VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

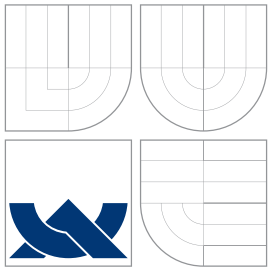
IMPLEMENTACE OLAP ANALÝZY NAD DATY KNIHOVEN VUT

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

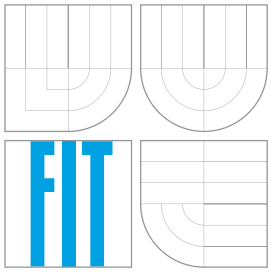
AUTOR PRÁCE
AUTHOR

Bc. JOSEF MAHDALÍČEK

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

IMPLEMENTACE OLAP ANALÝZY NAD DATY KNIHOVEN VUT

IMPLEMENTATION OF OLAP ANALYSIS FOR BUT LIBRARIES

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. JOSEF MAHDALÍČEK

VEDOUCÍ PRÁCE
SUPERVISOR

Mgr. MARTIN ŠÁRFY

BRNO 2008

Abstrakt

Cílem této práce je vytvořit nástroj pro OLAP analýzu nad provozními daty knihoven VUT. Tento nástroj odpovídá například na dotaz jak dlouho měli uživatelé půjčené knihy. Tyto dotazy mohou být specifikovány dle časového období (rok, měsíc), knihovny (např. knihovna FIT) a dalších dimenzí. Ze zdrojové databáze jsou použity jen některé, pro tuto aplikaci zajímavé, tabulky. Tabulky byly exportovány do textových souborů ve formátu csv. Dle požadavků ze zadání by OLAP nástroj měl být dodáván v licenci open source. Nástroj Mondrian splňuje tento požadavek a byl použit v této práci. Datový sklad je reprezentován relační databází MySQL. Nástroj ETL plní tento datový sklad daty z exportovaných souborů. Uživatelské rozhraní je ovládáno z webového prohlížeče a je implementováno komponentou JPivot. Výsledky dotazů jsou zobrazeny ve tabulkách a grafech.

Klíčová slova

OLAP analýza, Mondrian, Aleph VUT, ETL

Abstract

The aim of this project is to create tool for OLAP analysis over operational data of BUT libraries. This OLAP tool answers ie. query how long were users having loaned books. These queries could be specified by time period (year, month), library (ie. FIT library) and other dimensions. Only some, for this application interesting, tables from source database are used. Tables were exported into text files in csv format. According to project specification, system for OLAP analysis should be open source. Tool Mondrian accomplishes this requirement and was used in this work. Data warehouse is represented by relational database MySQL. ETL tool feeds data warehouse by data from exported files. User interface is used from internet browser and is implemented by component JPivot. Query results are displayed in tables and graphs.

Keywords

OLAP analysis, Mondrian, Aleph VUT, ETL

Citace

Josef Mahdalíček: Implementace OLAP analýzy nad daty knihoven VUT, diplomová práce, Brno, FIT VUT v Brně, 2008

Implementace OLAP analýzy nad daty knihoven VUT

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Mgr. Martina Šárfyho

.....
Josef Mahdalíček
16. května 2008

Poděkování

Tímto bych zde chtěl poděkovat vedoucímu diplomové práce Mgr. Martinu Šárfymu za konzultace zadání a odpovědi na mé dotazy.

Obsah

Kapitola 1

Úvod

Cílem této diplomové práce je vytvořit nástroj pro OLAP analýzu provozních dat knihoven VUT. Nástroj bude vytvářet statistiky ve formě tabulek a grafů. Podniky, banky a další instituce používají databázové nástroje pro ukládání dat z oblasti jejich působnosti. Například banka spravuje data o bankovních operacích, které její klienti provádějí, nebo podnik ukládá informace o výrobě a knihovna zase ukládá informace o dokumentech, výpůjčkách, platbách za pozdní vrácení atd. Databáze, které tyto instituce používají, spravují velké objemy podnikových dat. V poslední době se čím dál častěji rozšiřují nástroje pro analýzu těchto dat. Důvod je ten, že různé instituce by rády využily tato historická data pro podporu rozhodování o dalších krocích. Například podnik by mohl zjistit výkonnost jednotlivých oddělení. Nástroj vytvořený v rámci této práce poskytuje různé statistiky z historických dat knihoven VUT. Dotazy, na které tento nástroj odpovídá, jsou například následující. Jak dlouho mají průměrně zákazníci knihovny půjčené knihy? Nebo kolik uživatelé knihovny zaplatili za pozdní vrácení? Další statistikou bude statistika knih dle jazyka (např. české, anglické) nebo žánru (např. historický text, technický text) atd. Dotazy pro OLAP analýzu bude možné aplikovat na všechny knihovny nebo jen některou konkrétní. Také mohou být kladeny dotazy pro určité období (např. školní rok, měsíc). Tyto atributy mohou být kombinovány. Uživatel tohoto nástroje tedy bude moci klást předem neznámé dotazy, což je možné díky OLAP metodám bez znalosti dotazovacího jazyka. Bylo by možné vytvořit předem připravené dotazy v jazyce SQL, ovšem pak nastává otázka, co když chce uživatel spustit některý nový dotaz... Důležitým požadavkem na OLAP server, který bude program využívat, je licence. Server by měl být open source, tedy s dostupným zdrojovým kódem. Tento požadavek splňuje například server Mondrian, který bude v této práci použit. Z knihovní databáze Aleph budou pro OLAP analýzu použity jen některé tabulky, které budou exportovány do textových souborů. Každá tabulka je uložena v samostatném souboru, v němž je obvykle každá n-tice tabulky uvedena na samostatném řádku a atributy jsou odděleny mezerami. Pro naplnění datového skladu z exportovaných souborů byl v rámci této práce vytvořen ETL nástroj, který vykonává transformaci dat (převod hodnot, vynechání nepotřebných atributů a tak dále) a jejich nahrání do datového skladu.

Kapitola 2 obsahuje popis teorie budování datového skladu. Jsou zde rozebírány relační a multidimenzionální databázové modely jako základy pro datový sklad. Dále je v této kapitole popsáno rozdělení OLAP nástrojů dle uložení dat a licence. V sekci 2.3 je popsána fáze ETL (extrakce, transformace a nahrání), kterou se plní datový sklad, vybudovaný za účelem OLAP analýzy. V této sekci jsou popisovány tyto tři operace včetně kroků, které každá operace nejčastěji zahrnuje.

V kapitole 3 je popsáno schéma zdrojové databáze knihoven VUT, včetně významu vybraných tabulek, jejichž data budou použita k OLAP analýze. V této kapitole jsou také uvedeny příklady dotazů, které bude moci uživatel systému, vytvořenému v rámci této práce, klást. Příkladem může být statistika výpůjček pro jednotlivé knihovny. Dotazy se budou tvořit dle různých parametrů (knihovna, časové období a další). Dále jsou v této kapitole kladeny požadavky na ETL nástroj a OLAP server (např. licence).

Část kapitoly 4 představuje vybraný OLAP nástroj Mondrian. Také je v této kapitole představen návrh datového skladu, který bude sloužit jako úložiště dat pro OLAP analýzu. Datový sklad je reprezentován databází MySQL. V této kapitole je vysvětlen formát, kterým se definuje schéma pro systém Mondrian. Toto schéma mapuje kostky, dimenze, hodnoty a další prvky na tabulky a atributy relační databáze. Dále je v této kapitole popsán ETL nástroj, kterým se plní datový sklad ze souborů, exportovaných ze zdrojové databáze Aleph. Tento nástroj plní tabulky faktů a dimenzí a je implementován v jazyce Java.

Kapitola 5 obsahuje popis uživatelského rozhraní. Uživatelské rozhraní je implementováno komponentou JPivot a je ovládáno z internetového prohlížeče.

V kapitole 6 jsou zhodnoceny výsledky práce.

Dodatek A obsahuje popis instalace nástroje Mondrian v operačním systému Linux. Instalace nástroje Mondrian je popisována pro linuxovou distribuci Ubuntu ve verzi 6.06 Dapper Drake. Postup instalace v jiné distribuci by byl podobný. Mondrian je samozřejmě možné nainstalovat i v operačním systému Windows.

Dodatek B obsahuje popis používání tohoto nástroje.

Kapitola 2

Teorie budování datového skladu

Informace k této kapitole byly čerpány z [4], [5] a [6].

Datový sklad je subjektově orientovaná, integrovaná, časově neměnná, kolekce dat získaných v různém časovém období na podporu rozhodování. Data v datovém skladě je možné použít k různým účelům včetně toho případu, kdy zatím budou tato data uložena a použita budou někdy v budoucnu, protože nyní nejsou budoucí požadavky známy.

První vlastností datového skladu je *orientace na subjekty*. Klasické operační systémy jsou organizovány dle funkčních aplikací společnosti. Aplikacemi mohou být pro pojišťovací společnost pojištění auta, pojištění zdraví a řešení nehody. Oproti tomu jsou v datovém skladu ukládány informace spíše podle předmětu než podle funkce aplikace. Například subjekty pro výrobce by mohly být např. výrobek, objednávka, prodávající, seznam materiálu a suroviny. Každá společnost má vlastní množinu subjektů.

Druhou charakteristikou datového skladu je *integrovanost*. Integrovanost je ze všech hledisek nejdůležitější. Data se do datového skladu získávají z několika různých zdrojů. Data musí být často převedena, vyčištěna, a musí být sjednocen formát hodnot. Dříve aplikační programátoři při tvorbě aplikace neuvažovali, že data, se kterými pracují, budou integrována s jinými daty. V různých aplikacích není konzistence v kódování, konvencích pojmenování, fyzických atributech, měr atributů a tak dále. Například problém může být v různých jednotkách atributu vzdálenost. V prvním zdroji dat jsou rychlosti uloženy v jednotce km, ve druhém v mílích. Je třeba, aby byly jednotky atributu sjednoceny - tj. převedeny a v datovém skladu uloženy v jednom formátu (např. v km).

Třetí důležitou charakteristikou datového skladu je *neměnnost*. Data v operačním prostředí jsou vkládána, modifikována a mazána. Data v datovém skladu nejsou upravována. Data jsou nahrána do datového skladu a můžeme k nim přistupovat, ale nemění se. Z toho vyplývá, že většina metod optimalizace a normalizace údajů a transakční přístup k datům je v datovém skladě nepotřebná.

Poslední důležitá vlastnost datového skladu je *časová proměnlivost*. Časová proměnlivost znamená, že každá jednotka dat v datovém skladu je správná v určitém časovém okamžiku. V některých případech je k záznamu uloženo časové razítko (timestamp). V jiných případech záznam obsahuje datum transakce. Ale ve všech případech je některý způsob označení času, aby se určil okamžik, ve kterém je záznam správný.

Operační databáze obsahují *data s aktuálními hodnotami* neboli data, která jsou správná v momentu přístupu. Například letecká kancelář ví, kdo má rezervace na let v některém okamžiku, banka ví, kolik peněz má zákazník na účtě v určitém okamžiku atd. Současné hodnoty dat mohou být změněny tak, jak se mění stav. Například letecká společnost odstraní sedadlo ze seznamu dostupných sedadel, když se vytvoří rezervace. Data v datové skladu

ovšem nejsou jen aktuální data. Data v datovém skladu mohou být uvažována jako propracovaná série snímků (snapshot). Každý snímek je vytvořen v určitém časovém okamžiku. Efekt vytvořený sérií snímků je takový, že datový sklad má posloupnost historie aktivit a událostí, což není zjevné v prostředí aktuálních hodnot, kde mohou být nalezeny jen nejaktuálnější hodnoty.

Struktura operačních dat může nebo nemusí obsahovat nějaký časový údaj, jako např. rok, měsíc, den a tak dále. Struktura v datovém skladu vždy obsahuje některý údaj o čase. Zahrnutí časového údaje do záznamu datového skladu může mít několik podob, jako např. časové razítko (timestamp) každého záznamu, časové razítko pro celou databázi a tak dále.

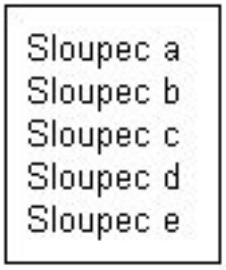
2.1 Relační a multidimenzionální databázový model

Jedna z otázek, kterým čelí tvůrci datových skladů, je základní model pro návrh databáze datového skladu (data warehouse). Existují dva základní modely pro návrh databáze, které jsou často uvažovány. Jsou jimi *relační model* a *multidimenzionální model*. Relační model je označován jako "Inmonův" přístup, zatímco multidimenzionální model je nazýván jako "Kimballův" přístup k návrhu datového skladu.

Tato podkapitola popisuje, jaké jsou přístupy k problematice datových skladů a jak se aplikují. Oba přístupy mají své výhody a nevýhody. Tyto výhody a nevýhody jsou diskutovány a závěr je vyvozen takový, že relační základ pro návrh datového skladu je nejlepší pro dlouhodobý přístup k budování datového skladu a pro případ, kdy je požadováno opravdové podnikové nasazení. Multidimenzionální model je dobrý pro krátkodobé datové sklady, kdy je limitovaná platnost datového skladu.

Relační model

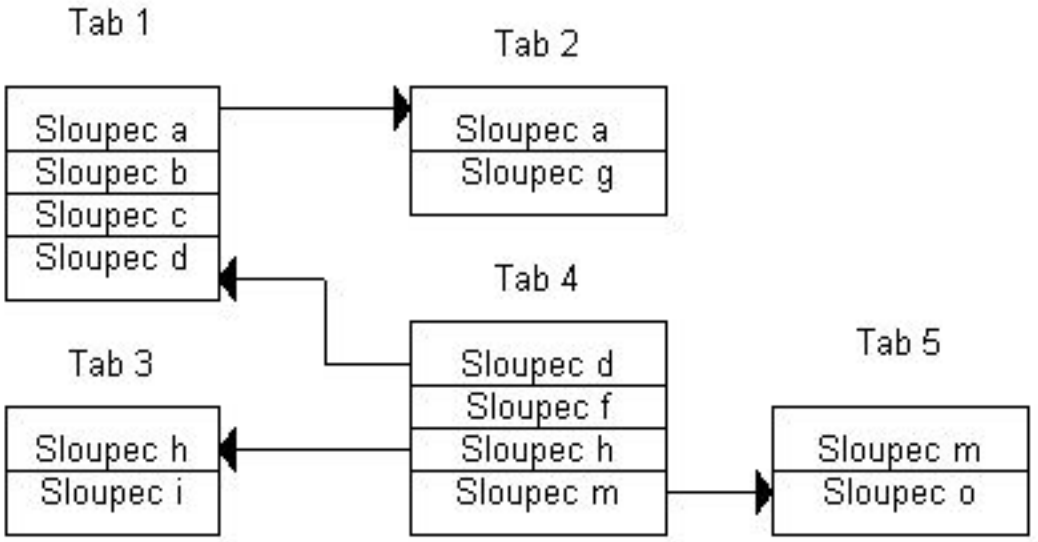
Relační přístup k návrhu databáze začíná s organizací dat v tabulce. Různé sloupce jsou v každém řádku tabulky. Obrázek 2.1 ukazuje jednoduchou tabulku.



Obrázek 2.1: Jednoduchá tabulka

Relační tabulka může mít různé vlastnosti. Sloupce dat mají odlišnou fyzickou charakteristiku. Různé sloupce mohou být indexovány a mohou fungovat jako identifikátory. Určité sloupce mohou mít hodnotu null. Všechny sloupce jsou definovány pomocí příkazů jazyka pro definování dat (Data Definition Language - DDL). Relační přístup k databázovému návrhu existuje od 70. let 20. století a osvědčil se v různých implementacích, jako např. systém DB2 od firmy IBM, systém pro řízení báze dat (SŘBD) firmy Oracle, SŘBD firmy Teradata a dalších. Relační technologie používá klíče a cizí klíče k vytvoření vztahů mezi

řádky dat. Relační technologie se o to stará pomocí jazyka SQL (Stuctured Query Language), který je široce používán jako jazyk rozhraní z programu k datům.



Obrázek 2.2: Návrh relační databáze

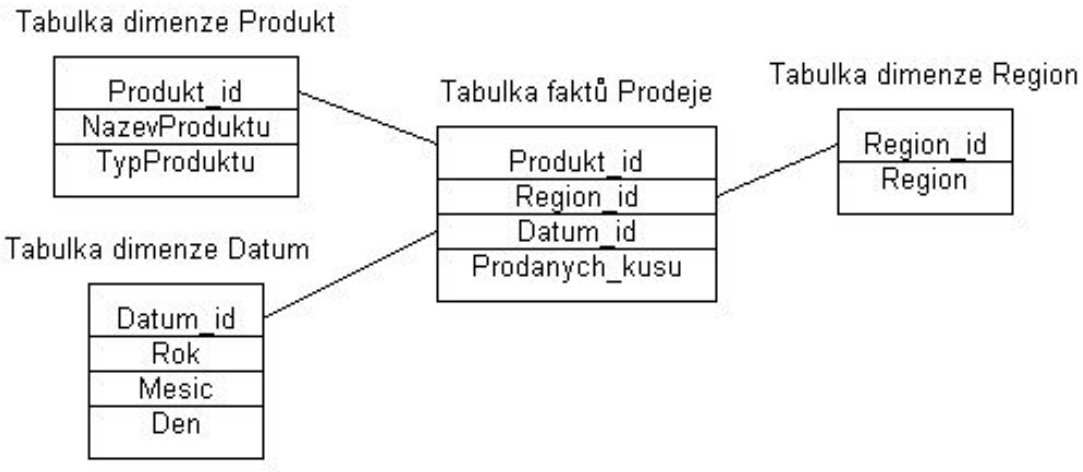
Obrázek 2.2 ukazuje příklad návrhu relační databáze. Tento obrázek obsahuje různé tabulky. Tabulky jsou propojeny ve smyslu série vztahů klíč-cizí klíč. Vztah klíč-cizí klíč je základní vztah, kdy se stejné jednotky dat nachází v obou tabulkách, a je v obrázku značen orientovanou šipkou.

Některé dva nebo více řádků jsou spojeny přes identickou jednotku dat. Například předpokládejme, že dva řádky mají hodnotu "Škoda Octavia" ve sloupci. Tyto dva řádky jsou spojeny výskytem stejné hodnoty. Data v relačním modelu jsou v "normalizované" formě. Normalizace dat má za následek, že databázový návrh způsobí, že data jsou rozdělena na velmi nízkou úroveň granularity. Pokud jsou tabulky normalizované, data uvnitř tabulek mají vztah jen s jinými daty, která jsou v tabulce. Normalizace se typicky uvažuje ve třech úrovních - první normální forma (1NF), druhá normální forma (2NF) a třetí normální forma (3NF).

Hodnota relačního modelu pro návrh databáze pro datový sklad je taková, že je řád ve způsobu, jak je databázový návrh vytvořen, jasnost významu a použití detailní úrovně normalizovaných dat. Jinými slovy relační model vytváří návrh pro datový sklad, který je velmi flexibilní. Na databáze se můžeme dle návrhu dívat nejprve jedním způsobem a potom jiným, když je návrh založen na relačním modelu. Datové elementy mohou být zobrazeny různými metodami. Flexibilita je potom silnou stránkou relačního modelu. Druhou silnou stránkou je univerzálnost. Protože detailní data jsou shromážděna a mohou být kombinována, mnoho různých pohledů na data může být podporováno, když je návrh datového skladu založen na relačním modelu.

Multidimenzionální model

Další přístup k návrhu databáze pro vybudování datového skladu je označován pojmem *multidimenzionální přístup*. Multidimenzionální přístup se někdy nazývá přístup star join. Zastáncem multidimenzionálního přístupu je Dr. Ralph Kimball. Základem multidimenzionálního přístupu k návrhu databáze je spojení tabulek, které připomíná hvězdu - viz obrázek 2.3. Toto schéma se nazývá hvězda (star), protože jeho reprezentace zobrazuje "hvězdu" se středem a několika okolními strukturami dat. Schéma hvězda obsahuje několik komponent. Ve středu tohoto schématu je tabulka faktů. Tabulka faktů je struktura, která obsahuje mnoho výskytů dat. Okolo tabulky faktů jsou tabulky dimenzí, které popisují jeden důležitý aspekt tabulky faktů. Tabulka dimenze má menší počet výskytů než počet výskytů v tabulce faktů.



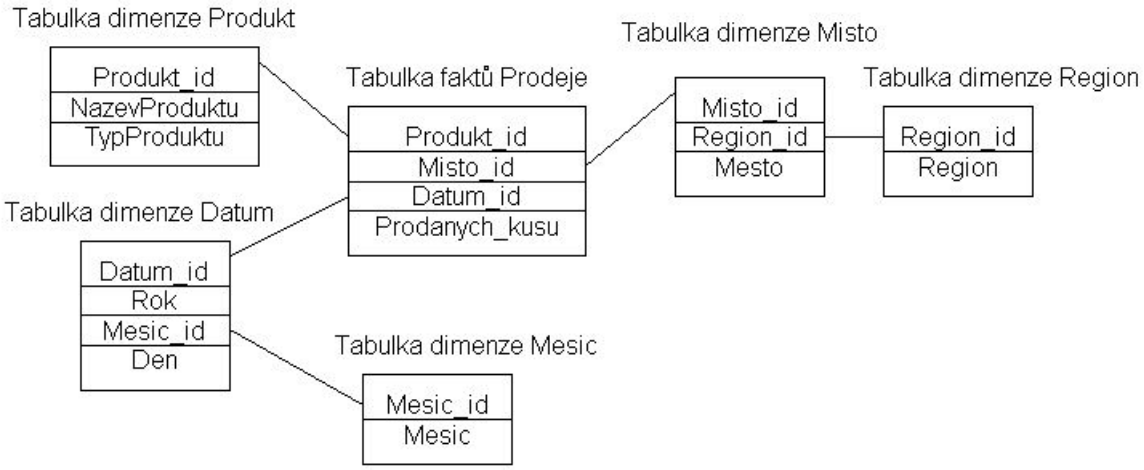
Obrázek 2.3: Schéma hvězda

Typický předmětem, majícím mnoho výskytů, který se může vyskytovat v tabulce faktů, mohou být objednávky. Nebo tabulka faktů by mohla obsahovat záznamy zákazníků. Nebo tabulka faktů může reprezentovat bankovní transakci. V každém případě tabulka faktů reprezentuje data, která se vyskytují mnohokrát. Tabulka dimenze obsahuje relevantní, přesto oddělenou, informaci (jako je kalendář, tabulky cen, umístění obchodů, prostředky dopravy objednávek, a tak dále). Tabulka dimenzí definuje důležitou, a přesto pomocnou, informaci, která se vztahuje k tabulce faktů. Tabulka faktů a tabulka dimenzí jsou vztaženy existencí společné jednotky dat. Například tabulka faktů může obsahovat data "týden 21." V tabulce dimenze je záznam s hodnotou "týden 21." Týden 21 může být například specifikován v tabulce dimenzí jako "Duben 19–26" . A tabulka dimenze může obsahovat informaci, že týden 21 neobsahuje svátky a byl třetím týdnem ve vykazovacím období společnosti.

Schéma sněhová vločka a souhvězdí

Schéma sněhová vločka vznikne ze schématu hvězda normalizací tabulek dimenzí. Nově vytvořené tabulky se nazývají lookup tabulky. Někdy se sdílené dimenze nazývají podřízené dimenze. Obrázek 2.4 obsahuje schéma sněhová vločka.

Někdy může být potřeba použít více tabulek faktů. Tabulky faktů sdílí tabulky dimenzí. Schéma, které takto vznikne, se nazývá souhvězdí.



Obrázek 2.4: Schéma sněhová vločka

Velká výhoda multidimenzionálního návrhu je jeho efektivita přístupu. Když je navržena správně, schéma hvězda je velmi efektivní v doručování dat koncovému uživateli. Aby bylo doručování informací efektivní, požadavky koncového uživatele musí být shromážděny a přizpůsobeny. Procesy koncového uživatele budou používat data, která jsou definována v multidimenzionální struktuře. Když jsou jednou požadavky koncového uživatele známy, jsou tyto použity k utvoření schématu hvězda do její konečné, nejoptimálnější struktury.

Rozdíly mezi modely

Rozdílů mezi schématem hvězda a relační strukturou jako báze pro návrh datového skladu je mnoho. Jeden nejdůležitější rozdíl je ve smyslu flexibility a výkonu. Relační model je velmi flexibilní, ale není optimalizován pro výkon pro některého uživatele. Multidimenzionální model je vysoce efektivní při uspokojení požadavků v prostředí jednoho uživatele, ale není dobrý, pokud jde o flexibilitu. Další důležitý rozdíl těchto dvou přístupů k návrhu databáze je v rozsahu působnosti návrhu. Multidimenzionální návrh nutně má limitovaný rozsah. Protože zpracování požadavků jsou použita k podobě modelu, návrh se začne hroutit, pokud je shromážděno mnoho zpracování požadavků. Jinými slovy návrh databáze může být optimalizován pro jednu a pouze jednu množinu požadavků ke zpracování. Když je úplně jiná množina požadavků na zpracování přidána do návrhu, optimalizace se stane spornou otázkou. Návrh databáze může být optimalizován pro výkon jen jedním způsobem. Když se použije relační model, pak není zvláštní optimalizace na výkon jedním způsobem nebo jiným. Protože relační model ukládá data na nejnižší úrovni granularity, nové prvky dat mohou být přidány "do nekonečna" (ad infinitum). V relačním modelu prostě není konec v přidávání dat. Z tohoto důvodu je relační model vhodný pro velmi široký rozsah dat (jako je podnikový model), zatímco multidimenzionální model je vhodný pro malý rozsah dat (jako je oddělení nebo dokonce pododdělení).

2.2 Nástroje OLAP

Dělení OLAP nástrojů dle uložení dat

OLAP nástroje se dle uložení dat v datovém skladu dělí (viz [6]) na

- MOLAP (Multidimenzionální OLAP)
- ROLAP (Relační OLAP)
- HOLAP (Hybridní OLAP)

Server MOLAP ukládá všechna jeho data na disku ve strukturách optimalizovaných pro multidimenzionální přístup. Během tohoto procesu se vypočítá co nejvíce možných agregovaných hodnot. Hlavní výhodou tohoto řešení je rychlost odpovědí na dotazy díky uložení v multidimenzionálních strukturách. Nevýhodou je omezený objem údajů, která může spravovat.

Server ROLAP ukládá svá data v relační databázi. Systémy ROLAP spouští SQL příkazy nad relační databází, aby se získaly údaje požadované uživateli. Výhodou těchto řešení je škálovatelnost (umožňuje ukládat velké množství dat). Nevýhodou může být vyšší čas potřebný ke zpracování dotazu.

HOLAP (Hybridní OLAP) je kombinací úložišť MOLAP a ROLAP, přičemž se využívají výhody obou úložišť a do značné míry se eliminují jejich nevýhody. Systém OLAP ukládá většinu dat v relační databázi, ale ukládá agregované hodnoty do multidimenzionálního formátu.

Dopředu vypočtené agregované (souhrnné) hodnoty jsou nezbytné pro velkou množinu dat, jinak určité dotazy nebudou moci být odpovězeny bez čtení určitého obsahu z databáze faktů. Agregáty jsou v systémech MOLAP často obrazem datových struktur z paměti, rozděleným na stránky a uloženým na disku. Systémy ROLAP ukládají agregáty v tabulkách. V některých systémech ROLAP jsou tyto explicitně spravovány serverem OLAP, v dalších jsou tabulky deklarovány jako materializované pohledy, a jsou implicitně použity, když OLAP server řeší dotaz se správnou kombinací sloupců v klauzuli group by.

Dělení nástrojů dle licence

Mezi komerční nástroje patří:

- Microsoft SQL Server 2005
- Cognos
- Informatica
- SAS
- a další.

Mezi OLAP nástroje vyvíjené v licenci open source například patří (viz [7]):

- Mondrian
- Palo
- Projekt Bee

Server PALO se řadí do kategorie MOLAP, je buňkově orientovaný, umožňuje hierarchické uspořádání a provádí OLAP operace nad daty v paměti. PALO pracuje jako OLAP server pro Excel.

Balík nástrojů projektu Bee slouží pro podporu rozhodování. Projekt Bee obsahuje nástroj ETL, OLAP server a tenký klient. Server se řadí do kategorie ROLAP (jako datový sklad používá relační databázi MySQL). OLAP server generuje SQL dotazy a spravuje paměť cache pro zrychlení provedení OLAP dotazů.

Důležitými faktory, které ovlivňují kvalitu OLAP nástrojů, jsou dokumentace a fóra, na kterých se řeší různé problémy při používání těchto nástrojů, případně jiná technická podpora. Výhodami open source OLAP nástrojů je dostupnost zdrojového kódu a také u většiny bezplatnost. Nevýhodou některých open source nástrojů je neexistence podrobné dokumentace, složitější instalace a konfigurování (např. u Mondrianu) a zejména u nových nespolehlivost. Nástroj JPivot používaný nástrojem Mondrian například neobsahuje uživatelskou dokumentaci, jen programátorskou dokumentaci generovanou ze zdrojových kódů. Některé nástroje jsou sice poskytované zdarma, ovšem za další služby musíme platit (např. dokumentace, technická podpora). Pak je sporné, zda je cena výhodou open source nástrojů. Výhodou komerčních nástrojů je jednoduchá instalace, spolehlivost, většinou podrobná dokumentace s příklady a také poskytovaná technická podpora. Tyto vlastnosti mohou být považovány za důležitější než cena a dostupnost zdrojového kódu, a mohou být důvodem použití komerčního nástroje. Nevýhodou komerčních nástrojů je nedostupnost zdrojového kódu a může také být cena.

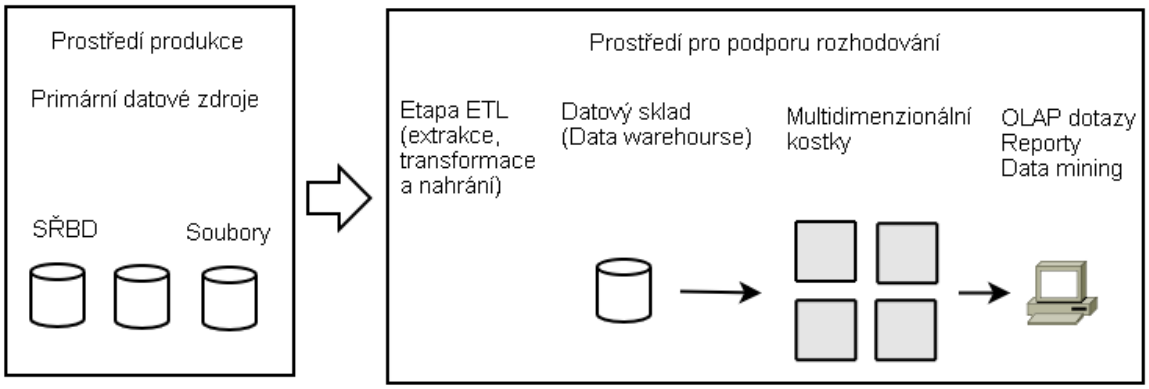
2.3 Fáze ETL

Extrakce, Transformace, Nahrání (angl. Extract, Transform, Load - ETL) je proces v problematice datových skladů (viz [5]), který zahrnuje

- extrakci dat z okolních zdrojů,
- transformaci dat, aby splnila obchodní potřeby(které může zahrnovat úrovně kvality), a konečně
- jejich nahrání do koncového cíle, např. datového skladu.

Do etapy ETL se často zahrnuje také fáze čištění (angl. cleaning nebo cleansing). ETL je důležitý proces, protože je to způsob, jak se data vlastně nahrávají do datového skladu. Na obrázku 2.5 jsou jednotlivé fáze budování datového skladu včetně důležité fáze ETL. Nad daty z primárních datových zdrojů vytvoříme datový sklad, který bude plněn etapou ETL. Zdroje dat mohou být různorodé. Některá data mohou pocházet z jedné databáze, jiná z druhé databáze a některá data jsou získávána z textových souborů (např. csv soubory). Úkolem etapy ETL je sjednotit formát těchto dat a uložit je do datového skladu. Data bývají uložena (u serverů MOLAP) nebo jsou transformována (ROLAP) na multidimenzionální struktury (datové kostky) tak, aby bylo umožněno klást dotazy, které nemusí být předem známy (např. dotazy upřesněné časovým obdobím nebo místem včetně jejich kombinací). Důvody, proč je pro data často vytvářen nový datový sklad a nevyužívá se skutečnosti, že data jsou již uložena v primární databázi transakčního prostředí, jsou požadavek na nezpomalení transakčního prostředí, různé formáty dat z několika datových zdrojů, z čehož plyne nutnost sjednocení, a také různorodost (multidimenzionálnost) dotazů, a tím pádem speciální uložení dat. Budeme předpokládat, že data jsou vždy nahrána do datového skladu,

zatímco pojem ETL může ve skutečnosti znamenat proces, který nahrává data do jakékoliv databáze. ETL může být také použit k integraci s dřívějšími systémy. Obvykle implementace ETL ukládá audit ke sledování pozitivních a negativních provedení procesu. V téměř všech návrzích není tento audit ke sledování na úrovni granularity, který by dovoloval reprodukovat výsledek ETL, když prvotní data nebyla dostupná.



Obrázek 2.5: Od zdrojových dat k OLAP dotazům

Extrakce

Úkolem první části procesu ETL je extrakce dat ze zdrojových systémů. Většina projektů z oblasti datových skladů slučuje data z různých zdrojových systémů. Každý jednotlivý systém může také používat různou organizaci nebo formát dat. Běžnými formáty datových zdrojů jsou relační databáze a soubory, ale mohou zahrnovat relační databázové struktury, které nejsou relační, jako jsou IMS nebo další datové struktury (např. VSAM nebo ISAM). Extrakce převádí tato data do formátu pro transakční zpracování.

Parsování extrahovaných dat je důležitou částí extrakce, a jeho výsledkem je kontrola, zda data vyhovují očekávaným vzorům nebo strukturám. Pokud ne, data jsou úplně odmítnuta.

Transformace

Transformační fáze aplikuje sérii pravidel nebo funkcí na extrahovaná data ze zdroje k odvození dat, která se nahrávají do koncového cíle. Některé datové zdroje budou vyžadovat velmi malou nebo dokonce žádnou manipulaci s daty. V jiných případech může tato fáze vyžadovat jeden nebo více následujících typů transformací ke splnění obchodních a technických požadavků konečných cílů:

- Vybrání pouze určitých sloupců k nahrání (nebo vybrání sloupců s hodnotou null, které se nebudou nahrávat).
- Převést kódované hodnoty (např. pokud zdrojový systém ukládá údaj o pohlaví jako 1 pro pro muž a 2 pro žena, ale datový sklad ukládá M pro muž a Ž pro žena), toto se nazývá automatizované datové vyčistění; žádné manuální vyčištění se neobjeví během ETL.
- Zakódování hodnot volného formátu (např. mapování “Muž”, “1” a “Pan” na M).

- Odvození nově vypočtené hodnoty (např. suma_prodejů = počet * cena_za_jednotku).
- Spojení dat z různých zdrojů (např. vyhledání, slučování atd.)
- Shrnutí několika řádků dat (např. celkové prodeje pro každý obchod a pro každý region).
- Generování náhradních klíčových hodnot.
- Transpozice nebo výběr hlavního prvku (pivoting) - převod několika sloupců na několik řádků nebo naopak.
- Rozdělení sloupce na několik sloupců (např. převod seznamu odděleného čárkou typu string v jednom sloupci na samostatné hodnoty v různých sloupcích).
- Aplikace některé jednoduché nebo podrobnější validace dat; pokud neúspěšná, plné, částečné nebo žádné odmítnutí dat, a tedy žádná, některá nebo všechna data se předají do dalšího kroku, v závislosti na návrhu pravidel a zpracování výjimek. Většina výše uvedených transformací může mít za výsledek výjimku, např. když převod kódu narazí na neznámý kód extrahovaných dat.

Čištění

Data, která získáváme v etapě extrakce, mohou pocházet z různých zdrojů, což je důvodem toho, že data obsahují chyby. To může být například dáno tím, že aplikace, která produkuje data v primárním transakčním prostředím, neošetřovala hodnoty vstupu. Chybným datům se mohlo předejít tak, že by atributy, které mohou nabývat jen některých hodnot (např. pohlaví), mohly být v aplikaci vybírány z rozbalovacího menu, přepínače či dalších prvků, nebo také upozorněním uživatele na chybu a vyzváním k opětovnému zadání. Ovšem v etapě ETL často přicházejí na vstup i špatná data a je třeba tento problém nějak vyřešit. Jednou z možností je změna hodnoty atributu na standardní hodnotu. Další možností by mohla být oprava hodnot uživatelem, který používá ETL nástroj, ovšem to by vyžadovalo znalost uživatele, které nové hodnoty použít. Ale vzhledem k tomu, že etapa ETL je automatizovaná a hodnot vyžadujících opravu by mohlo být mnoho, tak tento krok se téměř vůbec neprovádí. Ignorování záznamu, a tím jeho neuložení do datového skladu, je další možností. Záznam můžeme ignorovat zejména v případě, že těchto záznamů není mnoho. V prostředí datových skladů a OLAP analýzy není, na rozdíl od transakčního prostředí, správnost dat až tak důležitá. Datové sklady obsahují velké množství záznamů a několik špatných se ve výsledcích dotazů nemusí projevit. Některé ETL nástroje požadují data, která zajišťují kompletní data splňující integritní omezení.

Nahrání

Nahrávací fáze (load) nahrává data do koncového cíle, obvykle datového skladu. Tento proces se velmi liší v závislosti na požadavcích organizace. Některé datové sklady mohou týdně přepisovat existující informaci souhrnnými, aktualizovanými daty, zatímco jiné datové sklady (nebo dokonce jiné části stejného datového skladu) mohou přidávat nová data v historickém tvaru např. každou hodinu. Časování a rozsah přepsání nebo přidání jsou na výběr u strategického návrhu v závislosti na dostupném času a obchodních požadavcích. Složitější systémy mohou udržovat historii a audit ke sledování všech změn dat nahraných do datového skladu.

Protože fáze nahrání pracuje s databází, omezení (angl. constraints) definovaná v databázovém schématu stejně jako spouštěče (triggers) aktivované během nahrání dat se použijí (např. jedinečnost, referenční integrita, mandatorní pole). Tyto také přispívají k celkové kvalitě dat procesu ETL.

Inkrementální nahrání

Nahrání se dělí na plné (angl. full loading) a inkrementální (angl. incremental loading), někdy nazývané přírůstkové. Hlavním rozdílem je, že plné nahrání nahrává všechna data znovu, i když už bylo velké množství těchto dat nahráno v předchozích nahráních. Oproti tomu inkrementální nahrávání pouze nahrává změny (přírůstky) oproti minulému stavu. Plné a inkrementální nahrávání se rozlišuje především u plnění tabulek faktů, které mají více záznamů než tabulky dimenzí, a navíc se tabulky dimenzí po prvotním nahrání téměř nemění. Před plným nahráním se obvykle vyprázdní obsah tabulek faktů datového skladu. Rozdíl těchto druhů nahrání je v rychlosti provádění a efektivitě. Plné nahrání může znamenat ztrátu některých dřívějších dat, pokud se zdrojová data vymazala a nebyla vytvořena záloha. Je zřejmé, že inkrementální nahrání, protože nahrává jen přírůstky, a tím menší objem dat, trvá podstatně kratší dobu, a je také efektivnější, jelikož se neukládají data, která byla uložena při minulém nahrání.

Techniky inkrementálního nahrání jsou:

- Zachycení změn pomocí aplikace
- Zachycení změn pomocí databáze
  - Databázové triggery
  - Analýza databázových logů
  - Použití časových razítek (angl. time stamp)
  - Porovnání souborů (angl. file compare)

Zachycení změn pomocí aplikace znamená postup, při kterém se informuje ETL nástroj, že jsou k dispozici modifikace, a ten již provede nahrání změn do datového skladu. Tato technika vyžaduje u většiny aplikací jejich úpravu.

Inkrementální nahrání technikou databázových triggerů znamená, že se při změně databáze (nejčastěji při vložení nového záznamu) zavolají kroky etapy ETL. Zachycení změn pomocí databáze nevyžaduje změnu aplikace. Nevýhodou této techniky je nutnost podpory triggerů v SŘBD.

Některé SŘBD vytvářejí logy, což jsou informace o provedených databázových operacích (vložení, změna a smazání záznamů). Jedna z technik využívá tyto informace pro přírůstkové nahrání. Tento postup ovšem vyžaduje znalost logů, které SŘBD vytváří. Tato technika také nevyžaduje změnu aplikace používané v transakčním prostředí.

Pro účel rozpoznání změn v databázi může být vytvořen další atribut, který bude obsahovat časový údaj změny (time stamp, v překladu časové razítko). Tento atribut pak bude použit pro výběr záznamů, které se budou ukládat do datového skladu.

Poslední technikou je porovnávání souborů. Tato technika se používá, pokud není možné použít žádnou z předchozích technik. Provede se extrakce celého objemu dat do souborů a porovnají se změny souborů z předchozího stavu stavu databáze a současného stavu databáze. Operace porovnání souborů může trvat dlouho, pokud soubory obsahují velké objemy dat.

2.3.1 ETL nástroje

Důležitým rozhodnutím je, zda použijeme k plnění datového skladu některý již existující ETL nástroj nebo si vytvoříme svůj vlastní. Výhodou použití existujícího ETL nástroje je připravený ETL nástroj, spolehlivost a asi patrně také úspora času. Nevýhodou může být učení se práci v novém nástroji a také riziko, že tento nástroj nebude schopen provést některé operace, zejména z etapy transformace. Výhodou vlastního nástroje je kontrola nad celým ETL procesem, jelikož si vše naprogramujeme dle vlastních představ. Nevýhodou je čas potřebný k vytvoření takového nástroje, jelikož tato práce nás může stát velké množství času, a také požadavek na programovací znalosti.

Přehled některých ETL nástrojů (viz [7]):

- Octopus
- Scriptella
- Talend
- Clover.ETL
- Pentaho Data Integration
- JasperETL
- Oracle Warehouse Builder
- Microsoft SQL Server Integration Services

Prakticky všechny ETL nástroje pracují nad relační databází. Poslední dva jmenované produkty jsou zástupci komerčních nástrojů a zbylé jsou vyvíjené v licenci open source. Výhodou komerčních ETL nástrojů je většinou kvalitní dokumentace, poskytovaná podpora a také fakt, že jsou tyto ETL nástroje často součástí balíku pro OLAP analýzu a podporu rozhodování (např. Microsoft SQL Server Integration Services). Nevýhodou komerčních nástrojů může být cena. Výhodou open source ETL nástrojů je jejich cena (většina je poskytována zdarma). JasperETL a Pentaho Data Integration jsou open source ETL nástroje, které jsou součástí balíku pro podporu rozhodování od jedné firmy. Open source ETL nástroje se, alespoň co se týče funkčnosti, vyrovnají komerčním nástrojů. Společnou vlastností výše jmenovaných nástrojů je popis transformací jako sekvence kroků v textovém formátu nebo pomocí GUI (grafického uživatelského rozhraní), ale bez popisu implementace. Většina nástrojů má GUI rozhraní. Scriptella je ETL nástrojem, který GUI neobsahuje, a je třeba definovat kroky etapy ETL v XML souboru. Některé open source nástroje se inspirují jeden od druhého. Například nástroj JasperETL je založen na nástroji Talend.

Pentaho Data Integration Na webu je možné najít několik open source ETL nástrojů implementovaných v Javě. Balík Pentaho, do kterého patří také použitý OLAP server Mondrian, obsahuje jeden takový ETL nástroj zvaný Pentaho Data Integration (dříve Kettle, v překladu hrnec). Tento nástroj se skládá z několika nezávislých nástrojů. Ke každému nástroji je vytvořena podrobná dokumentace. Nástroj Spoon (česky lžíce) se používá k návrhu sekvencí jednotlivých kroků modelu v podobě toku dat ze vstupu přes transformace k výstupu pomocí grafického uživatelského rozhraní (GUI). Takový model se nazývá transformace. Tento nástroj obsahuje několik předem připravených objektů pro nejpoužívanější operace. Mezi ně například patří objekty `CSV file input` pro načtení vstupu

z CSV souborů (comma-separated values, v překladu čárkou oddělené hodnoty), `Select Values` pro výběr hodnot, `Filter rows` pro výběr hodnot, `Row denormalizer` pro denormalizaci řádků a XML output pro výstup do souboru XML. Výhodou tohoto nástroje možnost zobrazení si výsledků jednotlivých kroků etapy ETL. Pokud potřebujeme provést některou operaci, kterou tento nástroj nemá k dispozici, může si ji naprogramovat j jazyce Javascript. Na obrázku 2.6 je diagram vytvořený v nástroji Spoon. Diagram obsahuje kroky načtení dat z CSV souboru, výběr atributů a výstup do textového souboru. Nástroj Pan (česky pánev) je nástroj pracující v prostředí příkazové řádky a spouští transformace vytvořené v nástroji Spoon. Nástroj Chef (kuchař) je GUI nástroj, který se používá k modelovaní úkolů. Úkoly se skládají z například z kroků transformace a přenosu po síti, které jsou provedeny v toku řízení. Nástroj Kitchen (kuchyň) je nástroj příkazové řádky a používá se ke spuštění úkolů vytvořených v nástroji Chef. Tato koncepce několika nástrojů, z nichž každý provádí svůj úkol, je v oblasti open source nástrojů obvyklá.

Obrázek 2.6: Diagram části etapy ETL v nástroji Spoon

Kapitola 3

Popis problému

3.1 Aleph

Knihovní systém Aleph spravuje data knihoven VUT. Tento automatizovaný systém vyvíjí společnost Ex Libris. Aleph patří mezi systémy ILS (Integrated Library System, v překladu Integrovaný knihovní systém). Integrovaný systém Aleph používají akademické, výzkumné a národní knihovny po celém světě už přes 20 let. Aleph má vícevrstvou architekturu typu klient-server a poskytuje efektivní uživatelsky přívětivé nástroje a podporu toků řízení. Systém ILS spravuje informace o různých událostech, mezi které patří správa informací o dokumentech, jednotkách (konkrétní výtisk dokumentu), čtenářích, výpůjčkách, výpůjčkách mezi knihovnami, platbách, rezervacích a dalších událostech. Systém Aleph také umožňuje čtenářům vyhledat si v internetovém rozhraní informace o dokumentech. Data všech knihoven jsou obvykle uložena v jedné databázi. Na základě statutu čtenáře (student, zaměstnanec a další) se nastaví termín výpůjčky. Ke katalogizaci dokumentů se používá formát Marc (viz [2]). O dokumentech se evidují např. údaje o jazyku, žánru (číselný kód MDT - mezinárodní desetinné třídění), materiálu (např. knihy, časopisy) a další údaje.

Mezi výhody integrovaného systému Aleph patří:

flexibilita – systém Aleph je možné upravit dle požadavků institucí všech typů a velikostí od malé knihovny pro jeden obor po největší knihovny institucí a národní knihovny.

jednoduchost použití – uživatelsky příjemné uživatelské rozhraní.

škálovatelnost – do systému je možné přidat další funkce na základě požadavků prostředí.

podpora jazyků – plná podpora kódování Unicode zajišťuje možnost zvolení preferovaného jazyka uživatelského rozhraní, popřípadě vytvoření nových textů.

3.2 Schéma databáze knihoven

Knihovní systém Aleph (viz [3]) ukládá informace o knihách, výpůjčkách, poplatcích atd. všech knihoven VUT do jedné databáze Oracle. Atribut SUB_LIBRARY určuje, o kterou knihovnu se jedná. Databáze se skládá z mnoha tabulek, které obsahují více atributů, než je použito v tomto nástroji pro datový sklad pro OLAP analýzu. Tato aplikace bude pro vytvoření datového skladu využívat jen několik z nich:

Z00 – Dokumenty

Z30 – Jednotky

Z31 – Poplatky

Z35 – Události

Z36 – Výpůjčky

Z36H – Historie výpůjček

Z37 – Požadavek na půjčení knihy

Z37H – Historie požadavků na půjčení knihy

Z103 – Odkazy mezi záznamy

Tabulka **Z00** obsahuje bibliografické informace o dokumentech. Záznam dokumentu v systému ALEPH je základní informační záznam. Každý záznam je unikátně identifikován 9-ti číslicovým systémovým číslem přiřazeným sekvenčně systémem. Tabulka Z00 se nachází v databázi BUT01.

Tabulka **Z30** obsahuje záznamy (kopie) jednotek. Jednotka je jeden samostatný výtisk knížky. Jeden záznam může mít více jednotek. Každá jednotka, která je v oběhu (jednotka s čárovým kódem), musí mít záznam Z30. Záznam Z30 obsahuje informaci o identifikačním kódu záznamu (Z30_REC_KEY), o knihovně, ve které se kopie nachází (atribut Z30_SUB_LIBRARY), materiálu (Z30_MATERIAL), statutu jednotky (Z30_ITEM_STATUS), procesním statutu jednotky (Z30_PROCESS_STATUS), datu vložení (Z30_OPEN_DATE - číslo ve formát rrrrmmdd), datu změny (Z30_UPDATE_DATE), počtu výpůjček (Z30_NO_LOANS), ceně jednotky (Z30_PRICE) atd.

Tabulka **Z31** obsahuje záznamy o finančních transakcích uživatelů (poplatky a pokuty). Tato tabulka obsahuje atributy Z31_REC_KEY (identifikační číslo platby), Z31_DATE (datum platby), Z31_STATUS (status, který nabývá hodnot W, O, C, T atd.), Z31_SUB_LIBRARY (kód knihovny), Z31_TYPE (typ - kód pro výpůjčku, kopírování, pozdní vrácení, ztracenou knihu), Z31_DATE_X (datum platby), Z30_SUM (částka) a další.

Tabulka **Z35** obsahuje informace o různých transakcích (výpůjčky, vrácení, držení, žádosti o fotokopie a vyhledávání). Tato tabulka obsahuje atributy Z35_REC_KEY (číslo záznamu), Z35_ITEM_SEQUENCE (sekvenční číslo) Z35_MATERIAL (typ materiálu), Z35_SUB_LIBRARY (kód knihovny), Z35_EVENT_DAY a Z35_EVENT_HOUR (datum a čas transakce), s Z35_EVENT_TYPE (typ transakce), Z35_ITEM_STATUS (status jednotky), Z35_BOR_STATUS (status čtenáře), Z35_TIME_STAMP (časové razítko události). Atribut Z35_MATERIAL může mít hodnoty BOOK, VIDEO, ISSUE, AUDIO (kniha, video, číslo časopisu, audio) a další. Atribut Z35_EVENT_TYPE může například nabývat hodnot

61 (vrácení jednotky), 62 (výpůjčka) atd. Atribut Z35_TIME_STAMP je vhodné použit v inkrementálním nahrání v etapě ETL.

Tabulka **Z36** obsahuje údaje o výpůjčkách (angl. loans). Jejími atributy jsou Z36_REC_KEY (klíč záznamu), Z36_ID (čtenářovo ID), Z36_MATERIAL (typ materiálu), Z36_SUB_LIBRARY (kód knihovny), Z36_STATUS (status výpůjčky), Z36_LOAN_DATE a Z36_LOAN_HOUR (datum a čas výpůjčky), Z36_DUE_DATE a Z36_DUE_HOUR (datum a čas splatnosti), Z36_RETURNED_DATE a Z36_RETURNED_HOUR (datum a čas vrácení), Z36_STATUS (status výpůjčky), Z36_ITEM_STATUS (status jednotky), Z36_BOR_STATUS (status čtenáře) a další.

Tabulka **Z36H** obsahuje historie výpůjček (loans history). Tato tabulka obsahuje podobné atributy jako Z36 s tím rozdílem, že mají jiný prefix (Z36H místo Z36). Navíc má Z36H oproti tabulce Z36 atribut Z36H_TIME (datum a čas výpůjčky).

Tabulka **Z37** obsahuje informace o požadavcích na půjčení knihy nebo jiného média (hold requests). V této tabulce se uchovávají záznamy pro požadavky na půjčené jednotky (čekací požadavky - angl. waiting requests) a pro požadavky na doručení dostupných jednotek. Z37 obsahuje atributy Z37_REC_KEY (klíč záznamu), Z37_ID (ID žadatele), Z37_STATUS (stav požadavku na půjčení), Z37_SOURCE (zdroj požadavku), Z37_PRIORITY (priorita požadavku ve frontě), Z37_OPEN_DATE a Z37_OPEN_HOUR (datum a čas, kdy byla jednotka požadována), Z37_REQUEST_DATE (počáteční datum, od kdy má žadatel zájem obdržet materiál) a Z37_END_REQUEST_DATE (koncové datum čekání), Z37_HOLD_DATE (datum poslání oznámení čtenáři, že si může jednotku vyzvednout), Z37_END_HOLD_DATE (datum, do kdy se bude jednotka rezervovat pro čtenářovo vyzvednutí) a další.

Tabulka **Z37H** obsahuje historii požadavků na půjčení knihy (hold requests history). Rozdíl mezi Z37 a Z37H je přidání atributu Z37H_TIME na začátek tabulky Z37H. Když je požadovaná jednotka půjčena, záznam o požadavku je smazán z tabulky Z37 a je zapsán do tabulky Z37H, když je příznak (flag) TAB10_CREATE_Z36H nastaven na "Y".

Tabulka **Z103** obsahuje odkazy mezi záznamy (links between records). Odkaz je ukazatel mezi záznamy databáze, jako například mezi bibliografickými záznamy (Bibliographic Records) nebo mezi bibliografickým záznamem a administrativním záznamem (Administrative Record), nebo mezi bibliografickým záznamem a záznamem držení (Holdings Record). Tato tabulka obsahuje atributy Z103_REC_KEY (klíč záznamu), Z103_REC_KEY_1 (klíč odkazujícího záznamu - záznam "from"), Z103_LKR_LIBRARY (kód knihovny adresovaného záznamu - záznam "to"), Z103_LKR_DOC_NUMBER (systémové číslo adresovaného záznamu), Z103_LKR_TYPE (typ odkazu) atd.

3.3 Dotazy

Aplikace bude generovat statistiky pro zadané dotazy. Tyto statistiky bude možné omezit na všechny nebo jen pro jednotlivé knihovny, pro celou historii nebo pro časové období specifikované roky, měsíci, týdny a dny, podle statutů (status jednotky, čtenáře a události), typu (typ události), materiálu (knihy, časopisy) a tématu dokumentu (jazyková literatura, historické dokumenty, technické manuály atd.). Z tohoto plyne důležitá vlastnost OLAP analýzy, díky které je možné odpovídat na takové dotazy, a tou je dimenzionální orientace. Tyto atributy bude možné kombinovat a takové dotazy se nazývají analytické (OLAP). Uživatel tedy bude moci klást na aplikaci ad hoc dotazy, které nejsou předem známé, bez znalosti dotazovacího jazyka, a proto se pro tyto účely nepoužívá jazyk SQL nad daty transakční databáze. Tyto dotazy budou kladeny OLAP serveru, který bude dotazy vy-

hodnocovat a vypisovat výsledky dotazů. Aplikace by měla vytvářet statistiky pro tyto dotazy:

- Jak dlouho mají průměrně čtenáři knihovny půjčené knihy?
- Jaké jsou podíly výpůjček knih, CD-ROM, diplomových prací a dalších?
- Kolik bylo vypůjčených knih v daném období v dané knihovně?
- Kolik bylo požadavků na půjčení knih v daném období?
- Jaký je průměrný počet dnů pozdního vrácení pro danou kategorii čtenářů?
- Jaké byly poplatky za pozdní vrácení v daném období?
- Jaké je rozdělení knihovního fondu dle žánru (naučné, jazykové učebnice atd.) či jazyka (české, anglické a další)?
- Kolik je knih v knihovnách?

Výstupem odpovědí na tyto dotazy budou tabulky a také grafy.

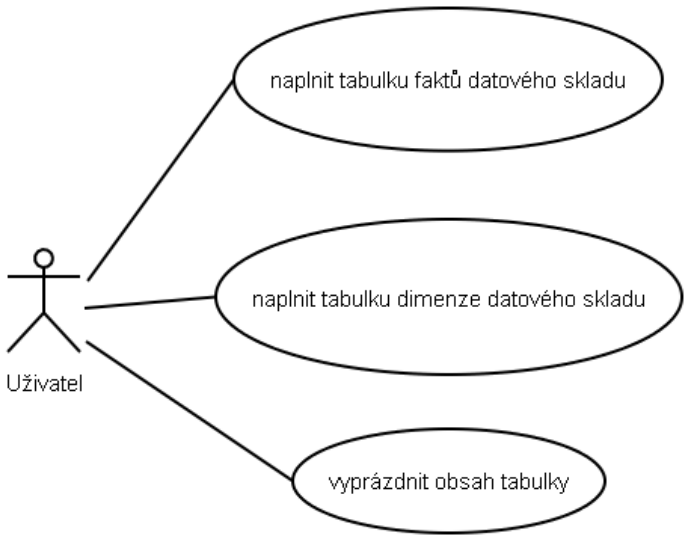
3.4 Požadavky na software

OLAP server, který bude v této práci použit, by měl být, jak plyne ze zadání, dodáván v licenci open source. Toto bude hlavní požadavek na OLAP server. Dalšími faktory, které budou vzaty v úvahu při výběru, jsou existence uživatelského rozhraní pro zobrazení výsledků OLAP operací (roll up a další), dokumentace, fóra a ukázkové aplikace. Ukázková aplikace by měla přiblížit práci s OLAP serverem, usnadnit počáteční kroky a minimalizovat riziko, že bude část práce v tomto nástroji vykonána a bude se přecházet na jiný nástroj. Uživatelské rozhraní musí umožňovat tvorbu libovolných dotazů dle požadavků uživatele bez znalosti dotazovacího jazyka.

Pokud bude OLAP server používat relační databázi jako datový sklad (kategorie serverů ROLAP), bude jako datový sklad použita databáze MySQL. Důvodem použití MySQL je široké rozšíření této databáze a typ licence (open source). Databáze MySQL se zdá vhodná pro tento projekt i proto, že obsahuje ekvivalenty pro datové typy atributů zdrojové databáze Oracle. Definice dat (vytvoření tabulek, definice integritních omezení) bude umístěna v sql souboru. Tento soubor bude obsahovat SQL příkaz CREATE TABLE.

Na obrázku 3.1 je diagram případů použití pro nástroj ETL, který bude vykonávat tyto funkce: plnění tabulek faktů datového skladu, plnění tabulek dimenzí a vymazání obsahu tabulek. Vymazání obsahu tabulek se bude využívat při prvotním nahrání dat do datové skladu. V systému je uvažován jeden typ uživatele. Zdrojová data budou uložena v textových souborech ve formátu csv. Nástroj bude provádět kontrolu hodnot dat. Plnění datového skladu by mělo být inkrementální, pokud to data v databázi umožňují. Inkrementální plnění znamená, že se nahrávají pouze přírůstky oproti minulému stavu, a tím se by se mělo znatelně urychlit nahrání. Požadavek inkrementálního nahrání je dán charakterem dat. Ze zdrojové databázi jsou vybírána především historická data, která se už nemění (např. výpůjčky v tabulce z36h nebo události v tabulce z35). Nejdříve bude proveden užší výběr již existujících nástrojů dodávaných v licenci open source, freeware nebo podobných. Pokud nebude žádný z nástrojů vyhovovat, začne se co nejdříve implementovat

vlastní ETL nástroj. Vlastní ETL nástroj by byl implementován v jazyce Java. Hlavním důvodem je přenositelnost na mnoho platforem. ETL nástroj bude pracovat jako konzolová aplikace. Parametry nástroje (funkce, cesta k vstupnímu souboru a další) budou předávány přes parametry příkazové řádky. Výhodou konzolové aplikace je možnost spuštění ze skriptu v terminálu. Pak je možné nastavit načasování spuštění etapy ETL příkazem cron. Inkrementální nahrání se bude volit uvedením přepínače v parametrech příkazové řádky. Tabulky dimenzí budou definovány v textových souborech ve formátu csv.



Obrázek 3.1: Diagram případů použití

Kapitola 4

Implementace

4.1 Mondrian

Mondrian je open source OLAP server, který bude použit k OLAP analýze dat knihoven VUT. V zadání této diplomové práce je napsáno, že by OLAP nástroj měl být ideálně open source, což znamená, že zdrojový kód je dostupný zájemcům. Právě tento požadavek byl hlavním požadavkem, dle kterého výběr probíhal. Dalšími důvody tohoto výběru jsou kvalitní dostupná dokumentace (viz [8]) a také flexibilita (možnost budoucího rozšíření datového skladu a aplikace). Oproti tomu například dokumentace jiného OLAP serveru Palo byla zdarma dostupná jen zčásti. Mondrian je OLAP server napsaný v programovacím jazyku Java a pracuje nad několika relačními databázemi, které slouží pro uložení datového skladu. K databázím se přistupuje přes datový zdroj JDBC. Mondrian umožňuje pomocí jazyka MDX interaktivně analyzovat velmi rozsáhlé množiny dat uložené v relačních databázích. Mondrian provádí interaktivní analýzu malých i velkých objemů informací. Umožňuje dimenzionální zkoumání dat, například analýzu prodejů dle typů produktu, regionu nebo časového období. Mondrian převádí zápisy v jazyce MDX na příkazy jazyka SQL (Standard Query Language) k získání odpovědí na dimenzionální dotazy. Jádro Mondrianu je JAR archív, který se chová jako "JDBC pro OLAP" (zajišťuje spojení a vykonání SQL příkazů nad relační databází). Tento archív může být spuštěn z okolních aplikací. V Mondrianu je možné používat rychlé dotazy prostřednictvím agregovaných tabulek v relačním databázovém systému. V tomto OLAP serveru lze také spouštět pokročilé výpočty použitím výrazů jazyka MDX. Podrobný popis instalace Mondrianu v prostředí Linux je v dodatku A.

4.1.1 MDX

MDX je zkratka pro MultiDimensional eXpressions (Multidimenzionální výrazy). Je to hlavní dotazovací jazyk implementovaný v Mondrianu. MDX byl uveden společností Microsoft v jejím produktu Microsoft SQL Server OLAP services okolo roku 1998 jako komponenta jazyka OLE DB pro OLAP API. Později se jazyk MDX objevil jako část XML for Analysis API. Microsoft navrhuje MDX na standard a jeho použití mezi programátory aplikací a jinými OLAP správci se pravidelně zvyšuje. V syntaxi jazyka MDX v Mondrianu a Microsoft SQL Serveru jsou drobné rozdíly. Ovšem několik funkcí (např. funkce Crossjoin, která vrací kartézský součin dvou množin) jazyka Visual Basic, které se používají v dotazech pro systém SQL server, je implementováno i v systému Mondrian. Výčet rozdílů je shrnut v [8].

Jedním ze způsobů, jak zapsat MDX dotaz pro server Mondrian, je zapsat tento do těla prvku `<mondrianQuery>`. Tento prvek má několik atributů. Prvním atributem je *jdbcDriver*, kterým se nastavuje ovladač, prostřednictvím kterého se bude přistupovat k datovému skladu. Další atribut *jdbcUrl* obsahuje adresu url k datovému JDBC zdroji. Atribut *catalogUri* obsahuje cestu k XML schématu kostek, dimenzí, hodnot a dalších prvků. XML schéma je popsáno v sekci 4.3. Připravený MDX dotaz, který chceme vykonat serverem Mondrian a jehož výsledek chceme zobrazit, může obsahovat chyby. Chyby mohou být, jak bývá i u programování v jiných jazycích, syntaktické nebo sémantické. Jak již bylo zmíněno výše, syntaxe jazyka MDX a SQL server se mírně liší. A to může vést k chybách. Jedna z chyb, která nastala při vývoji systému pro analýzu knihovních dat, byla způsobena různým kódováním primárních a cizích klíčů. Chybu zjistíme tak, že v prohlížeči otevřeme jsp stránku s MDX dotazem. Pokud nastane chyba, vyvolá se výjimka a na stránku se vypíše popis výjimky. Bohužel někdy chybová hláška moc nepomůže (např. Mondrian Internal Error, v překladu interní chyba Mondrianu). Většina chyb je dána chybou dotazu nebo schématu. A protože se při vykonávání MDX dotazu specifikuje soubor s multidimenzionálním XML schématem ve formátu XML, je vhodné nejprve opravit chyby v XML schématu.

Nastavení serveru Mondrian definujeme v souboru mondrian.properties. Příkazem `mondrian.result.limit=0` zrušíme omezení na velikost výsledku. Asi nejdůležitějším nastavením je nastavení výpisu dotazů SQL, které se generují z MDX dotazů. Pro toto chování je třeba vložit řádek `mondrian.trace.level=1` do tohoto souboru. Tohoto využijeme při vývoji a ušetří nám to čas při odstranění chyb. Generované SQL dotazy se, pokud spouštíme aplikační server Tomcat jako službu, vypisují do logu catalina_datum.log (např. catalina_2008-04-28.log) do adresáře logs. SQL dotaz se pak vypisuje při každém spuštění MDX dotazu (otevření jsp stránky nebo OLAP operace - roll up, drill down a další - spuštěné z uživatelského rozhraní JPivot). Vypsaný SQL dotaz můžeme spustit v databázové konzoli (příkaz mysql pro databázi MySQL) a můžeme tak zjistit chybu, která je většinou přesnější, než hláška Mondrianu. Pokud vše proběhne bez chyb, zobrazí se nám na stránce tabulka s výsledkem dotazu. Při výše zmíněné chybě byla například vypsána chyba Resultset overflow occured (mělo dojít k přetečení výsledku). Z MDX dotazu

```
select
  {[Measures].[Pocet vypujcek]} on columns,
  {[Materialy].[Vsechny materialy]} on rows
from [Vypujcky]
```

Mondrian vygeneruje tento SQL dotaz:

```
select
    `materialy`.`popis` as `c0`,
    count(`z36h`.`time`) as `m0`
from
    `materialy` as `materialy`,
    `z36h` as `z36h`
where
    `z36h`.`material` = `materialy`.`material`
group by
    `materialy`.`popis`
```

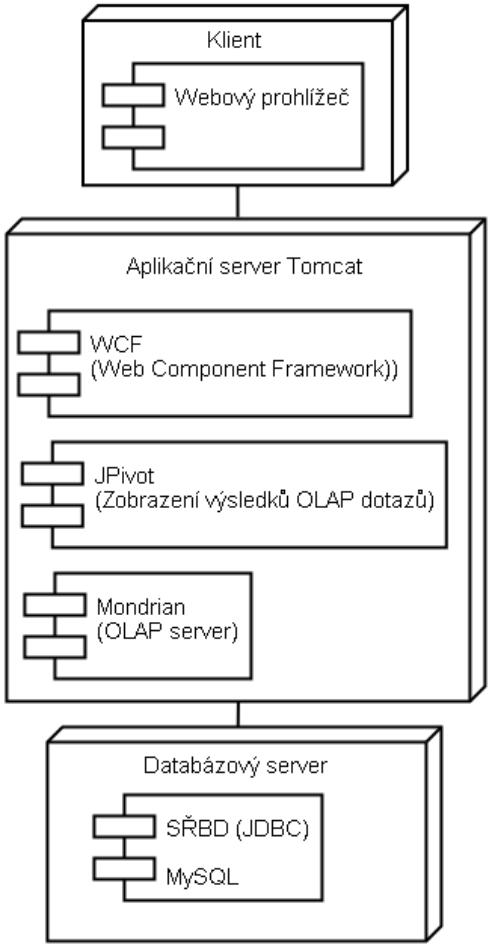
Spustíme tento příkaz v konzoli MySQL a necháme si vypsat chybu:

`(ERROR 1267 (HY000): Illegal mix of collations (utf8_general_ci,IMPLICIT) and (utf8_unicode_ci,IMPLICIT) for operation '=')`

Tento problém byl vyřešen nastavením stejné znakové sady pro obě tabulky příkazy: `alter table material convert to character set utf8;` a `alter table z36h convert to character set utf8;`.

Architektura

Architektura systému pro OLAP analýzu je na obrázku 4.1. Mondrian je distribuován jako binární balík WAR včetně komponenty JPivot, OLAP webového aplikačního frameworku a ukázkových dat ve dvou variantách. První varianta umožňuje nahrání do databáze dle našeho výběru (samozřejmě z podporovaných typů, kterých je mnoho). Druhá varianta obsahuje databázi Apache Derby a nevyžaduje žádné další nastavení před nasazením na aplikační server. Výsledek OLAP dotazů se zobrazuje na webové stránce komponentou JPivot. Knihovna JPivot je popsána v kapitole **??**. Jako databáze pro datový sklad byla použita volně dostupná MySQL. Apache Tomcat plní funkci aplikačního serveru.



Obrázek 4.1: Diagram nasazení

4.2 Schéma datového skladu

Datový sklad je uložen v relační databázi MySQL. Nad touto relační databází bude OLAP server Mondrian spouštět příkazy SQL (po převodu příkazů z jazyka MDX do SQL). Schéma datového skladu je na obrázku 4.2. Primární klíče jsou v obrázku vyznačeny tučně a cizí klíče jsou podtrženy. Tabulky faktů mají v pravém horním rohu písmeno F, tabulky dimenzí písmeno D a u hodnot je písmeno H.

Tabulka **z36h** obsahuje údaje o výpůjčkách. Atribut days_borrowed je hodnota počtu dnů, po kterou měl čtenář vypůjčenou jednotku, a počítá se jako rozdíl datumů výpůjčky a vracení ve dnech (atributy Z36H_LOAN_DATE a Z36H_RETURNED_DATE tabulky Z36H zdrojové databáze). Hodnoty datumů jsou uloženy jako číslo (datový typ INTEGER) ve formátu rrrrmmdd (rok, měsíc, den). Atribut days_from_due_date je hodnota počtu dnů návratu jednotky od termínu vrácení a vypočítá se jako rozdíl datumů termínu a vrácení (atributy Z36H_DUE_DATE a Z36H_RETURNED_DATE tabulky Z36H databáze Aleph). Tato hodnota je záporná, pokud čtenář vrátil jednotku před termínem, a kladná, pokud čtenář vrátil jednotku po termínu.

Tabulka **z35** ukládá události. Atribut type určuje, o jakou událost se jedná.

Tabulka **z37h** obsahuje požadavky na půjčení knihy nebo jiného materiálu.

V tabulce **but01** jsou uloženy bibliografické informace o dokumentech. Atribut mdt slouží pro kategorizaci dokumentů dle žánru (např. encyklopedie, jazyky) a obsahuje první část kódu mdt po desetinnou tečku ze zdrojové databáze. Atribut mdt je typu řetězec proměnné délky (max. 3 znaky). Tento atribut je zadán jen v některých záznamech databáze.

Tabulka **z30** obsahuje jednotky, což jsou konkrétní výtisky dokumentů.

Tabulka **z31** obsahuje údaje o platbách (např. za pozdní vrácení knihy).

Tabulka **z103** obsahuje odkazy mezi záznamy mezi dvěma knihovnami.

Tabulka **timedim** slouží k uložení data (den, měsíc, rok) a bude se využívat v OLAP analýze k určení časového období.

Tabulka **librarydim** obsahuje, stejně jako další jmenované tabulky dimenzí, dvojici atributů kód, popis (kód knihovny a název knihovny).

Tabulka **material** slouží pro uložení informací o materiálu dokumentů a je k ní odkazováno z tabulek z30, z35 a z36h.

V tabulce **item_status** najdeme informace o délce výpůjčky jednotek (1 týden, 4 týdny atd.).

Tabulka **bor_status** obsahuje údaje o typu čtenáře (student VUT, doktorand atd.). Tabulka **but01_fmt** ukládá informace o formátu dokumentu (např. knihy, periodika). Tabulka **but01_typ** obsahuje údaje o typu dokumentu (např. kniha, skriptum, CD-ROM). Tabulka **but01_jazyk** ukládá informace o jazyku dokumentu. Klíčem je řetězec o délce 3 znaky (např. cze, eng). Tabulka **but01_mdt** obsahuje číselný kód MDT (mezinárodní desetinné třídění) a název kategorie. Kód MDT slouží k zařazení dokumentů do různých kategorií na základě žánru (např. 001 - věda, 030 - encyklopedie). Jak název napovídá, tabulky dimenzí začínající řetězcem but01 se váží k tabulce but01 (dokumenty). Podobná konvence pro pojmenování je použita i pro další tabulky.

Tabulka **z31_status** ukládá údaje o stavu platby (např. C - zavřená, W - prominutá).

Tabulka **z31_type** obsahuje informace o typu platby (např. pozdní návrat, ztráta dokumentu).

Tabulka **z35_event_type** ukládá informace o typu události (např. návrat, výpůjčka).

Tabulka **z36h_status** obsahuje informaci o stavu výpůjčky (aktivní, ztráta, údajně vráceno).

Tabulka **z37h_status** ukládá údaje o stavu žádosti o výpůjčku. Tabulka **z37h_source** obsahuje informace o zdroji žádosti o výpůjčku (originální, generovaný).

Tabulky but01, z30, z31, z35, z36h a z37h jsou tabulky faktů. Zbývající tabulky jsou tabulky dimenzí. Z tabulek faktů se odkazujeme cizím klíčem přes hodnotu kódu (např. BK z tab. but01_fmt - formát) pro zjištění popisu (např. knihy) na primární klíč těchto tabulek dimenzí. Popis je uživateli jistě srozumitelnější než klíč v podobě krátké znakové zkratky. Atribut popis se zobrazuje v OLAP dotazech.

4.2.1 Vytvoření datového skladu v MySQL

Skript instalace.sh vytvoří databázi příkazem `mysqladmin create vutlib`, přiděluje práva příkazem `grant` a poté volá skripty CreateTables.sql a CreateIndexes.sql pro vytvoření tabulek a indexů. Schéma datového skladu je vytvořeno SQL skriptem CreateTables.sql, který jak název napovídá, vytváří tabulky dimenzí a faktů SQL příkazem CREATE TABLE. Ukázkový skript se nachází v demonstrační aplikaci FoodMart dodávané v distribuci serveru Mondrian. Jako datové typy atributů jsou použity CHAR pro znakové řetězce pevné délky, VARCHAR pro řetězce proměnlivé délky, INTEGER pro čísla, SMALLINT pro čísla, která nabývají menšího počtu hodnot, DATE pro datum a DECIMAL pro uložení částky. V příkazu CREATE TABLE jsou také definována integritní omezení (NOT NULL, tedy tento atribut musí být zadán). Aby se zrychlilo vykonávání SQL dotazů, které budou generovány při OLAP analýze Mondrianem, jsou vytvořeny indexy SQL skriptem CreateIndexes.sql. Indexy, které se budou vytvářet pro primární a cizí klíče a které se používají při spojení tabulek v části příkazu SELECT za klíčovým slovem WHERE, slouží ke zrychlení přístupu k datům. Skript DropTables.sql zruší všechny tabulky databáze včetně jejich obsahu. Tento skript se používal během implementace, kdy se přidávaly do tabulek nové atributy nebo se měnily jejich datové typy. Skript instalace.sh také nastavuje kódování všech tabulek na utf8. Pro správné zobrazení znaků je rovněž třeba nastavit konfigurační soubor my.cnf databázového serveru MySQL. Do sekce [client] přidáme řádek:

```
default_character_set=utf8
```

a do sekce [server] přidáme řádky

```
init-connect='SET NAMES utf8'
character-set-server=utf8
collation-server=utf8_general_ci
```

4.3 Schéma pro server Mondrian

Schéma definuje multidimenzionální databázi (datový sklad). Toto schéma obsahuje logický model, který se skládá z kostek, dimenzí, hierarchií a členů, a mapuje tento model na fyzický model (relace, atributy). Logický model obsahuje konstrukce používané k zapsání dotazů v jazyce MDX: kostky, dimenze, hierarchie, úrovně a členy.

Schéma pro Mondrian je definováno v XML souboru. Ukázkové schéma se nachází v ukázkové aplikaci FoodMart dodávané v balíku aplikace. Schéma datového skladu je možné vytvořit v programech Pentaho Cube Designer, Mondrian Schema Workbench a nebo v libovolném textovém editoru, pokud možno se zvýrazněním syntaxe. Ve schématu je třeba

dodržovat pořadí značek. Je dobré se inspirovat schématem ukázkové aplikace FoodMart, zkopírovat část textu a přejmenovat názvy tabulek, atributů a dalších prvků podle svých potřeb. Ukázkové schéma obsahuje většinu značek, které je možné používat. Především ze začátku se nám při definici schématu může stát, že toto obsahuje chyby. Správnost schématu zjistíme tak, že si vytvoříme MDX dotaz a vložíme jej do prvku `<mondrianQuery>` do stránky mondrian.jsp (Java Server Pages). Pokud máme to štěstí, že je schéma správné, zobrazí se nám ve webovém prohlížeči výsledek dotazu. Pokud schéma obsahuje chybu, vyvolá se výjimka mondrian.olap.MondrianException a na stránce se vypíše chybová hláška s popisem výjimky. Pokud má schéma špatnou syntaxi, nejčastěji se vypíše chybová hláška Mondrian Error:Internal error. Pak je vhodné se podívat do dokumentace ([8]), která obsahuje i příklady. Také je vhodné postupovat podle některého vzorového schématu (například již zmiňovaná aplikace FoodMart). Pokud má schéma správnou syntaxi, ale dotaz obsahuje chybu nebo došlo k jiné chybě, vypíše se hláška s popisem chyby. Příkladem chyby je výpis MDX cube 'Cash' not found, což znamená, že kostka Cash nebyla nalezena.

Nejdůležitějšími komponentami schématu jsou kostky (cubes), hodnoty (measures) a dimenze (dimensions):

Kostka je kolekcí dimenzí a hodnot z oblasti určitého předmětu a definuje se prvkem `<Cube>`. Definice kostky také obsahuje název tabulky, ze které se budou načítat hodnoty a dimenze, pokud není definováno jinak v dimenzích. Název tabulky se definuje v hodnotě atributu name prvku `<Table>`, který bývá prvním prvním prvek v definici kostky. Každá tabulka faktů je ve schématu definovaná jako kostka značkou `<Cube>`.

Hodnota je veličina, kterou chceme spočítat. Například počty prodaných kusů produktu nebo prodejní cena kusů ze skladu. Hodnota se zapisuje značkou `<Measure>`. Mondrian podporuje tyto typy agregátů: sum (součet), count (počet), min (minimální hodnota), max (maximální hodnota), avg (průměr), a distinct-count (počet různých hodnot). Agregáty jsou výsledné hodnoty pro všechny záznamy. Atributem formatString je možné nastavit formát výpisu hodnoty.

Dimenze je atribut nebo množina atributů, kterou rozdělujeme hodnoty na podkategorie. Například si můžeme přát rozdělit prodeje podle času (roky, měsíce, týdny, dny), pohlaví zákazníka a obchodu, ve kterém byl produkt prodán. Pak budou čas, pohlaví a obchod dimenzemi. Dimenze se definuje značkou `<Dimension>` a má definován primární klíč (atribut primaryKey značky `Hiearchy`), aby bylo možné se na ně odkazovat cizím klíčem z tabulek faktů (atribut foreignKey prvku `<Dimension>`). Schéma může obsahovat sdílené dimenze. Sdílené dimenze je možné deklarovat jednou a použít je v kostkách prvkem `<DimensionUsage>` a nacházejí na začátku schématu mezi značkou schema a první definicí kostky. Značka `<DimensionUsage>` má atribut foreignKey, který je atributem tabulky faktů. Jméno sdílené dimenze a dimenze pojmenované v kostce, z níž se na sdílenou dimenzi odkazujeme, musí být stejné.

Mondrian umožňuje hierarchii atributů. Hierarchie se definuje prvkem `<Hierarchy>`. Tato značka může obsahovat definici tabulky (prvek `<Table>`), ze které budou atributy vybrány, a úrovně (prvek `<Level>`). Prvek Level má atributy name (název), column (sloupec tabulky), type (datový typ) a další. Atribut type nabývá např. hodnot Numeric (číslo) a String (řetězec). Odlišení čísla a řetězce má vliv na to, zda Mondrian přidá do SQL dotazu apostrofy. Každý atribut je definován jako jedna úroveň. Pokud hierarchie neobsahuje žádnou definici tabulky, hledají se atributy v tabulce, která byla definována pro kostku. Dimenze může obsahovat více hierarchií. Toho se využívá například v definici časové dimenze

pro definici dvou hierarchií. První je hierarchie rok, měsíc. A druhou je rok, týden, den. V dotazech se použije jen jedna z těchto hierarchií.

Každá dimenze ve schématu XML standardně úroveň "All" (vše), která obsahuje jediný člen "all". Tento člen je předkem všech dalších členů hierarchie a tím obsahuje celkové přehledy (sumy). Tento člen je také standardním (angl. default) členem hierarchie, což znamená, že je členem, který je použit pro výpočet hodnot buněk, když není hodnota zahrnuta na ose nebo není vybrána operací slice. Pokud dimenze obsahuje atribut hasAll s hodnotou false, pak tento člen není vytvořen. Časová dimenze má hodnotu atributu type rovnu "TimeDimension". Úrovně časové dimenze mohou nabývat hodnot TimeYears (roky), TimeQuarters (čtvrtletí), TimeMonths (měsíce) a TimeDays (dny).

Pokud chceme aplikovat řízení přístupu (angl. access control), tj. zabránit někomu v přístupu k výsledkům OLAP analýzy (nebo jen například k některým kostkám), docílíme toho definicí značky `<Role>` a nastavením této role při připojení. Role se vyskytují za poslední kostkou ve schématu, jako potomek prvku `<Schema>`. Přístup ke schématu se definuje prvkem `<SchemaGrant>`. Prvkem `<CubeGrant>` můžeme definovat přístup ke kostkám. Prvkem `<HierarchyGrant>` definujeme přístup k hierarchiím dimenzí. Prvek `<SchemaGrant>` definuje výchozí přístup pro objekty schématu. Tyto prvky mají atribut access, který nabývá hodnot "all" (přístup ke všemu) a "none" (žádný přístup). Tento přístup může být pro určité objekty změněn.

Mapování na datový sklad s daty Alephu

V této části textu bude popsáno schéma pro server Mondrian mapující kostky, dimenze a hodnoty na datový sklad, jehož návrh je v sekci 4.2. Definice schématu je uvedena v souboru VUTLib.xml v adresáři WEB-INF/queries. Kostky, dimenze a hodnoty, definované v tomto schématu, se používají v OLAP dotazech (jazyk MDX).

Sdílené dimenze Dimenze Cas, Knihovna, Material, Jednotka, Ctenar a Dokument jsou sdílené mezi kostkami. Dimenze **Cas** je časová dimenze, obsahuje dvě hierarchie - rok, měsíc a rok, týden, den a mapuje se na tabulku timedim. Dimenze **Knihovna** slouží pro rozlišení faktů dle toho, ke které knihovně událost (např. výpůjčka) patří, a mapuje se na tabulku librarydim. Dimenze **Material** rozlišuje jednotky dle materiálu (např. knihy) a mapuje se na tabulku material. Dimenze **Jednotka** se používá pro statistiky dle statutu jednotky a mapuje se na tabulku item_status. Dimenze **Ctenar** rozlišuje čtenáře dle jejich statutu (např. student, zaměstnanec) a mapuje se na tabulku bor_status. Dimenze **Dokument** rozděluje dokumenty dle různých kategorií (např. jazyk, formát) a mapuje se na tabulku but01.

Kostky Každé tabulce faktů odpovídá jedna kostka. Schéma obsahuje kostky **Dokumenty** (tab. but01), **Jednotky** (tab. z30), **Platby** (tab. z31), **Udalosti** (tab. z35), **Vypujcky** (tab. z36h) a **Zadosti** (tab. z37h). Tabulka 4.1 zobrazuje seznam dimenzí jednotlivých kostek. Tabulka 4.2 obsahuje seznam hodnot, které obsahují jednotlivé kostky.

Část schématu pro ukázku vypadá takto:

```
<Schema name="VUTLib">
    ...
    <Dimension name="Knihovna">
        <Hierarchy hasAll="true" allMemberName="Vsechny knihovny"
            primaryKey="libcode">
            <Table name="librarydim">
            </Table>
            <Level name="Knihovna" column="name" type="String"
                uniqueMembers="true" levelType="Regular" hideMemberIf="Never">
            </Level>
        </Hierarchy>
    </Dimension>

    <Cube name="Vypujcky">
        <Table name="z36h">
        </Table>
        <DimensionUsage source="Knihovna" name="Knihovna"
            foreignKey="sub_library">
        </DimensionUsage>
        <Dimension foreignKey="status" name="Vypujcka">
            <Hierarchy name="Status vypujcky" hasAll="true"
                allMemberName="Vsechny statuty" primaryKey="status">
                <Table name="z36h_status">
                </Table>
                <Level name="Status vypujcky" column="popis" type="String"
                    uniqueMembers="true">
                </Level>
            </Hierarchy>
        </Dimension>
        <Measure name="Pocet vypujcek" column="time" formatString="Standard"
            aggregator="count">
        </Measure>
    </Cube>
    ...
</Schema>
```

Schéma se jmenuje VUTLib, obsahuje sdílenou dimenzi Knihovna, která získává data z tabulky dimenzí librarydim, která se nachází v datovém skladu, vytvořeném za účelem OLAP analýzy. Pro spojení s tabulkou faktů je specifikován primární klíč tabulky librarydim, který se jmenuje libcode. Spojením kostky s tabulkou dimenze vzniká schéma hvězda anebo sněhová vločka. Dimenze Knihovna, stejně jako další dimenze, musí mít specifikovánu hierarchii atributů. Atributem hierarchie hasAll s hodnotou "true" určujeme, že dimenze obsahuje člen "all" (hasAll=true), který v tomto případě znamená všechny knihovny. Tato dimenze obsahuje jediný atribut, který se definuje jako úroveň hierarchie prvkem `<Level>`. Důležitým atributem úrovně je jméno sloupce tabulky dimenze, které se určuje hodnotou atributu column. Ve schématu je definována kostka Vypujcky, které odpovídá tabulka

faktů z36h. Ve schématu se pro každou tabulku faktů definuje kostka. Kostka Vypujcky používá sdílenou dimenzi Knihovna, na níž se odkazuje cizím klíčem sub_library. Dimenze Knihovna se v MDX dotazech zapisuje jako `[Knihovna]`. Kostka Vypujcky obsahuje také dimenzi Vypujcka. Důvodem, proč není tato dimenze sdílená je ten, že tato dimenze se používá jen v této kostce. Dimenzi Vypujcka odpovídá tabulka dimenze z36h_status, má primární klíč status, obsahuje jediný atribut popis a odkazujeme se na ni z kostky cizím klíčem status. Kostka Vypujcka obsahuje hodnotu "Pocet vypujcek", která se objevuje v OLAP dotazech, pokud si ji uživatel vybere. V MDX dotazech tuto hodnotu zapíšeme jako `[Measures].[Pocet vypujcek]`.

4.4 ETL nástroj

Ve fázi extrakce jsou exportována data, která budou použita pro plnění datového skladu, z databáze knihovního systému Aleph do textových souborů. Každá tabulka byla exportována do jednoho souboru. Fáze transformace obsahuje výběr atributů, převody a výpočty hodnot (např. doba výpůjčky jednotky). Fáze nahrání reprezentuje nahrání dat do datové skladu.

Soubory jsou uloženy ve znakové sadě Unicode. Každý řádek souborů pro tabulky z30, z31, z35, z36h, z37h a z103 obsahuje jednu n-tici pozičních atributů (sloupců) tabulek. Atributy jsou v těchto souborech odděleny tabulátory. Hodnoty atributů v souborech jsou dány datovými typy, v jakém jsou v databázi ORACLE uloženy. Například když je pro uložení čísla je v databázi použit typ NUMBER s uvedenou délkou (např. NUMBER(8)), pak textová reprezentace v souboru je uvedené délky (doplnění zleva znakem "0"). Datový typ řetězec (CHAR) definované délky se exportuje na textový řetězec pevné délky (doplnění zprava mezerami). Datový typ VARCHAR definované délky se exportuje na textový řetězec délky takové, jakou hodnotu atribut obsahuje (řetězec tedy není doplněn mezerami).

Soubor but01, obsahující bibliografické údaje, je uložen v jiném formátu. Jednotlivé tokeny v tomto souboru jsou odděleny mezerami. První token je číslo dokumentu (DOC_NUMBER) o délce 9 znaků, druhý token je název atributu (např. FMT - formát dokumentu) pevné délky 5 znaků (text je doplněn zprava mezerami), třetí token je kód alpha (abeceda - vždy L - latinka) délky 1 znak, a čtvrtý token je hodnota atributu (např. BK - kniha). Údaje o jednom dokumentu jsou uvedeny na několika řádcích. Počet řádků, na kterých jsou údaje o jednom dokumentu, se liší. Některé dokumenty nemají zadané nějaké atributy (např. jazyk, kód MDT).

Jednou z možností provedení etapy ETL by bylo použít některý existující nástroj. Na výběr je poměrně mnoho nástrojů (viz 2.3.1). Balík Pentaho, do kterého patří také použitý OLAP server Mondrian, obsahuje pro etapu ETL nástroj Pentaho Data Integration (dříve Kettle). Nejdříve jsem si prošel ukázkové operace ETL dodávané s tímto nástrojem. Byl proveden pokus v tomto nástroji o import dat exportovaných ze zdrojové databáze knihoven. Byl vybrán importovaný soubor, oddělovač (tabulátor) a další parametry. Ovšem tento nástroj špatně rozpoznával jednotlivé části oddělené tabulátorem a bylo nutné specifikovat i délku jednotlivých řetězců. Při počtu hodnot v jednom řádku souboru a počtu souborů je to jistě zdlouhavé, a již tento první krok etapy ETL rozhodl o tom, že tento nástroj nebude použit. Další nástroje již nebyly testovány. Další možností je naprogramovat si vlastní nástroj ETL. A právě tato možnost byla zvolena. Důvody pro implementaci vlastního nástroje jsou nepodařený import dat knihoven ve zmiňovaném nástroji Pentaho Data Integration, kontrola nad použitými operacemi, pomocné výpisy v jednotlivých krocích

a kontrola ošetření chybějících hodnot.

Nástroj ETL je implementován v jazyce Java. Vstupní data získává z výše popsaných exportovaných souborů. Pro přístup k databázi je použito rozhraní JDBC (Java Database Connectivity). Toto rozhraní umožňuje pracovat se širokým spektrem SQL databází nebo souborů. Nástroj ETL je konzolová aplikace. Výhodou konzolové aplikace oproti aplikace s GUI rozhraním je možnost spuštění ve skriptu shellu a naplánování spuštění příkazem cron. Vstup je předáván přes parametry příkazové řádky. Nástroj buď nahrává transformované informace do prázdného skladu nebo inkrementálně. Inkrementální nahrávání ukládá do databáze pouze inkrement (přírůstek) oproti minulé verzi exportovaných dat. Toto výrazně ušetří čas a nejsou opakovaně nahrávána data, která už byla dříve zpracována. Inkrementální nahrávání je prováděno na základě porovnání vybraného atributu s nejvyšší hodnotou v databázi zjištěnou SQL příkazem SELECT max(atribut) FROM tabulka. Vybranými atributy jsou timestamp (časové razítko) a v případě but01 číslo dokumentu.

Po nahrávání dat do datového skladu bylo důležité zjistit hodnotu tabulek dimenzí. Seznam hodnot, kterých nabývá některý atribut v tabulce datového skladu, se zjistí příkazem SELECT DISTINCT - např.

```
select distinct item_status from z30.
```

Hodnoty tabulek dimenzí jsou uloženy v souborech exportovaných z databáze Aleph. Některé hodnoty bylo možné najít na internetu pomocí vyhledávače. Význam atributů tabulky but01 byl zjištěn z [2].

4.4.1 Třídy

Třída MyETL

Na obrázku 4.3 je diagram tříd ETL nástroje. Pro přehlednost jsou vynechány atributy třídy MyETL, parametry a návratové hodnoty metod. Třída `MyETL` reprezentuje hlavní třídu - konzolovou aplikaci. Tato obsahuje zpracování parametrů příkazové řádky, metodu pro výběr částí řetězce oddělených určitým znakem (např. mezerou, tabulátorem nebo čárkou), zpracování exportovaných souborů pro tabulky faktů, uložení n-tic do databáze, naplnění tabulek dimenzí a další pomocné metody.

Tabulky dimenzí jsou plněny daty z textových souborů CSV (hodnoty jsou oddělené čárkou). Tyto soubory obsahují na řádku dvojici hodnota a popis (např. kód knihovny a popis). Tyto hodnoty jsou umístěny v textových souborech kvůli požadavku na umístění těchto hodnot mimo program pro případ budoucí změny (přidání nové n-tice, změna hodnot).

Příkazy SQL, volané přes rozhraní JDBC, jsou definovány třídou `PreparedStatement` (v překladu připravený příkaz). Při inicializaci se do objektu třídy PreparedStatement přiřadí návratová hodnota metody preparedStatement třídy Connection, která má jako parametr řetězec, obsahující SQL příkaz (např. “INSERT INTO librarydim VALUES(?,?)”). Parametry SQL příkazu jsou v tomto řetězci reprezentovány otazníkem. SQL příkazy jsou reprezentovány třídou PreparedStatement, protože by volání takového příkazu mělo být údajně rychlejší než příkazu reprezentováného třídou Statement. Zrychlení se hodí především při plnění tabulek faktů (z30, z31 atd.). Metodami setXXX třídy PreparedStatement, kde XXX je typ (např. Int, String) se nastaví hodnota n-tého parametru SQL příkazu. Prvním parametrem je pořadové číslo parametru, druhým je hodnota. Příkazy upravující databázi (INSERT, UPDATE, DELETE) se spouští metodou executeUpdate. Tato metoda vrací počet upravených řádků. Příkaz SELECT se spouští metodou executeQuery.

Metoda `loadTimeDim` plní časovou dimenzi (tab. timedim) záznamy pro každý den od počátečního do koncového data, které jsou předány jako parametry. Tato metoda používá pro uložení data objekt třídy GregorianCalendar.

Metoda `createTimeStruct` plní objekt třídy TimeDimRec na základě data uloženého v objektu třídy GregorianCalendar. Určuje se klíč (formát rrrrmmdd - rok, měsíc den - je standardním formátem pro datum v databázi Aleph), datum typu java.sql.Date, den, měsíc, rok, týden, název dne v týdnu (pondělí atd.), název měsíce (leden atd.) a školní rok.

Metoda `connect` připojuje k databázi (datovému skladu). Metoda disconnect odpojuje od databáze.

Metoda `emptyTable` vymaže obsah tabulky, jejíž jméno je předáno v parametru typu String.

Metoda `parse` hledá v textovém řetězci, předaném jako první parametr, řetězce oddělené oddělovačem (mezerou, čárkou, tabulátorem atd.), který je předán jako druhý parametr, a výsledek uloží do seznamu řetězců (Vector¡String¿). Tato metoda používá pro nalezení dalšího tokenu třídu StringTokenizer. K i-tému prvku seznamu přistupujeme voláním metody get(i). Index se čísluje od nuly.

Metoda `checkLibrary` kontroluje, zda je kód knihovny obsažen v tabulce dimenze knihovna (tabulka library). Pokud je, vrátí se tato hodnota. Pokud není, vrátí se prázdný řetězec.

Metody `processBut01, processZ30, processZ31, processZ35, processZ36H, processZ37H a processZ103` zpracovávají řádek ze souborů faktů (but01 atd.) a rozpoznané hodnoty uloží do atributů tříd reprezentujících záznamy (But01Rec atd.).

Metody `loadBut01, loadZ30, loadZ31, loadZ35, loadZ36H, loadZ37H a loadZ103` uloží do tabulek faktů datového skladu jednu n-tici hodnot atributů tříd reprezentující záznam (např. But01Rec)metodou executeUpdate třídy PreparedStatement.

Metoda `getMaxValueFromDb` se používá pro zjištění maximální hodnoty určitého atributu (např. timestamp). Tato hodnota se používá při inkrementálním nahrání tabulky faktů (když timestamp > maxTimestamt, ulož záznam).

Metoda `loadTwoColDim` se používá pro naplnění tabulek dimenzí, které obsahují dva sloupce (kód, hodnota). Tato metoda načítá z csv souboru tyto dvojice a ukládá je do datového skladu.

Metoda `loadFirstColFromFile` načítá první hodnotu z csv souboru od dalších oddělenou čárkou a přidává ji do množiny. Množina se používá při kontrole hodnot (např. kódy knihoven) metodou contains.

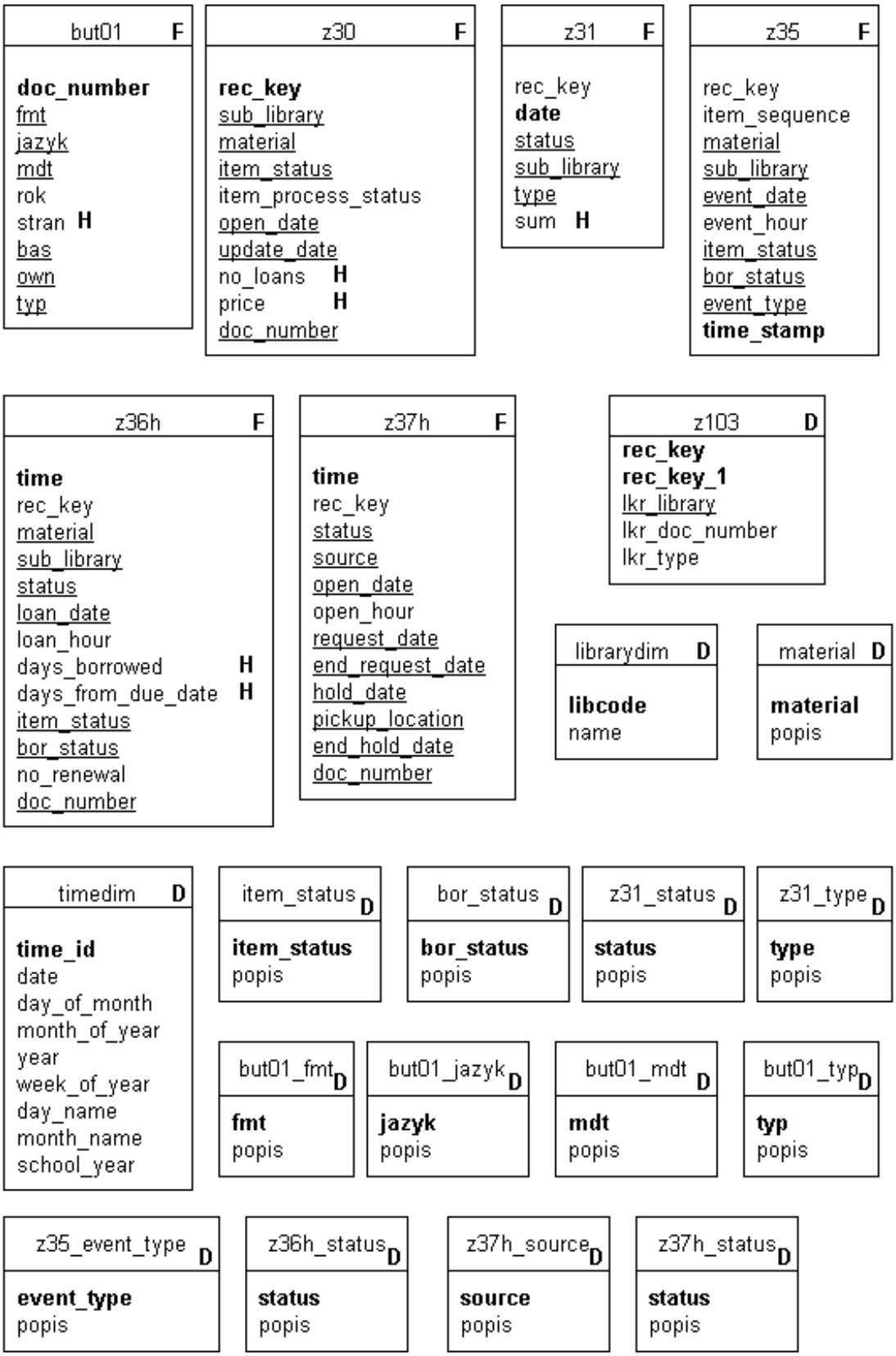
Metoda `parseFile` načítá řádek souborů faktů (but01, z30 atd.), volá metody pro zpracování řádku (processBut01 atd.) a uložení do datového skladu (loadBut01 atd.).

Další třídy

Třídy `But01Rec`, `TimeDimRec`, `Z30Rec`, `Z31Rec`, `Z35Rec`, `Z36HRec`, `Z37HRec` a `Z103Rec` reprezentují n-tici (záznam), která obsahuje vybrané atributy a bude se ukládat do datového skladu. Tyto třídy obsahují metody `vymazHodnoty` pro inicializaci hodnot záznamu a `vypisHodnoty` pro výpis hodnot.

4.4.2 Automatizace etapy ETL

ETL nástroj, reprezentovaný hlavní třídou MyETL, vykonává vždy jen určitou operaci (například plnění tabulky faktů z30). Aby se etapa ETL zautomatizovala, byl vytvořen skript load_data.sh. Tento skript má jeden povinný parametr a druhý volitelný. Prvním parametrem je buď cesta k adresáři, ve kterém se nachází exportovaná data, nebo parametr -dimensions. Třetím volitelným parametrem je -incremental, který se používá pro inkrementální (přírůstkové) nahrání. Inkrementální nahrání nahrává pouze změny oproti minulému stavu datového skladu a díky tomu je rychlejší. Spuštění skriptu s parametrem -dimensions, volá nástroj MyETL, který naplní atributy tabulky dimenzí hodnotami z csv souborů. Pokud je prvním parametrem skriptu jméno adresáře, provede se plnění tabulek faktů but01, z30, z31, z35, z36h, z37h a také pomocné tabulky z103 nástrojem MyETL, které je inkrementální, pokud je druhým parametrem přepínač -incremental. Pokud není plnění tabulek faktů inkrementální, pak se nejprve zavolá nástroj MyETL s parametrem -emptytable a jménem tabulky pro všechny tabulky faktů, a tím se vymaže jejich obsah a poté se provede plné nahrání (full load). Tabulky dimenzí se budou plnit obvykle jen při prvotní instalaci nástroje. Tabulky faktů se naopak budou plnit pravidelně v určitém časovém období. Prvotní naplnění všech tabulek faktů může trvat několik minut. Oproti tomu plnění tabulek dimenzí trvá podstatně kratší dobu.



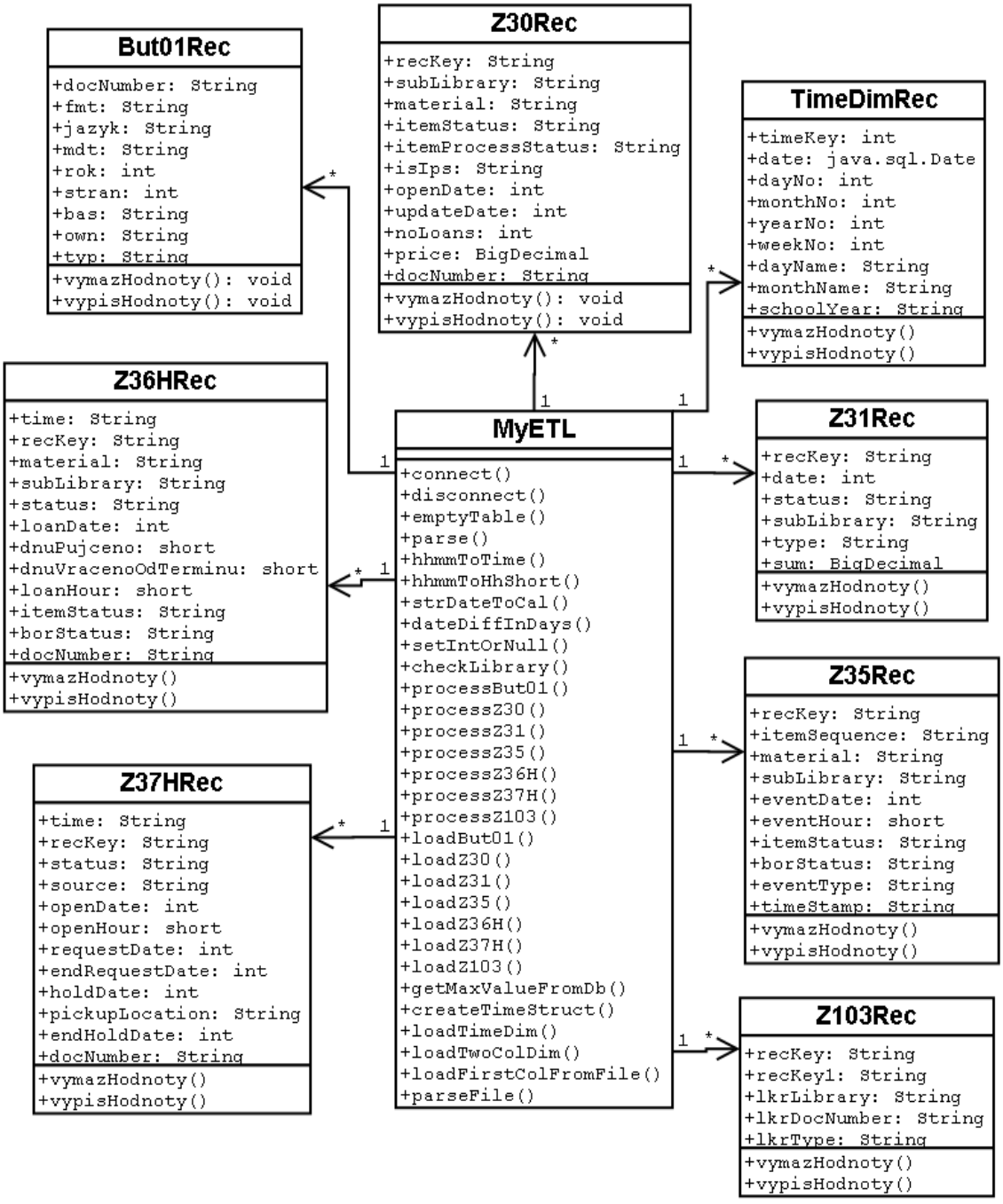
Obrázek 4.2: Schéma datového skladu

Kostka	**Dimenze**
Dokumenty	Knihovna Rok Format Jazyk Typ MDT
Jednotky	Knihovna Cas Material Status jednotky
Platby	Knihovna Cas Status platby Typ platby
Udalosti	Knihovna Cas Material Jednotka Ctenar Hodina Udalost
Vypujcky	Knihovna Dokument Cas Material Jednotka Ctenar Hodina Vypujcka
Zadosti	Knihovna Cas Status Zdroj

Tabulka 4.1: Seznam dimenzí pro jednotlivé kostky

Kostka	**Hodnoty**
Dokumenty	Prumerny pocet stran Pocet dokumentu
Jednotky	Prumerna cena Prumerny pocet jednotek Pocet jednotek
Platby	Pocet plateb Celkova castka Prumerna castka Maximalni castka
Udalosti	Pocet udalosti
Vypujčky	Pocet vypujcek Vypucejno prumerne dnu Vraceno prumerne dnu od terminu
Zadosti	Pocet zadosti

Tabulka 4.2: Seznam hodnot pro jednotlivé kostky



Obrázek 4.3: Diagram tříd

Kapitola 5

Uživatelské rozhraní

JPivot je knihovna v jazyce JSP, která vykresluje OLAP tabulky a umožňuje uživatelům spouštět typické OLAP operace slice a dice, drill down a roll up (viz [1]). Knihovna JPivot reprezentuje obsluhu uživatelského rozhraní a je navrhnuta pro práci s různými OLAP servery, speciálně XMLA a Mondrianem. XMLA (XML for Analysis) je pokus o standardizaci aplikačního programovacího rozhraní (API) v prostředí OLAP a systémů pro podporu rozhodování. JPivot nepoužívá API serveru Mondrian přímo, ale definuje svůj vlastní OLAP model v rozhraních (interface) balíků olap.model a olap.navi. Aby mohl JPivot pracovat s Mondrianem, musí být tato rozhraní implementována použitím tříd Mondrianu. JPivot používá balíky WCF (Web Component Framework), které podporují vytvoření komponent uživatelského rozhraní (UI) přes XML a XSLT. XSLT je jazyk pro transformaci XML dokumentů do jiných XML dokumentů. Třídy WCF pomáhají vytvořit, zobrazit a kontrolovat správnost (validace) HTML formulářů pomocí XML a XSLT. WCF zapouzdřuje znovu použitelné komponenty, které byly vyvinuty pro JPivot. Mezi komponenty patří prvky UI jako např. tabulka, strom a formulář. Pro jednoduché prvky, jako jsou tlačítko nebo vstupní textové pole, je možné použít komponenty JSF (JavaServer Faces). WCF se dobře integruje s JSF.

Uživatelské rozhraní se ovládá z internetového prohlížeče (Mozilla Firefox, Microsoft Internet Explorer a další). Multidimenzionální dotazy jsou umístěny v jsp stránce, jak již bylo uvedeno. V indexové stránce index.html jsou odkazy na jsp stránky s OLAP dotazy v jazyce MDX. Každá tabulka faktů (but01, z30, z31, z35, z36h a z37h) má svou jsp stránku. JPivot nechá serveru Mondrian spustit multidimenzionální dotazy v jazyce MDX, získá výsledek dotazu a ten zobrazí. Na obrázku 5.1 je ukázka výsledku dotazu

```
select
  {[Measures].[Pocet vypujcek], [Measures].[Vypujceno prumerne dnu],
  [Measures].[Vraceno prumerne dnu od terminu]} on columns,
  {[Cas].[Vsechny casy]} on rows
from [Vypujcky]
```

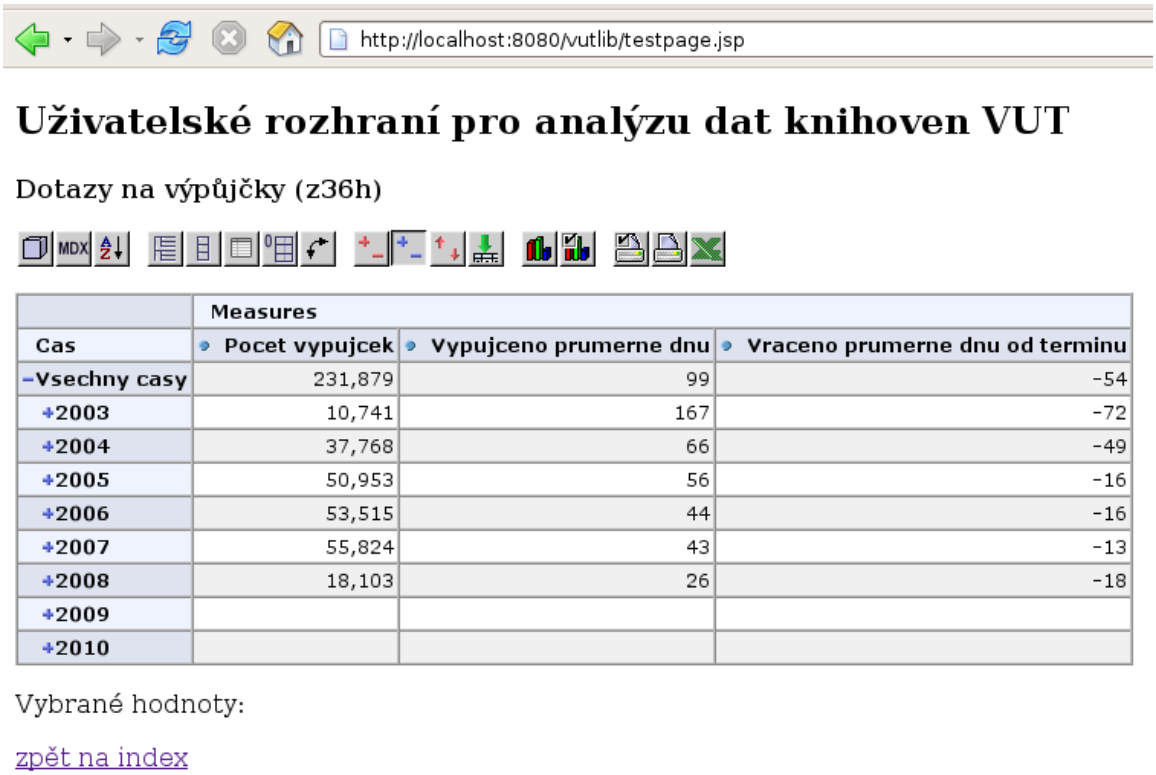
ve webovém prohlížeči. Tento dotaz zjišťuje počet výpůjček, kolik dnů průměrně trvala výpůjčka a kolik dnů byla jednotka vrácena od termínu (záporná hodnota určuje, že byla jednotka vrácena před termínem, a kladná po termínu) pro jednotlivé časy. Na jedné ose je zobrazen čas v hierarchii rok, měsíc a na druhé ose jsou jmenované hodnoty (angl. measures). Obě osy je možné otočit tlačítkem s dvěma šipkami na liště tlačítek.

Pro kontrolu si ověříme, zda jsou zobrazené hodnoty výsledku správné. Například si z datového skladu zjistíme SQL příkazem počet výpůjček v roce 2003:

```
select count(*) from z36h where (loan_date >= 20030101)
and (loan_date <= 20031231);
```

Tato hodnota se rovná hodnotě v tabulce s výsledkem MDX dotazu, tudíž výsledek, který vypočítal OLAP server Mondrian, je správný. Stejnou kontrolu bychom mohli provést i pro jiné dimenze a hodnoty.

Uživatelské rozhraní se skládá z lišty tlačítek, výsledku dotazu ve formě tabulky a vybraných hodnot dimenzí (hodnota za textem Vybrané hodnoty). Před tabulku se zobrazují formuláře pro upřesnění dotazu (řazení, výběr dimenzí, hodnot a dalších vlastností). Do zbývající části stránky za tabulkou se zobrazuje graf nebo seznam hodnot databáze, pokud chceme zjistit, jaké n-tice se do výsledku promítly. Podrobnější popis ovládání uživatelského rozhraní je v dodatku B.



Obrázek 5.1: Obrazovka uživatelského rozhraní v Mondrian/JPivot s dotazem na výpůjčky

Kapitola 6

Závěr

V semestrálním projektu proběhla příprava na implementaci OLAP analýzy. Nastudoval jsem problematiku budování datového skladu (kostky, dimenze, hodnoty, schéma hvězda a vločka) a seznámil jsem se se schématem databáze knihoven VUT. Tato databáze obsahuje mnoho tabulek, z nichž byly vybrány tabulky, které obsahují data zajímavá pro statistiky a budou použity k budování datového skladu. Tyto tabulky byly exportovány do textových souborů a byly mi dodány vedoucím této práce. Byla provedena analýza nástrojů pro OLAP analýzu. Hlavním parametrem, dle kterého byl vybrán OLAP nástroj, byla licence. V zadání bylo uvedeno, že by nástroj měl být ideálně dodáván v licenci open source. V této práci byl po úvaze vybrán nástroj Mondrian. Dalším nástrojem, který mohl být použit, je Palo, který je také vyvíjen v licenci open source. Nevýhodou Pala je, že je zdarma jen část dokumentace a pracuje s daty uloženými v Excelu. Mondrian používá jako datový sklad některou z relačních databází a je nasazen na aplikačním serveru Tomcat. Datový sklad je reprezentován relační databází MySQL.

Z vybraných tabulek byl vytvořen datový sklad a byla provedena analýza ETL existujících nástrojů pro jeho plnění. Z nich byl vybrán nástroj Pentaho Data Integration. Protože tento nástroj špatně importoval vstupní data ze souborů, rozhodl jsem, že si naprogramuji vlastní ETL nástroj. Tento nástroj plní tabulky faktů a dimenzí datového skladu. Bylo třeba provést kontroly hodnot. Je to dáno tím, že některé atributy nejsou zadány nebo obsahují špatná data. Aby mohl server Mondrian vykonávat OLAP dotazy, bylo nutné definovat schéma v xml souboru, které mapuje kostky, dimenze a hodnoty na tabulky a atributy datového skladu. Tento nástroj umožňuje klást dotazy na dokumenty, výpůjčky, jednotky, platby události a požadavky na půjčení bez znalosti jakéhokoliv dotazovacího jazyka. Tyto dotazy je možné specifikovat dle časového období (roků, měsíců atd.), knihovny, statusů atd. Tyto parametry je možné kombinovat, a tím může uživatel klást předem neznámé dotazy, což je možné díky definic kostek, dimenzí a hodnot a OLAP metodám. Samotný jazyk SQL není na takovou analýzu vhodný, protože nemusí být předem známy dotazy, které bude uživatel chtít odpovědět. Uživatelské rozhraní se ovládá z internetového prohlížeče a je implementováno komponentou JPivot. Toto rozhraní je uživatelsky přátelské. Byly kladeny dotazy nad jednotlivými tabulkami faktů (např. výpůjčky) a výsledky dotazů byly zkontrolovány s očekávanými výstupy zjištěnými příkazem SELECT jazyka SQL. Tyto výsledky se sobě rovnaly. Nástroje Mondrian a JPivot, použité v této práci, potvrdily, že bezplatné nástroje mohou vykonat podobnou službu jako komerční nástroje.

Literatura

[1] *JPivot Project Page*. http://jpivot.sourceforge.net, 2008.

[2] Anděrová, I.: *Metodika popisu článků ve Formátu MARC 21 (pracovní materiál)*. Národní knihovna České republiky, http://www.nkp.cz/pages/page.php3?page=oazp_metodika21.htm, 2006.

[3] Ex Libris: *Aleph 500 version 14.2 Documentation*. 2001.

[4] Inmon, W. H.: *Building the Data Warehouse*. Wiley Publishing, Inc., 2005, ISBN 0-7645-9944-5.

[5] Kimball, R.; Ross, M.: *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling (Second Edition)*. Wiley Publishing, Inc., 2002, ISBN 0-471-20024-7.

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

[7] Le Cannellier, C.: Les solutions Open Source pour le Décisionnel. Technická zpráva, EURIWARE, http://www.progilibre.com/index.php?preaction=download&id=&startdownload=6_Euriware.pdf, 2006.

[8] Pentaho, http://mondrian.pentaho.org/documentation/doc.php: *Mondrian Documentation*. 2007.

Dodatek A

Instalace Mondrianu v Linuxu

Mondrian byl instalován v operačním systému Linux v distribuci Ubuntu 6.06 (Dapper Drake). V jiných distribucích by měl být postup podobný. Při instalaci bylo postupován dle kroků v [8]. Mondrian je distribuován jako WAR archív ve dvou variantách:

1. bez přednastavené databáze. Tento archiv obsahuje ukázková data v databázi Access a SQL skript pro nahrání dat.
2. s databází Apache Derby. Tato varianta nevyžaduje žádné další nastavení před nasazením na aplikační server.

Zvolíme variantu 1, protože budeme používat databázi MySQL. Obě distribuce obsahují zdrojový kód Mondrianu.

Toto je základní postup, jak nainstalovat nainstalovat binární vydání:

1. Nainstalujeme Sun Java SDK (verze 1.4.2 nebo novější)
2. Stáhneme poslední binární vydání - soubor mondrian-verze.zip z adresy http://sourceforge.net/projects/mondrian a rozbalíme jej.
3. Nainstalujeme aplikační server (Apache Tomcat)
4. Nainstaluje databázi MySQL.
5. Nastavíme datový zdroj (balík obsahuje ukázkovou databázi FoodMart).
6. Nastavíme a spustíme webovou aplikaci.

Spustíme program Synaptic.

Vybereme sun-java-jdk, mysql-client, mysql-server a tomcat. A spustíme instalaci tlačítkem použít. Tím nainstalujeme Javu a databázi MySQL.

Při instalaci jsem narazil na problém s Javou. Instalace Ubuntu obsahovala distribuci Javy GNU JAVA (překladač gcj). Je třeba nastavit nastavit proměnnou JAVA_HOME na cestu k adresáři distribuce JDK od společnosti SUN.

Rozbalíme soubor mondrian.war do adresáře TOMCAT_HOME/webapps/mondrian.

Spustíme tyto příkazy z terminálu (vytvoření databáze foodmart a uživatele foodmart):

```
sudo mysqladmin create foodmart
sudo mysql

mysql> grant all privileges on *.* to 'foodmart'@'localhost'
 identified by 'foodmart';

Query OK, 0 rows affected (0.00 sec)

mysql> quit
Bye
```

A nahrajeme data:

```
java -cp "/usr/share/tomcat5/webapps/mondrian/WEB-INF/lib/mondrian.jar: \
/usr/share/tomcat5/webapps/mondrian/WEB-INF/lib/log4j-1.2.8.jar: \
/usr/share/tomcat5/webapps/mondrian/WEB-INF/lib/eigenbase-xom.jar: \
/usr/share/tomcat5/webapps/mondrian/WEB-INF/lib/eigenbase-resgen.jar: \
/usr/share/tomcat5/webapps/mondrian/WEB-INF/lib/eigenbase-properties.jar: \
/usr/share/mysql/mysql-connector-java-5.0.8-bin.jar" \
mondrian.test.loader.MondrianFoodMartLoader \
-verbose -tables -data -indexes \
-jdbcDrivers=com.mysql.jdbc.Driver \
-inputFile=./mondrian-2.4.2.9831/demo/FoodMartCreateData.sql \
-outputJdbcURL="jdbc:mysql://localhost/foodmart?user=foodmart& \
 password=foodmart"
```

U tohoto příkazu musíme dát pozor na zápis dotazu. Pokud je dotaz zapsán na několika řádcích, musí být na konci řádků znak zpětné lomítko ("\"). Je možné tento dotaz uložit do souboru a spustit jej příkazem `sudo sh skript.sh`. Dále je také třeba najít, kde se nacházejí soubory jar uvedené v dotazu, a případně změnit cestu. Měly by být v adresáři WEB-INF/lib. Také musíme nastavit hodnotu parametru `-inputfile`.

Nyní nastavíme datový zdroj. Spustíme editor textových souborů a změníme soubor mondrian.properties v adresáři TOMCAT_HOME/webapps/mondrian:

`sudo gedit /usr/share/tomcat5/webapps/mondrian/WEB-INF/mondrian.properties`

Přidáme řádek `mondrian.jdbcDrivers=com.mysql.jdbc.Driver` a uložíme soubor.

Editujeme soubor web.xml:

`sudo gedit /usr/share/tomcat5/webapps/mondrian/WEB-INF/web.xml`

Změníme Provider na `Provider=mondrian;Jdbc=jdbc:mysql://localhost/foodmart?user=foodmart&#38;password=foodmart;Catalog=/WEB-INF/queries/FoodMart.xml;JdbcDrivers=com.mysql.jdbc.Driver;`

A uložíme soubor.

Otevřeme soubory fourhier.jsp, mondrian, colors.jsp a arrows.jsp v adresáři TOMCAT_HOME/webapps/mondrian/WEB-INF/lib/queries.

Spustíme tento příkaz v terminálu (a podobně pro další výše jmenované soubory jsp):

```
sudo gedit /usr/share/tomcat5/webapps/mondrian/WEB-INF/queries/
fourhier.jsp
```

Změníme řádky s atributem jdbcDriver a jdbcUrl na tuto hodnotu u všech jmenovaných souborů jsp, a uložíme změny:

```
<jp:mondrianQuery id="query01" jdbcDriver="com.mysql.jdbc.Driver"
jdbcUrl="jdbc:mysql://localhost/foodmart?user=foodmart&password=foodmart"
catalogUri="/WEB-INF/queries/FoodMart.xml">
```

Nakopírujeme soubor s JDBC ovladačem do adresáře TOMCAT_HOME/common/endorsed:

```
sudo cp /usr/share/mysql/mysql-connector-java-5.0.8-bin.jar
/usr/share/tomcat5/common/endorsed/
```

Nakopírujeme soubor xalan.jar do adresáře TOMCAT_HOME/common/endorsed:

```
sudo cp /usr/share/tomcat5/webapps/mondrian/WEB-INF/lib/xalan.jar
/usr/share/tomcat5/common/endorsed/xalan.jar
```

V případě potřeby změníme obsah souboru TOMCAT_HOME/conf/tomcat-users.xml. Číslo portu můžeme zjistit a nebo změnit v souboru TOMCAT_HOME/conf/server.xml.

Spustíme tomcat:

```
sudo /etc/init.d/tomcat5 start
```

Otevřeme webový prohlížeč a zadáme tuto adresu: http://localhost:8180/mondrian

Ukázka spuštěné aplikace je na obrázku 5.1.

Dodatek B

Uživatelský manuál

Zadáme do prohlížeče adresu k souboru index.html.

Zobrazí se nám stránka, která obsahuje seznam položek, na které se můžeme dotazovat:

- Dokumenty
- Jednotky
- Platby
- Události
- Výpůjčky
- Požadavky na půjčení

Pokud se například chceme dotazovat dotazovat na Výpůjčky, klikneme na Výpůjčky. Pokud bychom se později chtěli dotazovat na něco jiného (např. na Jednotky), pak se vrátíme na stránku index.html, kde klikneme na požadovaný odkaz.

Zobrazí se nám stránka se jménem aplikace, pod ním text, na co pokládáme dotazy, dále lišta s tlačítky a tabulkou s výsledkem připraveného dotazu (viz obr. 5.1). Parametry jsou zobrazeny jako řádky (Čas, Knihovna nebo další). Parametrů může být více. Hodnoty, které chceme zjišťovat, jsou zobrazeny jako sloupce (např. počet výpůjček). Někdy se může stát, že se stránce se zobrazí zpráva JPivot is busy, pak je dobré počkat nebo zavřít prohlížeč a znovu jej spustit.

Pokud chceme získat pobrobnosti (např. hodnoty pro konkétní knihovny), pak klikneme na tlačítko +, kterým si zobrazíme hierarchii hodnot. Pokud naopak chceme získat méně podrobné výsledky, klikneme na tlačítko -. Pokud chceme změnit seřazení hodnot, klikneme na obrázek u jména hodnoty (např. Počet žádostí). Možnosti jsou neseřazené hodnoty (modré kolečko), nebo vzestupné či sestupné seřazení.

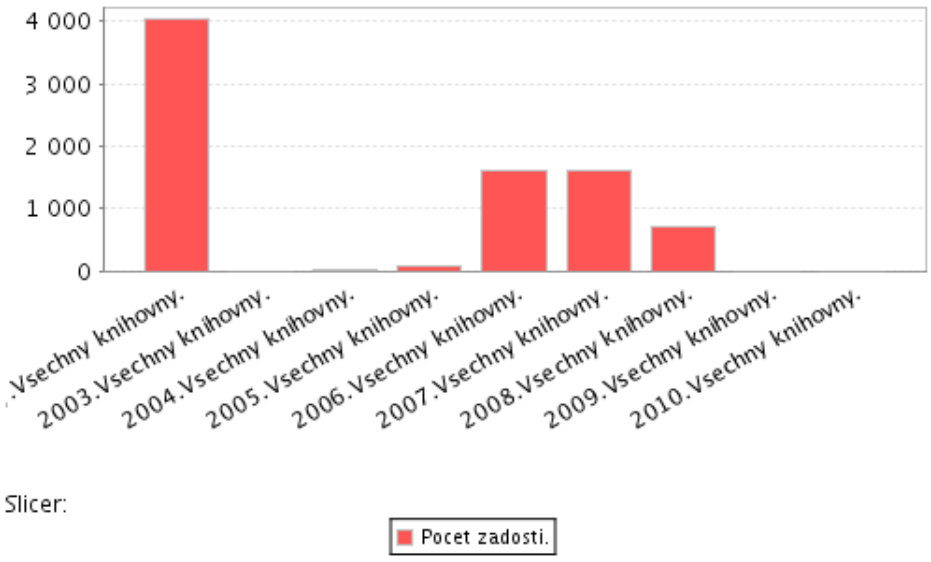
Tlačítka na liště:

- Výběr parametrů a hodnot
- Vytvoření vlastního MDX dotazu
- Volba řazení hodnot
- Zobrazit nebo nezobrazit předky členů

- Zobrazit nebo nezobrazit společné hodnoty
- Zobrazit nebo nezobrazit vlastnosti
- Zobrazit nebo nezobrazit prázdné řádky / sloupce
- Přehození řádků a sloupců
- 4 tlačítka pro nastavení zobrazení detailu
- 2 tlačítka pro zobrazení grafu a jeho nastavení
- 2 tlačítka pro nastavení a export do pdf
- Export do Excelu

Na první tlačítko klikneme, pokud chceme upravit tabulku (např. přidat knihovnu) - viz dále. Druhé tlačítko na liště moc používat nebudeme, museli bychom totiž znát jazyk MDX.

Pokud chceme zobrazit graf (obr. obrGraf), pak klikneme na tlačítko na liště s ikonou grafu (levé z nich). Tlačítkem vedle něj zobrazíme formulář pro změnu typu grafu (např. koláčový).

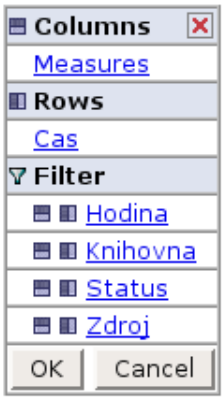


Obrázek B.1: Graf

Tlačítka s ikonami tiskárny na liště slouží pro nastavení a export do PDF. Poslední tlačítko na liště slouží k exportu do Excelu.

Úprava dotazu

Obr. B.2 obsahuje formulář pro nastavení dotazu (zde můžeme např. přidat parametr čas), kterým můžeme vytvořit dotaz dle našich požadavků. Tento formulář se zobrazí, pokud klikneme na první tlačítko na liště tlačítek. Pozn. tabulka s hodnotami, které si vybereme dle postupu dále, se překreslí, až klikneme na tlačítko ok pro projevení změn. Columns jsou sloupce, Rows řádky a Filter obsahuje seznam hodnot a parametrů, které můžeme vybrat a přidat je dotazu. U řádků a sloupců je ikona s půleným obrázkem, která nám pomáhá v orientaci, co je řádek a co sloupec.



Obrázek B.2: Formulář pro úpravu dotazu

Pokud chceme přidat parametr, který se má zobrazit v tabulce, do řádků, klikneme na malý obrázek se stejnou ikonou, jaká vedle textu Rows, vedle jména parametru, který chceme přidat (např. Knihovna). Pozor: dvě podobné hranaté ikony vedle jména parametru úplně vlevo se liší (jedna je pro zobrazení do řádků a druhá do sloupců).

Pokud chceme vybrat hodnoty (počet událostí), které se mají zobrazit, klikneme na odkaz Measures. Poté se nám zobrazí formuláře, ve kterém zaškrtnutím nebo nezaškrtnutím volíme, která hodnota se zobrazí ve výsledku

Pokud chceme umístit parametr nebo hodnotu, klikneme na ikonu s vodorovnou čárou vedle hodnoty, kterou chceme vybrat. Toto patrně nebude často používat, pokud nebudeme chtít vrátit hodnoty Measures mezi sloupce po našem překlepnutí.

Pokud chceme naopak parametr odebrat z řádků, tedy aby nebyl zobrazen v tabulce, klikneme na nálevku.

Pokud chceme změnit pořadí parametrů v tabulce, klikneme na šipku nahorů nebo dolů a tento parametr se v seznamů řádků posune výše nebo níže.

Pokud chceme upravit výsledek tak, že chceme vybrat jen některé hodnoty parametrů, klikneme na modrý odkaz se jménem tohoto parametru (např. knihovna). Zobrazí se nám formulář, kde si tlačítkem + zobrazíme seznam možných hodnot a klikneme na přepínač u položky, kterou chceme vybrat (např. Areálová knihovna FSI).

Pokud jsme se změnami skončili, klikneme na tlačítko ok, a tím se vykreslí tabulka v takovém tvaru, jaký jsme si vybrali.