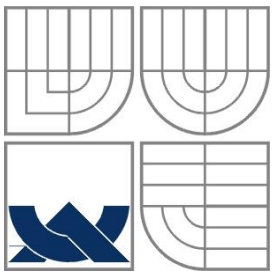
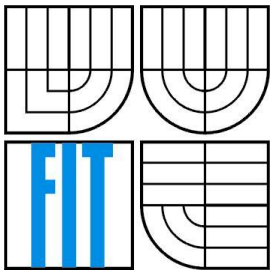


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

ZÍSKÁVÁNÍ ZNALOSTÍ Z DATOVÝCH SKLADŮ

KNOWLEDGE DISCOVERY OVER DATA WAREHOUSES

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. Ondřej Pumprla
AUTHOR

VEDOUCÍ PRÁCE — Ing. Lukáš Stryka
SUPERVISOR

BRNO 2009

Abstrakt

Diplomová práce se zabývá principy procesu získávání znalostí se zaměřením na asociační pravidla. Je vybudován teoretický aparát obecného popisu a principů tvorby datových skladů. Na základě těchto teoretických poznatků je implementována aplikace pro získávání asociačních pravidel. Aplikace očekává data buď v transakční nebo multidimenzionální podobě ve schématu hvězdy. Implementované algoritmy na hledání frekventovaných množin jsou Apriori a FP-strom. Systém umožňuje variantní nastavení parametrů dolování a byly provedeny ověřovací výkonnostní testy. Z pohledu podpory hledání asociačních pravidel se výsledná aplikace jeví robustnější než existující porovnávané systémy SAS a Oracle Data Miner.

Klíčová slova

Získávání znalostí, Dolování dat, Datový sklad, OLAP, Multidimenzionální model, Datové schéma hvězdy, ETL proces, Oracle Warehouse Builder 11*g*, Frekventované množiny, Asociační pravidla, Apriori, FP-strom

Abstract

This Master's thesis deals with the principles of the data mining process, especially with the mining of association rules. The theoretical apparatus of general description and principles of the data warehouse creation is set. On the basis of this theoretical knowledge, the application for the association rules mining is implemented. The application requires the data in the transactional form or the multidimensional data organized in the Star schema. The implemented algorithms for finding of the frequent patterns are Apriori and FP-tree. The system allows the variant setting of parameters for mining process. Also, the validation tests and efficiency proofs were accomplished. From the point of view of the association rules searching support, the resultant application is more applicable and robust than the existing compared systems SAS Miner and Oracle Data Miner.

Keywords

Knowledge discovery, Data mining, Data Warehouse, OLAP, Multidimensional model, Star schema, ETL, Oracle Warehouse Builder 11*g*, Frequent Patterns, Association rules, Apriori, FP-tree

Citace

Pumprla Ondřej: Získávání znalostí z datových skladů, diplomová práce, Brno, FIT VUT v Brně, 2009.

Získávání znalostí z datových skladů

Prohlášení

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

……
Ondřej Pumprla
19.5.2009

Poděkování

Děkuji svému vedoucímu Ing. Lukáši Strykovi za cenné rady a poskytnutí studijních podkladů pro tvorbu práce. Děkuji i svým rodičům za psychickou podporu při realizace práce.

Obsah

1 Úvod

Sběr a shromažďování dat lidé provádí již odedávna. Zvláště pak v posledních letech, tedy v době stále masivnějšího použití výpočetní techniky na evidenci obchodních a výzkumných procesů. Velkou měrou kumulativnosti dat přispívá téměř všudypřítomná internetová síť. Dochází k robustnímu a automatickému sběru dat z kamerových a obecně obrazově-snímatelných přístrojů (snímače čárových kódů). Nashromáždění velkého množství dat vynutilo rozvoje principů, které by napomohly, a to z velké části automaticky, přetransformovat tato data na relevantní znalosti.

Proces transformace dat na sémanticky ohodnocené informace není jednoduchý a přímočarý proces, ale skládá se z několika podprocesů, které je často nutné inkrementálně opakovat. Společnosti mají často data v heterogenních systémech a občas spolupracují s jinými orgány, ze kterých čerpají data taktéž. Pro proces získávání znalostí je nutné a efektivní mít tato data uložena v jednom komplexnějším úložišti. Ke sjednocování dat z různých zdrojů dochází k nekonzistentnosti, je nutné provádět dodatečné operace nad těmito daty a některá data jsou pro rozhodování irelevantní. Tyto problémy z velké části řeší datový sklad, jakožto integrované úložiště a komplexní nástroj na variantní transformační funkce. Jakmile jsou data v datovém skladu uložena, dochází k samotnému dolování dat.

Mezi nejvíce aplikovatelné principy dolování dat patří získávání asociačních pravidel. Účelem získávání asociačních pravidel je hledání často se společně vyskytujících položek. Hledání těchto frekventovaných vzorů může být účelné například pro obchodní řetězec, jenž má data o prodeji a zákaznících v heterogenních zdrojích a potřebuje zjistit, která zboží jsou společně často prodávána, v jakém období a kdo daná zboží často kupuje. Ze získaných znalostí pak může obchodní datový analytik provést návrh reorganizace regálů se zbožím a zasílat zákazníkům cílené letáky s nabízeným zbožím.

Cílem diplomové práce je navrhnout a implementovat vlastní systém pro získávání asociačních pravidel. Jako datovou podporu, možnost užití dat z heterogenních zdrojů a schopnost nad těmito daty provádět transformace bude implementovaný systém využívat datový sklad Oracle Warehouse Builder 11*g*. Aplikace, jež bude v rámci diplomové práce vyvíjena, si taktéž klade za cíl disponovat vlastnostmi, které nejsou součástí existujících systémů SAS a Oracle Data Miner. Vytvořený systém by měl být prakticky a intuitivně použitelný pro reálné nasazení.

1.1 Členění práce

Celá práce je členěná do kapitol s názvem podle tématického okruhu, které popisují. Po úvodní kapitole práce začíná seznámením s procesem získávání znalostí z databází, se základními principy a jejich využitím. Následující 3. kapitola uvádí obecnou koncepci a návrh datových skladů. Je taktéž představen datový sklad Oracle Warehouse Builder 11*g*. 4. kapitola pojednává o popisu a algoritmizaci jednodimenzionálních a vícedimenzionálních asociačních pravidel. Následující kapitoly pojednávají o praktické problematice diplomové práce. V 5. kapitole je vyvíjená aplikace popsána a je proveden architektonický a balíčkový/třídní návrh. Následující 6. kapitola obsahuje implementační a funkční popis aplikace. V rámci předposlední 7. kapitoly dochází k variantnímu testování aplikace a je provedeno srovnání se systémy SAS a Oracle Data Miner. Závěrečná kapitola zhodnocuje dosažené výsledky a jsou uvedeny návrhy na potenciální rozšíření systému.

1.2 Návaznost na Semestrální projekt

Diplomová práce navazuje na Semestrální projekt, který byl realizován v zimním semestru. V rámci Semestrálního projektu byla zpracována teorie, jenž tvoří podpůrný aparát pro implementaci systému, a bylo seznámeno s datovým skladem Oracle Warehouse Builder 11*g*. *Kapitoly 2* až *4* a *příloha A* byly po úpravě převzaty ze Semestrálního projektu a použity v této Diplomové práci.

2 Získávání znalostí z databází

Tato kapitola pojednává o vzniku, procesu a využití získávání znalostí z databází. Jsou zde vysvětleny obecné pojmy a nejčastěji používané techniky v oboru dolování z dat.

2.1 Úvod do získávání znalostí

Velké množství dat, které by bylo pozitivně účelné převést na sémantickou relevanci, vynutilo tvorbu a rozvoj principů a mechanizmů, jenž jsou komplexně nazývány **získávání znalostí z databází** (někdy též **dolování z dat**, datová archeologie, ..). Počátky tvorby pojmových základů a samotného využití a nasazení do praxe se datuje do konce 80. let [1].

Esenciální vlastností získávání znalostí je extrakce netriviálních vzorů. Za triviální dotazy se například považuje dotazování pomocí SQL jazyka (dle normy SQL-99 bez dalších rozšíření). Získané znalosti jsou předem neznámé a nesou potenciální sémantickou relevanci. Dochází tedy k extrakci skrytých a užitečných vzorů. Tento proces by měl být však z velké části automatizovaný, jelikož je zpravidla realizován na velkém množství dat.

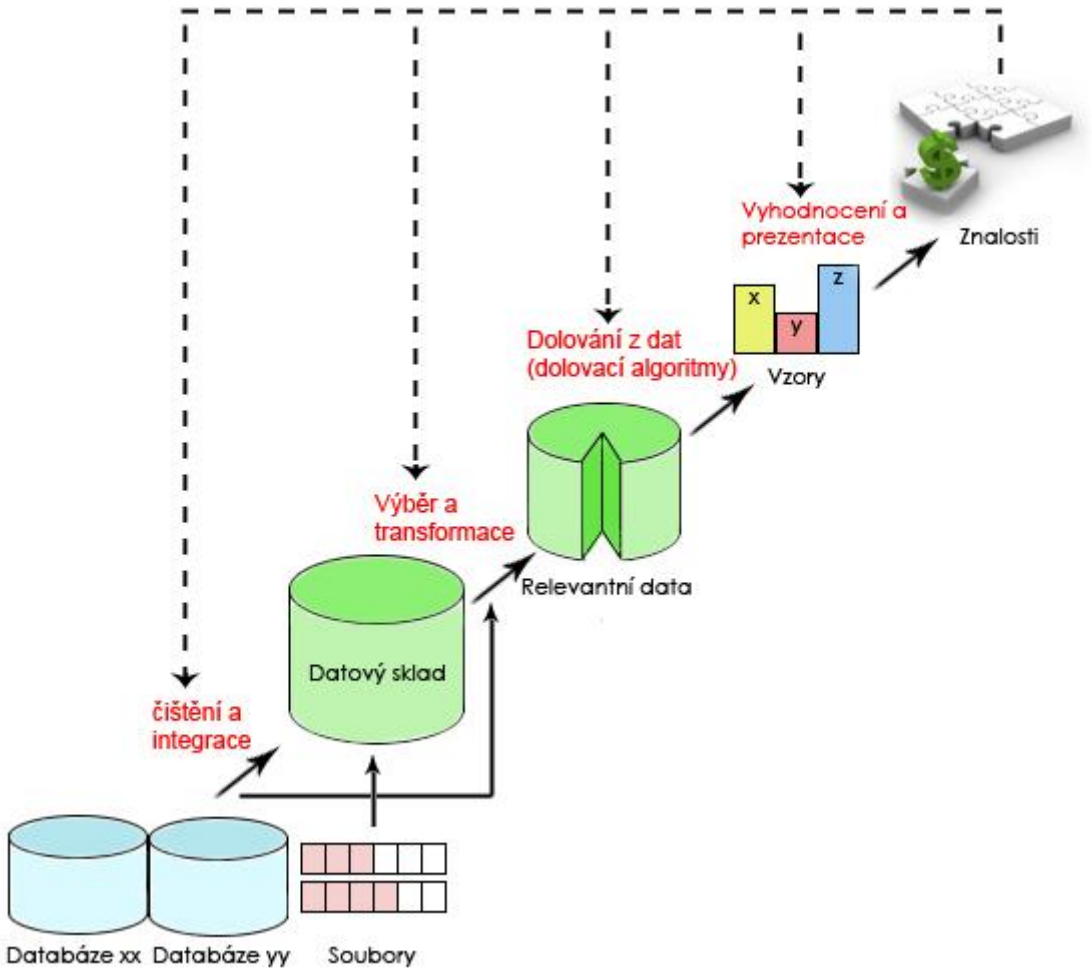
K analýze požadovaného problému cílový uživatel či organizace, pro kterou jsou potenciální výsledky určeny, používá zpravidla více zdrojů dat. Došlo tedy k vynucení tvorby komplexnějších nástrojů, které by umožnily efektivní správu dat a nad těmito daty provádět vhodné transformace. Tímto nástrojem se stal ve většině případů datový sklad (viz. kapitola *3. Datové sklady*).

Získávání znalostí v obecném pojetí zahrnuje a využívá plno vědních oborů, a to mimo jiné databázové technologie, pravděpodobnostní a statistickou matematiku, umělou inteligenci (evoluční algoritmy, neuronové sítě, shlukovou analýzu, ..), vyhledávaní informací, doménově specifickou problematiku (finanční a pojistná ekonomie, genové inženýrství, ..).

Mezi prakticky využívané obory patří například analýza nákupního košíku – které zboží se nakupuje často spolu; souslednost navštívených webových stránek; analýza emailu (spam); vhodnost poskytnutí úvěru – na základě poskytnutých úvěrů v minulosti; dolování v biologických datech – vazby mezi chorobami a geny; a mnohé další.

2.2 Proces získávání znalostí

Proces získávání a reprezentace znalostí z dat zahrnuje několik etap. Tyto etapy probíhají zpravidla v konkrétním uspořádání, přičemž se lze následně vracet do některé z etap předchozích (jsou k dispozici nová zdrojová data, nespokojenost s výsledky – špatně zvolená dolovací metoda a nastavení metrik). Pořadí etap není však předem dané a je určené typem dolovací úlohy či dostupnými prostředky (absence datového skladu). Jedna z typických forem procesu získávání znalostí je znázorněna na *obrázku 1.1.*



Obrázek 1.1: Etapy procesu získávání znalostí [2]

Vstupním zdrojem dat bývají zpravidla databázové entity nacházející se obecně na heterogenních databázových systémech (Oracle, MS SQL, DB2, ..), včetně datových skladů. Další častou alternativou bývají textové soubory, ve kterých jsou data formátována většinou jednoduchým souborovým formátem (csv, tsv, ..). Při tvorbě datového skladu v Oracle Warehouse Builder 11g lze za zdroj vstupních dat považovat i propojení s aplikační doménou (SAP, Siebel, ..) [3].

Jelikož jsou data získaná z různých zdrojů, je nutné pro efektivnější správu a aplikaci dolovacích algoritmů provést **integraci**. Tedy spojení dat z různých zdrojů do jediného datového úložiště. Častým jevem taktéž bývá, že jsou data zašuměná a je nutné tento šum odstranit **čištěním**. Šumem je například absence některé hodnoty domény (uživatel nevyplnil hodnotu), nebo vychýlení z reálné množiny hodnot dané domény (teplota pacienta je 380°C).

Integrovaná a vyčištěná data jsou postoupená do dalšího kroku, nebo dochází k uložení dat do **datového skladu**. Datový sklad ulehčuje proces rozhodování pomocí nástrojů pro OLAP analýzu, optimalizací pro rychlejší čtení informací z datového skladu (oproti relační databázi), vhodnějšímu internímu uložení (hvězda, sněhová vločka, souhvězdí) [4]. Detailnější popis datových skladů je v kapitole *3 Datové sklady*.

Další etapou je **výběr** a **transformace**. Obecně se v datovém skladu nachází velké množství dat, přičemž ne všechna data jsou pro danou úlohu relevantní. Taktéž dolovací algoritmy jsou

výpočetně relativně náročné a je tedy vhodné data redukovat. Vybírají se tedy takové atributy, které jsou pro danou úlohu užitečné. Některé algoritmy zase vyžadují data ve speciálním rozsahu hodnot. Například neuronová síť je určená pro data, jež nabývají hodnot z intervalu <0.0; 1.0>. Je tedy nutné provádět nad daty normalizaci, obecně transformaci. Transformace je *n*-ární operace nad daty.

Nyní jsou již data připravena pro samotné **dolování.** Neexistuje obecný dolovací algoritmus, který by se dal aplikovat na libovolný problém a je tedy nutné zvážit, který bude použit. Někdy je vhodné aplikovat více algoritmů a výsledky porovnat. Nemusí být totiž předem zřejmé, který bude pro daný problém ideální. Výsledkem této etapy jsou **vzory**, což jsou znalostní data, která nesou potenciálně relevantní informaci. Přehledovým popisem různých principů dolování se zabývá kapitola *2.3 Typy dolovacích úloh*.

Finální fáze je **vyhodnocení** a **prezentace.** Výstup této fáze je často prezentována koncovému uživateli (obchodní analytik, manažér) a je tedy nutné prezentovat výsledky ve vhodné formě. Pokud cílová skupina není s vydolovanými výsledky spokojena, probíhá návrat do některé z předešlých etap a proces získávání znalostí je částečně opakován.

Jak již bylo zmíněno, popsaný proces nemusí být vždy realizován v daném pořadí, nebo se mohou některé etapy lišit. Při tvorbě datového skladu v Oracle Warehouse Builder 11*g* dochází k výše popsanému čištění, výběru a transformaci v rámci etapy zvané jednotně **transformace.** Dochází k mapování transformačních funkcí (agregační funkce, dimenzionální funkce, ..) na zdrojová data a výsledky těchto transformací se ukládají do datového skladu. Celkový proces je pak označován jako **ETL** – Extraction (načtení dat ze zdrojových systémů), Transformation (transformace do podoby vhodné pro analýzu), Loading (uložení do databáze). [3]

2.3 Typy dolovacích úloh

V předchozí kapitole *2.2 Proces získávání znalostí* byla zmíněna etapa **dolování z dat**. Jedná se o samotné algoritmické hledání zajímavých vzorů a je stěžejní etapou procesu získávání znalostí z databází. V rámci této kapitoly budou popsány hlavní typy dolovacích úloh, které se v problematice získávání znalostí nejčastěji vyskytují. Bude se jednat o informativní, přehledový popis se zaměřením na účel použití, jelikož je diplomová práce zaměřena na získávání znalostí pomocí **asociačních pravidel**. Asociační pravidla budou podrobněji popsána v kapitole *4 Algoritmizace asociačních pravidel*, v této kapitole dojde pouze k abstraktnímu popisu – z důvodu systematické logiky.

2.3.1 Asociační pravidla

Hledání asociačních pravidel patří mezi nejčastěji aplikované dolovací mechanismy. Principem asociačních pravidel je v první fázi hledání **frekventovaných vzorů**, tedy položek, které se často vyskytují společně. Tyto položky jsou principiálně často uloženy (simulovány) ve **transakční databázi**. Ve druhé fázi se z těchto frekventovaných vzorů vytváří samotná **asociační pravidla**. Principem asociačních pravidel je hledání častého a současného výskytu jedné množiny položek s jinou množinou položek.

V nejzákladnějším případě dochází k dolování **jednodimenzionálních, jednoúrovňových booleovských pravidel**, kdy dochází k analýze pouze jedné nehierarchické entity, přičemž atribut nabývá pouze dvou hodnot (dvouhodnotová doména). Pokud entity tvoří hierarchii, dochází k dolování **víceúrovňových asociačních pravidel**. V případě, kdy dochází k participování více entit, jedná se o **vícedimenzionální asociační pravidla**.

Mezi hlavní aplikace hledání asociačních pravidel patří analýza nákupního košíku. Tento proces zkoumá zvyky nakupujících a určuje, které položky se nakupují současně, případně zjišťuje při koupi konkrétního zboží se kupuje současně jiné zboží. Z těchto získaných informacích může obchodník čerpat podklady pro rozmístění regálů v obchodě, případně při tvorbě katalogů a letáků. Dalším praktickým přínosem asociačních pravidel je analýza návštěvnosti webových stránek, kdy majitele webových stránek zajímá, které jeho stránky lidé často současně navštěvují, případně v jakém pořadí [5]. Jiné použití je v multimediálních databázích při popisu textury [6].

2.3.2 Klasifikace a predikce

Klasifikace a **predikce** jsou dva principy analýzy dat, které slouží pro zařazení dle charakteristiky dat do klasifikačních skupin a pro předvídání hodnot [1]. Jak již bylo zmíněno v kapitole *2.2 Proces získávání znalostí*, je nutné zdrojová data předzpracovat. Například klasifikační metoda založená na neuronové síti vyžaduje normalizaci vstupních dat na doménu o intervalu <0.0; 1.0>.

Klasifikace provádí analýzu dat, na základě které zařazuje objekt do jisté skupiny. Obecně probíhá klasifikace ve třech etapách. V první etapě dochází k učení klasifikátoru (učení probíhá na známých datech), který bude sloužit ke klasifikaci dat. Druhá etapa je zjištění kvality naučeného klasifikátoru a eventuální navrácení do první etapy, aby se klasifikátor mohl přeučit na vhodnější klasifikační parametry. Poslední etapa je již samotná aplikace klasifikátoru na předem neklasifikovaných datech. Mezi nejčastěji používané klasifikační algoritmy patří aplikace **rozhodovacího stromu**, **Bayesovská klasifikace**, **Bayesovské sítě**, **klasifikace založená na pravidlech**, **klasifikace založená na neuronových sítích** (**SVM metoda**), **klasifikace využívající asociační pravidla, fuzzy množiny** a jiné [1].

Predikce je oproti klasifikaci využívána k určení předem neznámé *obecně spojité (a uspořádané)* hodnoty atributu. Mezi hlavní prediktivní metody patří **lineární**, **nelineární** a **vícenásobná regrese** a **regresní stromy** [1].

Aplikací klasifikace a predikce je možnost předvídání vhodnosti poskytnutí úvěru klientovi na základě účelnosti poskytnutí úvěrů v minulosti. Dalším využitím je klasifikace osob, které nakupují podobné produkty – na základě této informace lze cíleně informovat charakteristické osoby produktově specifickými letáky, emaily, apod.

2.3.3 Shluková analýza

Shlukování je členění skupiny objektů do tříd tak, že objekty, které jsou ve stejné třídě, jsou ohodnoceny jako maximálně podobné, kdežto objekty ze tříd různých jsou ohodnoceny jako minimálně podobné. Při klasifikaci taktéž dochází k zařazení objektu do třídy, ve shlukování však není předem známa doména tříd, nebo nejsou známy trénovací množiny objektů. Shlukování je tedy z pohledu strojového učení charakteru učení bez učitele.

Shlukovacích algoritmů je velké množství a podle společných charakteristik se člení na následující skupiny: **metody založené na rozdělování** (metody **k-means**, **k-medoids**), **hierarchické metody**, **metody založené na hustotě**, **metody založené na mřížce**, **metody založené na modelech**, **metody pro shlukování visoce-dimensionálních dat**. [2]

Shluková analýza se může použít ve spojení s jinými typy metod, a to například při kategorizaci kvantitativních atributů ve vícedimenzionálních asociačních algoritmech, nebo pro předzpracování dat při klasifikaci a predikci. Dalším využití shlukové analýzy je na poli ekonomického trhu, kdy dochází k rozdělování zákazníků dle podobnostních charakteristik. Shluková analýza má taktéž velké využití v biologii, například při tvorbě evolučních stromů, nebo při analýze dat mikročipů pro hledání genů, které se chovají podobným způsobem [7].

2.3.4 Některé další dolovací mechanismy

Předchozí kapitoly zdaleka nepokrývají všechny principiální mechanismy dolování z dat. Dolování z dat je stále se vyvíjející disciplína a nedochází pouze k zefektivnění již existujících algoritmů, ale dochází k aplikaci dolování z dat do různých oborů.

Jeden z těchto oborů je **dolování z textu** a **na webu**. Předchozí metody uvažovaly strukturovaná data, což u dolování z textu a na webu nemusí být obecně splněno. Dochází k získávání dat z čistě textových souborů, emailů, vědeckých článků, webových stránek a jiných zdrojů. Data sice mohou mít jistou hierarchii, ale tato není předem jednoznačně známa a její sémantická hodnota bývá zpravidla na relativně nízké úrovni.

Na počátku druhého tisíciletí dochází k podmětům pro výzkum **získávání znalostí z proudu dat**. Jde o data vznikající souvisle v nějakém zdroji, která je třeba spravovat, ale také často analyzovat, a to zpravidla online [2]. Pro tato data je charakteristické, že jsou časově uspořádána, mají rychle a variabilně se měnící datový tok, a jejich množství je potenciálně nekonečné [1]. Příkladem využití získávání znalostí z proudu dat jsou informace o provozu z počítačové sítě, analýza telefonních hovorů, zpracování online transakcí ve finanční sféře a další [2].

Dalšími v poslední době se rozvíjejícími obory jsou: **dolování v grafových strukturách**, **analýza sociálních síti**, **dolování v objektových**, **prostorových** a **multimediálních datech**, a jiné.

3 Datové sklady

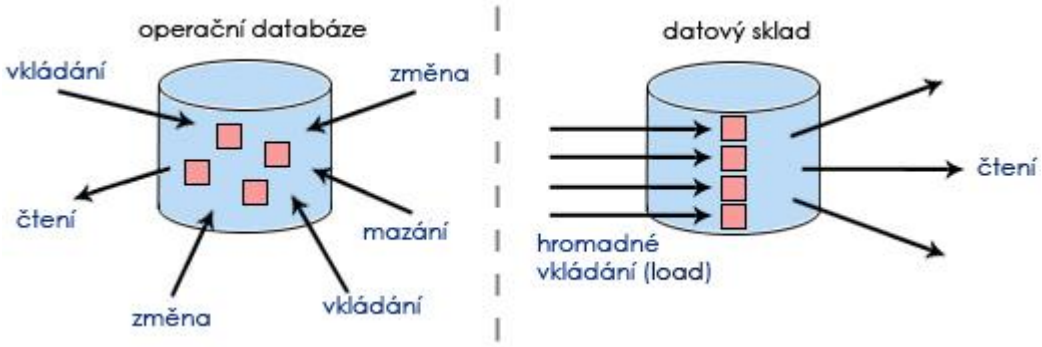
Tato kapitola je teoretickým východiskem k aplikování poznatků při tvorbě datového skladu a bude představena technologie Oracle Warehouse Builder 11*g*. Jsou zde popisovány obecné pojmy, které jsou typické pro datové sklady a bude zmíněn multidimenzionální model, jenž je důležitý při tvorbě návrhu datové struktury v datovém skladu. Dále jsou popsány OLAP operace, které slouží k efektivnější analýze dat. Na závěr této kapitoly dojde k seznámení a popisu architektury datového skladu Oracle Warehouse Builder 11*g*.

3.1 Co jsou datové sklady

Moderní trendy efektivní analýzy dat a získávání znalostí v podnikových (i v menších) sférách s sebou přinesly nový koncept k návrhu, tvorbě a přístupu k podnikovým (i nepodnikovým) datům. Firma má obecně data uložená v různých zdrojích – různorodých databázích, textových souborech s jednoduchým formátováním, XML souborech, dolovacích aplikacích, a tedy v heterogenních zdrojích. Byl vystaven problém přistupovat k těmto datům jednotně a efektivně. Dalším požadavkem bylo propojení vestavěných analytických nástrojů, které budou optimalizovány na podniková data. Pro rychlejší analýzu bylo taktéž vhodné data ukládat ve speciální metadatové formě, aby tato byla pro analytické a dolovací algoritmy optimální. Pro splnění všech těchto požadavků bylo navrhnuto speciální úložiště dat, a to **datový sklad**.

Mezi zakladatele a průkopníky datového skladu se řadí **Bill Inmon** a **Ralph Kimball** [4]. Od Billa Inmona pochází pravděpodobně nejznámější definice datového skladu: „*Datový sklad je subjektově orientovaná, integrovaná, stálá a časově variantní kolekce dat pro podporu strategického rozhodování. Data datového skladu obsahují členěná korporační data.*“ [8]. Následující klíčové subjekty jsou stěžejní pro datové sklady [8]:

- **Subjektová orientace:** Datový sklad je orientován na hlavní subjekty společnosti, zatímco klasické operační databázové systémy jsou zaměřeny na aplikační stránku firmy. Nejsou zde uvažována data, která nelze použít pro rozhodování. Mezi hlavní subjekty patří: zákazník, produkt, prodej, transakce, aktivita, účet.
- **Integrovanost:** Dochází k získávání dat z různých zdrojů a je nutné tato data uchovat v homogenním fyzickém úložišti. Data mohou být z heterogenních databázových systémů, jiných datových skladů (není moc vhodná praktika), textových souborů a jiných datových zdrojů.
- **Stálost:** Při integraci dat z heterogenních datových zdrojů dochází k ukládání těchto dat do datového skladu (*load*). Stěžejní operací, která se nad daty v datových skladech realizuje, je čtení. Dodatečně též ve větších časových intervalech dochází k integrování nových dat k datům stávajícím. Data v datových skladech zůstávají nadále uchována, a to předně z důvodu analýzy. Pro analýzu je nutné mít historická data. To je zásadní rozdíl oproti operační databázi, kde dochází často k aktualizaci dat novějšími údaji. Realizace hlavních databázových operací je schématicky znázorněna na *obrázku 3.1*.



Obrázek 3.1: Prezentace stálosti dat v datovém skladu [8]

- **Časová variantnost:** Jak již bylo řečeno v předchozím bodě, pro analýzu je důležité mít data historická. Tedy pokud nějaký atribut v reálném světě nabývá nové hodnoty, je stará hodnota (ve většině případů) v datovém skladu taktéž uchována. Z tohoto důvodu dochází k častému využití časových razítek.

Primární využití datových skladů je tedy integrace heterogenních zdrojů do zdroje jediného. Typický přístup (bez využití datového skladu) pro získávání dat z více zdrojů je odstínění uživatele od této heterogennosti díky adaptivní vrstvě. Tato adaptivní vrstva transformuje uživatelův dotaz na skupinu dotazů, které jsou specifické pro různé databázové zdroje, které budou v dotazu využity. Odpovědi na jednotné dotazy od různých databázových zdrojů jsou opět transparentně sloučeny adaptivní vrstvou a výsledek je předán dotazujícímu se uživateli. Tento kompletní proces je značně pomalejší než využití datového skladu, kde dochází ke kopií dat z heterogenních zdrojů, předzpracování (transformací), anotací, integrací, výpočtu agregačních funkcí a restrukturalizace do jediného sémantického úložiště. [1]

Informace z datových skladů pak mohou být využity k různým praktickým účelům. Nejčastěji se využívají pro analýzu chování zákazníka, vzájemných vztahů zákazníků, tvorba ad-hoc reportů a analýzu prodejnosti výrobků v závislosti na čase a místě prodeje. [2]

3.2 Porovnání datových skladů a operačních databází

Jak již plyne z předchozí kapitoly, primární účel **datových skladů** a **operačních databází** je rozdílný. Zatímco operační databáze jsou určeny pro vykonávání transakcí a online (obecně i modifikačních) dotazů (příkazů), datové sklady jsou určeny pro analytické a rozhodovací potřeby. Systémy založené na datových skladech se nazývají **OLAP** (*On-Line Analytical Processing*), systémy založené na operační databázi se pak nazývají **OLTP** (*On-Line Transaction Processing*).

Hlavní rozdíly mezi operačně-databázovými systémy a datovými sklady jsou následující [1]:

- **Návrh databáze**: Operační databázové systémy používají databázovou normalizaci (zaručení integrity dat na základě normálních forem - *Codd*) a entitně-relační model, tyto jsou pak aplikačně orientovány. Datové sklady jsou zpravidla založeny na multidimenzionálním datovém modelu (viz kapitola *3.3 Multidimenzionální datový model*) a jsou orientovány subjektově.

- **Datový obsah:** Operační databázové systémy zpravidla udržují pouze aktuální data. Zatímco datové sklady pro potřebu analýzy udržují taktéž data historická. Dále je vhodné uchovávat různě agregovaná a granulovaná data pro rychlou časovou odezvu při variantních analytických dotazech.
- **Přístup k datům:** Operační databáze jsou orientovány aplikačně a jsou tedy pro ně optimalizovány vkládací a obecně modifikační dotazy. V těchto systémech je velmi důležitá konzistence dat před a po provedení komplexnějších dotazů. Z tohoto důvodu jsou tyto komplexní dotazy prováděny jako transakční dotazy, což s sebou nese nutnou režii, která je pro dotazování nad datovými sklady nevhodná. U datových skladů převládají čtecí dotazy, které jsou však často výpočetně velmi náročné.
- **Uživatelé a orientace systému:** Operační databáze jsou převážně orientovány na zákazníky. Dochází k zpracování dotazů a transakcí, které často souvisí se zákazníkem jako jedincem. Systém je využíván řádově tisíci uživateli. V kontrastu jsou datové sklady využívány datovými analytiky, manažéry a zpravidla slouží ke zkvalitnění praktik firmy. Datové sklady jsou využívány jednotkami až stovkami uživateli.
- **Velikost databáze:** Velikost operační databáze se často pohybuje od 10MB do jednotek GB. Velikost datového skladu po kompletním prvním inkrementu pak bývá řádově od desítek GB do desítek TB. Například komplexně zpracovaný datový sklad bývalé firmy Eurotel na území České republiky měl v roce 2004 velikost 3.3 TB [9].

Z výše zmíněných důvodů je nutné uchovávat data pro různé účely v různém datovém úložišti (aplikačně orientovaná data uchovávat v operačních databázích, analytická data v datových skladech), jinak by docházelo k degradování výkonu. Dochází též k rozdílnému optimalizačnímu kroku indexace a hashování. Je nutné si však uvědomit, že předchozí popis rozdílů se mezi operačně databázovými systémy a datovými sklady v některých případech (a to zejména v poslední době) stírá. Objevují se již operačně databázové systémy, které disponují netriviálními nástroji pro analytické rozhodování.

3.3 Multidimenzionální datový model

Zatímco pro operační databázové systémy je typický entitně-relační model databáze, pro datové sklady se (většinou) používá **multidimenzionální datové modelování**. Tento typ modelování zobrazuje data ve formě *n*-rozměrné **datové kostky**. Datová kostka se skládá z **tabulek faktů** a **tabulek dimenzí**. Rozměr této kostky není shora nikterak omezen, zdola je omezen hodnotou 0 (součet všech položek).

Tabulka faktů je primární tabulkou v dimenzionálním modelu, kde jsou uchovány měrné jednotky o sledované veličině, takzvané **metriky**, a odkazy na záznamy v dimenzionálních tabulkách (faktová tabulka je v podstatě spojení many-to-many mezi dimenzionálními tabulkami). Nejčastěji se vyskytující se atributy jsou numerického, rychle se měnícího charakteru (nad kterými je definována aditivní relace). V obecném pojetí tabulky faktů obsahují více záznamů než tabulky dimenzí (tabulka dimenzí reprezentuje méně než 10% datového úložiště). [10]

Tabulka dimenzí představuje kontext, ve kterém jsou fakta prezentována a zkoumána. Obsahuje obecně pomalu se měnící a kvalitativní data (diskrétní a kategorická, zpravidla textová). Dimenzní tabulky jsou obecně tvořeny četnější relací než relace faktové tabulky (obsahují více sloupců, není neobvyklý počet 50 až 100). Dimenzní tabulky jsou typické menším počtem záznamů než tabulky faktů a vždy obsahují primární klíč (který je referencován tabulkou faktů). Hodnoty (domény) sloupců se volí čitelně. Pokud by bylo například potřeba ukládat v dimenzní tabulce *času* hodnotu, zda-li byl výrobek prodán v době svátků, či v pracovní den, nedochází k ukládání hodnot *A/N*, ale raději *svátek/pracovní_den*. Dimenzní tabulky jsou z principu na konkrétním datovém skladu nezávislé – existovaly by, i kdyby nebyl daný datový sklad implementován (jsou externí) a tvoří hierarchickou strukturu. [10]

Například kdyby prodejce v obchodním průmyslu chtěl provádět analýzy nad prodaným zbožím, pak by datový sklad (který by byl zdrojem analýz) jako dimenzní hodnoty mohl uchovávat *datum prodeje*; *charakteristiku produktu*; *místo obchodu*, kde došlo k prodeji; a jako faktové údaje by mohly být ukládány *ceny* těchto produktů a *množství* prodaných jednotlivých produktů.

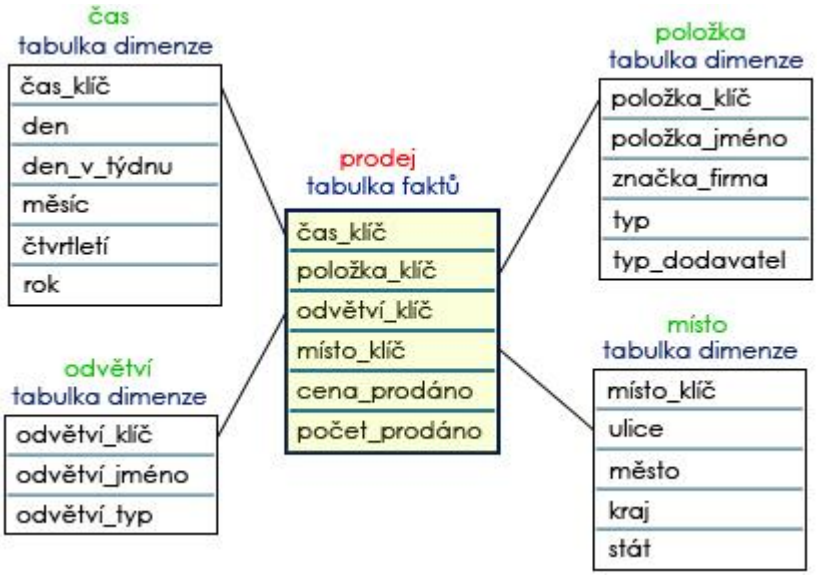
Je nutné podotknout, že **Inmon**, což je jeden ze zakladatelů pojmu datového skladu, uznává jako jediný návrh datového modelu prezentační vrstvy datového skladu entitně-relační modelování (ideálně ve 3. normální formě). Inmon uvádí, že jediné použití faktových a dimenzionálních tabulek je při návrhu **datových trhů** (*datamart*), a to stylem prvotního návrhu entitně-relačního modelu a z tohoto následně vyextrahovat fakta a dimenze. [8] Datový trh je dle Inmona *problémově orientovaná podmnožina dat prezentační vrstvy* [8]. Podle Kimballa je datový trh chápán jako *procesně orientovaná podmnožina tabulek prezentační vrstvy* [10].

3.3.1 Schéma hvězdy

Schéma hvězdy je nejzákladnější reprezentací dat pomocí multidimenzionálního datového modelu. Toto schéma je reprezentováno **jednou tabulkou faktů** a **více tabulkami dimenzí**. Tabulka faktů bývá obecně mnohem větší než tabulka dimenzí a neobsahuje redundantní data. Jak již bylo zmíněno v předešlé kapitole, obsahuje tabulka faktů měrné jednotky o sledované veličině a odkazy na záznamy v tabulkách dimenzí. Tabulky dimenzí obecně obsahují vysoce redundantní data. Je vysoce výhodné ukládat data v tabulce faktů a dimenzí na co nejnižší úrovni abstrakce, tedy co nejkonkrétnější data [10]. Ukládání velkého množství sesumarizovaných, obecně agregovaných dat, s sebou přináší snížení flexibility variantnosti ad-hoc dotazů.

Příklad schématu hvězdy o záznamech prodeje zboží je na *obrázku 3.2*. Firma si přeje provádět analýzu a získávat nové znalosti o prodejnosti svého zboží. Zboží je prodáváno v různém čase, na různém místě, spadá do různého kategorického odvětví, je prodáváno za jistou cenu a v různém množství. Jakožto dimenzionální tabulky se nabízí *položka* (zboží), *čas*, *odvětví* a *místo*. Za faktografické údaje by se dala považovat *cena* a *množství prodaného zboží*. V reálném případě (a je to vysoce žádoucí) by měly tabulky (zvláště pak dimenzionální) několik desítek sloupců.

Jak lze z tabulek dimenzí vyčíst, jsou redundantní. Například tabulka *čas* obsahuje sloupec *čtvrtletí*, který lze odvodit ze sloupce *měsíc*. Výpočet z hodnoty *měsíce* s sebou přináší zpomalení. Normalizací tabulky času taktéž dochází k následnému zpomalení při dotazování (provádí se spojení více tabulek, což má zajisté vliv na časovou odezvu), na druhou stranu však dojde k úspoře místa.

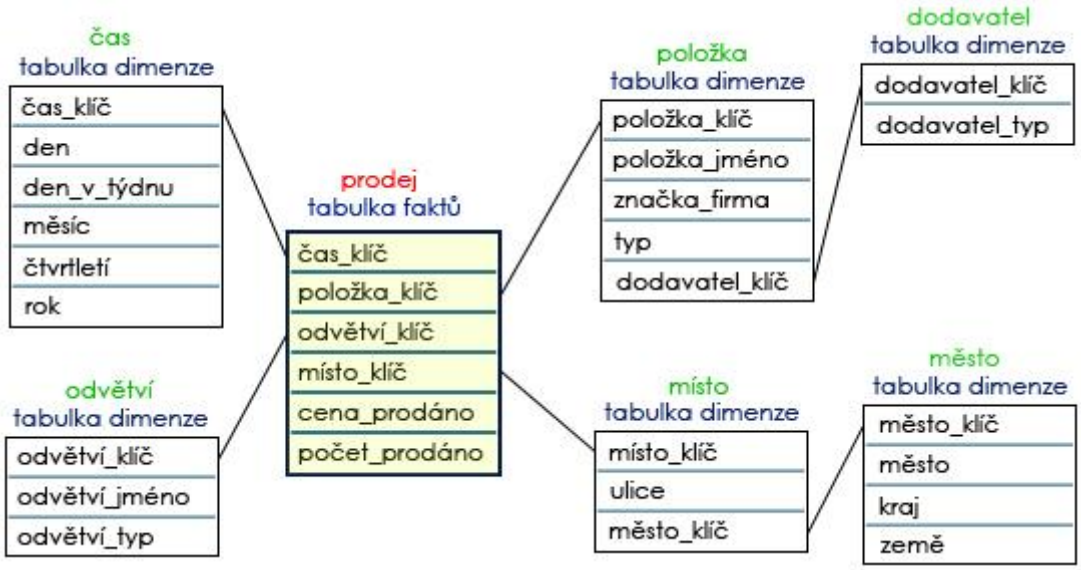


Obrázek 3.2: Schéma hvězdy prodeje zboží [1]

3.3.2 Schéma sněhové vločky

Jak bylo zmíněno na konci předešlé kapitoly, dochází občas k normalizacím tabulek dimenzí. Tato reprezentace se pak nazývá **schéma sněhové vločky**. Důvod tohoto činu je snížení redundance. Na druhou stranu však dochází ke zpomalení při vykonávání dotazů. V praxi se provádí jen ojedinělá normalizace, jelikož její paměťovou úsporou bývá přibližně pouze 1% (tabulka faktů často reprezentuje více než 90% datového úložiště) [10].

Na *obrázku 3.3 Schéma sněhové vločky prodeje zboží* došlo oproti předchozímu případu k rozdělení dvou tabulek dimenzí, a to tabulek *položka* a *místo*.

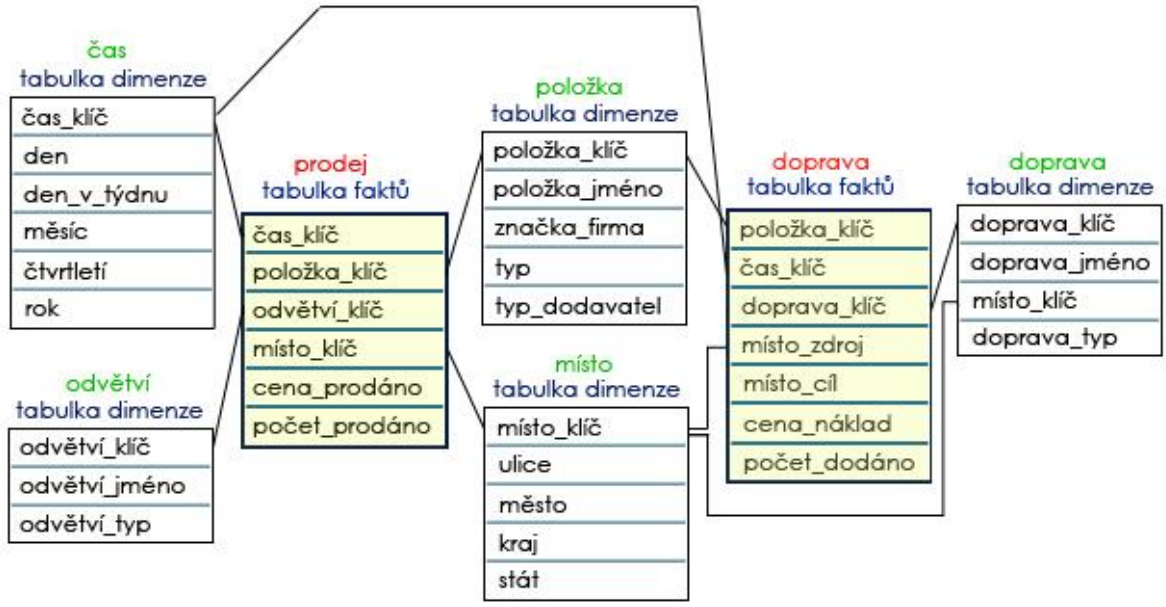


Obrázek 3.3: Schéma sněhové vločky prodeje zboží [1]

3.3.3 Schéma souhvězdí

Nasazení datového skladu se často provádí pro komplexnější problémy, kde bývá pravidlem více tabulek faktů. Komplexnější problémy se taktéž rozčleňují na více **datových trhů**. Tedy datový sklad se skládá z datových trhů. Logika členění datového trhu na datové sklady není jednotná a přímočará a vyžaduje hlubší analýzu problému. Již pánové, kteří stojí na špici problematiky datových skladů, nejsou v této problematice jednotní (viz kapitola *3.3 Multidimenzionální datový model*).

Schéma souhvězdí řeší koncepci, kdy je nutno využít více tabulek faktů. Tyto tabulky faktů mohou odkazovat na společné tabulky dimenzí. Na *obrázku 3.4* je znázorněna situace, kdy si obchodník přeje datovou reprezentaci *dopravy zboží*. Jak tabulka faktů *prodeje*, tak tabulka faktů *doprava* využívají společně tabulky dimenzí *místa* a *času* pro uchování lokalizačních dat.



Obrázek 3.4: Schéma souhvězdí prodeje zboží [1]

3.4 Operace OLAP

Datový sklad ukládá data nikoliv s ohledem na to, aby byly rychle prováděny editující operace, ale aby se efektivně prováděly složité (nemodifikační) dotazy. Mezi tyto složité dotazy patří **OLAP operace** (On-Line Analytical Processing).

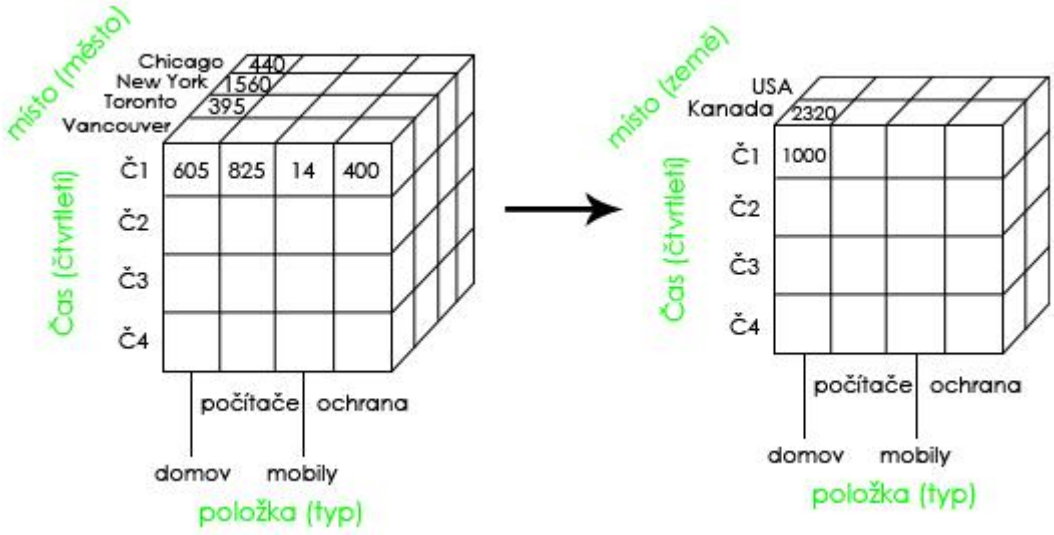
Jak již bylo zmíněno v předchozích článcích, datová struktura reprezentující data v datových skladech obsahuje často tabulky dimenzí. Tyto tabulky dimenzí obsahuji data, jež mají zpravidla konceptuální hierarchickou strukturu, a právě tato hierarchická povaha umožňuje realizovat OLAP operace, které slouží k variantním analytickým dotazům. Následuje popis hlavních OLAP operací, přičemž popis je realizován na příkladech, které jsou převzaty z [1].

3.4.1 Roll-Up

OLAP operace **Roll-UP** provádí posun nahoru v konceptuální hierarchii některé dimenze. Touto operací dochází k agregaci položek z tabulky faktů. Je možné, že při posunu v konceptuální hierarchii

nahoru dojde k odstranění dimenze (redukce), čímž dochází k analýze dat bez ohledu na danou dimenzi.

Na *obrázku 3.5* je zobrazení provedení operace Roll-Up nad dimenzí *město*. Dochází k hierarchickému posunu z úrovně *město* na úroveň *země*. Města *Chicago* a *New York* jsou sloučena v zemi *USA*, města *Toronto* a *Vancouver* jsou sloučena v zemi *Kanada*. Tímto dochází k agregování všech hodnot v jednotlivých městech.

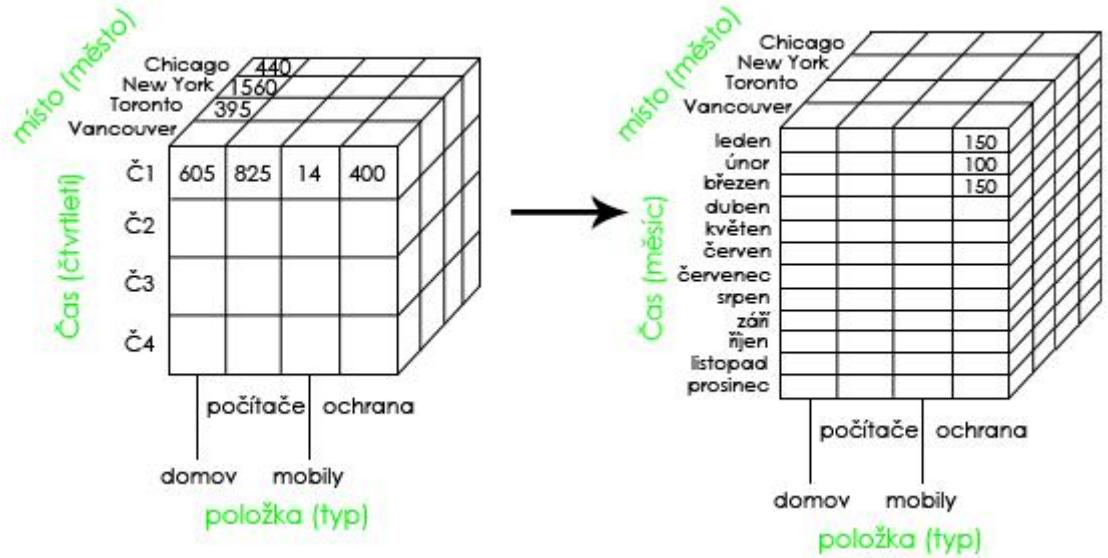


Obrázek 3.5: Operace Roll-Up nad dimenzí místo (město -> země) [1]

3.4.2 Drill-Down

Operace **Drill-Down** snižuje úroveň abstrakce dimenzionální hierarchie (méně detailní data jsou zobrazena na více detailní data). Je tedy opakem k operaci Roll-Up.

Na *obrázku 3.6* je zobrazení provedení operace Drill-Down nad dimenzí čas. Dochází k hierarchickému posunu z úrovně *čtvrtletí* na úroveň *měsíc*. Jedná se o detailnější pohled na prodej položek v jednotlivých měsících.



Obrázek 3.6: Operace Drill-Down nad dimenzí čas (čtvrtletí -> měsíc) [1]

3.4.3 Ostatní operace

Kromě **Roll-Up** a **Drill-Down** se mezi časté OLAP operace řadí **Slice**, **Dice** a **Pivot**.

Operace **Slice** provádí selekci nad jednou dimenzí. Výsledkem se stává podkostka, která splňuje danou selektivní podmínku (například prodej pouze ve městě Toronto).

Operace **Dice** na rozdíl od operace Slice vytváří selektivní podmínku nad více než jednou dimenzí.

Poslední zmíněná operace **Pivot** realizuje pouze změnu vizualizace, a to natočením osy (v libovolné dimenzi).

3.5 Návrh datového skladu

Tvorba datového skladu je komplexní problém, který nese relevanci celé firmy. Je tedy nutné provést na začátku zevrubnou analýzu všech potenciálních uživatelů a analýzu všech zdrojů, které přijdou do potenciální relace s firmou.

Ralph Kimball a **Bill Inmon** mají rozdílný přístup na tvorbu datových skladů a datových trhů. **Ralph Kimball** preferuje prvotní tvorbu datových trhů pro rychlou odezvu reportovacích a analytických informací firmy. Tyto datové trhy mohou být následně spojeny do datového skladu. Kombinace datových trhů je řízena skrz rozhraní architektury, kterou Kimball nazývá **Data Warehouse Bus**. Výhoda tohoto přístupu je rychlá odezva systému. **Bill Inmon** preferuje tvorbu komplexního datového skladu, a to způsobem prvotního návrhu pomocí entitně-relačního modelování (a z něj následně určit fakta a dimenze). Z tohoto datového skladu následně vznikají datové trhy. Tento přístup Inmon nazývá **Corporate Information Factory**. [4]

Jak již bylo zmíněno, tvorba datového skladu je rozsáhlý proces, a je nutno přistupovat **inkrementálně**. Tedy vyvarovat se přístupu „velkého třesku“. Z pohledu softwarového inženýrství jsou jednotlivé kroky inkrementu následující: plánování, studie požadavků, analýza problému, návrh, integrace dat, testování a samotné zavedení dat.

Podle Ralpha Kimballa jsou v návrhu multidimenzionálního datového úložiště důležité čtyři následující kroky [10]:

- **Výběr obchodního procesu**: Datový sklad je navržen pro analýzu zdrojů firmy a osob, které přijdou s firmou do relace. Je nutné provést předběžnou analýzu a relevanci těchto zdrojů firmy a samotných uživatelů. Přičemž komunikace s potenciálními uživateli je největším zdrojem obchodních informací.
- **Členění obchodních procesů**: Jedná se o ustanovení, co přesně budou obsahovat záznamy (řádky) v tabulce faktů. Členění by mělo být co nejdetailnější – uchovávat atomická data, a to z důvodu větší flexibility při variantních analytických dotazech.
- **Tvorba tabulek dimenzí:** Výběr dimenzí, které budou aplikovány na jednotlivé záznamy v tabulce faktů. Je nutné si položit otázku: „Jak uživatelé popisují data, která jsou výsledkem obchodních procesů?“. Atributy dimenzí jsou zpravidla textové, diskrétní hodnoty.

- **Tvorba tabulek faktů:** Identifikace numerických dat, které budou reprezentovány v tabulkách faktů. Výčet atributů zpravidla vznikne odpovědí na otázku: „Které hodnoty jsou měřitelné?“.

Při návrhu datového skladu je velmi důležitým aspektem, k čemu bude tento datový sklad sloužit. Mezi dvě základní využití slouží jednak k analýze a reportování firemních dat, a jednak k získávání nových znalostí. Zatímco pro analytické využití je velmi vhodné uchovávat velké množství agregovaných dat (pro rychlou odezvu ad-hoc dotazů), pro získávání znalostí se často nevyužívají vysoce agregované hodnoty – dochází sice k hierarchickému posunu, avšak velmi často ne na komplexní úroveň agregace. Pro získávání znalostí je datový sklad prospěšně využíván z důvodu potřeby integrovaných a transformovaných dat (čištění, konzistentnost, redukce), procházení dat na různé úrovni hierarchie a pro samotné řízení přístupu k datům.

3.6 Oracle Warehouse Builder

Teoretické poznatky z předchozích podkapitol budou aplikovány na datový sklad od firmy Oracle, a to v Oracle Warehouse Builder 11*g*. Tato kapitola obecně popisuje nástroj na tvorbu datových skladů Oracle Warehouse Builder 11*g* a jeho architekturu. Základní demonstrační popis tvorby datového skladu v Oracle Warehouse Builder 11*g* je uveden v *Příloze A – Tvorba datového skladu v Oracle Warehouse Builder 11g*.

3.6.1 Co je Oracle Warehouse Builder

Oracle Warehouse Builder je komplexní nástroj na datovou integraci. Oracle Warehouse Builder poskytuje tvorbu datových skladů, migraci dat z různých zdrojů, audit dat, plně integrované relační a dimenzionální modelování, transformační funkce a podporu tvorby metadat. [3]

Oracle Warehouse Builder vznikl v roce 2000 a je součástí aktuální verze Oracle Database Enterprise Edition 11*g*.

Pro tvorbu datového skladu je nutné mít vytvořenou databázi dle schématu *Data Warehouse*.

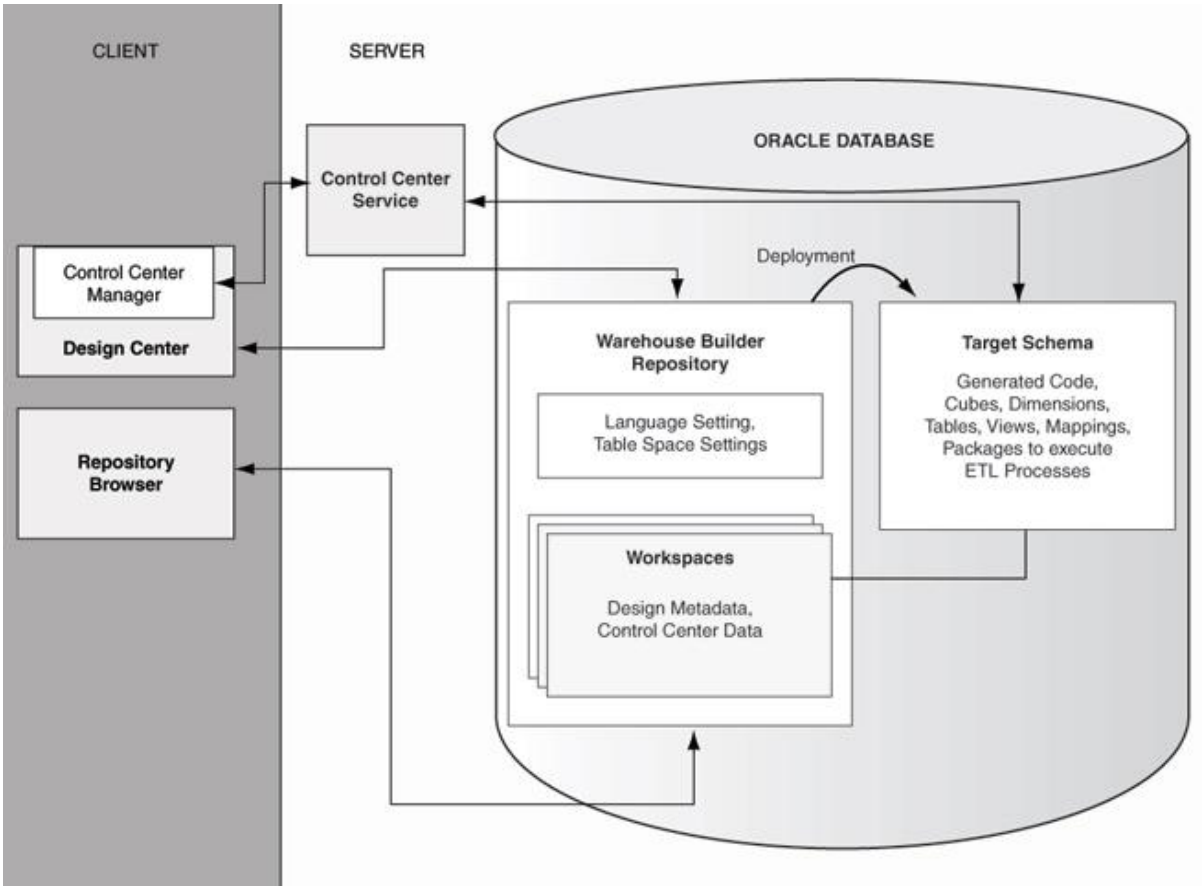
3.6.2 Architektura Warehouse Builder

Na *obrázku 3.7* jsou znázorněny komponenty architektury Oracle Warehouse Builder 11*g*. Architektura se přirozeně skládá z klientské a serverové části.

Hlavní komponentou **klientské části** je **Design Center**. Jedná se o grafické uživatelské rozhraní sloužící pro import zdrojových objektů, návrh **ETL** procesů, mapování a celkovou integraci heterogenních zdrojů. Mezi zdrojové objekty je možné zvolit databáze různých firem (Oracle, MS SQL, DB2, ..), soubory s předdefinovaným formátováním (csv, tsv), XML soubory, aplikační soubory jiných softwareových produktů. **ETL** (Extraction, Transformation, Loading) procesy slouží k mapování zdrojových dat na data cílová. Při tomto procesu dochází k realizaci různých transformačních funkcí. V Oracle Warehouse Builderu lze realizovat následující transformační funkce: agregace, operace nad multidimenzionální kostkou, matematické funkce, spojování, rozdělování, množinové operace a jiné. Výstupem **Design Center** je logický model - metadata. Další

důležitou komponentou je **Control Center Manager**, jenž slouží pro fyzickou tvorbu objektů, jejichž metadatový popis byl ustanoven v Design Center. Dochází ke komunikaci se serverem (prostřednictvím serverové komponenty Control Center Service). [3]

Hlavní komponentou na **serverové straně** je **Warehouse Builder Repository**. Tato komponenta slouží jako fyzické úložiště pro metadatové objekty, mapovací a transformační schémata (ETL procesy). Jsou zde též uložena provozní data již zmíněných komponent Control Center Manager a Control Center Service. K tvorbě tohoto speciálního úložiště slouží **Repository Assistant**. Další důležitou částí serveru je cílové úložiště, kde se nachází fyzické reprezentace objektů. Tedy například samotné tabulky, pohledy a mapované textové soubory do externích tabulek. [3]



Obrázek 3.7: Komponenty Oracle Warehouse Builder 11g [3]

4 Algoritmizace asociačních pravidel

Kapitola *2.3.1 Asociační pravidla* byla informativní kapitolou problematiky získávání znalostí pomocí asociačních pravidel. Tato kapitola se zabývá algoritmickým principem. Budou zde popsány algoritmy, které budou využity ve výsledné aplikaci. Nejdříve budou vysvětleny základní pojmy, dále budou popsány jednodimenzionální, jednoúrovňová pravidla, následně pak vícedimenzionální pravidla.

4.1 Základní pojmy

Jak již bylo uvedeno v kapitole *2.3.1 Asociační pravidla*, principem asociačních pravidel je nalézt frekventované vzory. Relevance těchto vzorů je pak dána **podporou** a **spolehlivostí.** Popis pojmů asociačního pravidla, podpory a spolehlivosti vystihuje následující definice [1]:

Buď $I = \{I_1, I_2, I_3, ..\}$ množina položek a nechť D je množina databázových transakcí, kde každá transakce T je množina položek takových, že platí $T \subseteq I$. Každá transakce je spojena s jednoznačným identifikátorem *TID*. Buď A množina položek. Říkáme, že transakce T obsahuje A, pokud platí $A \subseteq T$. **Asociační pravidlo** je implikace tvaru $A \Rightarrow B$, kde $A \subset I$, $B \subset I$, $A \cap B = \emptyset$.

Podpora je pak definována jakožto pravděpodobnost, že transakce obsahuje $A \cup B$, tedy:

$$podpora(A \Rightarrow B) = P(A \cup B)$$

Spolehlivost je podmíněná pravděpodobnost vyjadřující, že transakce obsahující A, taktéž obsahuje B, tedy:

$$spolehlivost(A \Rightarrow B) = P(B|A) = \frac{podpora(A \cup B)}{podpora(A)}$$

Množina položek, jež má podporu vyšší než požadovaná minimální podpora, se nazývá **frekventovaná množina**.

Asociační pravidlo, jež má podporu a spolehlivost vyšší, než jsou minimální hodnoty požadované podpory a spolehlivosti, je **silné asociační pravidlo**.

Obecný postup na výpočet silných asociačních pravidel je následující:

- Na základě minimální podpory vypočítat frekventované množiny
- Na základě minimální spolehlivosti z frekventovaných množin vypočítat silná asociační pravidla.

4.2 Jednodimenzionální, jednoúrovňová pravidla

Tato kapitola popisuje charakteristiku jednodimenzionálních, jednoúrovňových pravidel a dva algoritmy, které budou následně v aplikaci implementovány.

4.2.1 Charakteristika

Jednodimenzionálnost značí, že se budou získávat asociační pravidla z jedné entity. Mezi nejčastěji analyzovanou entitu patří *nákupní košík* (položky nákupu). Jednoúrovňovost vystihuje neuvažování hierarchického uspořádání položek (neexistuje tedy uspořádání).

4.2.2 Apriori

Mezi nejznámější algoritmy řešící problém hledání asociačních pravidel patří algoritmus **Apriori**. Tento algoritmus využívá předpokladu, že transakce, která obsahuje *n* položek, nemůže mít větší četnost než stejná transakce ochuzená o libovolnou položku (tedy zredukovaná transakce na *n-1* položek). Z toho plyne, že každá podmožina frekventované množiny musí být též frekventovaná. Jedná se o tzv. *Apriori vlastnost.*

Následuje popis algoritmu Apriori pomocí pseudokódu [1]:

Vstup: D – databáze transakcí
$min\_podpora$ – minimální podpora
Výstup: L – frekventovaná množina databáze D

Algoritmus:

```
L1 = najdi_frekventované_1-položky(D);
for( k = 2; Lk-1 ≠ ∅; k++){
  Ck = apriori_gen(Lk-1);
  for each transakce t ∈ D{
    Ct = podmnožina(Ck, t);
    for each kandidát c∈ Ct
      c.počet++;
  }
  Lk = {c∈ Ck | c.počet ≥ min_podpora};
}
return L = ∪ k Lk;

procedure apriori_gen(Lk-1: frekventované_(k-1)_položky)
  for each l1∈ Lk-1
    for each l2∈ Lk-1
      if (l1[1] = l2[1]) ∧ (l1[2] = l2[2]) ∧ .. ∧ (l1[k-2] = l2[k-2]) ∧ (l1[k-1] < l2[k-1]){
        c = l1 JOIN l2; //tvorba kandidátů
        if( má_nefrekventovanou_podmnožinu(c, Lk-1) ) then
          delete c;
        else přidej c do Ck;
      }
return Ck;

procedure má_nefrekventovanou_podmnožinu(c:kandidát; Lk-1:frekventované_(k-1)_položky)
for each (k-1)-podmnožina s of c
  if s∉Lk-1 then
    return true;
return false;
```

Algoritmus **Apriori** patří mezi nejméně výkonné algoritmy, nicméně se jeho základní princip využívá v jiných algoritmech.

4.2.3 FP-strom

Mezi nejrychlejší algoritmy současnosti se řadí algoritmus využívající konstrukce **FP-stromu** [12]. FP-strom slouží ke komprimovanějšímu uložení frekventovaných položek. Dochází ke sdílení společných položek, navíc více frekventované položky budou ve stromu dříve sdíleny, a tak se redukuje jejich vícečetný výskyt. Další výhodou je, že se databáze prochází pouze dvakrát, a to z důvodu výpočtu frekventovaných 1-množin a při konstrukci FP-stromu.

Následuje symbolický zápis konstrukce a využití FP-stromu [1]:

Vstup: *D* – databáze transakcí
min_podpora – minimální podpora
Výstup: Kompletní množina frekventovaných vzorů

Algoritmus:

1. ***Konstrukce FP stromu:***
   (a) Projdi jednou databází a zjisti frekventované 1-množiny, uchovej též jejich podporu. Tuto komplexní množinu nazvi *F*. Seřaď *F* podle hodnoty podpory s klesající tendencí. Vznikne seznam *L*.
   (b) Vytvoř kořen FP-stromu, označ jej jako *null*. Pro každou transakci *Trans* v *D* proveď následující kroky:
   Vyber a seřaď frekventované položky v *Trans* s ohledem na pořadí seznamu *L* (více frekventovaná položka v *L* bude v seřazeném *Trans* na přední pozici). Nechť je seřazený seznam v *Trans* nazván [*p*|*P*], kde *p* je první element a *P* je zbývající část seznamu. Nechť *vlož_do_stromu*([*p*|*P*], *T*) jsou nazvány následující kroky:
   Jestliže *T* má potomka *N* takového, že *N.jméno_položky* = *p.jméno_položky*, potom zvyš *N.počet* o jedničku; jinak vytvoř nový uzel *N* s hodnotou *N.počet* rovno 1, jeho otcovský uzel bude spojen do *T*. Jestliže *P* je neprázdné, zavolej *vlož_do_stromu*(*P*, *T*) rekurzivně.
2. ***Dolování v FP-stromu*** *zavoláním procedury FP_růst(FP_strom, null)**:***

```
procedure FP_růst(Strom, α)
if Strom obsahuje jednoduchou cestu P then
  for each kombinaci β uzlů v cestě P
    generuj vzor β ∪ α s podpora_počet = minimální podpora uzlů v β
else for each a_i v hlavičce ve Strom {
  generuj vzor β = a_i ∪ α s podpora_počet = a_i.podpora_počet;
  vytvoř podmíněný β strom FP-strom Strom_β ;
  if Strom_β ≠ ∅ then
    zavolej FP_růst(Strom_β, β);
}
```

Předchozí algoritmus využívá následující vlastnosti [12]:

Nechť α je frekventovaná množina v databázi, B je podmíněný základ α, a β je množina v B. Pak $\alpha \cup \beta$ je frekventovaná množina v databázi, jestliže β je frekventovaná v B.

4.3 Multidimenzionální pravidla

Kapitola pojednává o charakterizaci multidimenzionálních pravidel a popisu principů dolování z takového charakteru dat.

4.3.1 Charakteristika

Pro multidimenzionální pravidla je charakteristické, že dochází při dolování asociačních pravidel k uvažování více entit. Například při analýze nákupního košíku může mít obchodní relevanci znalost, které věkové kategorie, které profese a která pohlaví nakupují často, které položky (data jsou získána například z věrnostních kartiček – vyplnění dotazníku). Znalosti se pak dají zužitkovat například pro cílenou tvorbu reklamních letáků. Pro tento případ je užitečné využít datový model popsaný v kapitole *3.3 Multidimenzionální datový model*.

Zmíněné dimenzionální entity se taktéž nazývají predikáty. Asociační pravidlo, které obsahuje více predikátů, by mohlo vypadat následovně:

Věk(X, "30..39") $\wedge$ Povolání(X, "programátor") $\Rightarrow$ Kupuje(X, "Notebook IBM")

Dolovací algoritmy, které řeší problém hledání multidimenzionálních pravidel, mohou být založené na algoritmech aplikované na jednodimenzionálních pravidlech (například výše popsané algoritmy Apriori nebo FP-strom). Naskýtá se tu však potenciální problém výskytu kvantitativních atributů, které je nutné upravit pro potřeby dolování (například věk).

4.3.2 Úprava kvantitativních atributů

Jak již bylo uvedeno v závěru předchozí kapitoly, dochází při multidimenzionálním dolování u kvantitativních atributů k diskretizaci. Je to hlavně zapříčiněno z důvodu velkého počtu těchto hodnot (obor hodnot kvantitativního atributu je velký nebo nekonečný), a tím přílišnému snížení pravděpodobnosti nalezení frekventovaných vzorů. Techniky pro zpracování multidimenzionálních dat se pak dělí podle toho, jak jsou zpracovány kvantitativní atributy [1]:

- **Statická diskretizace kvantitativních atributů** – Před samotným dolováním probíhá statická diskretizace tohoto kvantitativního atributu, a to často diskretizace do hloubky nebo do šířky. Numerické hodnoty jsou tedy staticky nahrazeny intervaly. Dolování pak probíhá nad reprezentanty staticky určených intervalů.
- **Kvantitativní asociační pravidla** – Jde o pravidla, v nichž oproti předchozí metodě probíhá diskretizace dynamicky v průběhu dolování, a to tak, aby byla splněna jistá podmínka. U této metody dochází často k počáteční statické diskretizaci. V těchto diskrétních intervalech dojde ke zjištění podpory. Pokud sousední intervaly splňují minimální podporu, dojde k jejich dynamickému spojení (pokud není přesáhnuta maximální podpora).

- **Asociační pravidla založená na vzdálenosti** – Jedná se o princip, který nejlépe postihuje charakteristiku dat. Dolovací metody založené na této bázi využívají shlukovou analýzu dat. Principem dolování je v počátku nalezení shluků *n*-tic. Následně se vytvářejí asociační pravidla, která obsahují shluky hodnot. Tyto shluky hodnot berou v úvahu relativní vzdálenosti mezi hodnotami.

Jak již bylo zmíněno, algoritmy řešící multidimenzionální pravidla lze často převést na algoritmy řešící jednodimenzionální pravidla s tím, že pokud je nutné, dochází k úpravě kvantitativních atributů. Vyvíjený systém využívá výhod podpůrného datového skladu, v ramci kterého lze provádět různé transformační funkce (ETL proces). Bude tedy uvažováno, že jednotlivé predikáty nabývaji již předzpracovaných hodnot.

5 Návrh systému pro dolování

Zatímco předešlé kapitoly popisovaly spíše teoretické východisko a podpůrný aparát, tato kapitola je úvodní kapitolou do praktické problematiky diplomové práce. Jedná se o počáteční pohled na aspekty výsledného systému, kdy jsou popisovány vlastnosti a požadavky navrhované aplikace. Dále je zde popsána architektura na vysoké úrovni abstrakce a samotný logický návrh řešení aplikace.

5.1 Požadavky na systém

Zadání diplomové práce je dosti obecného charakteru. Dle stěžejních bodů zadání diplomové práce:

3. *Navrhněte aplikaci, která bude využívat zvolený algoritmus pro dolování asociačních pravidel z datových skladů*
4. *Aplikaci realizujte a její funkčnost ověřte na vhodném vzorku dat*

je požadováno, aby byl systém zaměřený na asociační pravidla a jako podpůrný nosič dat využíval datový sklad. Právě těmto problematikám se věnovala teoretická část realizována v rámci semestrálního projektu. Po konzultaci s vedoucím diplomové práce, po mém předběžném návrhu, bylo zadání dospecifikováno dle následujícího popisného odstavce.

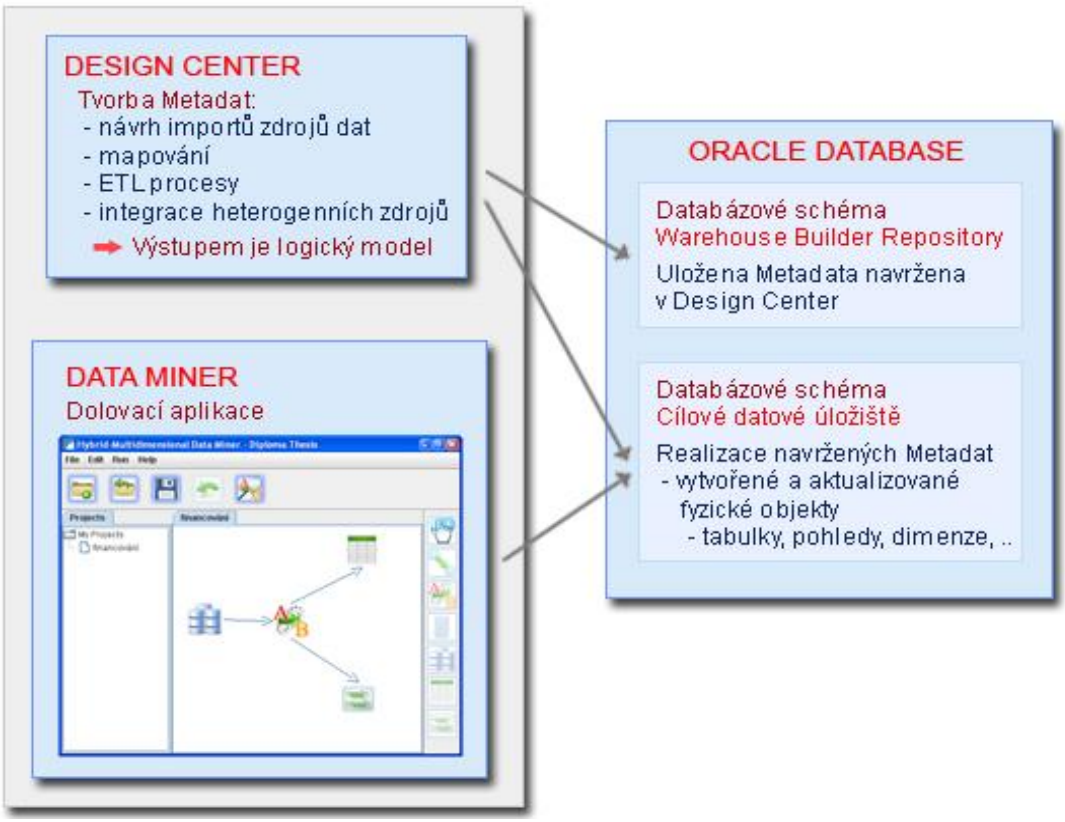
Výsledný systém bude sloužit jakožto obecný nástroj pro podporu dolování asociačních pravidel. Nástroj bude disponovat příjemným a intuitivním grafickým uživatelským rozhraním, aby bylo veškeré nastavení dolovacího procesu přímočaré. Nastavení asociačních parametrů bude obsahovat kromě základních vlastností (podpora, spolehlivost) možnost omezení či specifikování asociačních pravidel. Oproti většině existujících systémů bude systém, vyvíjený v rámci diplomové práce, obsahovat kromě algoritmu **Apriori**, viz *4.2.2 Apriori*, algoritmus další – byl zvolen algoritmus využívající **FP-strom**, viz kapitola *4.2.3 FP-strom*. Jakožto zdroj dat bude předpokládáno využití datového skladu firmy Oracle. Konkrétně v průběhu realizace práce bylo používáno Oracle Warehouse Builder 11*g*, viz kapitola *3.6 Oracle Warehouse Builder*. Dle obecného procesu dolování, viz *Obrázek 1.1*, je praktickou součástí tohoto procesu čištění, integrace, výběr a transformace, což je zajištěno právě datovým skladem Oracle Warehouse Builder 11*g* (ETL proces). V aplikaci dojde k připojení na zvolený datový sklad a uživatel vybírá již předzpracovaný zdroj dat ze svého databázového schématu, ze kterého jsou čerpána data pro následné získávání asociačních pravidel. Aplikace by měla být prakticky použitelná a otestována na reálných datech.

5.2 Architektura systému

Již v předcházejícím textu bylo zmíněno, že zdroj dat bude umístěn v datovém skladu Oracle. Pro tento účel je nutné mít vytvořenou Oracle databázi. Jelikož se bude pracovat s datovým skladem a v databázi se budou ukládat mimo jiné metadata datového skladu, je účelné vytvořit databázi dle šablony *Data Warehouse*. Tím dojde k zefektivnění ukládání dat a přístupové doby k datům. Pro práci s datovým skladem je nutné mít v databázi vytvořena dvě schémata – *Warehouse Builder Repository*

a schéma pro *cílové datové úložiště*. Ve *Warehouse Builder Repository* jsou uložena veškerá metadata tvorby dat v datovém skladu. Metadaty je myšlen návrh zdrojů pro import dat a provedení jejich integrace, čímž dojde k namapování obecně heterogenních zdrojů do jediného cílového úložiště a provádění komplexních ETL procesů. Tato metadata se navrhují v aplikaci *Design Center*. Výsledná data (namapovaná obecně z heterogenních zdrojů) jsou uložena do schématu *cílového datového úložiště*. Vyvíjená dolovací aplikace, *Data Miner*, přistupuje k tomuto *cílovému datovému úložišti*, kde již očekává integrovaná, selektovaná a transformovaná data. Proces tvorby datového skladu a realizace hlavních funkcí datového skladu (tvorba uživatelů, mapování zdrojů, integrace, nástin ETL procesů) je obsahem *přílohy A - Tvorba datového skladu v Oracle Warehouse Builder 11g*.

Při praktickém nasazení bude *Oracle databáze* se schématy *Warehouse Builder Repository* a *cílového datového úložiště* na jiném výpočetním stroji (server) než aplikace *Design Center* pro návrh metadat veškerých procesů a samotná *dolovací aplikace*. Dolovací aplikace pak přistupuje pouze ke schématu *cílového datového úložiště*, kde jsou uložena výsledná data. Základní koncept architektury je na *obrázku 5.1*.



Obrázek 5.1: Architektonický přístup aplikace

Použití datového skladu však není nezbytnou nutností. K databázovému schématu, kde se nachází integrovaná a transformovaná data, se programově přistupuje stejně jak k obecnému databázovému schématu. Vyvinutá dolovací aplikace očekává data ve formátu transakčním, či ve speciálním formátu schématu hvězdy (viz *3.3.1 Schéma hvězdy*). Schéma hvězdy, což je jedna z typických reprezentací dat v datových skladech, lze snadno nasimulovat spojením tabulek, viz kapitola *6.4 Zdroj dat*. Jak již bylo zmíněno, hlavní účel použití datového skladu je z důvodu využití

nástroje pro integraci a transformaci dat – tedy *předzpracování dat*, a není tedy nezbytnou nutností. Aplikace může pracovat i s generickou databází Oracle.

5.3 Návrh aplikace systému

Na *obrázku 5.2* se nachází UML schéma diagramu balíčků a důležitých tříd vyvíjené aplikace. UML diagram byl realizován v programu *Visual Paradigm for UML 7.0* [13]. Ve skutečnosti se zmíněné balíčky nachází v hlavním balíčku *net.pumprla.dataminer*. Nejedná se o detailní návrh aplikace, nýbrž rozčlenění logických celků a vzájemná komunikace těchto celků, s případným rozborem vnitřní třídní struktury balíčku. Detailnějšímu popisu důležitých komponent systému bude věnována následující kapitola *6 Implementace systému*. Následuje popis systému z pohledu strukturálně logického návrhu s případnou okrajově implementační vazbou.

Vstupním bodem programu je třída *Miner*, jež se nachází v balíčku *miner*. Tato třída má přirozeně vazbu na *gui* balíček, který má na starosti grafické uživatelské rozhraní. Grafické uživatelské rozhraní je vstupním bodem pro uživatele, který řídí činnost běhu programu.

V rámci uživatelského rozhraní má uživatel možnost graficky grafově realizovat dolovací proces. Tento dolovací proces se skládá z jednotlivých kroků dolování – uzly grafu (na plátno přetahovat jednotlivé komponenty a spojovat je – tím vytvářet dolovací proces). Je tedy účelné vytvořit balíček, nazvaný *tool*, který obsahuje jednotlivé třídy zastupující jednotlivé kroky dolování (zdroj dat, asociační pravidla, XML výstup, ..), a tím uchovávat jednotlivé charakteristické vlastnosti. Jak již bylo zmíněno, dolovací proces je graficky realizován uživatelem ve formě grafu, a tedy každá komponenta musí uchovávat obrázek a souřadnice. Tím vzniká třída, která je třídním předkem všech ostatních tříd z uzlů procesu dolování, a sice třída *Tool*. Uživatel má na výběr, zda-li chce dolovat z transakčního zdroje dat nebo multidimenzionálního a tím vzniká odlišné nastavení pro výsledná asociační pravidla. Některé vlastnosti asociačních pravidel (podpora, spolehlivost, maximální počet položek v pravidle) jsou však společné, a proto je vytvořena generická asociační třída *ToolAssocationRules*, z níž dědí třídy *ToolAssociationMultidim* a *ToolAssociationTransaction*. Třída *ToolsInRelation* reprezentuje grafové schéma dolovacího procesu, přičemž je nutná kontrola, zda-li mohou některé uzly tvořit relaci (nelze například spojit zdroj dat přímo s vizualizačním výstupem výsledků). Třída *ToolType* je využívána v rámci celého systému a obsahuje pouze statické členy a metody (konstanty pro jednotlivé uzly grafu, omezení každého uzlu na počet vstupů/výstupu).

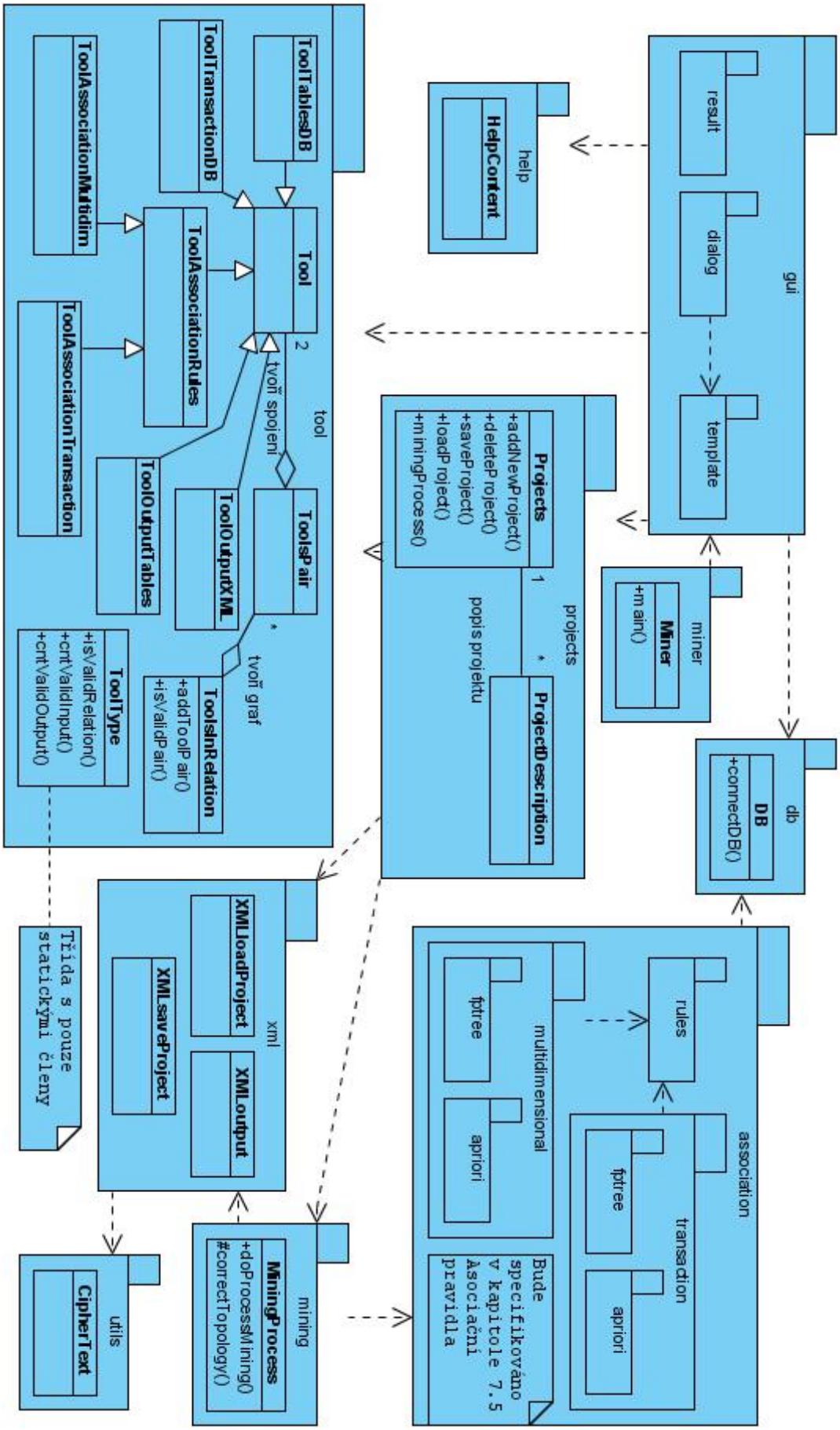
Balíček, který má na starosti správu projektů, se nazývá *projects*. Třída *Projects* uchovává seznam všech otevřených projektů a každý projekt je popsán v rámci třídy *ProjectDescription*. Balíček *projects* má přímou vazbu na balíček *tool*, jelikož projekt je charakteristický nastavením dolovacích parametrů (které jsou uloženy právě v jednotlivých třídách balíčku *tool*).

Aplikace umožňuje ukládat a znovu ustanovit uložené projekty. Projekt – dolovací proces – je uložen ve speciální XML struktuře, proto balíček *projects* využívá služeb balíčku *xml*.

V rámci jednotlivých projektů dochází k požadavku procesu dolování. Je nutné zjistit, zda-li je dané grafové schéma, které vytvořil uživatel, validní a zda-li splňuje všechny náležitosti (nastavení nezbytných parametrů). Pokud je dolovací schéma v pořádku, dochází k dolování. Balíček *projects* k tomu využívá služby balíčku *mining*, konkrétně třídu *MiningProcess*.

Funkčním jádrem aplikace je balíček *association,* v rámci kterého jsou realizovány asociační dolovací algoritmy pro výpočet frekventovaných množin a z těchto množin tvorba asociační pravidel. Frekventované množiny, které jsou získávány z transakčních zdrojů, jsou vytvářeny v rámci vnitřního balíčku *transaction*. Frekventované množiny, které jsou získávány z multidimenzionálních zdrojů, jsou pak vytvářeny v rámci vnitřního balíčku *multidimensional*. Každý z těchto balíčků následně obsahuje balíčky *apriori* a *fptree,* kde je realizace konkrétních algoritmů. Funkci tvorby asociačních pravidel zajišťuje balíček *rules*, jenž je využíván balíčky *transaction* i *multidimensional*. V průběhu implementace se ukázalo, že je užitečné vytvořit balíčky *apriori* a *fptree* i v rámci balíčku *association* (není součástí UML grafu na *obrázku 5.2*) – tyto balíčky nesou vlastnosti pro jednotlivé algoritmy, které jsou společné v rámci transakčního a multidimenzionálního dolování frekventovaných vzorů. Bližšímu popisu realizace asociačních pravidel je věnována kapitola *6.5 Asociační pravidla*.

Posledním důležitým balíčkem je balíček *db* s třídou *DB*. Balíček *db* má na starost komunikaci s databází (konkrétně databází Oracle). Tento balíček je využíván balíčky *gui* a *association*. Balíček *gui* využívá databázi k uživatelskému – formulářovému nastavení zdroje dat pro dolování (výběr tabulek, tvorba hvězdy). Balíček *association* využívá databázi k čerpání samotných dat, nad kterými se provádí dolovací proces.



Obrázek 5.2: Návrh stěžejních balíčků a tříd aplikace

6 Implementace systému

Tato kapitola popisně rozšiřuje některé aspekty, které byly zmíněny v kapitole *5 Návrh aplikace systému*. Jedná se o implementační a funkční popis se zaměřením na realizaci uživatelského rozhraní, mapování zdrojů dat a na tvorbu asociačních pravidel.

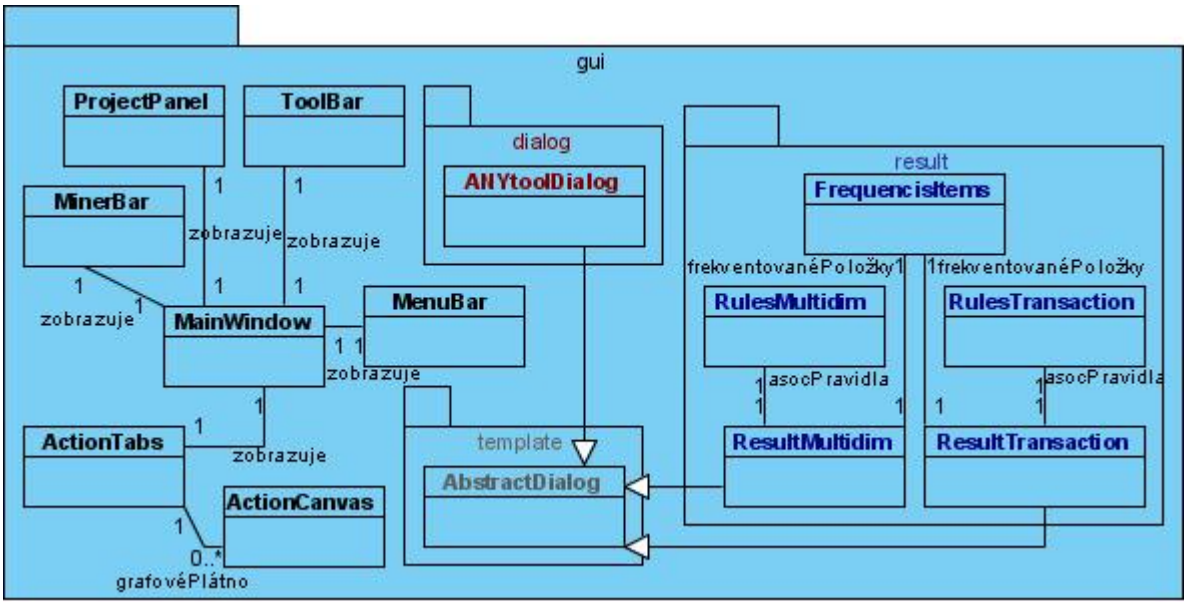
6.1 Implementační prostředí

Z důvodu vyšší přenositelnosti byla aplikace implementována v jazyce **Java SE**. Jako vývojové prostředí bylo použito **NetBeans IDE 6.5**. Zdrojem dat byla databáze Oracle (schéma Warehouse Builder Repository, schéma cílového datového úložiště s daty), která byla spravována lokálním databázovým serverem **Oracle 11*g***.

Jako alternativa k datovému skladu Oracle se nabízelo využití datového skladu MS SQL Server. Firma Oracle byla upřednostněna z důvodu zjištění většího zastoupení využití na poli datových skladů. Aplikace, které jsou vyvíjeny jako podpůrné nástroje pro správce a uživatele databázového systému Oracle, jsou v poslední době často implementovány v jazyce Java, proto byl tento jazyk zvolen i pro aplikaci vyvíjenou v rámci diplomové práce.

6.2 Uživatelské rozhraní

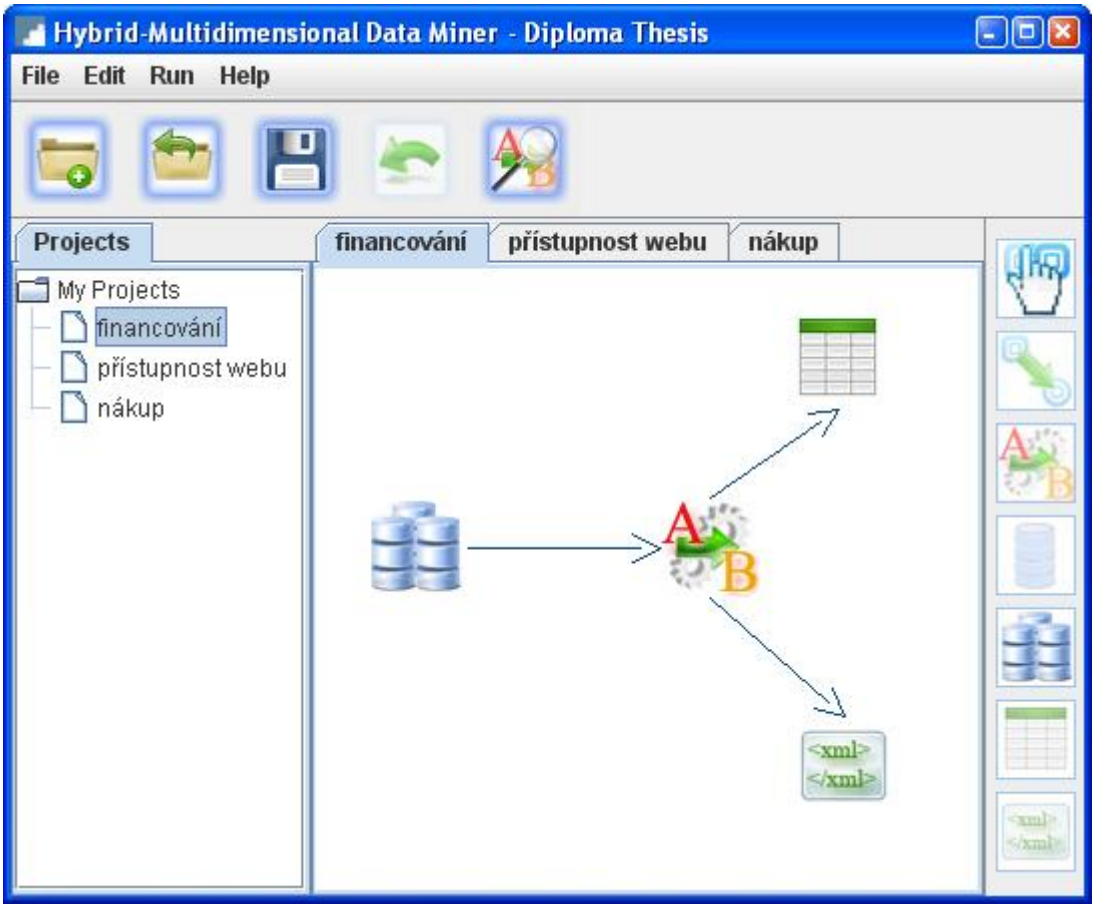
Aplikace má být dle požadavků intuitivní na ovládání. Veškerá interakce se systémem probíhá pomocí grafických komponent. Celá vizuální reprezentace je implementována v balíčku *gui*, viz *obrázek 6.1*.



Obrázek 6.1: UML diagram návrhu gui balíčků a tříd

Tento balíček přímo obsahuje třídy, které realizují hlavní grafické rozhraní aplikace. Kmenovou třídou je třída *MainWindow*, která stmeluje ostatní třídy *MenuBar* (menu aplikace),

ToolBar (rychlé menu - ikony rychlého spuštění), *ProjectPanel* (strom projektů), *MinerBar* (ikony dolovacího procesu) a *ActionTabs* (kontejner pro schémata dolování). Třída *ActionTabs* obsahuje vazbu na třídu *ActionCanvas*, která implementuje zobrazení a manipulaci s uživatelem navrženým grafem dolování. Jelikož může mít uživatel rozpracovaných více grafových, dolovacích schémat, je vazba typu (1 – 0..*). Všechny tyto třídy nesou zpětnou referenci na třídu *MainWindow* pro účel výpisu chybových hlášek a jakožto prostředníka pro komunikaci dvou grafických komponent. Realizované hlavní grafické uživatelské rozhraní je na *obrázku 6.2*.



Obrázek 6.2: Grafické uživatelské rozhraní hlavní části aplikace

Uživatel vytváří dolovací proces ve formě grafu. Každý uzel grafu s sebou nese charakteristické vlastnosti (výběr zdrojových tabulek, nastavení podpory, ..), které jsou typické pro jednotlivé uzly. Tyto parametry lze nastavit prostřednictvím formulářů. Správa těchto grafických komponent je ve vnitřním balíčku *dialog*. Ke každé dolovací komponentě koresponduje dialogová třída. Tyto třídy mezi sebou nemají žádnou vazbu a všechny dědí z obecné dialogové třídy *AbstractDialog* (vnitřní balíček *template*), která zajišťuje obecné dialogové vlastnosti. Tyto třídy nejsou na UML schématu *obrázku 6.1* zaznamenány. Třídy se jmenují *AssociationMultidimDialog*, *AssociationTransactionDialog*, *TransactionDbSetDialog*, *TablesDBSetDialog* a *SetXMLDialog*. Dále se zde nachází dialogová třída na tvorbu nových projektů *NewProjectDialog*.

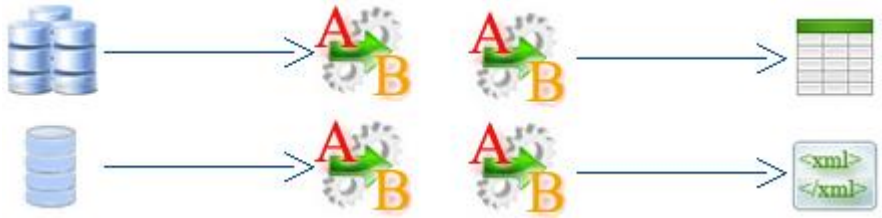
Posledním vnitřním balíčkem je balíček *result*, který obsahuje třídy pro vizualizaci asociačních pravidel a frekventovaných položek ve formě tabulky. Menším úskalím momentálního řešení je, že při požadavku na zobrazení výsledků ve formě tabulky, dochází ke kompletní tvorbě všech tabulkových dat. Pokud je výsledných asociačních pravidel velké množství, je spotřebováno větší

množství paměti. Jelikož v tabulce dochází k momentálnímu zobrazení relativně malého počtu položek, stačilo by do tabulkové podoby převést pouze aktuálně zobrazená data. Ve výsledku by došlo k lepšímu hospodaření s pamětí (proces zobrazení asociačních pravidel by byl však pomalejší).

6.3 Dolovací schéma

Jak již bylo několikrát zmíněno, uživatel vytváří prostřednictvím grafického rozhraní schéma dolovacího procesu. Vytváří se grafová (momentálně stromová) topologie. Pro tuto topologii je přirozené uvažovat některá omezení, která jsou implikována z obecných principů dolování. Dle schématu *obrázku 5.2* má dolovací proces na starost třída *MiningProcess*, jež je v balíčku *mining*. Pokud jsou splněna následující omezení, je dolovací proces započat a jeho běh je realizován v rámci nového samostatného vlákna, aby bylo možné proces dolování předběžně ukončit.

- Jako z prvních omezení je předpokládáno, že každý uzel grafu je v relaci s některým jiným uzlem grafu (těchto uzlů může být obecně libovolný počet). Neexistuje tedy uzel bez relace spojení.
- Binární relace mezi uzly grafu mohou být jen ty, které jsou na *obrázku 6.3*.



Obrázek 6.3: Validní binární relace mezi uzly grafu pro dolování

- V grafu se musí vyskytovat zdroj dat a uzel řešící problém dolování dat (nyní pouze asociační pravidla), přičemž zdroj dat musí být (obecně nepřímým) předchůdcem. Určení zdroje dat jako (obecně nepřímého) předchůdce dolovacího uzlu se řeší pomocí algoritmu backtracking. Hledání zdroje dat pomocí algoritmu backtracking je momentálně zbytečným zesložitěním (jelikož v aktuální verzi je zdroj dat přímým předchůdcem dolovacího uzlu), avšak jedná se o zobecnění pro eventuální rozšíření topologie dolovacího grafu.
- Pokud se v grafu vyskytuje uzel, který reprezentuje výstup (formou tabulky, XML soubor), je zjištěno, zda-li má jakožto předchůdce dolovací uzel. Opět je použit algoritmus backtracking.
- Pro každý uzel je dáno omezení na maximální počet vstupů a výstupů následující *tabulkou 6.1*.

Uzel	Max. počet vstupů	Max. počet výstupů
Transakční zdroj	0	1
Multidimenzionální zdroj	0	1
Asociační pravidla	1	2
Výstup – tabulka	1	0
Výstup – XML soubor	1	0

Tabulka 6.1: Omezení uzlů na maximální počet vstupů a výstupů

- Každý uzel musí mít nastaveny všechny povinné parametry. Pro zdroj dat je to určení zdrojových tabulek s daty. Pokud je u asociačních pravidel nastavena specifikace pravidla, je nutné nastavit levou a pravou stranu tohoto pravidla.

6.4 Zdroj dat

Jako zdroj dat pro dolovací proces lze využít **transakční** nebo **multidimenzionální** formu.

6.4.1 Transakce

V kapitole *4.1 Základní pojmy*, zabývající se asociačními pravidly, byl zmíněn pojem transakční zdroj dat. Transakci lze chápat jako množinu hodnot daného atributu opatřené jednoznačným ID, tzv. TID. *Tabulka 6.2* reprezentuje základní podobu transakcí. Například pro paralelu reprezentace nákupu zboží v supermarketu se TID jeví jako pořadové číslo nákupu provedené v daném zkoumaném období a položky transakce jsou jednotlivá nakoupená zboží.

TID	Položky transakce
T100	I1, I4, I5
T200	I1, I5
T300	I2, I5

Tabulka 6.2: Příklad transakční podoby dat

Pro databázovou reprezentaci dat se však jeví jako účelná jiná podoba vyjádření transakcí. Program předpokládá, že se data v databázi nachází dle podoby *tabulky 6.3*.

TID	Položky transakce
T100	I1
T100	I4
T100	I5
T200	I1
T200	I5
T300	I2
T300	I5

Tabulka 6.3: Reprezentace transakcí v databázi

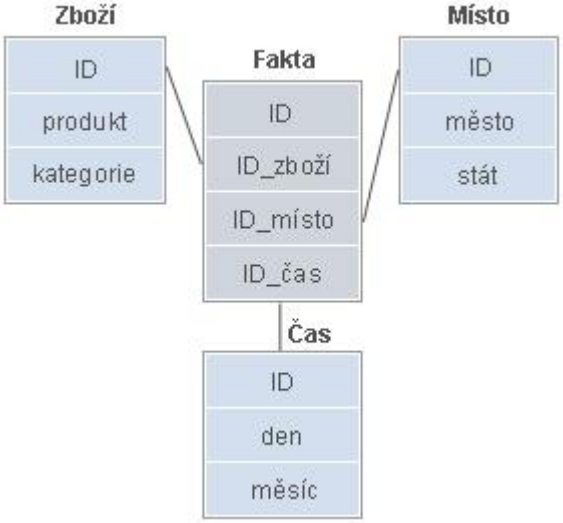
Pro dolovaní z transakční databáze je tedy nutné určit název tabulky a sloupce, ve kterém se nachází transakční identifikátory. Po uživatelském přihlášení do své databáze je nabídnut seznam tabulek, které má uživatel ve svém databázovém schématu. Je nutné, aby tabulka měla dva sloupce. Jeden sloupec je uvažován jako identifikace transakce (TID), druhý sloupec jako datový (položky transakce). Uživatel volí, který sloupec je identifikační. Se všemi hodnotami je dále nakládáno jako s řetězci (data mohou být v databázi tedy uložena v libovolné nebinární podobě). Možným doplněním k nabídnutým tabulkám uživatele by bylo taktéž nabídnutí databázových pohledů (View), které má uživatel ve svém schématu vytvořeny.

6.4.2 Multidimenzionální hvězda

Provádět dolování asociačních pravidel nad jedním atributem je omezující. Pokud je třeba hledat frekventované vzory nad různými položkami (například místo bydliště, čas nákupu, položky nákupu), je nutné mít k tomu potřebný datový aparát. *Kapitola 3.3* se zabývala *multidimenzionálním datovým modelem* datových skladů a byly specifikovány různé přístupy reprezentace multidimenzionálních dat: **hvězda**, **sněhová vločka** a **souhvězdí**. Pro účely reprezentace dat pro tvorbu **hybridně-dimenzionálních asociačních pravidel** byla zvolena **hvězda.**

Hvězda se skládá z **tabulky faktů** a **tabulek dimenzí.** Tabulka faktů obsahuje jednak data metrická (zpravidla měrné veličiny, například cena zboží, počet prodaného zboží) a jednak odkazy na záznamy v dimenzních tabulkách. Metrická data faktové tabulky nejsou v programu v procesu dolování uvažována. Tabulky dimenzí obsahují kontext, ve kterém jsou fakta prezentována a zkoumána. Reprezentují trvalejší charakteristiku entit, které do procesu vstupují. A právě tato data budou pro dolování relevantní.

V programu uživatel po přihlášení do databáze vybírá (jsou opět nabídnuty všechny tabulky, které má uživatel ve svém databázovém schématu) tabulku faktů a tabulky dimenzí. V tabulce faktů se vybírají cizí klíče, které jsou mapovány na privátní klíče do tabulek dimenzí. Metrické hodnoty tabulky faktů nejsou v aplikaci uvažovány. Tato metrická data jsou spíše užívána při statistických přehledech a OLAP dotazech. Dále uživatel provádí restrikci nad atributy dimenzních tabulek, čímž určuje, které atributy budou pro tvorbu asociačních pravidel podstatné. Pokud se zvolí jediný atribut, výsledek asociačních pravidel bude shodný, jako by byla data uložena v transakčním schématu a provádělo se dolování nad tímto schématem. Důvod tvorby (zdánlivě zbytečně) transakčního zdroje je efektivnější zpracování dat pro transakční účel. Příklad datové hvězdy pro analýzu nákupu je na *obrázku 6.4*.



Obrázek 6.4: Reprezentace datové hvězdy analýzy nákupu

Na *obrázku 6.5* lze pak vidět tabulkovou reprezentaci této hvězdy v databázi, ve které program vstupní data očekává. Jelikož se jedná o vyjádření dat formou hvězdy, dochází k redundanci dat. Položka *ID* ve faktové tabulce značí identifikaci konkrétního zkoumaného elementu (například konkrétní nákup). Tato položka se obecně může v tabulce vyskytovat se shodnou ID hodnotou vícekrát (nejedná se o privátní klíč tabulky), čímž se zajistí, že daný element může nabývat pro jistý

atribut (v asociačních pravidlech se používá pojem *predikát*) více hodnot. Tímto vznikají **hybridně**-dimenzionální asociační pravidla. Z důvodu reprezentace dat hvězdou však vzniká, jak již bylo zmíněno, značná redundance. Alternativou k hvězdě se nabízí sněhová vločka, kde lze redundanci zabránit. Důvod zvolení hvězdy bylo prioritou rychlejšího získávání dat z databází, což je u dolování zásadní aspekt. Možností tvorby dat ve formě sněhové vločky se jeví jako vhodné rozšíření současné práce.

Zboží

ID	produkt	kategorie
1	banán	ovoce
2	hruška	ovoce
3	jahody	zelenina

Místo

ID	město	stát
1	Brno	ČR
2	Brno	ČR
3	Holešov	ČR

Čas

ID	den	měsíc
1	pondělí	srpen
2	úterý	srpen

Fakta

ID	ID_zboží	ID_místo	ID_čas
1	1	1	1
1	2	1	1
2	1	3	2
2	2	3	2
2	3	3	2

Obrázek 6.5: Tabulky, které v databázi reprezentují hvězdu.

Na data lze pak v programu pohlížet jako na množinu transakcí, přičemž každý predikát může nabývat více hodnot. Každý predikát je v programu reprezentován jako *hashovací množina* (*HashSet*), což zajišťuje vyhledávání těchto položek v obecně konstantním čase. Pro úsporu paměti se jeví vhodné použít *seznam* (*Vector*, *ArrayList*), jehož časová složitost pro vyhledávání je však lineární. Z pohledu časové úspory byla tedy použita *hashovací množina*. Pokud by však predikáty nabývaly velmi málo různých hodnot, bylo by z pohledu kompromisu časová / paměťová složitost vhodnější použít *seznam*. Výsledné hybridně-dimenzionální transakce, které vznikly z datové reprezentace hvězdy z tabulek *obrázku 6.5*, jsou v *tabulce 6.4*.

ID	produkt	kategorie	město	stát	den	měsíc
1	banán, hruška	ovoce	Brno	ČR	pondělí	srpen
2	banán, hruška, jahody	ovoce, zelenina	Holešov	ČR	úterý	srpen

Tabulka 6.4: Ukázka reprezentace hybridně-dimenzionálních transakcí, se kterými je v programu pracováno.

6.5 Asociační pravidla

Dolování asociačních pravidel je stěžejní funkční částí aplikace. Implementační kostra byla realizována dle popisu kapitoly *4 Algoritmizace asociačních pravidel*. V aplikaci byly implementovány algoritmy **Apriori** a **FP-strom**. Každý algoritmus se realizoval se specifikací

na dolování dat z transakčního a multidimenzionálního zdroje (hvězda). Je možné provádět dolování *offline* či *online*, což značí, že jsou veškeré záznamy z databáze načteny do operační paměti a je nad němi prováděno hledání asociačních pravidel (*offline*), nebo jsou pravidla dle potřeby postupně načítána do operační paměti (*online*). Samozřejmostí je nastavení základních vlastností *podpory* a *spolehlivosti*. Dále lze specifikovat tvar asociačního pravidla dle výskytu predikátů a omezení na počet těchto predikátů. Následující odstavce popisují některé společné implementační vlastnosti všech algoritmů.

Položky, které jsou čteny z databáze a jsou z nich tvořeny frekventované množiny a následně asociační pravidla, mohou být libovolného nebinárního datového typu. Implementované algoritmy pracují nad řetězcovými typy, a proto jsou čtené položky automaticky převáděny na řetězce. Je tomu tak provedeno z důvodu obecnosti programu. Tímto však dochází ke zvýšení času provádění algoritmů pro menší datové typy položek (například *tinyint, smallint*), kdy se může časové zpoždění projevit relativně značně. Optimalizací by se jevilo implementování algoritmů vícekrát pro různé datové typy v databázi. Například pro Oracle typ *tinyint* se jeví jako vhodný v algoritmech používat typ *byte* a přizpůsobit tomu jednotlivé metody těchto algoritmů (porovnání, řazení, ..).

Pokud se pracuje nad menší množinou dat, je vhodné načíst veškerá data do operační paměti počítače, kde se provádí dolování (*offline* dolování). V tomto režimu dochází k načtení veškerých dat z databáze pouze jednou. V případě, kdy je dat mnoho, je z důvodu zahlcení paměti vhodnější načítat data z databáze v momentě potřeby algoritmu (*online* dolování). To v obecném případě ale znamená, že jsou shodná data načítána vícekrát (což je u algoritmu Apriori dosti znát). *Offline* režim je oproti *online* režimu obecně rychlejší, zvláště pak v případě, kdy je Oracle server dostupný s velkým síťovým zpožděním. Při čtení dat z databáze se používá technologie JDBC.

Pokud dochází ke čtení dat z transakčního zdroje, jsou přečtená data, která tvoří jednotlivé prvky transakce, uchována v kolekci *Vector<String>*. Je-li zdrojem dat multidimenzionální hvězda, jsou aktuálně čtená data uchována v typu *Set[][]*, kde řádky tvoří jednotlivé dimenze, sloupce tvoří jednotlivé atributy/predikáty těchto dimenzi a prvky množiny jsou jednotlivé hodnoty těchto predikátů. Položky typu *Set[][]* si však lze představit jako transakci, kde jednotlivé hodnoty predikátů tvoří jednotlivé prvky transakce.

6.5.1 Implementace Apriori

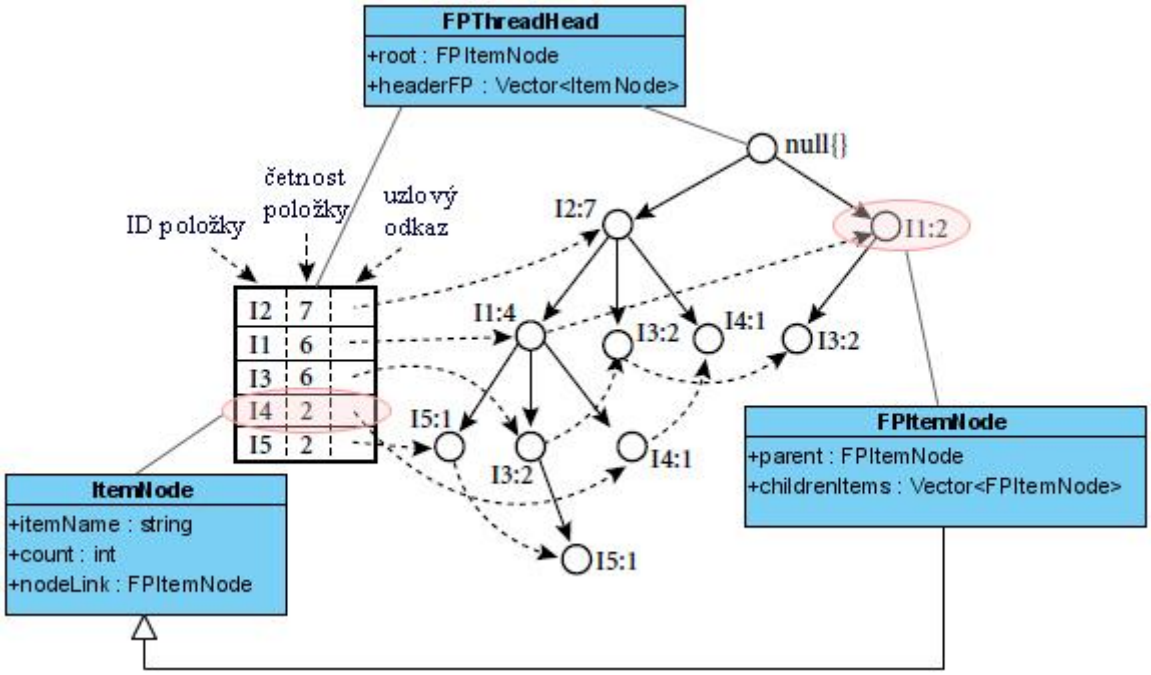
Implementace algoritmu Apriori byla provedena plně v souladu s kapitolou *4.2.2 Apriori*. Jednotlivé frekventované množiny o všech možných mohutnostech jsou uchovávány v kolekci *Vector* s prvky datového typu *String*. Sémanticky byla tedy množina změněna na seznam, a to z důvodu uchování těchto prvků v seřazené posloupnosti. Vysvětlením je to, že manipulace se seřazenými frekventovanými položkami je časově efektivnější, zvláště pak při tvorbě frekventovaných množin z frekventovaných množin o menší mohutnosti. Zbylé relevantní, implementační informace byly již zmíněny v předešlé kapitole a následná tvorba pravidel z těchto frekventovaných množin je popsána v kapitole *6.5.3 Tvorba pravidel*.

Balíčky, ve kterých je implementace algoritmu Apriori, jsou následující (v rámci hlavního balíčku *net.pumprla.dataminer*): *association, association.transaction, association.multidimensional, association.transaction.apriori , association.multidimensional.apriori*.

6.5.2 Implementace FP-stromu

Implementační kostra algoritmu FP-stromu byla provedena v souladu s kapitolou *4.2.3 FP-strom*. Jelikož je tento algoritmus oproti Apriori komplexnější a byly provedeny některé kroky, které nejsou v nástinu algoritmu v kapitole *4.2.3 FP-strom* zmíněny, bude tento algoritmus popsán, oproti Apriori, podrobněji.

Pro následující popis bude sloužit *obrázek 6.6*, kde se nachází již vytvořený FP-strom se seznamem frekventovaných jednoprvkových množin, které jsou se stromem provázány, a třídní popis, který postihuje základní prvky této stromové struktury.



Obrázek 6.6: FP-strom s návazností na třídy programu. Obrázek převzat a upraven z [1].

Algoritmus FP-strom má výhodnou vlastnost, že prochází databázi pouze dvakrát. Nejdříve se zjistí četnost výskytu všech položek v databázi (první čtení všech položek databáze). Položky, které splňují minimální podporu, utvoří seznam a tento seznam je seřazen klesajíc, nechť je nazván *L*. Vytvoří se proměnná *fpTreeHead* typu *FPTreeHead* uchovávající stromovou strukturu, a to jednak seznam všech frekventovaných položek, které jsou seřazeny klesajíc, a samotný FP-strom. Seřazený seznam *L* je použit v proměnné *fpTreeHead* jako seznam frekventovaných položek provázaných se stromovou strukturou, jedná se o *Vector* položek typu *ItemNode*. Klesající posloupnost těchto položek má důvod zefektivnění procesu dolování, kdy se nejdříve zpracovávají méně frekventované položky a při následném zpracování více frekventovaných položek nejsou již tyto méně frekventované uvažovány – stromová struktura se prochází zespodu.

Následuje tvorba samotného FP-stromu – metoda *createFPTree()* ve třídě *FPTree* balíčku *transaction.fptree*, respektive ve třídě *MultiFPTree* balíčku *multidimensional.fptree*. Databáze se prochází napodruhé (a naposledy), přičemž dochází ke zpracování transakcí (pro multidimenzionální zdroj se dají data též považovat za transakce). Každá transakce se porovnává se seznamem *L* (již uložený seznam všech frekventovaných položek). Pokud se prvek nacházející se v transakci vyskytuje i v seznamu *L*, je dále použit a zařazen do nového seznamu. Jelikož dochází k postupnému porovnání

položek se seznamem *L*, kde je klesající tendence četnosti výskytu, je nově vznikající seznam též klesající. To je pro tvorbu stromu důležité, jelikož se nejčetnější položky vyskytují blíže ke kořeni stromu. FP-Strom je v cyklu přes všechny transakce vytvářen dle popisu kapitoly *4.2.3 FP-strom*, algoritmu *Konstrukce FP-stromu*, části *(b)*. Takto vytvořený FP-strom je uchován, spolu se seznamem *L*, v proměnné *fpTreeHead*.

Nyní následuje proces získávání frekventovaných množin – metoda *miningFPTree()* ve třídě *FPTree* balíčku *transaction.fptree*, respektive ve třídě *MultiFPTree* balíčku *multidimensional.fptree*. Metoda *miningFPTree()* je volána rekurzivně. Nejdříve je zjištěno, zda-li není strom tvořen pouze seznamem – pokud ano, všechny kombinace tohoto seznamu tvoří potenciální frekventované množiny. Pro zjištění všech kombinací jsou vypočítány všechny indexy, které značí prvky, jež tyto kombinace tvoří. Výpočet indexů je popsán v následující kapitole *6.5.3 Tvorba pravidel*, kde indexy slouží pro výběr prvků levé strany asociačního pravidla. Právě vypočítané indexy jsou uchovány pro potenciální znovupoužití a tím dochází ke zrychlení procesu dolování. Pokud strom není tvořen pouze seznamem, je zpracován dle popisu kapitoly *4.2.3 FP-strom*, algoritmu *dolování v FP-stromu*.

V průběhu získávání frekventovaných množin dochází k dodatečné kontrole na potenciální omezení, které může uživatel prostřednictvím aplikace zadat (omezení na velikost pravidla, hodnotu predikátů).

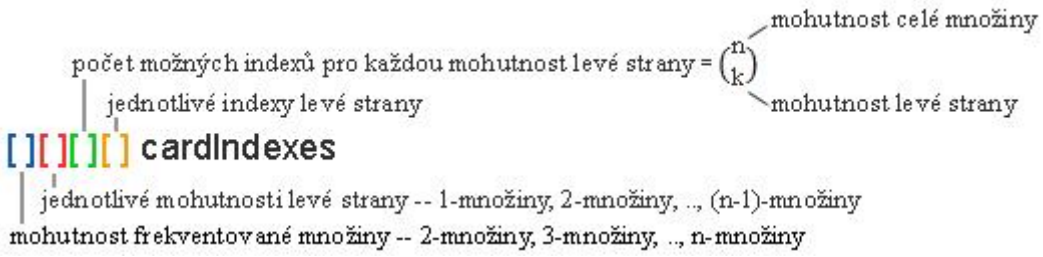
Záznamy seznamu klesajících frekventovaných položek, které jsou provázány se stromem, jsou reprezentovány třídou *ItemNode*, jež uchovává název položky, se kterou je svázána, počet výskytů těchto položek ve stromu a první výskyt této položky. Uzly FP-stromu jsou reprezentovány třídou *FPItemNode*, jež dědí z třídy ItemNode, a je dodatečně uchováván odkaz na otcovský uzel a odkazy na následující uzly ve stromu (synovské uzly).

Balíčky, ve kterých je implementace algoritmu FP-strom, jsou následující (v rámci hlavního balíčku *net.pumprla.dataminer*): *association, association.transaction, association.multidimensional, association.transaction.fptree , association.multidimensional.fptree*.

6.5.3 Tvorba pravidel

Jelikož jsou jednotlivé algoritmy (Apriori, FP-strom, ..) specifické pro získávání frekventovaných vzorů, je postup tvoření asociačních pravidel z těchto vzorů pro všechny algoritmy shodný. Respektive dochází k odlišnostem pro transakční a multidimenzionální zdroj dat.

Před samotnou tvorbou asociačních pravidel z jednotlivých frekventovaných množin dochází k předvypočítání veškerých potřebných indexů pro výběr prvků z těchto množin, které budou tvořit levou část pravidla. Indexy se počítají pro každou mohutnost frekventované množiny zvlášť. Levá část pravidla má maximální mohutnost o 1 menší než mohutnost celé množiny pravidla. Následně se jednotlivé indexy levé strany počítají pro $\binom{n}{k}$ možností, kde *n* je mohutnost celé frekventované množiny a *k* je mohutnost levé strany výsledného pravidla. Předchozí popis je demonstrován *obrázkem 6.7*, kde je zobrazena proměnná, která v programu uchovává jednotlivé indexy prvků frekventovaných množin levé strany pravidla.



Obrázek 6.7: Uchování indexů pro výpočet asociačních pravidel.

Indexy jsou použity pro výběr prvků levé strany asociačního pravidla z frekventovaných množin a následně je zjištěno, jaká je hodnota podpory pro levou stranu pravidla. Tato hodnota není znovu počítána a je pouze dohledána, jelikož již byla spočítána při tvorbě frekventovaných množin. Tímto se určí, zda-li pravidlo splňuje minimální spolehlivost.

Při tvorbě asociačních pravidel z multidimenzionálního zdroje dat je nutné uvažovat potenciální omezení na pravidlech, které uživatel mohl zvolit. Pokud je vynucena minimální mohutnost pravidel, jsou indexy pro položky levých stran počítány až od jisté hodnoty (minimální počet predikátů v pravidle). Navíc dochází ke kontrole počtu jednotlivých položek, které budou tvořit predikáty asociačního pravidla.

Balíčky, ve kterých je implementace asociačních pravidel z frekventovaných množin, jsou následující (v rámci hlavního balíčku *net.pumprla.dataminer*): *rules* (*AssociationRule*, *AssociationRules*), *multidimensional.rules* (*MultiFilterRule*, *RuleCreator*), *transaction.rules* (*RuleCreator*).

6.6 Výstup dolovacího procesu

Jednou z etap dolovacího procesu je vizualizace a reprezentace výsledků. Jelikož je program zaměřen na asociační pravidla, jako hlavním mechanismem reprezentací výsledků byla zvolena forma tabulky. Jak již bylo zmíněno v kapitole *6.2 Uživatelské rozhraní*, veškerá výsledná asociační pravidla jsou automaticky uložena do tabulkové reprezentace. Pro obrovské množství pravidel dochází k nutnému alokování většího množství paměti (výsledky je nutné převést do tabulkové formy – nutná nová paměť), což může u výsledku několika desítek milionových pravidel způsobit přetečení paměťové části *heap* (halda). Při testování byly cíleně nastaveny parametry na vydolování obrovského množství pravidel (bylo vydolováno cca 10 000 000 pravidel) a při požadavku tabulkového zobrazení došlo k přetečení paměti *heap*. Je však nepraktické dolovat několik desítek milionů asociačních pravidel, ze kterých nelze usuzovat (téměř) žádný reálný závěr. Při nižším počtu výsledných asociačních pravidel (maximálně jednotky milionů, což je též neúčelně mnoho) již k přetečení paměti heap nedocházelo.

Alternativou k zobrazení výsledků asociačních pravidel ve formě tabulky je XML soubor. Účel tvorby XML souboru je možnost relativně pohodlného využití výsledných dat v externích programech. XML soubor obsahuje metadata pro dolovací proces a samotná výsledná asociační pravidla. Na tvorbu dokumentu XML nebylo použito žádného programového Java rozhraní (SAX) či modelu (DOM, JDOM, ..), jelikož tyto zpravidla (DOM, JDOM, ..) uchovávají složitý XML model náročný na paměť a pro větší množství asociačních pravidel by došlo k alokování velkého množství paměti. Dochází tedy k manuální tvorbě XML struktury uchované v řetězcových proměnných

(StringBuffer) a následný zápis do souboru. DTD pro XML soubor dolovacího procesu s výsledky pro multidimenzionální zdroj je následující (ukázka XML souboru předpisu procesu dolování a vydolovaných dat je v *příloze B*):

```
<!ELEMENT miningProcess (metadata, associationRules)>
<!ELEMENT metadata (algorithmProcess, associationParameters, onlineDB, tableStarScheme, constraints)>
<!ELEMENT algorithmProcess (kind, algorithm)>
<!ELEMENT kind (#PCDATA)> <!ELEMENT algorithm (#PCDATA)>
<!ELEMENT associationParameters (support, confidence)>
<!ELEMENT support (#PCDATA)> <!ELEMENT confidence (#PCDATA)>
<!ELEMENT onlineDB (#PCDATA)>
<!ELEMENT tableStarScheme (factTable, dimensionalTables, tableStarScheme)>
<!ELEMENT factTable (#PCDATA)>
<!ELEMENT dimensionalTables (dimensionalTable+)>
<!ELEMENT dimensionalTable (name, usedColumns)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT usedColumns (column+)>
<!ELEMENT column (#PCDATA)>
<!ELEMENT constraints (specificRule|ruleConstraints)>
<!ELEMENT specificRule (leftSide, rightSide)>
<!ELEMENT leftSide (column+)> <!ELEMENT rightSide (column+)>
<!ELEMENT column (#PCDATA)> <!ATTLIST column count CDATA #REQUIRED>
<!ELEMENT ruleConstraints (maxItems, cantTogether)>
<!ELEMENT maxItems (#PCDATA)>
<!ELEMENT cantTogether (pair*)>
<!ELEMENT pair (column1, column2)>
<!ELEMENT column1 (#PCDATA)> <!ELEMENT column2 (#PCDATA)>
<!ELEMENT associationRules (associationRule*)>
<!ELEMENT associationRule (left, right)>
<!ELEMENT left (#PCDATA)> <!ELEMENT right (#PCDATA)>
```

6.7 Správa projektů

V aplikaci je možné projekty ukládat a znovu ustanovovat. Projektu je po celou dobu přiděleno jednoznačné jméno, které zadává uživatel při jeho tvorbě. Tímto jménem je projekt identifikován a je kontrolováno, zda nejsou v systému souběžně dva projekty se shodným názvem. Perzistence projektu se provádí ve formátu XML (ačkoliv má soubor koncovku **.sav*). Jelikož nedochází k ukládání asociačních pravidel, je nyní při analýze uloženého projektu při jeho znovu ustanovení použit objektový model DOM, který je součástí standardní edice Javy (nemělo by již dojít k problému alokace velkého množství operační paměti).

V souboru se ukládá jednak vizuální rozmístění a propojení komponent/uzlů dolovacího grafu, jednak samotné nastavení těchto komponent. Pokud je uživatel připojen k databázi, jsou v souboru uloženy uživatelské připojovací údaje k této databázi. Heslo není možné uchovávat v otevřené podobě a je proto použito symetrické šifrování AES s 128-bitovým klíčem (balíček *utils*, třída *CipherText,* viz *obrázek 5.2*). Z důvodu blokového módu AES šifrování dochází k zarovnání hesla

na násobek 128-bitové délky. Nevýhodou momentálního řešení je uchovávání privátního klíče ve zdrojovém kódu, což může být nasnadě některým útokům. Bylo by zřejmě vhodnější, kdyby uživatel při požadavku na uložení projektu tento privátní klíč osobně zadával (respektive část klíče).

7 Testování a praktické nasazení aplikace

Tato kapitola zahrnuje variantní testování a je proveden rozbor praktického nasazení systému. Jsou jednak získávána asociační pravidla, čímž dochází k ověřování správnosti a získávání nových informací, a jednak je zjišťována efektivnost jednotlivých algoritmů pro různá nastavení charakteristických parametrů. Dochází též k testování online/offline přístupu dolování. Dále je rozebíráno řešení případové studie za použití datového skladu a vyvinutého systému s multidimenzionálním zdrojem dat. Na závěr kapitoly je uvedeno srovnání se systémy SAS a Oracle Data Miner.

7.1 Návštěvnost webových stránek

V rámci této kapitoly dojde k získávání znalostí z návštěvnosti internetového obchodu a analýze výkonnostních testů.

7.1.1 Získávání nových znalostí

Jako jedno z možných použití aplikace systému pro získávání asociačních pravidel je sledovanost návštěvnosti webových stránek. Například pro majitele internetového obchodu se jeví účelné mít přehled o tom, které kategorie nebo zboží si návštěvník webových stránek často společně prohlíží (nebo které zboží kupuje). Pokud si uživatel následně některé zboží při nákupu prohlíží, jsou mu nabídnuta jako vhodná kombinační zboží právě ta, která spadají do kategorií, jež jsou často s pravě prohlíženým zbožím/kategorií zobrazována.

Následující úloha si klade za cíl zjistit, které kategorie internetového obchodu jsou uživateli často společně navštěvovány. Sledovaný internetový obchod, který jsem pro jistou firmu vytvořil, je na adrese www.glamourworld.cz. Je zde nabízeno oblečení a módní doplňky glamour charakteru (Ačkoliv je na stránkách i erotické oblečení, není zde žádný sexuální podtext!). Zboží je řazeno do kategorií a subkategorií a právě tyto jsou zkoumány.

Počet subkategorií je 41 a jsou zastřešeny 8 kategoriemi. Návštěvnost těchto subkategorií internetového obchodu byla sledována dva týdny a stránky byly navštíveny 1275 různými zákazníky. Tito zákazníci dohromady zobrazili 5642 subkategorií. Pokud zákazník navštívil stejnou subkategorii vícekrát, byla počítána jednou. Zkoumaná data ve formě skriptu *strankySubkategorie.sql* se nachází na přiloženém CD nosiči v adresáři *Data*.

Dle skriptu je vytvořena tabulka *Stranky* se sloupci *TID* a *subKat*. *TID* je identifikace návštěvníka a jedná se o transakční ID. Jelikož jsou data v čistě transakčním charakteru, jako zdroj dat se vybere *tool* s názvem *Transaction Database*. V tomto *toolu* dojde k nastavení právě popsaného zdroje dat. Další povinný tool *Association Rules* je nastavován experimentálně. Následují experimenty a analýza výsledků.

Vydolované výsledky budou využívat zkrácení hlavních kategorií pro úspornější zobrazení asociačních pravidel: *Spodní prádlo* = SPOD, *Erotické spodní prádlo* = EROT, *Punčochy* = PUN, *Móda* = MOD, *Rukavičky* = RUK, *Šperky* = SPE, *Noční prádlo* = NOC, *Plavky* = PLA. Za asociačním pravidlem bude vždy zobrazeno [podpora, spolehlivost].

Následující 3 asociační pravidla mají nejvyšší podporu (ze symetrického pravidla bylo vybráno to, které mělo vyšší spolehlivost):

EROT.erotické_kostýmy → EROT.komplety [15.87, 71.33]
EROT.pin-up-kostýmy → EROT.komplety [15.15, 67.14]
EROT.erotické_kostýmy → EROT.pin-up-kostýmy [14.35, 64.52]

Lidé tedy nejčastěji zobrazují položky, které se společně vyskytují v hlavní kategorii *Erotické spodní prádlo*. I mnoho ostatních zde neuvedených pravidel bylo právě z této kategorie.

Následující pravidla jsou vybrána s ohledem na potenciální zajímavost pro obchodníka (seřazeny dle podpory):

EROT.komplety → SPOD.kosilky [13.72, 40.86]
EROT.erotické-kostýmy → EROT.pin-up-kostýmy & EROT.komplety [11.48, 51.61]
NOC.noční-košilky → EROT.komplety [11.24, 61.84]
SPOD.košilky → NOC.noční-košilky [9.57, 42.70]
EROT.erotické-kostýmy → NOC.noční-košilky [9.25, 41.58]
EROT.erotické-kostýmy → NOC.noční-košilky & EROT.komplety [7.18, 32.26]
MOD.šaty → SPOD.tanga [6.94, 43.22]

Cíleně byla převážně vybírána pravidla, ve kterých se nenacházely všechny hlavní kategorie stejné. Lze totiž dedukovat, že zákazník často zobrazuje položky, jež se nachází ve společné hlavní kategorii. Bylo též bráno ohled na spolehlivost a byla preferována pravidla, kde je na levé straně právě jeden predikát.

Závěrem tedy bylo zjištěno, že návštěvníci internetového obchodu www.glamourworld.cz často zobrazují položky, které jsou společně v hlavní kategorii *Erotické spodní prádlo* a byla nalezena zajímavá pravidla, která implikují, že je přínosné zobrazovat u daných zboží jako doporučená zboží i ta, jež jsou z jiných hlavních kategorií (a ze kterých). Například pravidlo EROT.komplety → SPOD.kosilky [13.72, 40.86] je čtvrtým nejčetnějším pravidlem s ohledem na podporu, přitom jsou hlavní kategorie různé. Jak již bylo zmíněno, tyto výsledky lze použít pro nabídnutí vhodných kombinačních produktů pro návštěvníka internetového obchodu a tím zvýšit prodejnost.

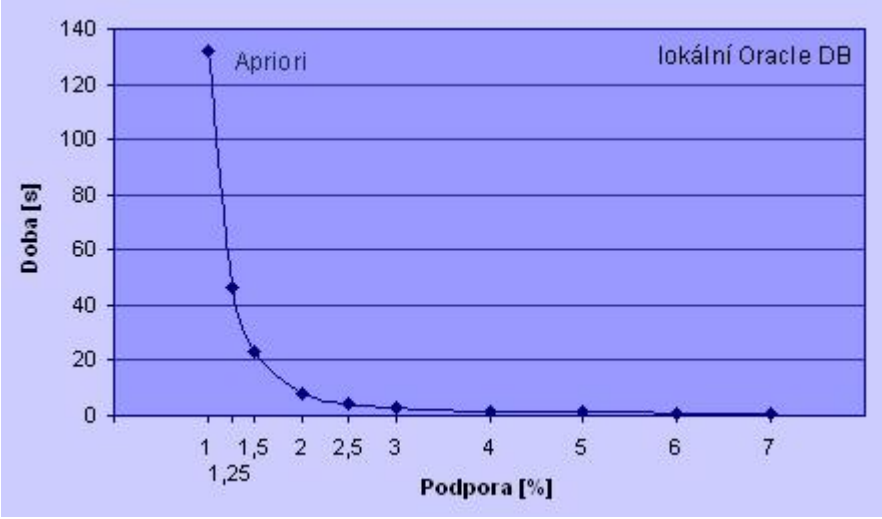
7.1.2 Výkonnostní testování

Data popsaná v předchozí kapitole byla podrobena výkonnostním testům. Nejdříve bylo 5642 transakčních záznamů použito pro zkoumání efektivnosti vykonávání algoritmů **Apriori** a **FP-strom**, které vytváří frekventované množiny. Získané hodnoty z těchto testů jsou viděny v *tabulce 7.1* Naměřené hodnoty jsou v sekundách, přičemž každý případ byl měřen třikrát a průměrován.

Podpora	7.00	6.00	5.00	4.00	3.00	2.50	2.00	1.50	1.25	1.00
Apriori	0.688	0.797	1.093	1.453	2.938	4.500	7.938	23.485	46.782	131.984
FP-strom	0.237	0.251	0.247	0.255	0.315	0.344	0.359	0.485	0.482	0.656
Nejdelší frekv. mn.	3	4	4	5	5	5	6	7	8	?
Počet pravidel	88	184	412	832	2820	5728	14028	64322	160 886	?

Tabulka 7.1: Porovnání výkonnosti algoritmů Apriori a FP-strom na základě změny podpory.

Jsou zde uvedeny časy algoritmů Apriori a FP-strom pro různé hodnoty vlastnosti podpory. Též jsou zmíněny počty pravidel, která vznikají při dané podpoře a stejné hodnotě spolehlivosti (největší možný počet pravidel pro danou podporu) a velikost nejdelší frekventované množiny – mohutnost množiny (otazníky v tabulce značí nezměřenou hodnotu pro obrovské množství pravidel, které nebylo možno uchovat v paměti v *offline* režimu). Databázový server s daty se nacházel na lokálním počítači. Zatímco doba zpracování algoritmu FP-strom je pro všechny uvedené hodnoty jen nepatrně odchýlena a dochází pouze k malému zpomalení, algoritmus Apriori vykazuje razantnější zpomalení s klesající hodnotou podpory. Při postupné změně podpory ze 7.00 na 1.00 (%) došlo u Apriori ke zpomalení času až 191x, zatímco u FP-stromu došlo ke zpomalení pouze méně než 3x. Tímto je FP-strom oproti Apriori přibližně 200x rychlejší při zpracování zkoumaných transakcí při podpoře rovno jedné. Pro názornější představu závislosti doby hledání frekventovaných množin vůči parametru podpory jsou naměřené hodnoty algoritmu Apriori vyneseny do grafu, viz *obrázek 7.1*. Jelikož je doba provádění algoritmu FP-strom vůči algoritmu Apriori velmi nízká, není pro přehlednost algoritmus FP-strom zobrazen na grafu.



Obrázek 7.1 Závislost času hledání frekventovaných množin vůči parametru podpory pro algoritmus Apriori.

I jiné testy též ukázaly, že algoritmus Apriori výrazně časově zaostává za FP-stromem, zvláště pak pro nízké hodnoty podpory (z grafu na *obrázku 7.1* lze vidět rychlý nárůst doby provádění pro menší hodnoty podpory). Přímý důvod vychází z popisu algoritmů v této diplomové práci. FP-strom uchovává relevantní data ve komprimované podobě (transakce, které obsahují většinu shodných prvků, ne-li všechny, jsou ve stromu sdíleny) a oproti Apriori prochází celý zdroj dat pouze dvakrát. Pokud by však byla zkoumána data, jež obsahují malé množství podobných transakcí,

FP-strom by dával mnohem horší výsledky, než ve právě prováděném případě. Důvodem by byla velmi rozsáhlá stromová struktura pro malý počet sdílených uzlů tohoto stromu.

Stejnému testu byla též podrobena aplikace *Oracle Data Miner*. Pro hodnotu podpory 4 až 7 (%) byly naměřeny časy přibližně 3 až 8 sekund. Tyto časy byly měřeny s velkou nepravidelností (pro konkrétní hodnotu podpory byla jednou změřena hodnota přibližně 3 sekundy, napodruhé 7 sekund, napotřetí 4 sekundy, ..). Pro hodnotu podpory nižší než 3% již nedocházelo ke zdárnému ukončení dolovacího procesu (po 30 minutách bylo dolování zrušeno) ani po několikanásobném restartu aplikace či databáze. Přibližná doba 3 sekund je pravděpodobně přisouzena úvodní režii, kterou *Oracle Data Miner* při každém dolovacím procesu provádí .

Následující test byl zaměřen na porovnání časů dvou přístupů čtení zdrojových dat. **Offline** režim uvažuje, že jsou veškerá data přečtena z databáze pouze jednou a jsou následně uchována v operační paměti. **Online** režim při každém požadavku na data čte tato z databáze. Výsledky jsou zobrazeny v následující *tabulce 7.2* (tabulka má mnoho sloupců, proto je složena ze dvou částí). Veškeré časy jsou opět v sekundách.

Podpora	15.00	12.00	10.00	8.00	7.00	6.00	5.00
Apriori:online (s)	42.609	38.672	65.359	63.062	59.890	94.547	91.954
Apriori:offline (s)	20.150	18.859	21.453	19.265	21.679	21.719	21.826
FP-strom:online (s)	40.516	42.391	41.312	41.297	41.843	39.172	42.750
FP-strom:offline (s)	21.172	19.578	18.031	20.500	21.438	19.787	20.629
Nejdelší frekv. mn.	2	2	3	3	3	4	4
Počet pravidel	4	8	20	52	88	184	412

Podpora	4.00	3.00	2.50	2.00	1.50	1.25
Apriori:online (s)	102.688	105.031	106.482	115.094	158.172	214.375
Apriori:offline (s)	22.039	23.840	24.638	28.331	42.924	67.582
FP-strom:online (s)	42.594	43.328	42.319	41.726	42.336	42.649
FP-strom:offline (s)	21.005	20.672	19.592	20.539	20.189	21.527
Nejdelší frekv. mn.	5	5	5	6	7	8
Počet pravidel	832	2 820	5 728	14 028	64 322	160 886

Tabulka 7.2: Porovnání online a offline režimu činnosti algoritmů. Databázový Oracle server je na FIT VUT Brno, rychlost připojení Internetu 2Mbit/s (17 hopů routerů).

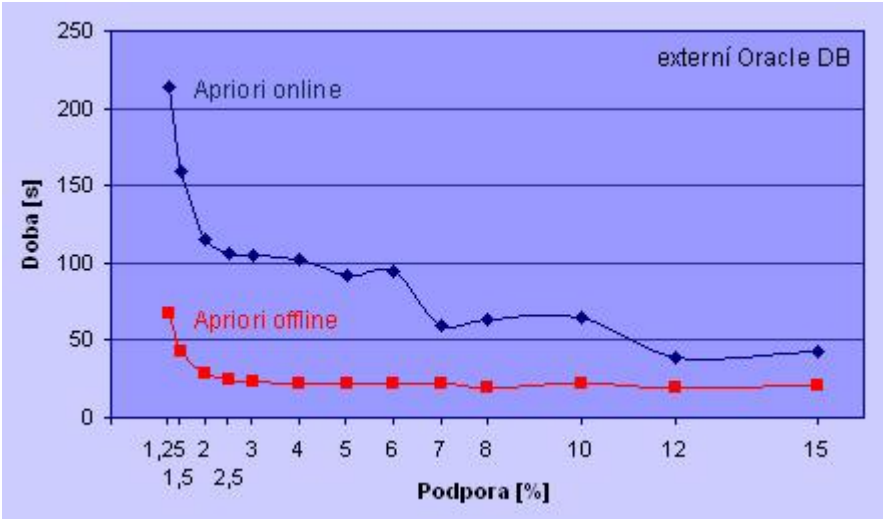
Zatímco v předchozím testu se databázový Oracle server se zdrojovými daty nacházel na lokálním počítači (tedy na shodném počítači s aplikací pro dolování), byla nyní data uchována na Oracle serveru na FIT VUT Brno. Data byla čtena internetovým připojením s rychlostí 2Mbit/s (17 hopů routerů na cestě lokální počítač – Oracle server). Samotné čtení všech dat, tedy 5642 transakčních záznamů, z databáze do lokálního počítače trvalo přibližně 18 až 21 sekund s případnými časovými odchylkami.

Jak bylo poukázáno v kapitole *6.5.2 Implementace FP-stromu*, pro algoritmus FP-strom je typické, že je nutné procházet databázi právě dvakrát (pokud jsou data v databázi) a relevantní data jsou uchována ve vytvořeném stromu. Proti tomu algoritmus Apriori vyžaduje čtení celé databáze

tolikrát, kolik je mohutnost nejdelší frekventované množiny. Zmíněné vlastnosti vysvětlují časy uvedené v *tabulce 7.2*.

Jak již bylo uvedeno v *tabulce 7.1*, FP-strom řešil všechny zmíněné případy *podpory* v čase menším než 1 sekunda. Při čtení dat z externího Oracle serveru je tedy téměř veškerá časová režie právě v procesu čtení dat z databáze. Pro *offline* režim se čtou data pouze jedenkrát, zatímco v *online* režimu dvakrát. Pokud je uvažována doba čtení veškerých dat z externí databáze v přibližném časovém rozpětí 18 až 21 sekund (s případnými časovými odchylkami), odpovídají časy v *tabulce 7.2* předpokladům algoritmu FP-strom. U algoritmu Apriori je celková přibližná doba běhu předpokládána: (*mohutnost_nejdelší_frekventované_množiny* * *doba_čtení_dat_z_DB* + *režie_algoritmu_Apriori*). Jak je vidět v *tabulce 7.2*, zatímco pro vyšší hodnoty *podpory* je dominantní režie čtení dat z databáze, pro nižší hodnoty *podpory* již dochází ke zvýraznění časové složky (zpomalení) samotným algoritmem Apriori.

Popsanou charakteristiku též znázorňuje graf na *obrázku 7.2*. Je zde opět zobrazen pouze algoritmus Apriori (online a offline režim), jelikož pro zkoumaný vzorek dat je téměř veškerá režie provádění algoritmu FP-stromu v procesu čtení dat z databáze – fluktuace grafů pro FP-strom by byla kolem hodnot 20 (offline) a 40 sekund (online).



Obrázek 7.2: Závislost času hledání frekventovaných množin vůči parametru podpory pro algoritmus Apriori v online a offline režimu. Databázový Oracle server je na FIT VUT Brno.

Tento test potvrdil charakteristické vlastnosti pro čtení dat z databáze jednotlivých algoritmů. Pokud zdrojová databáze neobsahuje velké množství dat a je umístěna na vzdálenějším serveru, je užitečné použít *offline* režim algoritmů (zvlášte pak u algoritmu Apriori). Opět bylo potvrzeno, že je účelnější při hledání frekventovaných množin a asociačních pravidel používat algoritmus FP-strom.

7.2 Multidimenzionální dolování s podporou datového skladu

V této kapitole bude demonstrováno použití datového skladu Oracle Warehouse Builder 11*g* za účelem přípravy multidimenzionálních dat pro aplikaci vytvořenou v rámci diplomové práce. Bude

se jednat o případovou studii, jak by se dala aplikace s podporou datového skladu prakticky využít a nebudou použita reálná data (není tedy nyní účelem hledat nové a skryté znalosti).

7.2.1 Řešený problém

Fiktivní obchodní řetězec FITmarket chce provést analýzu prodeje svého zboží za účelem zvýšení zisku. Zajímá se jednak o to, která zboží jsou často společně prodávána a v jakém období, a jednak kdo dané zboží kupuje.

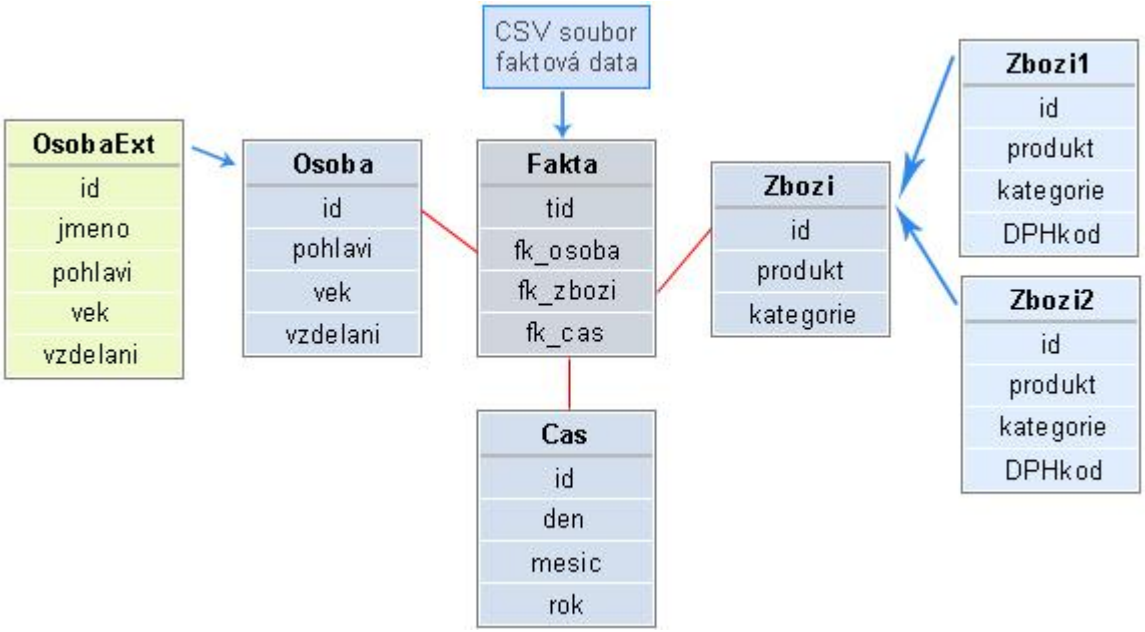
FITmarket vydává svým zákazníkům věrnostní kartičky, které lidé dostanou za vyplnění formuláře. Formulář obsahuje dotazy na jméno, pohlaví, přibližný věk (vybírá se interval), nejvyšší dosažené vzdělání a případný kontakt. Při platbě je na pokladně zaznamenáno identifikační číslo nákupu, identifikace osoby (číslo věrnostní kartičky), identifikace koupeného zboží a doba nákupu (pořadové číslo do tabulky variantních dnů/měsíců/roků). Program, který zaznamenává tyto údaje umožňuje výstup pouze ve formě CSV souboru (záznamy oddělené čárkou). Data o zboží jsou uložena ve více tabulkách (například dvě tabulky) na lokálním databázovém serveru. Data o uživatelích jsou uložena na externím databázovém serveru (může být i například Microsoft Database).

Obchodní řetězec FITmarket chce získané znalosti zužitkovat pro přeskládání regálů se zbožím, adaptace počtu nabízených zboží na dané období (možnost nabízet větší množství daného zboží v jistých dnech) a zasílání cílených letáků.

7.2.2 Realizace řešení

Řešením popsaného problému bude hledání vhodně zvolených asociačních pravidel. Vytvořený systém pro dolování asociačních pravidel umožňuje variantní specifikování hybridně-dimenzionálních pravidel a bude se jevit pro tento problém jako užitečný. Jelikož program očekává vstupní data ve formě hvězdy, bude nutné prvotní data nejdříve upravit. Pro úpravu dat je vhodné použít již několikrát zmíněný Oracle Warehouse Builder 11*g*.

Na *obrázku 7.3* je databázové schéma hvězdy, která je složena z faktové tabulky *Fakta* a dimenzionálních tabulek *Osoba*, *Cas* a *Zbozi*. Jsou zde též naznačeny požadované datové zdroje pro naplnění dimenzionálních tabulek.



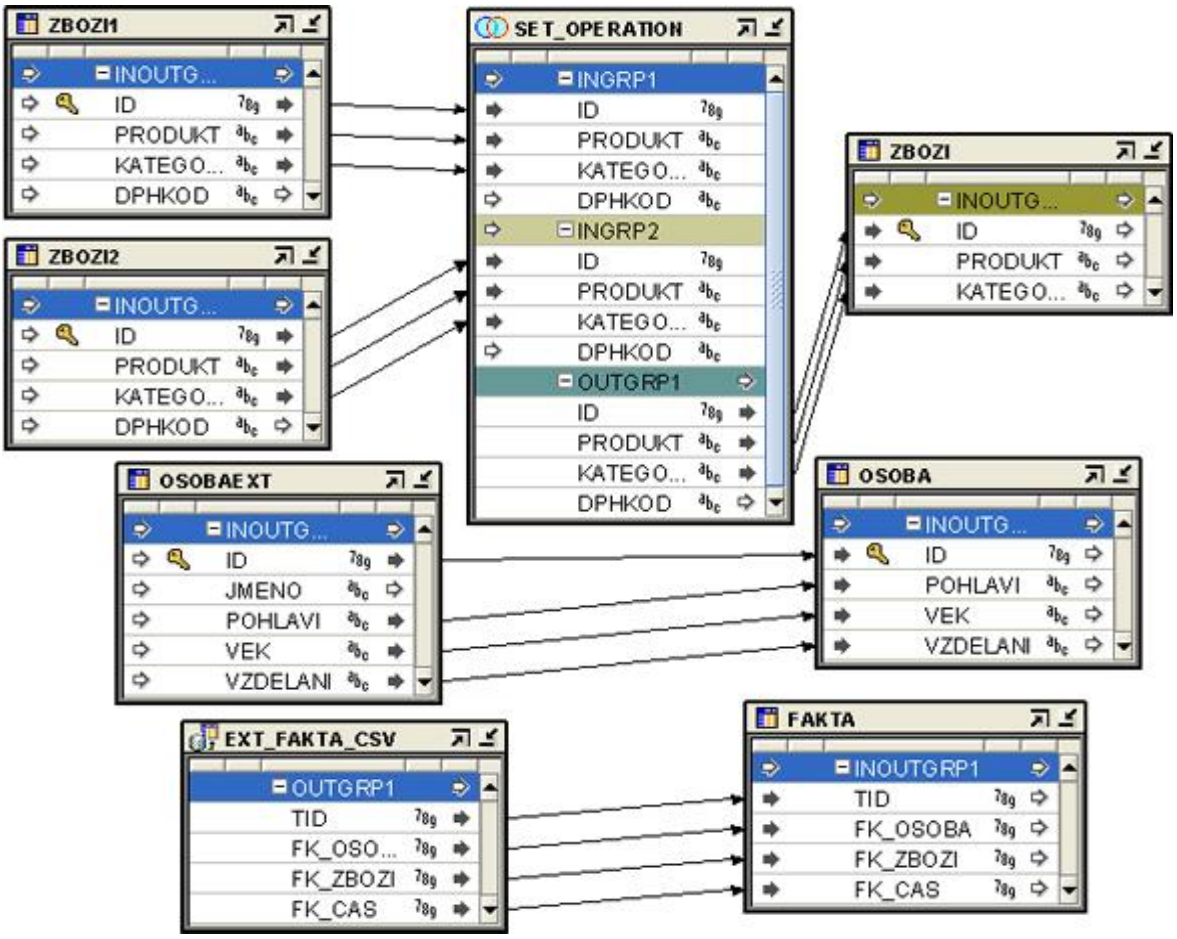
Obrázek 7.3: Databázové schéma hvězdy a dodatečných zdrojů dat pro analýzu FITmarket.

Nejdříve je nutné mít vytvořené lokální tabulky – faktová (*Fakta*) a dimenzionální tabulky (*Osoba*, *Zbozi*, *Cas*) a datové tabulky *Zbozi1* a *Zbozi2*. V reálném případě by tabulky *Cas*, *Zbozi1* a *Zbozi2* již v systému existovaly. K tomu slouží soubor *OracleWarehouse\market\local\database-create.sql*, jenž je uložen na přiloženém CD nosiči. Data k naplnění tabulek, která by v reálném systému již existovala, jsou v souboru *OracleWarehouse\market\local\database-insert.sql*. Externí data (tabulka *OsobaExt*), jež jsou uvažována, že se nachází na jiném databázovém systému, se vytvoří prostřednictvím skriptů *OracleWarehouse\market\external\database-create.sql a OracleWarehouse\market\ external \database-insert.sql*, a to na dostupné externí databázi.

Pro práci s datovým skladem je nutné vytvořit *Warehouse Builder Repository* a uživatele, který má k tomuto schématu přístup. Postup k vytvoření *Repository* a uživatele je uveden v *Příloze A*. Nadále v textu bude manipulace s datovým skladem popisována ve stručnosti, jelikož jsou jednotlivé kroky podrobněji zmíněny v rámci *přílohy A*.

Prostřednictvím programu *Design Center* bude nutné vyřešit tři problémy. Mapování externí databázové tabulky *OsobaExt* do lokální tabulky *Osoba*, mapování dvou tabulek *Zbozi1* a *Zbozi2* do tabulky *Zbozi*, a převést data z CSV souboru do faktové tabulky *Fakta*.

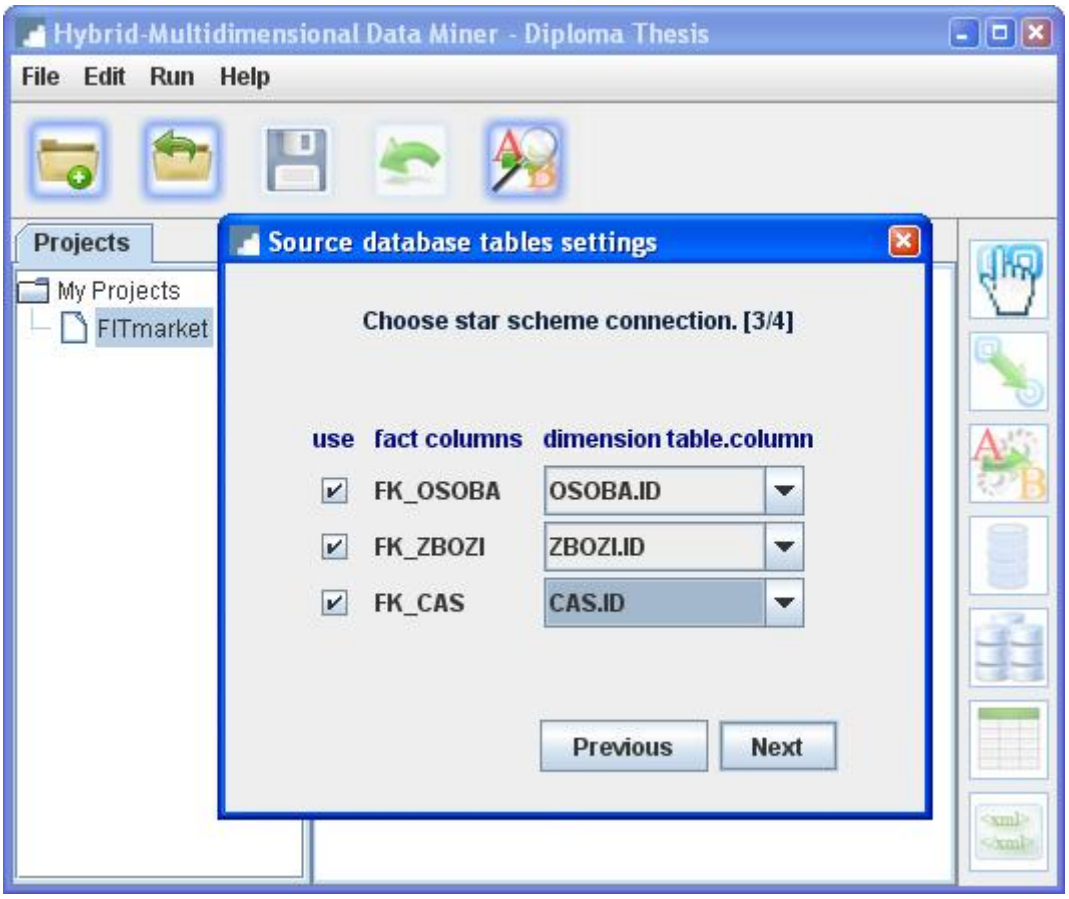
Nejdříve dojde k připojení cílového úložiště, nazvané například *lokálníDataDB*, a naimportují se tabulková metadata pro *Fakta*, *Osoba* a *Zbozi*. Jedná se o tabulky, které budou výstupní v ETL procesu. Nyní dojde k importu metadat faktového CSV souboru a vytvoření (opět v *lokálníDataDB*) metadat externí tabulky *extFaktaCSV*, jež je s tímto souborem provázána. Prostřednictvím *Control Center Manager* dojde k fyzickému vytvoření a naplnění externí tabulky *extFaktaCSV* (bude naplněna daty z CSV souboru). Dalším krokem je připojení zdrojového úložiště (shodné umístění databáze jak bylo uvedeno pro *lokálníDataDB*), nazvané například *lokálníZdrojDB*, kde se nachází tabulky *Zbozi1* a *Zbozi2* a provést jejich import tabulkových metadat. Poslední připojení zdrojového úložiště, nazvané například *externíDataDB*, je pro tabulku *OsobaExt*. Závěrem se vytvoří mapovací schéma a ETL proces, jehož znázornění je na *obrázku 7.4*.



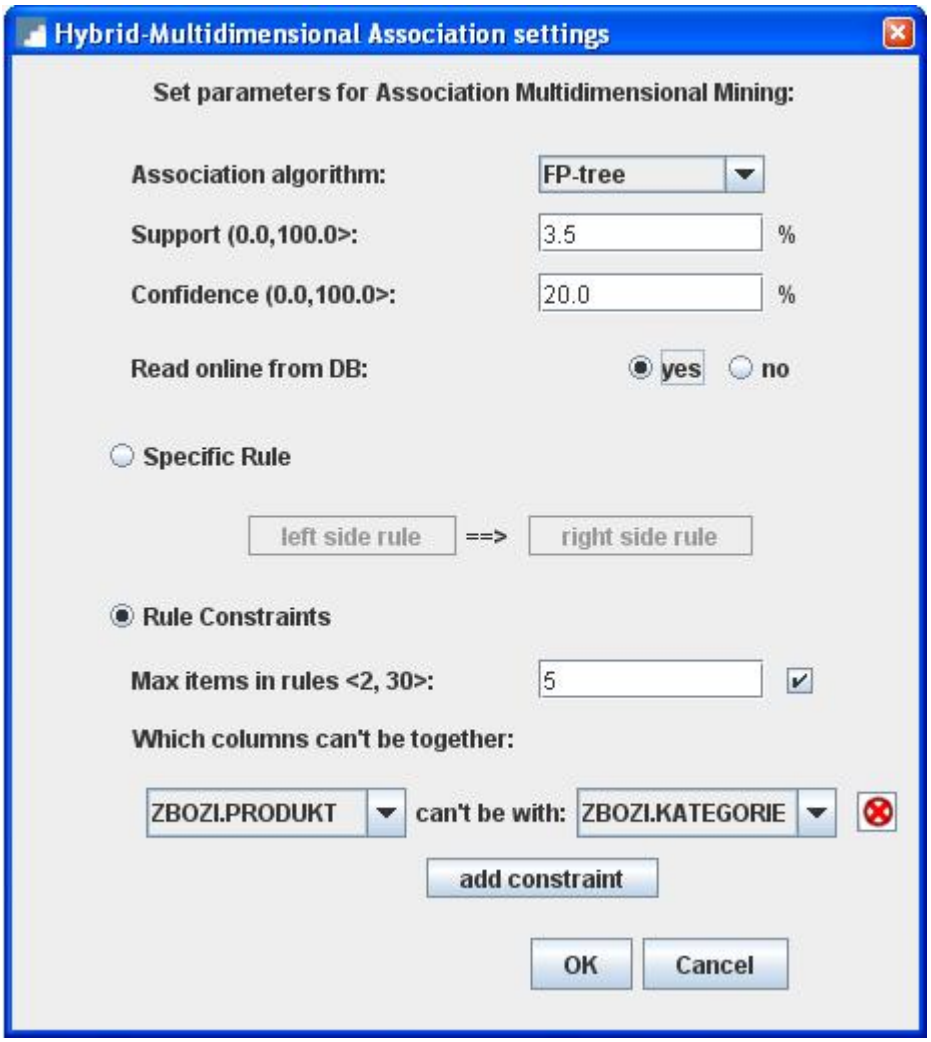
Obrázek 7.4: Návrh ETL procesu přípravy dat pro tvorbu hvězdy v Oracle Warehouse Builder 11g.

Z obrázku lze vidět, že pro ETL proces bylo dvakrát vytvořeno přímé mapování (přičemž jednou byla provedena restrikce nechtěného sloupce *jmeno* tabulky *osobaExt*) a jedno množinové sjednocení dvou tabulek *Zbozi1* a *Zbozi2* do tabulky *Zbozi*. K fyzickému vytvoření dat opět dojde spuštěním mapovacího ETL procesu v *Control Center Manager*.

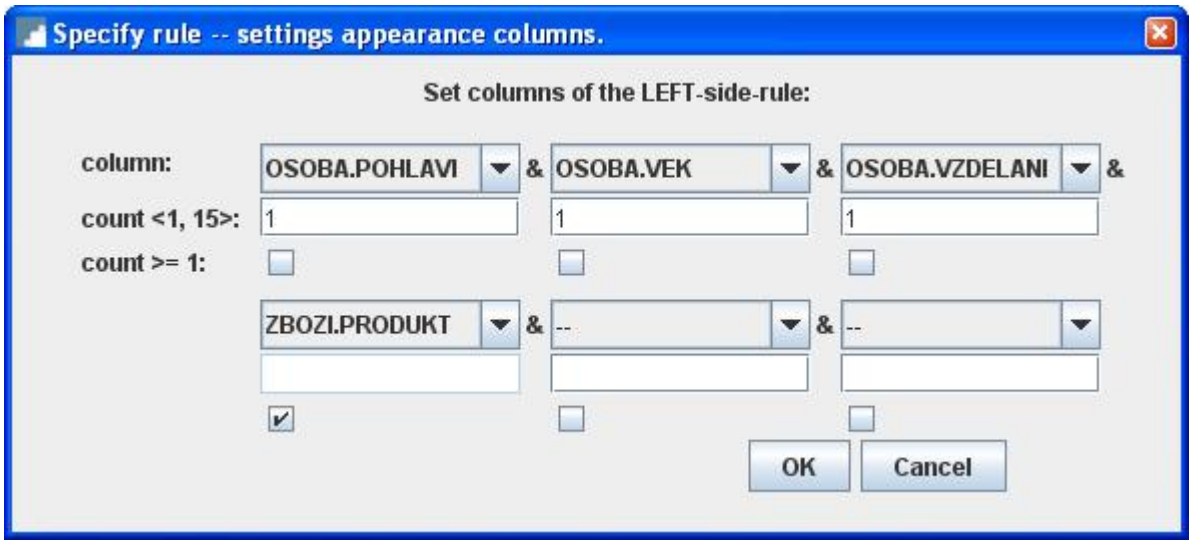
V tomto stavu jsou již veškerá data připravena (existuje požadovaná hvězda) pro program na získávání asociačních pravidel, který byl vytvořen v rámci diplomové práce. V programu se vytvoří dolovací schéma s multidimenzionálním zdrojem a v nastavení se postupně provede propojení faktové tabulky s dimenzionálními tabulkami, čímž vznikne hvězda, jež je zdrojem dat pro hledání asociačních pravidel. Tvorbu hvězdy v programu demonstruje *obrázek 7.5*. Na *obrázcích 7.6* a *7.7* jsou zobrazeny formuláře pro nastavení a omezení asociačních pravidel, přičemž je kromě základních vlastností možné nastavit, které predikáty nemohou být spolu v pravidle, které v pravidle musí být, na jaké straně pravidla a určení počtu těchto predikátů v pravidle. Datový analytik obchodního řetězce FITmarket pak realizuje variantní nastavení těchto parametrů a provádí analýzu získaných pravidel.



Obrázek 7.5: Mapování zdrojové hvězdy v programu pro dolování dat.



Obrázek 7.6: Nastavení omezení asociačních pravidel.



Obrázek 7.7: Konkrétní specifikace asociačních pravidel (určení výskytu a četnosti predikátů pro levou stranu asociačního pravidla).

7.3 Srovnání s ostatními systémy

V této kapitole bude naimplementovaný systém porovnán s již existujícími systémy. Bude se jednat o porovnání z pohledu možností tvorby asociačních pravidel. Systémy, se kterými jsem se dostal do kontaktu, jsou *SAS Enterprise Miner, verze 4.3* [14] (v rámci předmětu *Získávání znalostí z databází* na FIT VUT Brno) a *Oracle Data Miner, verze 11.1.0.1* [15]. Program, jenž byl implementován v rámci diplomové práce, bude dále nazýván *Association Data Miner*.

Co se týče dostupných algoritmů na tvorbu asociačních pravidel *SAS* i *Oracle Data Miner* obsahují pouze **Apriori**, zatímco *Association Data Miner* obsahuje navíc algoritmus **FP-strom**. Jak je známo a bylo otestováno, Apriori je prakticky nepoužitelné pro hledání frekventovaných množin pro rozsáhlé zdroje dat, nebo pro nastavení malé hodnoty *podpory*.

SAS očekává zdrojová data pouze v transakční formě, což je značně omezující pro robustnější nasazení systému. *Oracle Data Miner* umožňuje kromě transakčního přístupu pracovat nad daty uloženými v relačních tabulkách. Tyto tabulky je však nutné před samotným dolováním neintuitivně převést do transakční tabulky a ve výsledku není možné tvořit pravidla s vícečetným výskytem predikátů. *Association Data Miner* očekává vstupní data buď v transakční podobě, nebo multidimenzionální – ve formě hvězdy. Hvězda je jedna z přirozených reprezentací multidimenzionálních dat a je taktéž typická pro datové sklady.

Jediné vlastnosti, které lze v *SAS*u nastavit, jsou *podpora, spolehlivost* a maximální délka pravidla. Je možné dolovat pouze *jednodimenzionální booleovská asociační pravidla*.

V aplikaci *Oracle Data Miner* je možné před procesem dolování nastavit *podporu, spolehlivost* a maximální délku pravidla. Po procesu dolování, kdy jsou již veškerá pravidla nalezena, je možné omezit tabulkový výpis pravidel na specifikování, které predikáty mají být na levé a pravé straně pravidel. Výsledkem dolování jsou *jednodimenzionální booleovská asociační pravidla* nebo *mezi-dimenzionální asociační pravidla*.

Aplikace *Association Data Miner* umožňuje před procesem dolování nastavit výběr *dolovacího algoritmu, podporu, spolehlivost, online* čtení dat z databáze (*online*/*offline* režim přístupu dat). Dále je možné klást omezení na dolovací pravidla, jakožto určit maximální délku pravidla, které predikáty nemohou být společně v pravidlech a jaké predikáty v jakém počtu se mají vyskytovat na levé a pravé straně pravidel. Výsledkem dolování jsou *jednodimenzionální booleovská asociační pravidla* nebo *hybridně-dimenzionální asociační pravidla*.

Z pohledu asociačních pravidel lze usuzovat, že aplikace *Association Data Miner*, jež byla vyvinuta v rámci diplomové práce, disponuje robustnějšími a efektivnějšími vlastnostmi než zbylé porovnávané systémy *SAS* a *Oracle Data Miner*.

8 Závěr

Diplomová práce se v úvodních kapitolách zabývá teoretickým východiskem z oblasti získávání znalostí. Stěžejní částí této teoretické části bylo zaměření se na obecný pohled procesu získávání znalostí, obecných principů návrhů datový skladů s popsanými doporučeními (někdy až kontrastními) průkopníků této problematiky Inmona a Kimballa, a specializace se na asociační pravidla.

Teoretické poznatky vytvořily silný aparát pro realizaci praktické části diplomové práce. V rámci implementační části byla realizována aplikace pro získávání jednodimenzionálních booleovských asociačních pravidel a hybridně-dimenzionálních asociačních pravidel. Jako zdroj hybridně-dimenzionálních pravidel je uvažována datová hvězda, která vykazuje rychlejší přístupovou dobu k datům oproti sněhové vločce, což je u procesu dolování dat významným aspektem. Pro zvýšení rychlosti dolování a pro cílenější hledání asociačních pravidel, je možné nastavit variantní omezení na tato pravidla. Pro další možné zrychlení procesu dolování byly algoritmy implementovány pro práci nad daty buď přímo v operační paměti (offline režim), nebo ve smyslu čtení právě požadovaných dat z databáze (online režim).

Implementovanými algoritmy na hledání frekventovaných množin jsou Apriori a FP-strom. FP-strom je momentálně nejefektivnější algoritmus na tvorbu frekventovaných množin, což bylo porovnáním s algoritmem Apriori pro různé zdroje dat testováno. FP-strom se jeví mnohem efektivnější, zvláště pak pro nízké hodnoty podpory, kdy Apriori bylo řádově stokrát pomalejší a pro velmi nízké hodnoty podpory prakticky nepoužitelné. Dalšími testy docházelo k porovnání offline a online režimu ze vzdáleného databázového serveru. Prokázalo se, že pro menší množství zdrojových dat je vhodnější pracovat nad daty v operační paměti. Offline režim se jevil a jeví význačným přínosem právě pro metodu Apriori, kde v online režimu dochází ke čtení celé databáze do aplikace pro dolování četnostně přímo úměrně s mohutností nejdelší frekventované množiny. To se může nepříjemně projevit zvláště v případě, kdy se nachází databázový server daleko od aplikace pro dolování (velké zpoždění toku dat).

Zdrojová data byla uchovávána na lokálním databázovém serveru *Oracle Database 11g, Enterprise Edition* (pro testování online/offline režimu byl použit databázový *Oracle* server na FIT VUT Brno), jenž obsahuje podporu pro tvorbu datových skladů (*Oracle Warehouse Builder*). Vyvinutá aplikace datový sklad účelně využívá, neboť pomocí mapování a ETL procesů je možné připravit data z různých datových zdrojů, provádět nad němi transformace a převést do podoby datového schématu hvězdy, která je programem pro hledání hybridně-dimenzionálních asociačních pravidel očekáván. Tímto se aplikace, vytvořená v rámci diplomové práce, stává ještě robustnější a prakticky použitelnější v mnoha sférách, kde je vhodné provádět asociační analýzu dat.

Přínosem oproti zkoumaným existujícím systémům (*Oracle Data Miner, SAS*) se jeví implementace algoritmu FP-strom a možnosti variantního omezení asociačních pravidel před samotným dolováním, což umožňuje tato data efektivněji zpracovávat. Dalším přínosem je možnost tvorby hybridně-dimenzionálních asociačních pravidel, jež v *SAS*u není umožněno a v *Oracle Data Miner* je tvorba dimenzionálních pravidel velmi neintuitivní (a není umožněno mít více hodnot daného predikátu v jednom pravidle).

V rámci rozšíření systému se jeví účelné vytvořit dodatečné algoritmy pro získávání víceúrovňových asociačních pravidel, jež umožňují nastavení podpory a spolehlivosti pro každou úroveň v dimenzích. Pro redukci datové redundance, která je pro hvězdu typická, by bylo vhodné jako alternativu umožnit uchovávat zdrojová data ve schématu sněhové vločky.

Tato diplomová práce úspěšně navázala na semestrální projekt, jenž byl realizován v rámci zimního semestru. Obsahem Semestrálního projektu bylo důkladné zpracování teoretických poznatků (kapitoly 2 až 4) a seznámení se s datovým skladem firmy *Oracle* (Příloha A).

Práce vychází ze studia řady podkladů, které jsou citovány v použité literatuře.

Literatura

[1] Han, J., Kamber, L.: Data Mining: Concepts and Techniques. Second Edition. Elsevier Inc., 2006. ISBN: 1-55860-901-3.

[2] Zendulka, J., Bartík, V., Lukáš, R., Rudolfová, I..: Získávání znalostí z databází, studijní opora do předmětu ZZN. FIT VUT Brno [online]. 2006-10-31 [cit. 2008-12-21]. Dostupné z WWW: <https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/ZZN-IT/texts/ZZN.pdf>

[3] Oracle Warehouse Builder. User's Guide. 11*g* Release 1. Oracle [online]. 2008-11 [cit. 2008-12-21]. Dostupné z WWW: <http://download.oracle.com/docs/cd/B28359_01/owb.111/b31278.pdf>

[4] Data Warehouse. Wikipedia [online]. 2008-12-20 [cit. 2008-12-23]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Data_warehouse>

[5] Segmentace návštěvníků webových stránek. StatSoft CR [online]. 2008 [cit. 2008-12-22]. Dostupné z WWW: <http://www.statsoft.cz/page/index2.php?segmentace>

[6] Chmelář, P.: Mining Association Rules from Texture Databases. FIT VUT Brno [online]. 2005 [cit. 2008-12-22]. Dostupné z WWW: <http://www.fit.vutbr.cz/~chmelarp/public/05artexture.pdf>

[7] Rudolfová, I.: Shluková analýza. Přednášky do předmětu ZZN. FIT VUT Brno [online]. 2007-11-28 [cit. 2008-12-22]. Dostupné z WWW: <https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/ZZN-IT/lectures/07_Shlukova_analyza.pdf>

[8] Inmon, W., H..: Building the Data Warehouse. Third Edition. Wiley Computer Publishing, 2002. ISBN: 0-471-08130-2.

[9] Datové sklady Microsoftu pomáhají Eurotelu zvyšovat obrat a zisk na zákazníka. Microsoft [online]. 2004-08-09 [cit. 2008-12-23]. Dostupné z WWW: <http://www.microsoft.com/cze/casestudies/Eurotel_BI.mspx>

[10] Kimball, R., Ross, M.: The Data Warehouse Toolkit. Second Edition. The Complete Guide to Dimensional Modeling. Wiley Computer Publishing, 2002. ISBN: 0-471-20024-7.

[11] Kotásek, P.: DWH / BI systémy, datové sklady. Přednášky do předmětu ZZN odborníka z praxe (Home Credit International). FIT VUT Brno [online]. 2007-11-13 [cit. 2008-12-26]. Dostupné z WWW: <https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/ZZN-IT/lectures/P6_DWH_BI.pdf>

[12] Bartík, V.: Získávání frekventovaných množin a asociačních pravidel. Přednášky do předmětu ZZN. FIT VUT Brno [online]. 2008-10-23 [cit. 2008-12-29]. Dostupné z WWW: <https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/ZZN-IT/lectures/04_AR.pdf>

[13] Visual Paradigm for UML 7.0. Visual Paradigm [online]. 2009-04-20 [cit. 2009-05-09]. Dostupné z WWW: <http://www.visual-paradigm.com/news/vpsuite40/vpuml70.jsp>

[14] SAS Enterprise Miner 4.3. SAS [online]. 2009 [cit. 2009-05-18].
Dostupné z WWW:
<http://support.sas.com/documentation/whatsnew/91x/emguiwhatsnew900.htm>

[15] Oracle Data Miner 11.1.0.1. Oracle [online]. 2009-08 [cit. 2009-05-18].
Dostupné z WWW: <http://www.oracle.com/technology/products/bi/odm/odminer.html>

Příloha A – Tvorba datového skladu v Oracle Warehouse Builder 11*g*

Obsahem této přílohy je základní demonstrace aplikace **Oracle Warehouse Builder 11*g***, jenž slouží na tvorbu datového skladu. Bude zde popsán postup, jak vytvořit datový sklad, a jaké možnosti jsou v tomto datovém skladu k dispozici. Následující informace vychází z technické dokumentace Oracle 11*g* [3].

Tvorba Warehouse Builder Repository

Prvním nutným úkonem k tvorbě datového skladu je vytvoření datového úložiště – **Warehouse Builder Repository**. Jsou zde uloženy metadatové objekty, mapovací a transformační schémata (ETL procesy). K tvorbě tohoto speciálního úložiště slouží **Repository Assistant**. Je taktéž nutné mít vytvořenou Oracle databázi, kde se ukládají veškerá data. Při spuštění *Repository Assistant* se zobrazí úvodní obrazovka – je možné přeskočit. Dále následují tyto kroky:

1. Přístupové údaje k Oracle databázi - adresa serveru (*Host Name*), port (*Port Number*) a jméno instance databáze (*Oracle Service Name*).
2. Výběr akce, která se má provést – zvolí se *Manage Warehouse Builder workspaces.*
3. Tvorba úložiště pro datový sklad - *Create a new Warehouse Builder workspace.*
4. Možnost tvorby *repository* s tvorbou nového uživatele, nebo tvorby *repository* s již existujícím uživatelem. Zvolí se volba *Create a workspace with a new user as workspace owner*, tedy bude se vytvářet i nový uživatel.
5. Vložení informací o uživateli, který má oprávnění tvorby nových uživatelů – uživatel skupiny DBA (implicitně je to uživatel *SYSTEM* s heslem *oracle*).
6. Tvorba nového uživatele – vlastníka *repository*. Zadává se jméno uživatele (*Workspace Owner's User Name*), heslo (*Workspace Owner's Password*), ověření hesla (*Workspace Owner's Password Confirmation*) a jméno *repository* (*Workspace Name*).
7. Vyplnění hesla uživatele s oprávněním OWBSYS (implicitně heslo uživatele *SYSTEM*)
8. Výběr *tablespace* pro schéma OWBSYS – je možné nechat implicitní volbu.
9. Výběr jazyka objektů vytvořených v *repository* – implicitní volba jazyka nastaveného v operačního systému.
10. Výběr a tvorba uživatelů, kteří budou s *repository* asociováni.

Nyní po potvrzení údajů (poslední okno) dochází k tvorbě *repository*.

Tvorba uživatelů Repository

Pokud je nutné vytvořit nového uživatele, docílí se toho opět aplikací **Repository Assistant**. Po úvodní obrazovce následují tyto kroky:

1. Přístupové údaje k Oracle databázi - adresa serveru (*Host Name*), port (*Port Number*) a jméno instance databáze (*Oracle Service Name*).

2. Výběr akce, která se má provést – zvolí se *Manage Warehouse Builder workspace users*.
3. Vložení přihlašovacích údajů vlastníka – název vlastníka (*Workspace Owner/Admin User Name*), heslo vlastníka (*Workspace Owner/Admin Password*).
4. Volba *repository* (*Workspace Name*) – zvolí se implicitní.
5. Výběr *Register Warehouse Builder workspace users*.
6. Zde se zvolí již vytvořený uživatel, nebo se klikne na tlačítko *Create New User*, čímž se vytvoří nový uživatel. Zde se zadává – uživatelské jméno (*User Name*), heslo (*Password*), ověření hesla (*Re-enter Password*), jméno uživatele spadajícího do skupiny DBA (*DBA User Name*), heslo uvažovaného uživatele skupiny DBA (DBA User Password). Dojde k vytvoření nového uživatele.

Po potvrzení předchozích údajů vznikne nový uživatel (obecně více nových uživatelů), který bude asociován se zvolenou *repository*.

Připojení a nastavení zdrojů k integraci dat

Přihlášení do Design Center

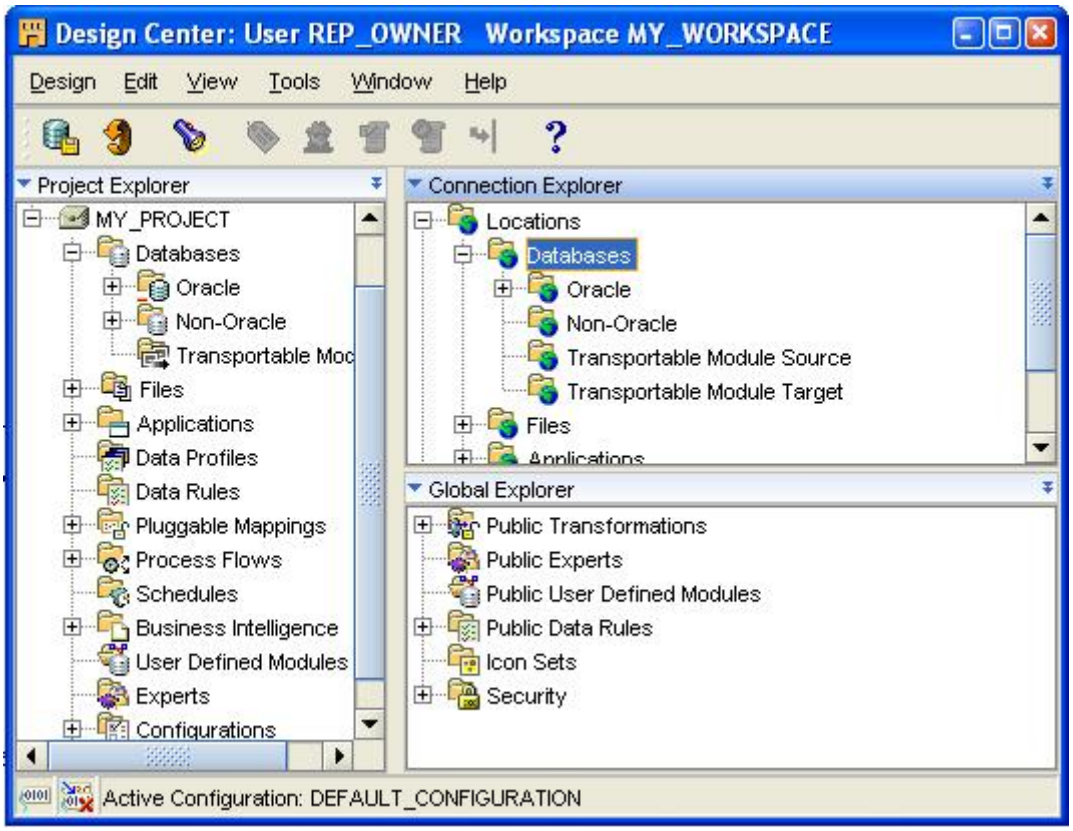
V tuto chvíli je vytvořené datové úložiště pro metadata datového skladu, mapovací a transformační schémata – *repository*. Je taktéž vytvořený vlastník tohoto *repository*. Tento vlastník (respektive jiný uživatel, jenž je asociován s *repository*) se může nyní připojit a konfigurovat datový sklad. Jakožto hlavní aplikace pro správu Warehouse Builder je **Design Center**.

Pro přihlášení ke konfiguraci Warehouse Builder se spustí program Design Center. Zobrazí se okno **Design Center Logon**. Zde se zadává jméno uživatele *repository* (*User Name*), heslo tohoto uživatele (*Password*), je taktéž nutné zadat přístupové údaje k Oracle databázi – adresa serveru (*Host*), port (*port*) a jméno instance databáze (*Service Name*). Po úspěšném přihlášení se zobrazí hlavní okno Desing Center (*obrázek A-1*).

Připojení cílového úložiště

Nyní je nutné vytvořit informaci o umístění cílového úložiště, kde budou ukládány fyzické objekty (tedy to, co bude popsáno metadaty a ETL procesy). Bude popsán postup pro tvorbu cílového úložiště v Oracle databázi:

1. Pravý klik myší na *Databases-Oracle*, výběr *new*.
2. Přeskočit uvítací obrazovku (poklik na *Next*).
3. Do *Name* vepsat libovolný (již neexistující) řetězec, *Identify the module type* zvolit *Warehouse Target*.
4. U kolonky *Location* zvolit *Edit*.
5. Vyplní se přihlašovací údaje k Oracle databázi.
6. Zvolí se *Finish*, čímž dojde k vytvoření informací o cílovém úložišti, kde budou fyzicky ukládány objekty datového skladu.



Obrázek A-1: Hlavní okno Design Center

Tvorba metadat cílových objektů datového skladu

Datový sklad vzniká integrací několika zdrojů, přičemž tyto zdroje jsou obecně heterogenní (databáze, textové soubory, ..). Tato integrovaná data se ukládají do datového skladu do předem vytvořených tabulek. Nyní bude popsán postup, jak informovat Warehouse Builder, kam má cílová data ukládat (tvorba metadat pro cílové tabulky).

1. Ve vytvořeném cílovém úložišti (bylo vytvořeno v předchozích krocích) vybrat *tables*, poklik pravým tlačítkem myši, vybrat *Import*. Otevře se *Import Metadata Wizard*.
2. Přeskočí se úvodní obrazovka (*Next*).
3. Zvolí se typ objektu (*Object Type*) pro import metadat. Jedná se o filtraci typů objektů.
4. Výběr konkrétních objektů a potvrzení (*Next*, *Finish*).

Připojení úložiště zdroje dat

Oracle Warehouse Builder nabízí širokou volbu výběru variantních zdrojů dat – heterogenní databázové systémy, textové soubory s jednoduchým formátováním (csv, tsv, ..), XML soubory, externí aplikace a další. Nejčastějším zdrojem dat jsou databázové systémy. Postup vytvoření připojení úložiště zdroje dat je velmi podobný již popsaným postupům tvorby úložiště cílového.

1. Postup připojení cílového úložiště je totožný s postupem zmíněným v **Připojení cílového úložiště** s tím rozdílem, že jakožto *Identify the module type* se uvede *Data Source*.
2. Je opět nutné provést import metadat, jak bylo popsáno v **Tvorba metadat cílových objektů datového skladu**. Nyní se však uvádí metadata těch objektů, které budou sloužit jako zdroj dat.

Tvorba ETL procesů a integrace dat

V tuto chvíli je již kompletní informace o úložišti zdrojových dat, o metadatech zdrojových dat, o úložišti cílových dat a metadatech cílových dat. Nyní následují ETL procesy, které slouží k propojení různých zdrojových dat a k provádění transformačních funkcí (agregace, čištění, redukce hodnot, tvorba multidimenzionálních kostek, ..).

Tvorba mapovacího schématu

K tvorbě návrhu ETL procesů slouží mapovací schéma. Mapovací schéma (*Mappings*) se vytváří (za předpokladu, že cílové schéma je na databázi Oracle, což je doporučené) klikem pravého tlačítka myši na *Databases-Oracle-Úložiště_cílového_schématu-Mappings* a výběrem z kontextového menu položky *New*.

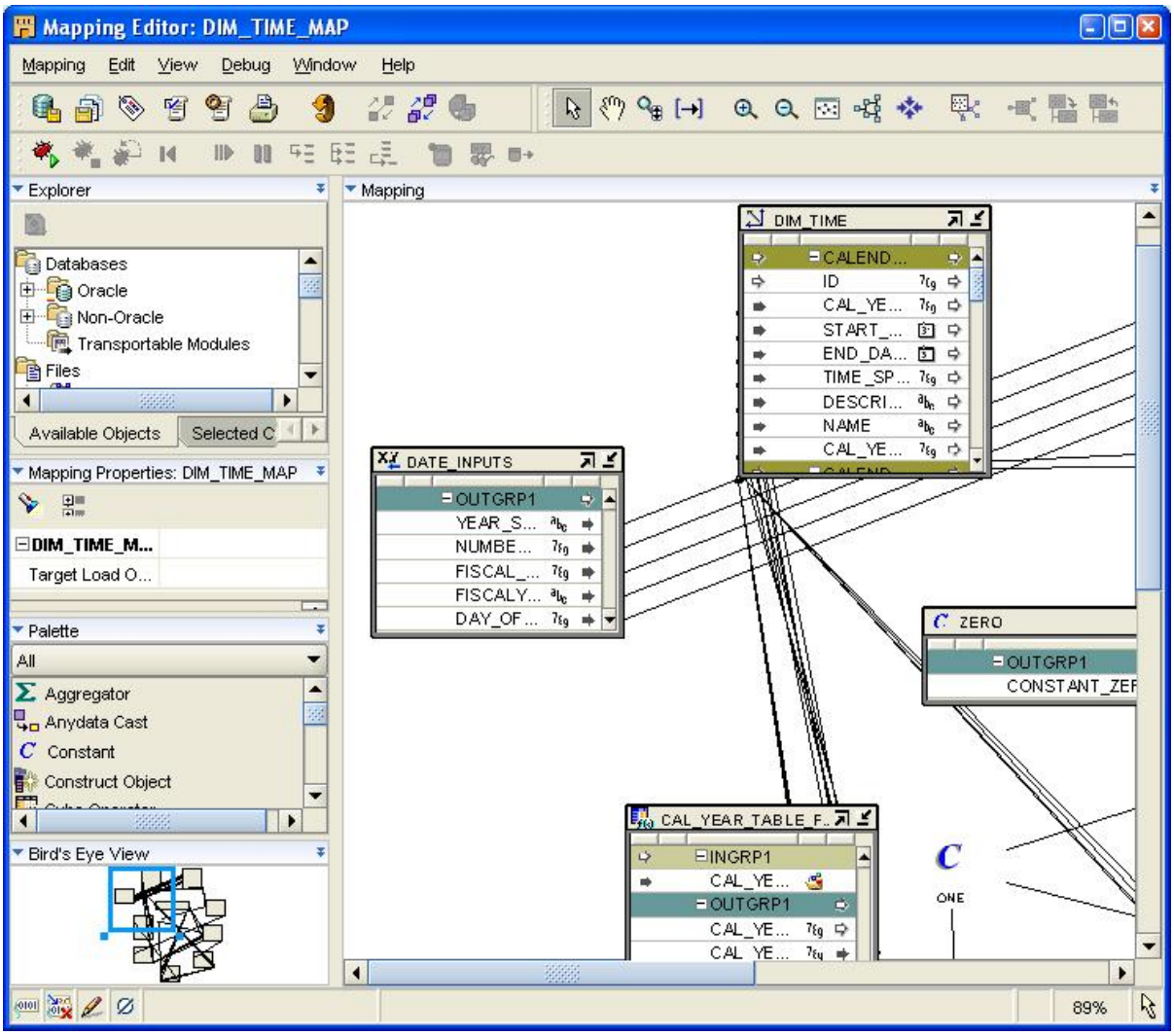
Dojde k zobrazení **Mapping Editor**, kde probíhá veškerý ETL návrh (*Obrázek A-2*). Jsou zde významná okna *Explorer*, *Palette* a *Mapping*. V okně *Explorer* se vybírají zdrojové a cílové objekty, mezi kterými dojde k mapování. Okno *Pallete* slouží k výběru různých transformačních funkcí, jakožto například agregování, aplikace matematických výrazů, filtrování nežádoucích dat, multidimenzionální operátory, množinové operace a další. Poslední zmíněné okno *Mapping* se využívá pro samotnou tvorbu mapovacího schématu – jsou zde přemístěna zdrojová, cílová data a transformační funkce.

Tvorba dat z návrhu mapovacího schématu

Jakmile je schéma pomocí **Mapping Editor** vytvořené (tato metadata jsou uložena v *repository*), je možné tato data v databázi vytvořit. K tvorbě navržených dat slouží **Control Center Manager**. Spouští se v Design Center – v menu vybrat *Tools*, dále vybrat *Control Center Manager*. K tvorbě dat dojde docílením následujících kroků:

1. V cílovém schéma vybrat položku *mappings* a v ní vybrat vytvořené mapovací schéma. Klepnout na tlačítko *Default Actions*.
2. Klepnutí na ikonu *Deploy*. Vygeneruje se kód mapovacího schématu (metadat) a uloží se na server.
3. Klepnutí na ikonu *Start*. Dochází k načtení a uložení dat objektů. Pokud vše proběhlo v pořádku, v okně *Control Center Jobs* bude u vykonané úlohy s názvem mapovacího schématu zelené zatržítko.

V tento moment jsou již na data aplikovány veškeré ETL procesy, jenž byly navrženy v *Mapping Editor*, a výsledná data jsou uložena v databázi.



Obrázek A-2: Tvorba mapovacího schématu – návrh ETL procesů

Příloha B – Ukázka XML schématu výstupu multidimenzionálního dolování

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE miningProcess SYSTEM "../dtd/miningProcess.dtd">

<miningProcess>
 <metadata>
  <algorithmProcess>
   <kind>HybridMultidimensional</kind>
   <algorithm>FP-Tree</algorithm>
  </algorithmProcess>

  <associationParameters>
   <support>3.5</support>
   <confidence>20.0</confidence>
  </associationParameters>

  <onlineDB>false</onlineDB>

  <tableStarScheme>
   <factTable>FAKTA</factTable>
   <dimensionalTables>
    <dimensionalTable>
     <name>OSOBA</name>
     <usedColumns>
      <column>POHLAVI</column>
      <column>VEK</column>
      <column>VZDELANI</column>
     </usedColumns>
    </dimensionalTable>
    <dimensionalTable>
     <name>ZBOZI</name>
     <usedColumns>
      <column>PRODUKT</column>
      <column>KATEGORIE</column>
     </usedColumns>
    </dimensionalTable>
    <dimensionalTable>
     <name>CAS</name>
     <usedColumns>
      <column>DEN</column>
     </usedColumns>
    </dimensionalTable>
   </dimensionalTables>
  </tableStarScheme>
```

```
  <constraints>
   <ruleConstraints>
    <maxItems>5</maxItems>
   </ruleConstraints>
  </constraints>
 </metadata>

<associationRules>
  <associationRule>
   <left>muz & 26-30 & VS</left>
   <right>pocitac & LCD</right>
  </associationRule>

  <associationRule>
   <left>muz & vyucen</left>
   <right>sada_sroubovaku</right>
  </associationRule>
        .
        .
        .
</associationRules>
```