují v tomto případě prodej nápojů, koření, cukroví, mořských produktů



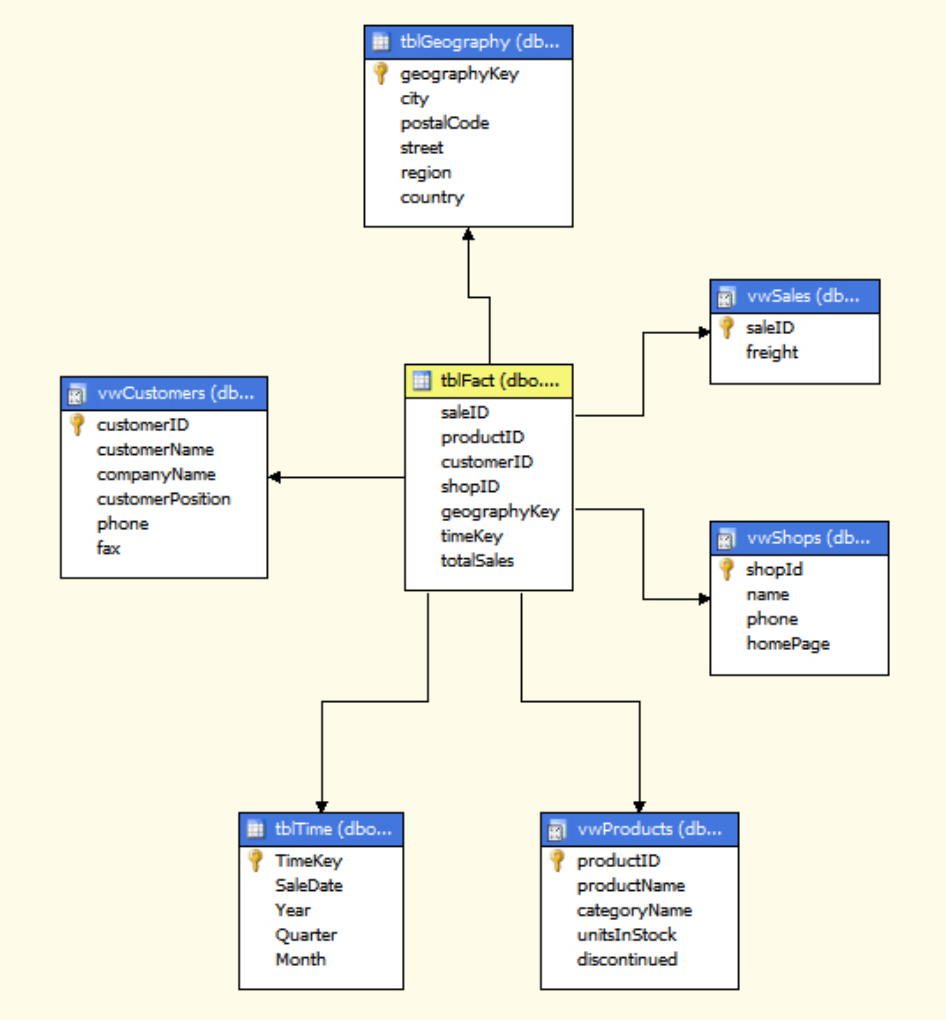
Obrázek 8.5: Identifikace položek časové dimenze

aj. Obsahuje základní operace se zákazníky, údaje o internetových obchodech, v nichž se konkrétní nákupy realizovaly. Dále v IS uchováváme informace o obchodních transakcích a o položkách, produktech, které byly koupeny konkrétním zákazníkem při transakci. Každý výrobek náleží do kategorie výrobků, čehož pak v dolování můžeme využít v přechodu o úroveň výše, jestliže nás nezajímá prodej konkrétního typu špaget, ale těstovin jako celku.

Informační systém obsahuje i přihlašování uživatelů, pro administrátora aplikace jsou dostupné všechny funkce, pro ostatní uživatele představující konkrétní manažery nebo obchodníky je funkčnost omezena. Je to z toho důvodu, aby se konkrétní uživatelé dostali jen k těm informacím, na které mají opravdu nárok, a také je to pro ně pohodlnější, jelikož většinou budou provádět omezenou množinu dotazů. Fakt, jestli bude moci obyčejný uživatel zapisovat nové informace do IS, pak bude záležet na konkrétní obchodní a bezpečnostní politice firmy. V našem informačním systému mu tyto funkce umožníme.

Aplikace se skládá z několika modulů vytvářejících funkčnost celého systému. Jedná se o moduly implementované v rámci IS realizujícího přidávání, editaci a odebírání jednotlivých entit z produkční databáze a dále pak o modul realizující dolování asociačních pravidel z datové kostky. Práce s jednotlivými moduly je poměrně intuitivní, uživatel je přehlednou formou informován o právě probíhajích operacích, ve formulářích jsou uvedeny povinné položky, jenž musí být při přidávání vyplněny, jinak není operace provedena, o čemž je zákazník samozřejmě informován.

Struktura jednotlivých modulů je vždy rozdělena mezi JavaBean třídu uchovávající data o objektu, a ActionBean třídu, plnící službu kontroleru a zajišťující zpracování akce od uživatele. Jednotlivé operace s daty uchovanými v JavaBean objektu, jako je vyhledání



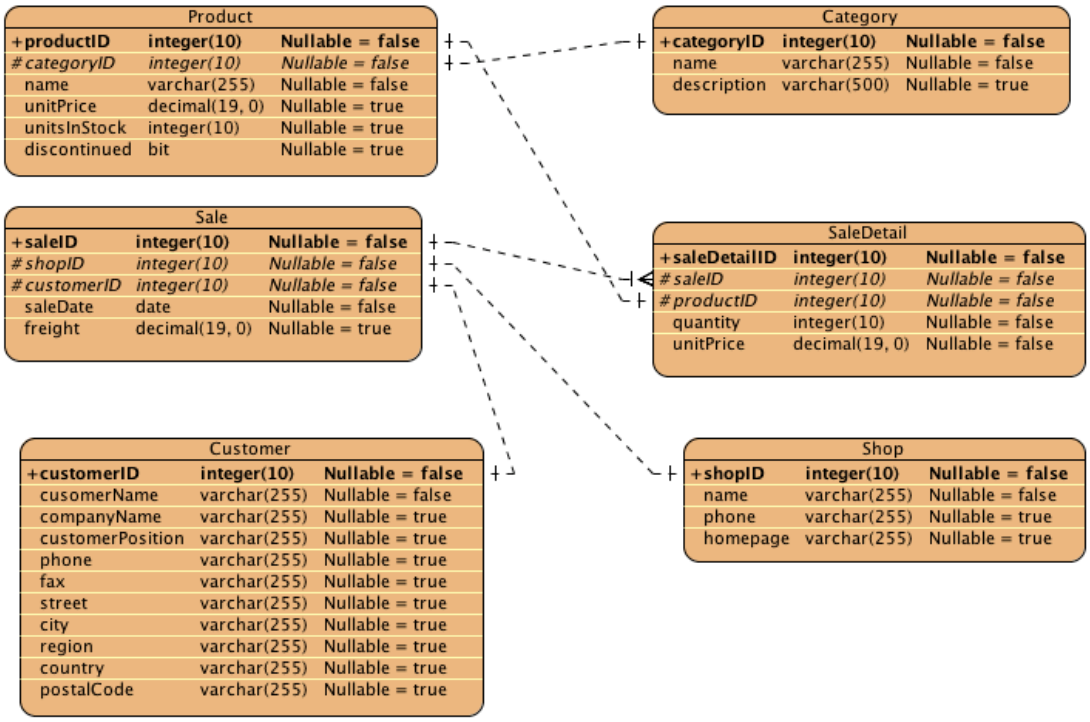
Obrázek 8.6: Struktura datové kostky v nástroji SQL Server Business Intelligence Development Studio

příslušné entity v databázi podle konkrétního parametru, uložení, aktualizace nebo smazání, jsou implementované konkrétním DAO(Data Access Object) objektem. Ty jsou soustředěny v balíčku model. Pro prezentaci dat v prohlížeči je použita JSP(JavaServer Pages) vrstva, kdy je oddělena logika aplikace od prezentační vrstvy. Pro připojení a komunikaci s databází moduly používají ORM(Object-relational mapping) mapování objektů na databázové tabulky knihovny EclipseLink a standardu JPA(Java Persistence API). Díky použití JPA se komunikace s databází hodně zjednodušila, jelikož se data z databáze automaticky načítají do objektů jazyka Java a stejně tak i zpátky ukládají do databáze.

Moduly realizující informační systém

Obchod(Shop) - realizuje přidávání obchodů, v nichž k obchodní transakci došlo, jejich editaci nebo vymazání ze systému. Pro obchod uchováváme např. jméno, telefon nebo webovou adresu.

Zákazník(Customer) - slouží k práci se zákazníkem, jenž transakci provedl. Jedná se



Obrázek 8.7: Schéma databáze (pouze část)

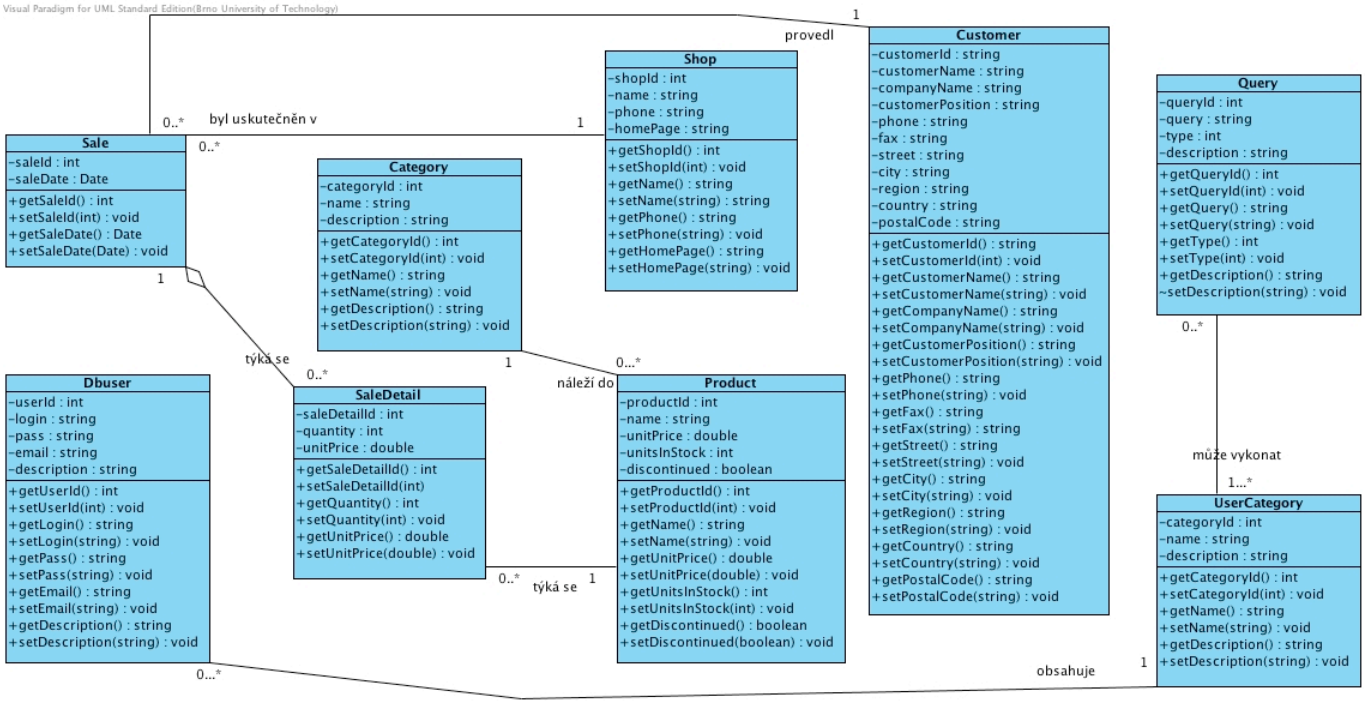
o firemní zákazníky nakupující v obchodech a jednotlivé firmy jsou reprezentovány svým zástupcem. Je tedy u něj uvedeno jméno, příjmení, název firmy, pozice ve firmě a adresa firmy. Adresu pak využíváme jako data pro geografickou dimenzi v datové kostce.

Prodej(Sale) - zabezpečuje všechny operace týkající se konkrétní transakce a přidávání jednotlivých položek, které si během jednotlivého prodeje zákazník koupil. U prodeje uchováváme například čas transakce, obchod, v němž byla transakce uskutečněna, zákazníka realizujícího nákup a poštovné, jenž si obchod účtuje za dopravu k zákazníkovi. U jednotlivých položek transakce pak konkrétní produkt, množství a jednotkovou cenu.

Produkt(Product) - uchovává informace o produktech, jenž si zákazníci během transakcí koupili. Pro produkt uchováváme informaci o kategorii, v které se produkt nachází, o názvu produktu, ceně za jednotku a o počtu jednotlivých kusů v balení, jestliže se produkt neprodává samostatně. Uchováváme i informaci o tom, zda se daný produkt ještě prodává nebo neprodává.

Kategorie(Category) - slouží k ukládání kategorií, do které každý produkt náleží. V tomto případě se jedná o položky kategorie jídel jako jsou nápoje, cukrovinky, mořské produkty nebo mléčné výrobky.

MDX dotaz(Query) - pro uživatele je předdefinovaná sada multidimenzionálních do-



Obrázek 8.8: Class diagram modelu uloženého v databázi

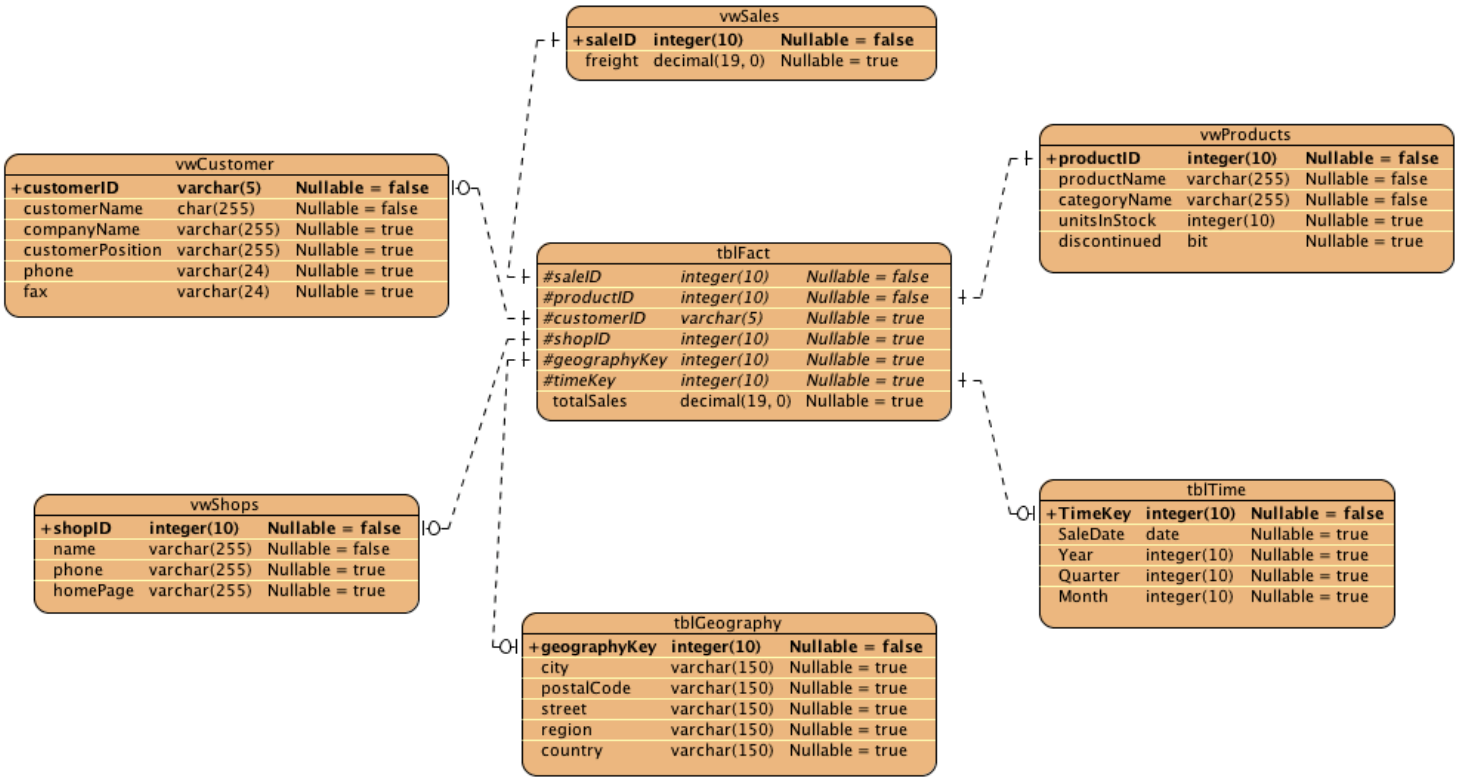
tazů, jenž si v rámci své kategorie, do níž náleží, může spouštět. Tato funkce je zde obsažena pro větší uživatelský komfort obchodníka využívajícího aplikaci a také kvůli bezpečnostní politice firmy, aby si uživatelé mohli spouštět jen dotazy, na které mají přidělená práva. Pro multidimenzionální dotaz uchováváme mimo jiné kategorii uživatelů s právem tento dotaz spouštět a typ dotazu, zda se jedná o dotaz pro dolování asociačních pravidel z charakteristiky nákupu nebo o analýzu nákupního košíku.

Uživatel(User) - tento modul zahrnuje práci s informacemi o uživatelích informačního systému, uchovává se přihlašovací jméno, emailová adresa a kategorie uživatelů, do níž náleží.

Kategorie uživatelů(User category) - obsahuje informace o jednotlivých kategoriích, v nichž se uživatelé nacházejí. Podle této informace, v jaké kategorii se uživatel nachází, pak k uživateli přiřazujeme množinu MDX dotazů, jenž má právo spouštět.

Modul realizující dolování asociačních pravidel

Modul se skládá z několika částí, povětšinou soustředěných do balíčku minerModule. Jedná se o třídy OlapConnectionManager, starající se o připojení k analytickým službám MS SQL Serveru 2005 a dále pak o třídu MinerModule, ve které probíhá většina operací s výsledkem vráceného dotazu od SQL serveru, jeho zpracování, dále je na výsledek proveden určený algoritmus. Po analýze a vyhodnocení jsou pak vyhodnocené závěry prezentovány do webového prohlížeče uživatele. O odchytávání uživatelských příkazů a akcí se v případě dolovacího modulu stará MinerModuleActionBean z balíčku miner a prezentaci výsledků zajišťují příslušné jsp soubory.



Obrázek 8.9: Podrobné schéma datové kostky

Důležitou informací pro uživatele je fakt, že dotazy nad datovou kostkou mohou být dvojího typu. Zaprvé analýza nákupního košíku, kdy nás zajímá, které produkty se prodávají nejčastěji společně. Příkladem takového dotazu může být:

```
SELECT NONEMPTY([Products].[Products].MEMBERS) ON COLUMNS,
NONEMPTY([Sales].[Sales].MEMBERS) ON ROWS
FROM [Minerdb]
WHERE [Customers].[Company Name].&[Blauer See Delikatessen]
```

Na osu COLUMNS - do sloupců jsme umístili názvy produktů, na osu ROWS - do řádků jsme umístili jednotlivé nákupy a to vše jsme omezili názvem firmy zákazníka, umístěného za klauzuli WHERE. Zajímá nás tedy, jaké produkty si ve svých transakcích kupovala Firma Blauer See Delikatessen.

Druhý způsob dotazu se týká dolování z charakteristik nákupů, v jehož důsledku může být zjištěno, že např. zákazníci z Německa si v lednu v obchodě Speedy Express nejčastěji kupovali cukrovinky. Příkladem takového multidimenzionálního dotazu může být:

```
SELECT NONEMPTY([Time].[Year]) ON COLUMNS,
 NONEMPTYCROSSJOIN(
 [Products].[Products].MEMBERS,
```

```
 [Geography].[City].MEMBERS,
  [Shops].[Name].MEMBERS)
 ON ROWS FROM [Minerdb]
  where [Geography].[Country].&[Spain]
```

V dotazu jsme specifikovali skutečnost, že chceme dolovat z názvů produktů, jmen měst, názvů obchodů, prodávající zboží. To vše jsme omezili za klauzulí WHERE položkou z geografické dimenze a to konkrétně Španělskem. Jako výsledek můžeme dostat, že zákazníci z Madridu kupovali v obchodě Federal Shiping omáčku Louisiana Fiery Hot Pepper sauce.

Zadání dotazu pro uživatele s administrátorskými právy probíhá následujícím způsobem. Uživatel najede do modulu pro dolování a vybere si buď konkrétní hierarchie, ze kterých chce dolovat, nebo jen konkrétní prvky z dané hierarchie, čímž si omezí výsledek dotazu opravdu jen na to, co ho skutečně zajímá. Dále si už vybere typ dotazu tj. již zmiňované dolování nad charakteristikami nákupu nebo analýzu nákupního košíku. Zvolí si podporu a spolehlivost jako vstup do dolovacího algoritmu. Poslední věc, kterou musí provézt, je volba konkrétního algoritmu. Poté již spustí konkrétní dotaz a v prohlížeči je mu prezentován požadovaný výsledek v podobě frekventovaných množin a vydolovaných asociačních pravidel. V případě administrátora je mu nabídnuta i možnost uložení právě provedeného dotazu.

Co se týká implementace modulu, pak mezi nejdůležitější procedury patří:

getDimensionsNames() - vrací jména všech unikátních dimenzí v datové kostce.

parseGetQuery() - na vstup dáme všechny parametry, které jsme postupně vzali z webového formuláře a vytvoříme MDX dotaz, jenž se vrátí ve formě SelectNode.

getOlapDataFromMDX() - spustí přímo MDX dotaz z předpřipravených dotazů, aniž by se díval na obsah vstupních formulářů.

getHierarchiesNames() - vrací hierarchie k dané dimenzi.

getSchema() - získáme konkrétní schéma databáze pomocí metody objektu OlapConnectionManager.

getCubes() - získáme jednotlivé kostky v databázi pomocí metody objektu OlapConnectionManager.

Jak už bylo zmíněno, práce s knihovnou OLAP4J je podobná práci s JDBC. Pro ukázku si uvedeme pár příkladů.

Pro připojení k databázi využíváme url v podobném tvaru jako k JDBC, ale připojujeme se přes knihovnu msmdpump.dll, na konci musíme specifikovat jméno katalogu, k němuž se chceme připojit:

```
url=jdbc:xmla:Server=http://localhost/olap/msmdpump.dll;Catalog=Miner
```

Spojení vytvoříme následujícím způsobem:

```
Connection test = DriverManager
.getConnection("jdbc:xmla:Server=http://localhost/olap/msmdpump.dll;" +
        "Catalog=Minerdb");
```

A dotaz provedeme jako:

```
OlapStatement st = connection.createStatement();
CellSet result = st.executeOlapQuery(sql);
```

Kapitola 9

Testy doby výpočtu aplikace a algoritmů

Testy byly prováděny s pomocí již zmiňovaných tří algoritmů a na dvou typech úloh. Zaprvé na úloze typu analýza nákupního košíku, kdy procházíme všechny transakce a hledáme množiny položek, které se v transakcích nejčastěji opakují, a zadruhé, kdy dolujeme z charakteristik nákupů, tedy odvozujeme fakta, že např. v červnu se v Německu prodávaly z oblečení nejvíce fotbalové dresy německé reprezentace(tj. kvůli mistrovství světa ve fotbale).

Pokud chceme dále pojednávat o testování, rychlosti zpracovávání a rychlosti běhu aplikace, potom je nutné uvézt architekturu, na níž byly testy prováděny:
Počítač Mac Book Pro 2.2 GHz Intel Core 2 Duo s operační pamětí 4 GB 667 MHz DDR2 SDRAM a operačním systémem Mac OS X version 10.5.7.
Dále pak ve WMware Fusion spuštěné Microsoft Windows XP Professional SP3 a Microsoft SQL Server 2005 - verze 9.00.4035.00, Developer Edition, SP3.

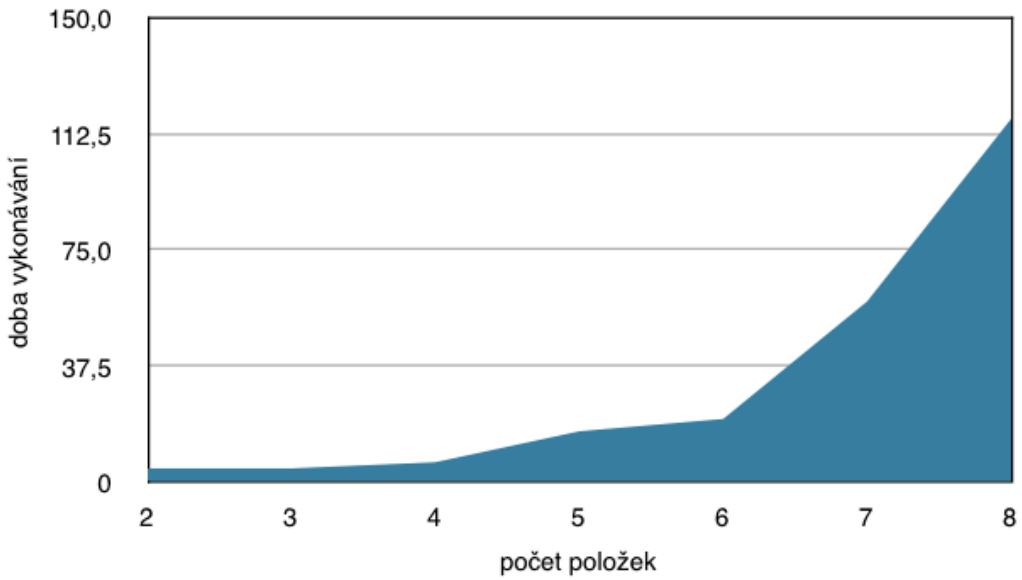
V našem případě jako vstup do analýzy nákupního košíku bereme 811 transakcí a celkově 77 prodávaných položek, které se v transakcích vyskytují. Při použití algoritmu AprioriT a při nastavení podpory na hodnotu 5 a spolehlivosti na hodnotu 5 jsme nedostali žádné asociační pravidlo, ale dostali jsme 11 jednoprvkových frekventovaných množin. Čas provedení operace byl téměř zanedbatelný, algoritmus provedl operaci za zhruba 0,01 vteřiny. Pokud snížíme podporu a spolehlivost, obě na hodnotu 2, opět jsme nedostali žádné asociační pravidlo, ale 57 jednoprvkových frekventovaných množin. Znovu algoritmus proběhl poměrně rychle, kolem 0,01 sekundy. A nakonec, když jsme nastavili podporu a spolehlivost na hodnotu 1, také algoritmus proběhl rychle a my jsme tentokrát dostali 12 asociačních pravidel a 77 jednoprvkových frekventovaných množin. Pokud tento test provedeme s použitím ostatních algoritmů, tak výsledky se nemění a rychlosti algoritmů zůstavají řádově stejné. To platí ovšem v tomto příkladu.

Při odvozování asociačních pravidel z charakteristik nákupů jsme prováděli dotazy, jejichž popis následuje. Při této úloze ovšem v aplikaci z výsledku dotazu zároveň generujeme i asociační pravidla, kdežto v MSSQLSMS jen zobrazíme výsledek dotazu.
Začali jsme od dolování ze dvou položek charakteristik. Nejprve to bylo zkoumání závislosti mezi firmou zákazníka a zemí, ve které zákazník bydlí, tedy na osu *ROWS* jsme umístili *[Customers].[Company Name].MEMBERS* a *[Geography].[Country].MEMBERS*. Na osu

COLUMNS jsme v tomto i v ostatních případech, popisovaných dále, přidali položku roku z časové dimenze (*[Time].[Year]*). Je jasné, že takovýto dotaz nebude mít moc užitek pro vydolování relevantních dat, ale snažil jsem se postupovat od začátku a měřil, kolik času daný dotaz zabere. Nastavení podpory a spolehlivosti bylo vždy na hodnotě 1 a jako použitý algoritmus jsem zvolil AprioriT. V aplikaci trval dotaz 4 vteřiny a v Microsoft SQL Server Management Studiu (MSSQLSMS) trval 1 vteřinu, prohledávali jsme 190 položek.

Pro tři položky v dotazu, kdy jsme přidali položku zákazníkovu pozici ve firmě (*[Customers].[Customer Position].MEMBERS*), to byly opět 4 vteřiny v aplikaci a jedna vteřina v MSSQLSMS, 450 řádků celkově, dvě vygenerovaná asociační pravidla.
Po přidání tentokrát čtvrté položky, jednající se o položku města, ve které zákazník bydlí (*[Geography].[City].MEMBERS*), pak 6 vteřin pro aplikaci(8 vygenerovaných asociačních pravidel) a 1 vteřina pro MSSQLSMS při 1100 řádcích.

Po přidání další položky se jménem kategorie výrobku



Obrázek 9.1: Závislost doby vykonávání aplikace(ve vteřinách) na počtu položek v MDX dotazu

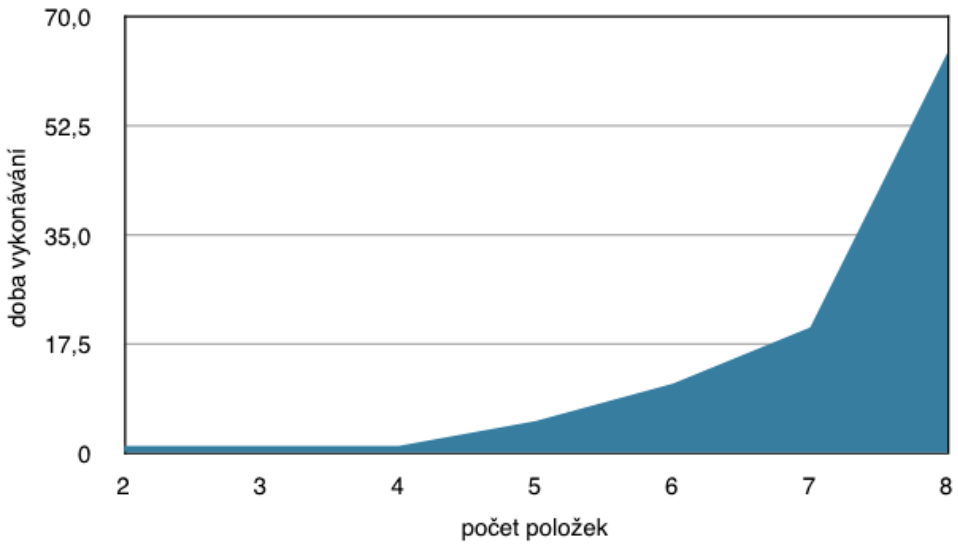
(*[Products].[Category Name].MEMBERS*), už máme položek dohromady pět, běh aplikace trval 16 vteřin a MSSQLSMS 5 vteřin, celkově 8596 řádků a v aplikaci vygenerovaných 30 asociačních pravidel.
Přidáváme další položku, tentokrát se jedná o jméno obchodu, v němž byla transakce uskutečněna (*[Shops].[Name].MEMBERS*). V aplikaci proběhla analýza za 20 vteřin se 128 vydolovanými asociačními pravidly, v MSSQLSMS za 11 vteřin při počtu 25818 transakcí.

Po přidání položky region, ve kterém zákazník bydlí (*[Geography].[Region].MEMBERS*), se v aplikaci dotaz zpracoval za 58 vteřin, běh aprioriT trval 0,03 vteřiny při 176 vygenerovaných asociačních pravidlech a v MSSQLSMS proběhlo zobrazení výsledku za 20 vteřin při počtu 53093 transakcí(řádků).

A při přidání poštovního směrovacího čísla(*[Geography].[Postal Code].MEMBERS*) jsme

v aplikaci nejdříve dostali java.lang.OutOfMemoryError: Java heap space, i když jsme předtím zvýšili velikost paměti pro Apache Tomcat na 1024 MB, při opětovném navýšení na 1524 MB jsme v aplikaci za 1 minutu a 57 vteřin dostali výsledek, algoritmus aprioriT zpracoval za 0,09 vteřin 116373 řádků(transakcí) při výsledku 180 vygenerovaných asociačních pravidel a celkově 141 jedno, dvou a tří prvkových frekventovaných množin. V MSSQLSMS trval výpočet 1 minutu a 4 vteřiny.

A konečně, při zahrnutí devaté položky, kdy se jedná o informace o tom, zda se daný



Obrázek 9.2: Závislost doby vykonávání našeho MDX dotazu (ve vteřinách) v MSSQLSMS na počtu položek

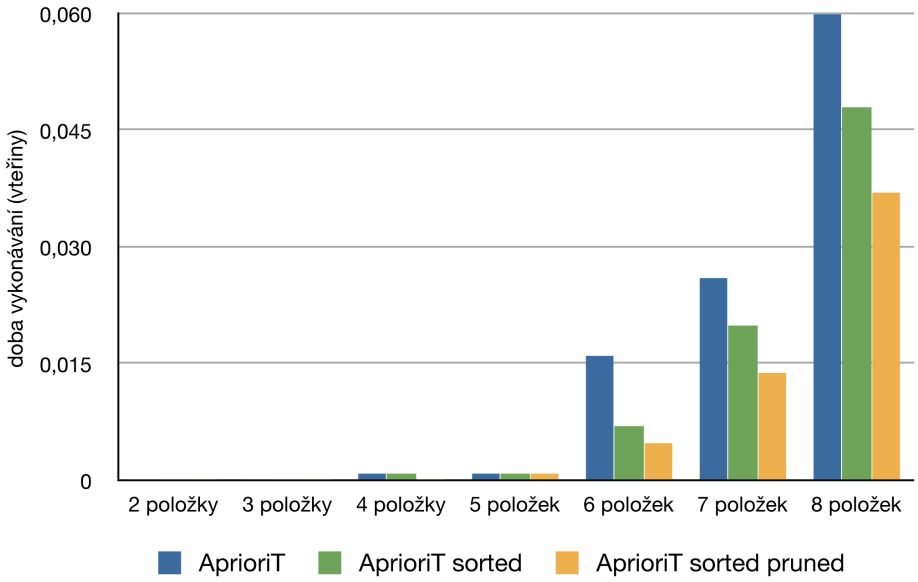
produkt ještě prodává(*[Products].[Discontinued].MEMBERS*) jsme překročili možnosti naší aplikace v testovacím prostředí. Důkazem je to, že se prováděný dotaz v MSSQLSMS zpracovával více než 10 minut a stejně se neprovedl. Otázka je, nakolik by toto ovlivnil výkonnější hardware, na kterém by byl nainstalován databázový server.

Co se týká rychlosti jednotlivých algoritmů, pak naměřené hodnoty jsou uvedeny v grafu na obrázku 9.3. Z něho můžeme vyčíst fakt, že doba vykonávání u všech tří algoritmů je prakticky stejná a tudíž by nemelo velký smysl v aplikaci nasazovat všechny tři algoritmy najednou. Pro zákazníka je při charakteru aplikace, kdy největší dobu stráví čekáním na vrácení výsledků dotazu ze serveru, uspořená setina vteřiny při volbě jiného algoritmu zanedbatelná.

Výsledný MDX dotaz, jenž jsme předchozím postupem dostali, vypadá následovně:

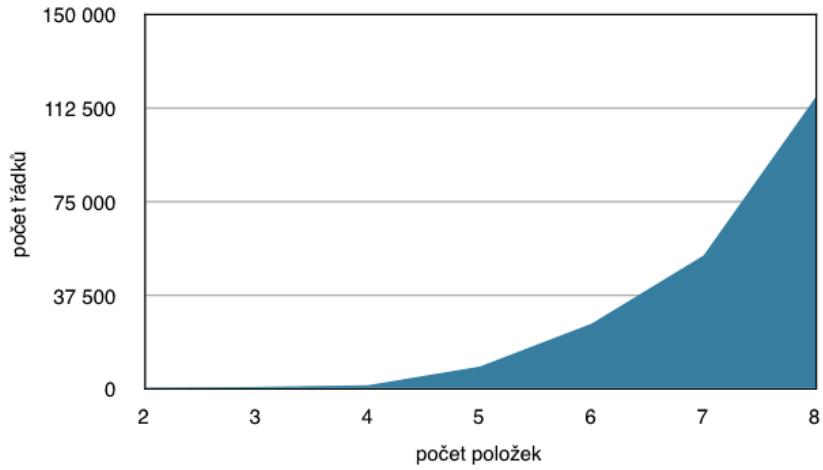
```
SELECT NONEMPTY([Time].[Year]) ON COLUMNS,
NONEMPTYCROSSJOIN([Customers].[Company Name].MEMBERS,
[Customers].[Customer Position].MEMBERS,
[Geography].[City].MEMBERS, [Geography].[Country].MEMBERS,
[Geography].[Postal Code].MEMBERS, [Geography].[Region].MEMBERS,
[Products].[Category Name].MEMBERS, [Shops].[Name].MEMBERS) ON ROWS
FROM [Minerdb]
```

Na závěr je nutné uvést, že všechny prezentované testy byly zkouškou maximálního výkonu aplikace, protože jsme vždy brali všechny položky obsažené v dané hierarchii, tzn. brali jsme vždy všechny státy, vždy všechny města, vždy všechny názvy obchodů. V reálném rostředí by ovšem byla situace jiná, protože obchodníka zajímají převážně charakteristiky obchodů spadajícího do oblasti jeho zájmu, tzn. obchodníka, specializujícího se ve firmě na prodej pečiva bude zajímat pečivo, obchodník specializující se na prodej mořských produktů se bude zajímat o ryby a jiné mořské živočichy.



Obrázek 9.3: Závislost doby vykonávání algoritmů (ve vteřinách) na počtu položek v dotazu

Taktéž pro ně bude mít větší vypovídací hodnotu konkrétní časové období než vybírání celého časového intervalu obsaženého v datovém skladu. Bude se zajímat, které produkty se nejvíce prodávají v posledním měsíci a nebude ho zajímat, které produkty se prodávali před třemi lety. Obchodníci budou mít k dispozici předpřipravené dotazy, jenž si budou moci jednoduše spouštět. Tyto dotazy pravděpodobně již budou připraveny jednotlivému obchodníkovi a manažerovi na míru a již optimalizovány pro efektivní dobu výpočtu tak, aby nemuseli u počítače zbytečně dlouho čekat na výsledky.



Obrázek 9.4: Závislost počtu řádků vygenerovaného naším MDX dotazem v MSSQLSMS na počtu položek v MDX dotazu

Kapitola 10

Závěr

Hlavní náplní diplomové práce bylo seznámit se s problematikou Business Intelligence a začlenit modul pro dolování asociačních pravidel do informačního systému.
Teoretická část práce se zabývá procesem Business Intelligence a návazností moderních technologií na sféru obchodního rozhodování. V dnešní době se jedná o velice aktuální téma, jelikož velké společnosti, působící v širokém spektru odvětví, investují nemalé prostředky do zdokonalování rozhodovacích řešení. Právě správné uplatnění a zavedení BI systému může firmě napomoci k lepším výsledkům. Firma bude moci díky odhalení skrytých vztahů a skutečností z dat, jež má ve svých produkčních systémech k dispozici, lépe oslovit zákazníky. Při správném vyhodnocení všech získaných informací může zlepšit účinnost marketingových kampaní, podpořit prodej svých produktů, minimalizovat rizika nebo například optimalizovat zásobovací činnost. Vše uvedené může mít za následek zvyšování jejího podílu na trhu.

V úvodních kapitolách teoretické části jsou popsány hlavní pojmy, které s pojmem Business Intelligence souvisí. Jsou uvedena témata budování datového skladu, OLAP analýzy a průběhu procesu získávání znalostí z databází. Dále jsou popsány metody získávání asociačních pravidel z databází a jejich možné využití v BI procesu.

Praktická část diplomové práce je věnována tvorbě informačního systému a následnému začlenění modulu pro dolování asociačních pravidel. Jsou zde popsány i technologie využité při tvorbě aplikace. Velká část je věnována MS SQL Serveru 2005, jeho relační databázi a analytickým službám, které informační systém a modul pro dolování využívá. V práci jsou uvedeny i konkrétní problémy, jež při tvorbě aplikace nastaly, například při realizování připojení aplikace pomocí knihovny olap4j k analytickým službám SQL Serveru. Po vyřešení všech úskalí je však modul využitelný pro práci nad většinou OLAP Serverů podporujících standard XMLA.
Poslední kapitola praktické části se věnuje testům aplikace a zhodnocení konkrétních výsledků, dosažených při dolování asociačních pravidel. Uvedeny jsou porovnání doby běhu modulu při zpracovávání dotazů, jenž by mohl uživatel při reálném nasazení do praxe využít.

Co se týká dalšího rozšíření práce a případného nasazení aplikace do praxe, bylo by vhodné se dále zabývat upravením a přidáním další funkcionality do informačního systému, jenž sbírá data, která pak analytické služby SQL Serveru a dolovací modul využívají. Modul by se mohl zapojit například do internetového obchodu a umožnil by tedy zpracovávat data z reálného provozu, jenž by byla zavedena do databáze a následně do datové kostky. Dále by bylo možné otestovat funkčnost modulu i nad jinými typy OLAP serverů.

Literatura

[1] The Eclipse Foundation: EclipseLink [online]. 2009, [cit. 2009-4-20].
URL http://www.eclipse.org/eclipselink/

[2] LUCS-KDD research team [online]. 2009, [cit. 2009-5-12].
URL http://www.csc.liv.ac.uk/~frans/KDD/

[3] Bartík, V.: Datové sklady a technologie OLAP pro získávání znalostí [online]. 2008, [cit. 2008-12-8].
URL https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/ZZN-IT/lectures/02_DWOLAP.pdf

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

[5] Bernard, B.: Novinky SQL Serveru 2005 v oblasti Business Intelligence [online]. 2008, [cit. 2008-12-27].
URL http://www.borber.com/files/4IT435-Novinky-SQL-Serveru-2005.pdf

[6] Brust, A. J.; Forte, S.: *Mistrovství v programování v SQL Serveru 2005*. Brno: Computer Press, 2007, ISBN 978-80-251-1607-4.

[7] Coenen, F.: The LUCS-KDD Apriori-T Association Rule Mining Algorithm [online]. Technická zpráva, The University of Liverpool, UK, Duben 2004, [cit. 2009-5-19].
URL
http://www.csc.liv.ac.uk/~frans/KDD/Software/Apriori-T/aprioriT.html

[8] Fennell, T.; aj.: Stripes [online]. 2009, [cit. 2009-5-12].
URL http://stripesframework.org/display/stripes/Home

[9] Han, J.; Kamber, M.: *Data Mining: Concepts and Techniques*. USA: Elsevier, 2006, ISBN 1-55860-901-6.

[10] Humphries, M.; Hawkins, M. W.; Dy, M. C.: *Data warehousing - návrh a implementace*. Praha: Computer press, 2002, ISBN 80-7226-560-1.

[11] Hyde, J.; aj.: olap4j: Open Java API for Olap [online]. 2009, [cit. 2009-4-15].
URL http://www.olap4j.org

[12] InterDynBMI: Microsoft SQL Server 2005 [online]. 2008, [cit. 2009-1-7].
URL http:
//www.interdynbmi.com/solutions/sql-server-2005.htm#server_reporting

[13] Kotásek, P.: DWH / BI systémy, datové sklady [online]. 2008, [cit. 2009-1-20].
URL https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/ZZN-IT/lectures/P6_DWH_BI.pdf

[14] Lacko, L.: *Datové sklady, analýza OLAP a dolování dat*. Brno: Computer press, 2003, ISBN 80-7226-696-0.

[15] Lacko, L.: Datový sklad v Yukonu podruhé [online]. 2005, [cit. 2009-1-12].
URL http://www.dbsvet.cz/view.php?cisloclanku=2005070601

[16] Lacko, L.: Datový sklad v Yukonu poprvé [online]. 2005, [cit. 2009-2-3].
URL http://www.dbsvet.cz/view.php?cisloclanku=2005070502

[17] Lacko, L.: *Business Intelligence v SQL Serveru 2005*. Brno: Computer press, 2006, ISBN 80-251-1110-5.

[18] Lee, D.; Pawlowski, L.: Report Server Catalog Best Practices [online]. 2006, [cit. 2009-5-2].
URL http://sqlcat.com/technicalnotes/archive/2008/06/26/report-server-catalog-best-practices.aspx

[19] Melomed, E.; Gorbach, I.; Berger, A.; aj.: *Microsoft SQL Server 2005 Analysis Services*. USA: Sams publishing, 2007, ISBN 0-672-32782-1.

[20] Microsoft: MS SQL Server 2005 - Informace o produktu [online]. 2008, [cit. 2008-12-15].
URL http://www.microsoft.com/cze/windowsserversystem/sql/prodinfo/default.mspx

[21] Microsoft: Přehled produktu SQL Server 2005 [online]. 2008, [cit. 2008-12-16].
URL http://www.microsoft.com/cze/windowsserversystem/sql/prodinfo/overview/default.mspx

[22] Microsoft: Configuring HTTP Access to SQL Server 2005 Analysis Services on Microsoft Windows XP [online]. 2009, [cit. 2009-5-7].
URL http://technet.microsoft.com/cs-cz/library/cc917712(en-us).aspx

[23] Microsoft: Northwind and pubs Sample Databases for SQL Server 2000 [online]. 2009, [cit. 2009-4-6].
URL http://www.microsoft.com/downloads/details.aspx?FamilyID=06616212-0356-46A0-8DA2-EEBC53A68034&displaylang=en

[24] Microsoft: OK button unavailable when you add application mappings [online]. 2009, [cit. 2009-2-4].
URL http://support.microsoft.com/?id=317948

[25] Mráz, J.: Podnikové informační systémy [online]. 2007, [cit. 2008-12-18].
URL https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/AIS-IT/lectures/2007/10_ICT-Podniku.pdf

[26] Sack, J.: *Velká kniha T-SQL a SQL Server 2005 - Kompendium znalostí pro začátečníky i profesionály*. Brno: Zoner Press, 2007, ISBN 978-80-86815-57-2.

[27] Zendulka, J.; Bartík, V.; Lukáš, R.; aj.: Získávání znalostí z databází, ZZN, Studijní opora [online]. 2006, [cit. 2009-1-9].
URL https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/ZZN-IT/texts/ZZN.pdf

Seznam použitých zkratek a symbolů

MSSQLSMS – Microsoft SQL Server Management Studio

VS – Visual Studio

BI – Business Intelligence

OLAP – Online Analytical Processing

SOAP – Simple Object Access Protocol

XML – Extensible Markup Language

XMLA(XML/A) - XML for Analysis

SSIS – SQL Server Integration Services

WMI – Windows Management Instrumentation

URL – Uniform Resource Locator

HTTP – Hypertext Transfer Protocol

OLE DB – Object Linking and Embedding, Database

TCP/IP – Transmission Control Protocol/Internet Protocol

ADO MD – ActiveX Data Objects Multi-dimensional

OLAP – Online Analytical Processing

OLTP – Online Transaction Processing

DAO – Data Access Object

ORM – Object-relational mapping

JPA – Java Persistence API

JSP – JavaServer Pages

UDM – Unified dimensional model