

MASARYKOVA UNIVERZITA
FAKULTA INFORMATIKY



Implementace OLAP analýzy v prostředí knihoven MU

DIPLOMOVÁ PRÁCE

Lucie Navrátilová

Brno, podzim 2007

Prohlášení

Prohlašuji, že tato diplomová práce je mým původním autorským dílem, které jsem vypracovala samostatně. Všechny zdroje, prameny a literaturu, které jsem při vypracování používala nebo z nich čerpala, v práci řádně cituji s uvedením úplného odkazu na příslušný zdroj.

Vedoucí práce: Mgr. Martin Šárfy

Poděkování

Na tomto místě bych ráda poděkovala Mgr. Martinu Šárfymu za cenné rady a konzultace, věnovaný čas a v neposlední řadě poskytnutí exportů tabulek knihovního systému Aleph-MU.

Shrnutí

Diplomová práce se zabývá *implementací OLAP analýzy v prostředí knihoven MU*. *Hodnotí možné způsoby návrhu a implementace jednotlivých částí OLAP systému, který bude použit pro realizaci zmíněné analýzy. Vybírá nejvhodnější řešení návrhu a implementace OLAP systému vzhledem k operačnímu prostředí knihoven MU.*

Nejdříve je celý problém podrobně popsán. Je vysvětlen pojem OLAP systém, OLAP analýza, datový sklad a etapa ETL. Jsou uvedeny příklady řešení OLAP systému a možné nástroje a technologie, které lze použít pro OLAP analýzu, pro návrh datového skladu a realizaci etapy ETL. Po objasnění řešení celého problému implementace OLAP analýzy v prostředí knihoven MU se přistupuje k popisu samotného návrhu a implementace OLAP systému ve zmíněném prostředí.

Cílem diplomové práce bylo v prostředí knihoven MU navrhnout OLAP systém, který poskytne knihovnám uživatelsky příjemné prostředí, pomocí něhož mohou knihovníci či správci knihoven získat důležité informace pro rozhodování při řízení a správě knihovního systému. V neposlední řadě si klade za cíl, aby byla tato informace aktuální, nezkreslená a včasná.

Klíčová slova

report, OLAP, metadata, Data warehouse, Jasper, BEE, ETL, Beard, Osito, cache, Perl, XML, MDX, Drill-down, Roll-up, Pivoting, OLAP systém, ROLAP, OLAP krychle, tabulka faktů, tabulka dimenzí, hvězdicové schéma, schéma „sněhové vločky“, OWC, ORACLE, Aleph-MU

Obsah

1	Úvod	1
2	Popis problematiky	3
2.1	KKD a DWH technologie	3
2.2	OLAP systém	4
2.3	OLAP krychle systému	6
2.3.1	Tabulka faktů	8
2.3.2	Tabulky dimenzí	9
2.4	Možné přístupy k databázím OLAP z klientských aplikací	11
2.5	Prezentace poznatků OLAP analýzy klientovi	12
2.5.1	Hotové komponenty OWC (Office Web Components)	13
2.5.2	Konzolová aplikace nad dotazy jazyka MDX	13
2.5.2.1	Popis jazyka MDX	13
2.5.2.2	Stránky ASP pomocí ADOMD	17
2.6	Popis etapy ETL	18
2.7	DWH systému	20
2.7.1	Metadata – logický model datového skladu	22
2.7.2	Budování datového skladu	24
3	Návrh OLAP systému v prostředí knihoven MU	27
3.1	Popis systému Aleph–MU	27
3.2	Návrh datového skladu systému Aleph–MU	29
3.3	Výběr nástroje pro OLAP analýzu provozních dat Aleph–MU	34
3.3.1	JasperSoft Business Intelligence Suite	34
3.3.2	BEE Project Insight Strategy	35
4	Implementace OLAP systému v prostředí knihoven MU	37
4.1	Popis použitého ROLAP serveru Beard	38
4.2	Popis použitého nástroje Osito	39
4.3	Popis DWH metadat OLAP systému	40
4.4	Popis ETL skriptů	42
4.5	Instalace OLAP analyzátoru a spuštění systému	44
4.6	Popis uživatelského rozhraní	47
4.7	Ukázka tvorby reportu	50
5	Závěr	53
A	Schéma datového tržiště „sledování výpůjček“	54
B	Schéma datového tržiště „vyhodnocení jednotek“	55
C	Schéma datového tržiště „sledování událostí v systému“	56
D	metadata DWH MU	57
E	Obsah CD	61
	Literatura	63

Kapitola 1

Úvod

Současnost lze z hlediska nasazení informačních technologií označit jako dobu „datové exploze“. To proto, že máme sice velké množství nahromaděných dat, ale nemáme zpravidla k dispozici informaci použitelnou k rozhodování. Jsme stále ještě daleko ideálu informační společnosti „mít správnou informaci ve správný čas na správném místě“.

Každá organizace dnes disponuje nebývalým množstvím dat. V přeneseném smyslu slova tvoří tato data základ určité "paměti" organizace. Podobně jako inteligentní jedinec má schopnost optimalizovat své chování využitím zapamatovaných informací, tak je také třeba hledat způsoby, jak nalézt a využít informace nacházející se v „paměti“ organizace viz [1].

Systém řízení knihoven MU lze také v jistém smyslu chápat jako zmíněnou organizaci, jejíž data lze analyzovat a vhodnou analýzou takových dat získat důležité informace ohledně provozu a řízení provozu knihovního systému MU. Mezi informace tohoto typu patří generování různých statistik, týkajících se provozu knihoven MU, jako například počty výpůjček, počty majitelů nevrácených knih, přehled prošlých čtenářských registrací, celková cena knih, vytiženost knihovního systému, počty jednotek různého typu v knihovně, vícenásobné výpůjčky apod., to vše vztaženo například k vybranému období nebo konkrétnímu čtenáři nebo konkrétnímu knihovníkovi nebo konkrétní publikaci nebo skupině publikací nebo čtenářů nebo knihovníků nebo k vybraným fakultám nebo ke kombinaci těchto kritérií. Takovýchto kritérií a druhů informací je nepřehledné množství. K tomu abychom získali zmíněné informace, je nutné mít k dispozici nástroje, které by umožnily objevit v datech informaci, pomáhající optimalizovat rozhodování knihovníků na všech úrovních řízení knihovního systému MU.

Diplomová práce se zabývá problematikou OLAP systémů, OLAP analýzy a datových skladů. Hodnotí možná řešení implementace OLAP analýzy a datových skladů a na základě těchto hodnocení vytváří OLAP systém v prostředí knihoven MU. Popisuje návrh a implementaci jednotlivých částí zmíněného OLAP systému pro analýzu a vyhodnocování provozních dat knihoven MU.

Práce je rozdělena do čtyř kapitol. První kapitola je úvodem do problematiky OLAP systémů. Snaží se nalézt možné řešení návrhu a implementace OLAP systému v prostředí knihoven MU. Jednotlivé etapy tvorby OLAP systémů a jeho částí jsou zde vysvětleny a ukázány na příslušných tabulkách databáze knihovního systému Aleph-MU či tabulkách navrženého datového skladu pro OLAP analýzu provozních dat Aleph-MU. Konkrétně tedy popisuje problematiku OLAP krychlí a možné přístupy k databázím OLAP z klientských

aplikací. Seznamuje čtenáře s jazykem MDX, nástrojem používaného pro přístup k OLAP krychlím a pro výpis vybrané podmnožiny údajů z krychle v podobě dvojrozměrné kontingenční tabulky. Popisuje způsoby, kterými je možné prezentovat výsledky dotazů jazyka MDX v přehledné a srozumitelné formě (tabulky, grafy, jiné hierarchické struktury) směrem k uživateli. Vybírá nejvhodnější variantu prezentace výsledků OLAP analýzy v prostředí knihovního systému MU.

Zabývá se možnými metodami budování datového skladu a provozem datového skladu. Vybírá nejvhodnější metody pro návrh datového skladu pro OLAP analýzu provozních dat systému Aleph–MU. V neposlední řadě popisuje etapu ETL, tedy extrakci, transformaci a zavedení dat do datového skladu, která je nedílnou a velmi podstatnou součástí celého OLAP systému. Volí vhodnou variantu řešení etapy ETL pro nástroj OLAP analýzy provozních dat knihoven MU.

Druhá kapitola se věnuje návrhu jednotlivých částí systému pro OLAP analýzu provozních dat knihovního systému MU. Knihovní systém Aleph–MU je zde popsán z pohledu technologií, na kterých stojí, a z pohledu jeho možného rozšíření o datový sklad, který bude součástí OLAP systému budovaného nad provozními daty zmíněného knihovního systému. Je popsán návrh datového skladu a dostupné nástroje a technologie, které je možné použít pro OLAP analýzu dat v prostředí knihoven MU.

Třetí kapitola se věnuje implementaci navrženého OLAP systému v prostředí knihoven MU. Podrobněji je zde popsán nástroj, který byl vybrán jako vhodné řešení OLAP analýzy provozních dat knihovního systému Aleph–MU. Vysvětluje význam navržených metadat, na jejichž základě se vytváří pracovní prostředí pro nástroj provádějící OLAP analýzu provozních dat knihoven MU a význam ETL skriptů provádějících extrakci, transformaci a zavedení dat do datového skladu MU.

Dále seznamuje čtenáře s instalací a spuštěním navrženého systému realizujícího OLAP analýzu v prostředí knihoven MU. Popisuje uživatelské rozhraní nástroje a tvorbu reportů, jejichž výsledkem jsou užitečné informace, které jsou předmětem zájmu uživatele, předané uživateli v přehledné a srozumitelné formě (tabulka, graf, jiná hierarchická struktura).

Kapitola 2

Popis problematiky

Implementovat OLAP analýzu v prostředí knihoven MU znamená vytvořit OLAP systém, který poskytne knihovnám uživatelsky příjemné prostředí, pomocí něhož mohou knihovníci či správci knihoven získat informace důležité pro rozhodování při řízení a správě knihovního systému. Aby bylo použití systému efektivní, musí být tato informace, poskytnutá knihovníkům prostřednictvím systému, aktuální a včasná. Informace zde představuje jakousi znalost určité oblasti prostředí, která se mění podle toho, jak se mění samotné prostředí a lze ji pozorovat v závislosti na různých omezujících či konkretizujících kritériích. V případě knihovního systému jsou takovými znalostmi, odpovědi na otázky typu kolik bylo vypůjčených publikací typu sborník v měsíci červnu roku 2005 na fakultě informatiky nebo kolik existuje publikací typu kniha majících víc než 150 stran vydaných do roku 2000 nebo kolik publikací půjčil konkrétní knihovník fakulty informatiky v podzimním semestru nebo kolik rezervací provedl příslušný uživatel knihovního systému v lednu roku 2008 apod. Hledáním takové znalosti se zabývají KDD (*Knowledge Discovery in Databases*) technologie, s nimiž úzce souvisejí DWH (*Data warehousing*) technologie.

2.1 KKD a DWH technologie

KDD je mezioborová disciplína zahrnující použití statistických metod, vizualizace dat, expertních systémů, technologie databází, metod strojového učení a v neposlední řadě nástrojů pro tzv. OLAP (*On-Line Analytical Processing*). KDD lze chápat jako "proces netriviálního objevování implicitních, dopředu neznámých a potenciálně použitelných znalostí v datech" viz [18].

Proces KDD můžeme rozdělit do několika fází – selekce dat, příprava dat (čištění, transformace a kontrola kvality dat), objevování znalostí, prezentace znalostí.

1. **selekce dat** – zahrnuje činnosti, pomocí nichž lze získat základní představu o datech, která jsou k dispozici (posouzení kvality dat, první „vhled“ do dat, vytipování zajímavých podmnožin záznamů v databázích). Obvykle se zjišťují různé deskriptivní charakteristiky dat (četnosti hodnot různých atributů, průměrné hodnoty, minima, maxima apod.), s výhodou se využívají i různé vizualizační techniky viz [15].
2. **příprava dat** – zahrnuje činnosti, kterými jsou data speciálně extrahována, upravována, transformována, strukturována a uložena do datového skladu – DW (*Data*

warehouse). Tímto speciálním uložením dat se zabývají DWH (*Data warehousing*) technologie. Problematika DWH je blíže popsána v kapitole 2.7. Samotná příprava dat, která odpovídá tzv. ETL etapě je blíže popsána v kapitole 2.6.

3. **objevování znalostí** – jedná se o zpracování dat uložených v datovém skladě speciálními analytickými metodami. Jde o metody použité v rámci tzv. OLAP analýzy. Tato problematika je blíže popsána v kapitole 2.2.
4. **prezentace znalostí** – zahrnuje metody, které vyhodnocují dosažené výsledky z pohledu uživatelů, tedy z pohledu zda byly splněny cíle kladené na systém. Vytvořením vhodného modelu, celý návrh systému obecně nekončí. Získané znalosti je třeba upravit do podoby použitelné pro podporu rozhodování. Zavádí se speciální části OLAP systému pro automatickou klasifikaci nových případů vytvořených na základě prezentace výsledků analýzy ve srozumitelné formě koncovým uživatelům a jejich zpětné vazby kladení požadavků na systém. Prezentace výsledků OLAP analýzy směrem ke klientovi jsou blíže popsány v kapitole 2.5.

viz [15]

2.2 OLAP systém

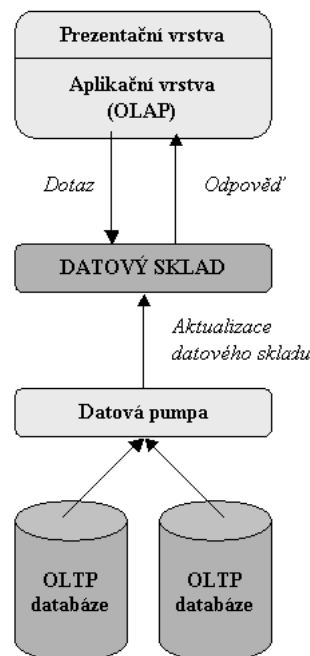
OLAP neboli „on-line analytical processing“, znamená okamžité zpracování dat. Spíše se tímto termínem rozumí pružné (rychlé) zpracování dotazů a analýz. Typicky se OLAP využívá pro analýzu velkého množství údajů. Výsledkem analýzy jsou souhrny a reporty, které slouží jako podklady pro podporu rozhodování, ať už v oblasti řízení firmy, řízení ekonomických a technologických procesů a podobně viz [1].

Rozlišujeme tyto základní typy operací analýzy OLAP:

1. **Drill-down** – umožňuje uživateli ve zvolených instancích jisté agregační úroveň nastavit nižší (jemnější) agregační úroveň viz [21].
2. **Roll-up** – jde o opak předešlé operace. Ve zvolených instancích jisté agregační úroveň nastavuje vyšší (hrubší) agregační úroveň viz [21].
3. **Pivoting** – umožňuje „otáčet“ datovou krychli, tj. měnit úhel pohledu na data na úrovni prezentace obsahu datového skladu viz [21].
4. **Slicing** – dovoluje provádět řezy datovou kostkou, tj. nalézt pohled, v němž je jedna dimenze fixována v jistých instancích jisté agregační úrovně. Jinými slovy tato dimenze aplikuje filtr na instance příslušné agregační úrovně dané dimenze viz [21].
5. **Diviny** – je obdobou „slicingu“, jenž umožňuje nastavit takový filtr pro více dimenzí viz [21].

OLAP analýza je nedílnou součástí OLAP systémů.

OLAP systém chápeme jako třívrstvou architekturu.



1. **spodní** – do této vrstvy patří server skladu, na kterém jsou uloženy relační databáze. Vrstvě odpovídá položka „Datový sklad“ viz [19].
2. **prostřední** – tato vrstva zahrnuje OLAP server, který obvykle implementuje buď relační OLAP model (ROLAP), což je rozšířený relační DBMS, který převádí operace nad multidimenzionálními daty na standardní relační operace. Druhou možností je multidimenzionální OLAP (MOLAP), který přímo umí pracovat s multidimenzionálními daty a operacemi. Tato vrstva koresponduje s aplikační vrstvou viz [19].
3. **vrchní** – vrchní vrstvu označujeme jako klienta. Obsahuje nástroje pro provádění dotazů a vytváření zpráv a analýzy, dále zahrnuje dataminingové nástroje určených pro analýzy trendu, predikce, apod. Shoduje se s prezentační vrstvou viz [19].

Celý systém lze rozdělit na dvě základní části. První z nich je OLAP systém. Na druhé straně stojí klasické databázové systémy, které se označují jako OLTP systém, což je zkratka „on-line transaction processing“ neboli „okamžité zpracování transakcí“.

Rozdílnost mezi OLAP a OLTP spočívá v tom, že OLTP systémy uchovávají záznamy o jednotlivých uskutečněných (typicky obchodních) transakcích a jsou obvykle realizovány pomocí dnes nejběžnější relační databázové technologie. Data uchovávaná v OLTP databázovém systému jsou (zpravidla periodicky) agregována (typicky sumarizována) a poté ukládána do datového skladu, nad nímž se posléze podle potřeb provádí okamžité zpracování analýz pomocí vrstvy OLAP viz [13].

OLAP systém využívá multidimenzionální přístup. Umožňuje přistupovat k obrovskému množství kombinací informací (řádově desítky miliónů) tak, že se nepracuje s jednotlivými údaji, ale s o mnoho menšími rozměry dimenzí těchto údajů. Údaje tak mají větší vypovídací schopnost.

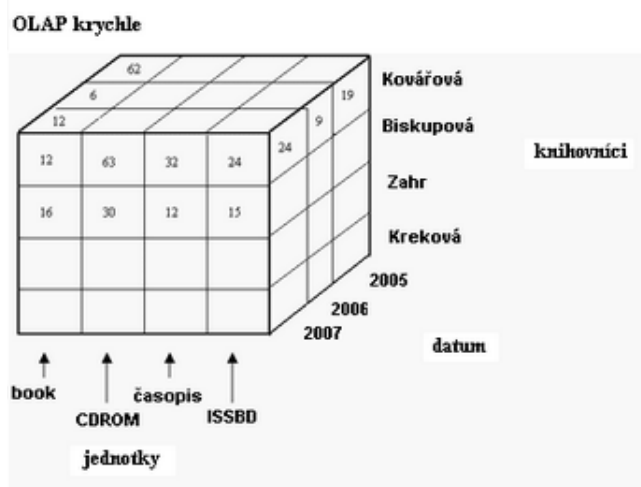
Systém lze dále rozdělit podle typu ukládání dat do datového skladu a následných analýz, do tří skupin:

1. **MOLAP** – je multidimenzionální způsob uložení dat s vysokým výkonem. V tomto přístupu jsou data ukládána na OLAP server. MOLAP poskytuje nejlepší výkon ve fázi dotazování (analýzy), neboť je právě pro mnoha dimenzionální dotazy speciálně optimalizován. Je vhodný pro malé až středně velké objemy dat, kdy kopírování všech dat do multidimenzionálního formátu nevyžaduje výrazně dlouhou dobu nebo nespotebovává příliš mnoho diskového prostoru viz [21].
2. **ROLAP** – při použití tohoto způsobu uložení dat, zůstávají data v původních relačních databázích. Oddělená sada relačních tabulek je použita k uložení agregací. ROLAP je vhodný pro rozsáhlé databáze nebo na stará data, která nejsou často analyzována viz [21].
3. **HOLAP** – slučuje prvky z předešlých dvou přístupů. Ponechává původní data v relačních tabulkách, ale ukládá agregace v multidimenzionálním formátu. HOLAP poskytuje propojení mezi rozsáhlými objemy dat v relačních tabulkách a zároveň nabízí výhodu rychlejšího výkonu multidimenzionálně uložených agregací viz [21].

Při návrhu nástroje pro OLAP analýzu provozních dat knihovního systému MU využívám ROLAP. Použití bylo zvoleno vzhledem k tomu, že se jedná o knihovní systém a tedy rozsáhlé databáze nebo archivní data, která nebudou tak často analyzována (tak často ve smyslu každý den nepřetržitě každou hodinu). HOLAP nebyl použit, není třeba zpracovávat a vyhodnocovat údaje knihovního systému v reálném čase, tak jak jsou ukládána do databáze. Vzhledem k rychlosti provozu knihovny a k nízké míře celkového zkreslení informací, ve srovnání aktuálnost dat na začátku dne a na konci dne a váhy reálné informace za delší období než je jeden den, kterou potřebujeme znát, byl systém navržen tak, že jsou všechna data potřebná pro analýzu zavedena jednorázově do datového skladu, který tak do příštího zavedení dat zůstává, co se týče zdroje dat, statický. Analýza je prováděna nad tímto v reálném čase neměnným datovým skladem, co se týče zavádění dat ze zdrojových systémů, a proto byl zvolen přístup ROLAP.

2.3 OLAP krychle systému

OLAP systém pracuje tak, že vytváří multidimenzionální OLAP krychle a využívá přitom datové hierarchie.



Datové krychle OLAP (Cube) představují výkonný nástroj pro vytváření sestav a pro komplexní OLAP analýzu dat. Obsahují informace o úkolech, zdrojích, projektech, přiřazeních, problémech, rizicích a závazcích. Jsou základním stavebním kamenem multidimenziálních databází, nad kterými se OLAP analýzy provádí. Tato krychle může mít na rozdíl od běžného fyzikálního modelu více dimenzí (až 64). Dimenze kostky reprezentují rozdílné kategorie pro analýzu dat. Kategorie jako například čas, geografické umístění nebo různé výrobní řady jsou typickými dimenzemi v datových kostkách. Dimenze jsou obvykle uspořádány do hierarchií tak, že mapují sloupce v relačních databázích. Hierarchie dimenzí jsou seskupovány do úrovní obsahujících hodnoty dané dimenze. Každá úroveň v dimenzi může být sumarizována, aby vytvořila hodnoty pro vyšší úroveň viz [1].

Např. v dimenzi času, sumarizací hodnot v úrovni den, získáme hodnoty pro vyšší úroveň měsíc. Vlastní výsledek neboli fakta se nachází v průniku jednotlivých dimenzí. Fakta nebo také míry jsou kvantitativní hodnoty v databázi, které mají být analyzovány viz [1].

Typickými mírami bývají prodeje, náklady a rozpočty. Míry jsou analyzovány oproti různým kategoriím dimenzí datové kostky. Např. analýza prodejů (míra) určitého výrobku (dimenze) v různých zemích (konkrétní úroveň dimenze geografická poloha) během dvou určitých roků (úroveň dimenze čas). Fakta se mohou kombinovat nebo vypočítat pomocí jiných faktů viz [1].

Výpočet krychle je časově a výkonnostně náročný proces, prochází se velké množství dat, nad nimiž se provádějí jednotlivé výpočty a agregace. Krychle jsou zpravidla uloženy někde na disku, kde a jak jsou uloženy, ví jen speciální program, který se nazývá server OLAP. Můžeme je prohlížet pomocí různých klientských aplikací, například známého tabulkového programu Microsoft Excel viz [23].

Blíže bych pojem OLAP krychle vysvětlila na jednoduchém příkladě, využitém při OLAP analýze knihovního systému MU. K vysvětlení použiji OLAP krychli výpůjčky (s názvem vypucka).

Nechť chceme vysledovat a přehledně zachytit stav, jak se mění počet výpůjček v přísluš-

ném roce, měsíci či dni v závislosti na typu vypůjčené publikaci a uživateli, který provedl výpůjčky.

Údaj rok, semestr, měsíc či konkrétní den nám v krychli představuje *tzv. časovou dimenzi*. Charakterizuje, kdy byl vypůjčen příslušný počet publikací v knihovně. Mějme dán příslušný počet výpůjček za konkrétní rok 2006, tento počet se postupně „rozmělní“, rozdělí do skupin o menších počtech výpůjček, tak jak se postupně zanořujeme hlouběji v roce 2006 do jednotlivých měsíců či dnů v příslušném měsíci.

Údaj typu vypůjčené publikace představuje *tzv. produktovou dimenzi*. Charakterizuje blíže typ příslušných publikací daných výpůjček. Má tři všeobecné hierarchické úrovně: druh, skupina, produkt.

Údaj typu uživatele, který provedl výpůjčky představuje *tzv. subjektovou dimenzi*. Charakterizuje, kdo konkrétně si vypůjčil či kdo konkrétně půjčil. I tato dimenze může mít všeobecné hierarchické úrovně: druh, skupina, subjekt.

Dimenze a fakta jsou zaznamenána v tabulkách dimenzí a faktů. Tyto tabulky a jejich speciální propojení tvoří multidimenzionální model, nad kterým se OLAP analýza provádí.

2.3.1 Tabulka faktů

Obsahuje numerické měrné jednotky, které jsou výsledkem OLAP analýzy a představují skutečnost, kterou sledujeme. Tabulka faktů je největší tabulkou v databázi a je součástí určitého schéma, například hvězdicového schéma nebo schéma sněhové vločky. Hvězdicové schéma obsahuje jen jednu tabulku faktů, jiná schémata mohou obsahovat i více tabulek faktů.

Tabulka faktů v případě sledování počtu výpůjček v příslušném roce, měsíci či dni v závislosti na typu vypůjčené publikaci a uživateli, který provedl výpůjčky, by vypadala následovně.

tabulka lu_vypujcka

Sloupec	Typ	Porovnávání	Vlastnosti	Nulový	Výchozí
<u>id</u>	smallint(5)		UNSIGNED	Ne	0
status_ctenare_id	tinyint(3)		UNSIGNED	Ne	0
status_jednotky_id	tinyint(4)			Ne	
typ_ctenare_id	tinyint(4)			Ne	
knihovnik_id_pujcujici	mediumint(8)		UNSIGNED	Ne	0
knihovnik_id_prijimajici	mediumint(8)		UNSIGNED	Ne	0
fakulta_id	tinyint(4)			Ne	
material_id	tinyint(4)			Ne	
datum_id_pujceni	date			Ne	0000-00-00
datum_id_vraceni	date			Ne	0000-00-00
datum_id_expirace	date			Ne	0000-00-00
datum_id_prodlouzeni	date			Ne	0000-00-00
pocet	smallint(5)			Ne	

2.3.2 Tabulky dimenzí

Jsou obvykle menší než tabulka faktů a data v nich se nemění častěji než je tomu u tabulky faktů. Tabulky dimenzí obsahují úhly pohledu na data. Obvykle mají stromovou strukturu.

Tabulky dimenzí, v případě sledování počtu výpůjček v příslušném roce, měsíci či dni v závislosti na typu vypůjčené publikace a uživateli, který provedl výpůjčku, by vypadaly následovně:

1. tabulka lu_datum

Sloupec	Typ	Nulový	Výchozí
<u>id</u>	date	Ne	0000-00-00
typ_dne_id	tinyint(3)	Ne	0
mesic_id	mediumint(6)	Ne	0
semestr_id	mediumint(6)	Ne	0
rok_id	smallint(4)	Ne	0

2. tabulka lu_status_jednotky

Sloupec	Typ	Nulový	Výchozí
<u>id</u>	smallint(4)	Ne	
status	char(2)	Ne	

3. tabulka lu_material

Sloupec	Typ	Nulový	Výchozí
<u>id</u>	smallint(4)	Ne	
nazev	char(5)	Ne	

4. tabulka lu_knihovnik

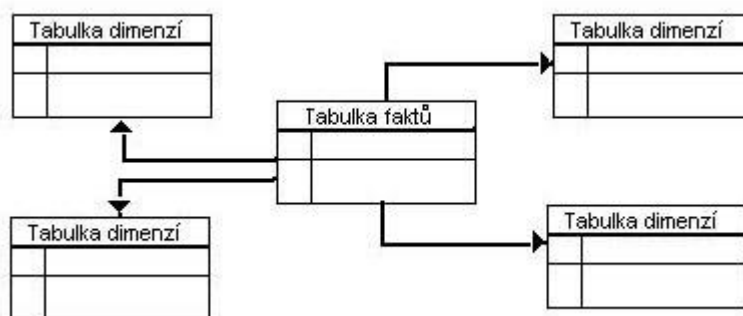
Sloupec	Typ	Nulový	Výchozí
<u>id</u>	mediumint(20)	Ne	
jmeno	char(50)	Ne	
typ_knihovnika_id	tinyint(3)	Ne	0

Dimenzionální model má určité topologické uspořádání tzv. schéma. Nejčastěji se používá hvězdicové schéma (star schema) nebo schéma „sněhové vločky“ (*snowflake schema*).

1. Hvězdicové schéma

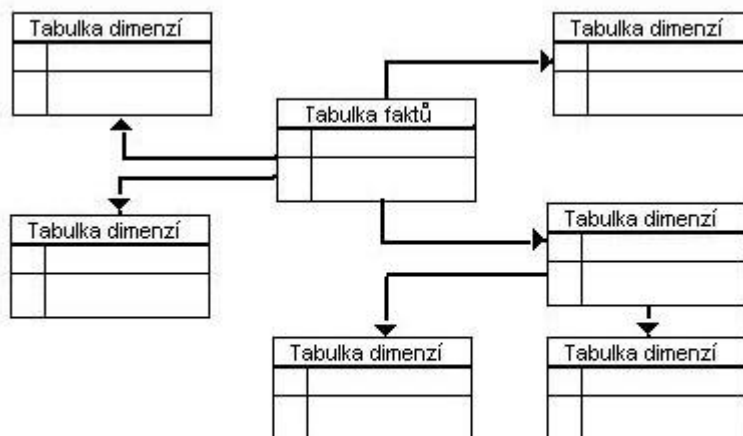
Skládá se z tabulky faktů obsahující cizí klíče, které se vztahují k primárním klíčům v tabulkách dimenzí. Hvězdicové schéma nemá normalizované dimenze ani

relační propojení mezi tabulkami dimenzí (tzn. nedodržují pravidla normálních forem), proto je velmi lehce pochopitelné. Ale v důsledku nenormalizovaných dimenzí je vytvoření takového modelu relativně pomalé, ale na druhé straně tento model poskytuje vysoký „dotazovací výkon“, protože všechny údaje získáme najednou a nemusíme je skládat z relačních tabulek. Grafické vyjádření schématu připomíná hvězdu, s tabulkami dimenzí zobrazenými v paprskovité struktuře okolo centrální tabulky faktů viz [1].



2. Schéma „sněhové vločky“

Obsahuje některé dimenze složené z mnoha relačně svázaných tabulek. Tento model umožňuje rychlejší zavedení údajů do normalizovaných tabulek, ale má podstatně nižší dotazovací výkon, neboť obsahuje větší množství spojení tabulek. Proto může být výkon systému nepříznivě ovlivněn. Z tohoto důvodu není schéma sněhové vločky tak časté při návrhu datového skladu jako hvězdicové schéma viz [1].



Při návrhu nástroje pro OLAP analýzu provozních dat knihovního systému MU využívám vločkové schéma. Z velké části se chová jako hvězdicové schéma, pouze dimenze času je

2.4. MOŽNÉ PŘÍSTUPY K DATABÁZÍM OLAP Z KLIENTSKÝCH APLIKACÍ

zde, tak jak se hierarchicky dělí, reprezentována příslušnými tabulkami a relacemi mezi těmito tabulkami. Místa, která by mohla být kritická z hlediska dotazovacího výkonu, jsou doplněna o předpřipravené agregace, proto není dotazovací výkon nijak snižován. Takto vzniklý model tedy poskytuje vysoký „dotazovací výkon“ směrem k uživateli a navíc je dobře hierarchicky strukturován pomocí postupných relací příslušných tabulek a proto není zavedení údajů do datového skladu tak časově náročné, jako by tomu bylo u prostého hvězdicového schématu.

2.4 Možné přístupy k databázím OLAP z klientských aplikací

Datové sklady a analytické databáze jsou použitelné a efektivní, když je mohou využívat příslušní klienti (manažeři, analytici). Analytické služby jsou proto k dispozici jako systém klient–server. V každém prostředí datového skladu se uživatelé a jejich požadavky na aplikační prostředí a typy dotazů liší v závislosti na jejich přístupu k předmětu podnikání nebo na jejich postavení ve firemní hierarchii. Každý potřebuje jiný segment údajů, jiný typ přístupu a jinak prezentované údaje. Při výběru existující nebo nové klientské aplikace je třeba brát v úvahu požadavky a potřeby jejich budoucích uživatelů viz [1].

V prostředí datového skladu dominují architektury typu klient–server a přístup k údajům přes web. Procesy pro přístup k údajům a jejich zpracování se tak rozdělí mezi servery a klientské počítače, na kterých se vykonává lokální zpracování.

Klientské aplikace se dělí podle toho, zda jádro aplikace běží na serveru nebo na klientském počítači. Dle tohoto kritéria rozlišujeme tzv. „tenkého“ a „tlustého“ klienta viz [1].

1. „Tenký“ klient

Pod pojmem „tenký klient“ myslíme zpravidla počítač, kde se pro přístup k údajům používá prohlížeč webových stránek. To znamená, že celá aplikační logika běží na serveru a výpočetní možnosti klientského počítače se víceméně nevyužívají, s výjimkou použití pro zobrazení. „Tenký“ klient je zpravidla navržen tak, aby byl malý, proto se hromadné zpracování dat vykonává na serveru. Chápeme pod tímto pojmem nejen počítač, ale už kapesní počítač, nejvýkonnější server nebo low-endové sestavy, ale i například mobilní telefony s podporou protokolu WAP, tzv. mobilního „tenkého“ klienta viz [1].

Výhodou „tenkého“ klienta je především podpora nejširšího spektra zařízení a platform a přístup prostřednictvím prohlížeče webových stránek. Dále jsou výhodou i nízké náklady na hardware na jednoho uživatele a také nízké licenční náklady na software, protože „drahé“ jádro softwaru je na serveru. Hlavní nevýhodou je, že takové aplikace z principu nepracují v režimu offline, ve většině případů málo využívají výkon lokálního zařízení viz [1].

2. „Tlustý“ klient

Pod pojmem „tlustý“ klient si lze představovat klientskou aplikaci, která běží na lokálním počítači, a tedy může naplno využívat výkon jeho hardwaru a možnosti

2.5. PREZENTACE POZNATKŮ OLAP ANALÝZY KLIENTOVI

operačního systému. Pro svoji činnost ale využívá údaje a službu nějakého serveru. „Tlustý“ klient na klientském počítači vykonává operace zpracování a vyhodnocení údajů. Samotné údaje se ukládají na server. Výhodou takového řešení je, že takové aplikace mohou pracovat v obou režimech offline i online, při čemž se může klientský počítač během připojení synchronizovat, případně replikovat údaje se serverem. Nevýhodou je, že pro každou platformu (Windows, Linux, iMAC, Pocket PC) musíme aplikaci vyvíjet zvlášť. Tato nevýhoda se částečně stírá použitím tzv. řízeného kódu, který je pro více platforem stejný viz [1].

Dle režimu ve kterém „tlustý“ klient pracuje, rozlišujeme připojeného a odpojeného „tlustého“ klienta.

- **Připojený „tlustý“ klient**

Pokud aplikace typu „tlustý“ klient pracuje v síťovém prostředí a má neustálou konektivitu na server, mluvíme v takovém případě o připojeném „tlustém“ klientovi. Když pro připojení použijeme protokol http, můžeme aplikace tohoto typu využívat prakticky všude, kde je připojen Internet viz [1].

- **Odpojený „tlustý“ klient**

Filosofie aplikací typu „tlustý“ klient je založena na poznatcích z praxe, kdy často potřebujeme výsledky analýz, hlavně sumární a parciální, a to i v takových situacích, kdy nemáme možnost připojení k síti (například na prezentacích u zákazníka). V takovém případě pracujeme s údaji v lokálních křehkých, uložených v souborech viz [1].

Při návrhu nástroje pro OLAP analýzu provozních dat knihovního systému MU byl použit přístup „tenkého“ klienta. Sám knihovní systém funguje jako systém klient-server, k systému jednotliví uživatelé (klienti) přistupují pomocí webového rozhraní. Tento model byl zachován i v případě prezentace výsledků OLAP analýzy. Důvodem a faktorem, který ovlivnil zvolení tohoto modelu je zvyklost uživatelů na webové rozhraní, podpora nejširšího spektra zařízení a platforem pro přístup prostřednictvím prohlížeče webových stránek a také to, že se data budou analyzovat vždy online a tedy není nutné volit například offline „tlustého“ klienta.

2.5 Prezentace poznatků OLAP analýzy klientovi

Při vývoji aplikací, které budou prezentovat poznatky OLAP analýzy v přehledné a srozumitelné formě směrem ke klientovi, existují v zásadě dvě možnosti jejich realizace. Buď se dají použít hotové komponenty OWC (*Office Web Components*) nebo vyvíjet vlastní webovou aplikaci. Při vývoji vlastní aplikace lze použít několik serverových skriptovacích systémů (*JSP, PHP, ASP, ASP.NET, PERL*). S ohledem na největší dostupnost se nejvíce používají pro případ prezentace OLAP analýzy stránky ASP viz [1].

2.5.1 Hotové komponenty OWC (*Office Web Components*)

Poměrně komfortní a na vývojářské práce velmi nenáročný způsob zobrazování výsledků analýzy OLAP. Výhodou těchto komponent, konkrétně PivotTable, Chart a Spreadsheet je, že je lze použít i bez uživatelského rozhraní například na stránkách ASP. Další výhodou je i skutečnost, že mohou generovat statický obsah, například obrázky ve formátu GIF, stránky HTML, případně soubory XML viz [1].

1. OWC PivotTable

Komponenta OWC Pivot Table odpovídá kontingenční tabulce, ta má na rozdíl od klasické tabulky několik speciálních vlastností. Například umožňuje určitou rotaci, tedy výměnu sloupců a řádků, kombinaci a hierarchickou strukturu řádků a sloupců. Objektová struktura je poměrně složitá a skládá se ze tří objektů OCCCommands, PivotData a PivotView, které se dále hierarchicky člení. Práce s komponentou je následující: umístí se na stránku HTML, její parametry se nastaví pomocí skriptu JavaScript a tento kód skriptu se bude interpretovat na počítači klienta. Parametry které se nastaví jsou připojovací řetězec na server OLAP a název krychle. Dále se nadefinuje které měrné jednotky a dimenze se kde zobrazí a zda se vůbec zobrazí a seznam polí, které lze dle potřeby přesunout na příslušné pole kontingenční tabulky. V neposlední řadě je třeba nadefinovat objekty objView.RowAxis, objView.ColumnAxis a objView.DataAxis viz [1].

2. OWC Chart

Komponenta totožná s kontingenčním grafem. Pracuje se s ní velmi podobně jako s komponentou PivotTable, stačí změnit classid, jinak je kód pro inicializaci stejný viz [1].

2.5.2 Konzolová aplikace nad dotazy jazyka MDX

Další možností, jak prezentovat výsledky OLAP analýzy v přehledné a srozumitelné formě směrem k uživateli, je vytvořit metadata napsaná v jazyce MDX, která budou načítána a interpretována pomocí například již existující konzolové aplikace. Příkladem takové konzolové aplikace je OWC, která přistupuje k dotazům jazyka MDX pomocí rozhraní ADODB nebo stránky ASP, které přistupují k dotazům jazyka MDX pomocí rozhraní ADOMD. Více rozšířeným způsobem interpretace OLAP analýz jsou stránky ASP s přístupem ADOMD, které následně podrobněji popíši.

Vzhledem k tomu, že nástroj pro OLAP analýzu provozních dat v prostředí knihoven MU používá při práci s OLAP krychlemi jazyk MDX, uvádím dále bližší popis tohoto jazyka.

2.5.2.1 Popis jazyka MDX

Je určitým ekvivalentem jazyka SQL v relačních databázích. Jeho hlavním cílem je navigace v multidimenzionálních údajích. Pomocí dotazu v MDX lze vypsát vybranou podmnožinu

2.5. PREZENTACE POZNATKŮ OLAP ANALÝZY KLIENTOVI

údajů z multidimenzionální struktury (krychle OLAP) do dvojrozměrné tabulky, která obsahuje množinu buněk tzv. „Cellset“ viz [1].

Úplná syntaxe jazyka MDX je následující:

```
SELECT [<specifikace_osy> [,<specifikace_osy>...]]  
  
FROM   [<specifikace_krychle>]  
  
[WHERE [<specifikace_rezu>]]
```

Na již zmíněném a používaném modelu OLAP kostky vypůjčka ukáží a vysvětlím postupně význam a použití jednotlivých příkazů a terminologii jazyka MDX viz [1].

1. **Prvek (Member):** může se jednat o kterýkoli prvek v hierarchii.

Př: [nazev].[BOOK]

2. **N-tice (tuple):** jeden prvek, případně průnik dvou i více prvků

Př: ([nazev].[CDROM], [fakulta_id].[1])

3. **Množina (set):** množina n-tic nebo prvků nebo také mnoha buněk ve výsledné tabulce. Každý prvek množiny musí mít stejný počet dimenzí.

Př: SELECT {[nazev].[BOOK], [nazev].[CDROM]} ON COLUMNS
FROM lu_vypujcka JOIN lu_material ON material_id= lu_material.id

4. **Dimenze:** množina elementů stejného typu.

```
Př: SELECT {[nazev].[BOOK], [nazev].[CDROM], [nazev].[ISSBD]}  
      ON COLUMNS,  
      {[datum_id_pujceni].[2005], [datum_id_pujceni].[2006], [datum_id_pujceni].[2007]}  
      ON ROWS  
FROM ((lu_vypujcka JOIN lu_material ON material_id= lu_material.id) JOIN  
      lu_datum ON datum_id_pujceni = lu_datum.id)
```

Vykreslí dvourozměrnou tabulku, která bude mít tři sloupce s názvy BOOK, CDROM, ISSBD a tři řádky s názvy 2005, 2006, 2007. V průniku tabulky bude počet výpůjček konkrétní publikace v daném roce.

Místo COLUMNS a ROWS lze také použít AXIS(0) a AXIS(1). Obecně lze použít až 128 dimenzí (AXIS(0) ... AXIS(127)).

Kromě absolutního definování os buněk můžeme buňky definovat i relativně pomocí klauzulí PrevMember a NextMember. Pokud máme například dotaz MDX pro údaje za kvartál datum_id_pujceni[2001], pak datum_id_pujceni[2001].PrevMember odpovídá údajům za datum_id_pujceni[2000] a datum_id_pujceni[2001]. NextMember odpovídá údajům za datum_id_pujceni [2002]. Ty samé příkazy lze zapsat pomocí funkcí LAG(1) a LEAD(1).

Dimenze os můžeme měnit pomocí klauzule MEMBERS na určité hierarchické úrovni.

2.5. PREZENTACE POZNATKŮ OLAP ANALÝZY KLIENTOVI

```
Př: SELECT {[nazev].[BOOK], [nazev].[CDROOM], [nazev].[ISSDB]} ON COLUMN
      {[datum_id_pujceni].MEMBERS} ON ROWS
FROM ((lu_vypujcka JOIN lu_material ON material_id= lu_material.id) JOIN lu_datum ON
      datum_id_pujceni = lu_datum.id)
```

Vykreslí dvourozměrnou tabulku, která bude mít tři sloupce s názvy BOOK, CDROM, ISSDB a v řádcích budou všechny možné navzájem různé roky, které obsahuje tabulka lu_datum. V průniku tabulky bude počet výpůjček konkrétní publikace v daném roce.

5. **Řezy:** používá se klauzule WHERE, která definuje výrazy údajů.

```
Př: SELECT {[nazev].[BOOK], [nazev].[CDROM], [nazev].[ISSDB]} ON COLUMNS,
      {[datum_id_pujceni].MEMBERS} ON ROWS
FROM ((lu_vypujcka JOIN lu_material ON material_id= lu_material.id) JOIN lu_datum ON
      datum_id_pujceni = lu_datum.id)
WHERE ([fakulta_id].[2], [status_jednotky_id].[44])
```

Vykreslí dvourozměrnou tabulku, která bude mít tři sloupce s názvy BOOK, CDROM, ISSDB a v řádcích budou všechny možné navzájem různé roky, které obsahuje tabulka lu_datum. V průniku tabulky bude počet výpůjček konkrétní publikace v daném roce, ale omezené fakultou a statutem jednotky (publikace).

6. **Specifikace intervalu os:** používá se pokud je na některé z os potřeba vymezit určitý interval.

```
Př: SELECT {[nazev].[BOOK], [nazev].[CDROM], [nazev].[ISSDB]} ON COLUMNS,
      {[datum_id_pujceni].[2005] : {[ datum_id_pujceni].[2007] } ON ROWS
FROM ((lu_vypujcka JOIN lu_material ON material_id= lu_material.id) JOIN lu_datum ON
      datum_id_pujceni = lu_datum.id)
```

Vykreslí dvourozměrnou tabulku, která bude mít tři sloupce s názvy BOOK, CDROM, ISSDB a v řádcích budou roky ležící v intervalu <2005,2007>, které obsahuje tabulka lu_datum. V průniku tabulky bude počet výpůjček konkrétní publikace v daném roce.

7. **Fakta:** měrné jednotky, které se zobrazí na průsečících dimenzí.

8. **Vypočtené prvky – klauzule WITH:** pomocí klausule WITH a aritmetických operátorů a funkcí, lze vypočítat potřebné prvky, které se nám hodí a nejsou vždy přímo v hierarchii prvků.

```
Př: WITH
  MEMBER [1 čtvrtletí] AS 'Sum({[mesic_id].[24001]], [mesic_id].[24002], [mesic_id].[24003]})',
  MEMBER [2 čtvrtletí] AS 'Sum({[mesic_id].[24004]], [mesic_id].[24005], [mesic_id].[24006]})',
  SELECT {[nazev].[BOOK], [nazev].[CDROM], [nazev].[ISSDB]} ON COLUMNS,
      {[datum_id_pujceni].[ 1 pulrok] : {[ datum_id_pujceni].[ 2 pulrok] } ON ROWS
FROM ((lu_vypujcka JOIN lu_material ON material_id= lu_material.id) JOIN lu_datum ON
      datum_id_pujceni = lu_datum.id)
```

Vykreslí dvourozměrnou tabulku, která bude mít tři sloupce s názvy BOOK, CDROM, ISSDB a v řádcích bude 1. čtvrtletí a 2. čtvrtletí. V průniku tabulky bude počet výpůjček konkrétní publikace za první čtvrtletí a za druhé čtvrtletí.

9. Vyloučení prázdných buněk – klauzule NON EMPTY

```
Př: WITH
  MEMBER [1 čtvrtletí] AS 'Sum({[mesic_id].[24001]], [mesic_id].[24002], [mesic_id].[24003]})'
  MEMBER [2 čtvrtletí] AS 'Sum({[mesic_id].[24004]], [mesic_id].[24005], [mesic_id].[24006]})'
  SELECT NON EMPTY {[nazev].[BOOK], [nazev].[CDROM], [nazev].[ISSDB]}
    ON COLUMNS,
    {[datum_id_pujceni].[1 pulrok] : {[datum_id_pujceni].[2 pulrok]}
    ON ROWS
  FROM ((lu_vypujcka JOIN lu_material ON material_id= lu_material.id) JOIN lu_datum ON
    datum_id_pujceni = lu_datum.id)
```

Vykreslí dvourozměrnou tabulku, která bude mít tři sloupce s názvy BOOK, CDROM, ISSDB a v řádcích bude 1. čtvrtletí a 2. čtvrtletí. V průniku tabulky bude počet výpůjček konkrétní publikace za první čtvrtletí a za druhé čtvrtletí, průnik nebude obsahovat prázdnou hodnotu.

10. Podmínka IIF: Podmínka vztažená na fakta.

```
Př: WITH
  MEMBER [BIG COUNT] AS 'IIF([pocet > 10000], "Yes", "No")'
  SELECT NON EMPTY {[nazev].[BOOK], [nazev].[CDROM], [nazev].[ISSDB]} ON COLUMNS,
    {[pocet, [BIG COUNT]]} ON ROWS
  FROM ((lu_vypujcka JOIN lu_material ON material_id= lu_material.id) JOIN lu_datum ON
    datum_id_pujceni = lu_datum.id)
```

Vykreslí dvourozměrnou tabulku, která bude mít tři sloupce s názvy BOOK, CDROM, ISSDB a v řádcích bude pocet, BIG COUNT. V průniku řádku pocet a libovolného sloupce tabulky bude počet výpůjček konkrétní publikace a v průniku řádku BIG COUNT a libovolného sloupce tabulky bude hodnota "Yes" pokud je počet větší než 10000 a hodnota "No" pokud je tomu naopak.

11. Křížové spojení – operátor CrossJoin(): slouží ke spojení dvou různých množin a výpisu všech možných kombinací.

```
Př: SELECT NON EMPTY CROSSJOIN({[nazev].[BOOK], [nazev].[CDROM], [nazev].[ISSDB]},
  {[datum_id_pujceni].[2005]: {[datum_id_pujceni].[2007]}) ON COLUMNS,
  {[fakulta_id].MEMBERS} ON ROWS
  FROM ((lu_vypujcka JOIN lu_material ON material_id= lu_material.id) JOIN lu_datum ON
    datum_id_pujceni = lu_datum.id)
```

Vykreslí dvourozměrnou tabulku, která bude mít dělené sloupce s názvy BOOK, CDROM, ISSDB, dělit se budou do menších sloupců s názvy roků 2005, 2006, 2007 a v řádcích budou jednotlivé fakulty MU. V průniku tabulky bude počet výpůjček konkrétní publikace v příslušném roce u dané fakulty, Průnik nebude obsahovat prázdnou hodnotu.

12. Redukce údajů – operátor Filter(): v závislosti na požadavcích zobrazení údajů je často potřeba omezit množinu údajů.

Pro tento účel se používá operátor Filter a operátory porovnání (=, <, >, <=, >=, <=) a logické operátory AND, OR, NOT

2.5. PREZENTACE POZNATKŮ OLAP ANALÝZY KLIENTOVI

```
Př: SELECT {[nazev].[BOOK], [nazev].[CDROM], [nazev].[ISSDB]} ON COLUMNS  
      FILTER {[datum_id_pujceni]. MEMBERS > 2006} ON ROWS  
      FROM ((lu_vypujcka JOIN lu_material ON material_id= lu_material.id) JOIN lu_datum ON  
            datum_id_pujceni = lu_datum.id)
```

Vykreslí dvourozměrnou tabulku, která bude mít tři sloupce s názvy BOOK, CDROM, ISSDB a v řádcích budou všechny možné navzájem různé roky, mladší než je rok 2006, které obsahuje tabulka lu_datum. V průniku tabulky bude počet výpůjček konkrétní publikace v daném roce.

13. **Setřídění množiny – operátor Order():** třídění údajů v případě (DESC) sestupně a v případě (ASC) vzestupně.

Třídění absolutní bez ohledu na hierarchii

```
Př: SELECT {[pocet]} ON COLUMNS,  
      NON EMPTY ORDER( {[nazev]. MEMBERS}, [pocet], BDESC) ON ROWS  
      FROM ((lu_vypujcka JOIN lu_material ON material_id = lu_material.id) JOIN lu_datum ON  
            datum_id_pujceni = lu_datum.id)
```

Vykreslí dvourozměrnou tabulku, která bude mít jeden sloupec s názvem pocet a v řádcích budou všechny možné typy publikací, které obsahuje tabulka lu_vypujcka. V průniku tabulky bude počet výpůjček konkrétní publikace, ale setříděné absolutně.

Třídění hierarchicky

```
Př: SELECT {[pocet]} ON COLUMNS,  
      NON EMPTY ORDER( {[nazev]. MEMBERS}, [pocet], DESC) ON ROWS  
      FROM ((lu_vypujcka JOIN lu_material ON material_id = lu_material.id) JOIN lu_datum ON  
            datum_id_pujceni = lu_datum.id)
```

Vykreslí dvourozměrnou tabulku, která bude mít jeden sloupec s názvem pocet a v řádcích budou všechny možné typy publikací, které obsahuje tabulka lu_material. V průniku tabulky bude počet výpůjček konkrétní publikace, ale setříděné hierarchicky. Publikace se dělí do podskupin dle hodnoty materiál_id.

2.5.2.2 Stránky ASP pomocí ADOMD

Pro stránky ASP není zapotřebí, kromě operačního systému a webového serveru, žádné vývojové prostředí. Stránkami ASP jsou soubory umístěné v příslušném adresáři na serveru s příponou ASP. Jsou to v principu stránky HTML (s příponou ASP) doplněné o kód ve skriptovacím jazyce. Pokud řádek na stránce ASP obsahuje kód skriptu, interpretuje se a jeho výstupy jsou vloženy do stránky HTML pro klienta. Stránky ASP přistupují k multidimenzionálním databázím přes rozhraní ADOMD viz [1].

Rozhraní ADOMD (ActiveX Data Objects Multidimensional) je rozšířením rozhraní ADO o podporu práce s multidimenzionálními údaji. Rozšíření spočívá v zavedení objektů CATALOG a CELLSET viz [1].

Objekt CATALOG slouží k získávání metadat, tedy informací o multidimenzionální databázi.

Objekt CELLSET obsahuje výsledky dotazů jazyka MDX v multidimenzionální formě.

Údaje se zobrazují na HTML stránce u klienta ve formě tabulky. Počet a názvy dimenzí pro vytvoření záhlaví tabulky se nemusí navrhovat „napevno“, získají se z objektu

ADOMD.Catalog. Což je nesporná výhoda a dynamická adaptace na změny v dotazech, kladených jazykem MDX viz [1].

Při návrhu nástroje pro OLAP analýzu provozních dat knihovního systému MU byl použit, pro prezentaci výstupů OLAP analýzy v přehledné a pochopitelné formě směrem ke klientovi, open source nástroj Osito, který je součástí projektu BEE. Bližšímu popisu se věnuje kapitola 4.2.

2.6 Popis etapy ETL

Údaje které budou analyzovány, pocházejí zpravidla z různých nehomogenních zdrojů. Mohou to být údaje ze souborových databází (*Access, dBase*), údaje z databází spravovaných některým databázovým serverem (*Oracle, Informix, Microsoft SQL Server, Sybase, Interbase, Ingres*). Mohou to být údaje vyexportované některou z databázových platforem do tzv. „flat file“ a podobně. Příprava a zavedení údajů je důležitou součástí každého řešení datového skladu. Údaje z operačního prostředí je potřebné před zavedením do datového skladu vyextrahovat, vyčistit, upravit a až následně ve vhodné formě do datového skladu zavést. Tato příprava údajů se označuje jako etapa ETL (*Extraction, Transformation, Loading*) viz [18].

Hlavním cílem etapy je centralizace údajů, jejich shromáždění z mnoha zpravidla nehomogenních a různorodých zdrojů z databází OLTP systému, a naplnění datového skladu určenými údaji v požadovaném čase. Údaje se v této etapě nejen přenášejí, ale i zpracovávají, například indexují, sumarizují, zjišťují se případné změny struktur zdrojových údajů potřebných pro datový sklad, podle potřeby se mění struktura klíčů a udržují se metadata, tedy data o datech, v tomto případě předpisů a definic pro přenos a zpracování údajů.

Etapu ETL v jistém smyslu předchází tzv. vynášení údajů do jakéhosi „staveniště“ datového skladu, kterým je oblast pro vynášení údajů. Z hlediska implementace se může jednat například o paměť operačních dat, adresát textových nebo flat souborů, tabulky v relační databázi nebo vlastní struktury údajů, které používají nástroje určené pro vynášení dat. Z hlediska modelů se dají použít dva modely vynášení: viz [1]

- **Model lokálního vynášení**

Procesy úpravy a transformace údajů se v tomto případě vykonávají nejdříve, a to lokálně v operačním prostředí a až po tom se přenášejí do vynášecí oblasti. Tento způsob může zatěžovat operační paměť počítače, takže je ho možné využívat jen u méně vytížených systémů viz [1].

- **Model vzdáleného vynášení**

V tomto případě se data nejdříve přenesou z operačního systému do vynášecí oblasti, nebo dokonce přímo do prostředí datového skladu, a až tady se zpracují viz [1].

Poté již následuje Etapa ETL, skládající se z extrakce dat, transformace dat a přenosu dat do datového skladu.

1. Extrakce

Údaje, které chceme přenést do datového skladu, mohou být umístěny v různých nehomogenních operačních prostředích, hardwarových platformách, operačních systémech, archivních systémech a mohou se vyskytovat v rozličných formátech. Úlohou extrakce je získat údaje z takových zdrojů.

Existují dvě hlavní extrakční metody – celkové extrakce a replikace založené na změnách viz [1].

V prvním případě je celý obsah datového skladu pravidelně obnovován daty ze zdrojových systémů a získávají se z nich všechna použitelná data. Tento případ klade velké nároky na síťové připojení mezi zdrojovými a cílovými databázemi, ale datový sklad je snadno připravitelný a spravovatelný viz [1].

Ve druhém případě jsou do datového skladu nahrána pouze nově přidaná či změněná data ve zdrojových systémech. Tento přístup znamená menší síťový přenos (díky menšímu objemu přenášených dat), ale vyžaduje složitější programování a učení, kdy musí být uložen do databáze nový záznam či kdy musí být existující záznam aktualizován viz [1].

2. Transformace

Data v datovém skladu musí být kvalitní. Když jsou data v datovém skladu nekvalitní, snižuje to důvěru v taková řešení a datový sklad se oprávněně nevyužívá. Jádrem problému je v tom, že nekvalitní data se dost často vyskytují ve zdrojových systémech. Použití nekvalitních údajů vede k chybným nebo minimálně nepřesným sestavám a následně též k chybným rozhodnutím nebo opatřením nápravy stavu dané situace. Údaje z externích zdrojů mají určitou kvalitu, která je buď postačující nebo nepostačující pro jejich zavedení do datového skladu viz [1].

Pro zvýšení kvality údajů dochází k transformaci údajů. Sjednocuje se formátování údajů, přiřazení datových typů, jednotek míry a peněžních měn. Údaje v databázích často obsahují různě kódované údaje nebo i primární klíče, které se skládají z více částí. Při zavedení těchto údajů do datových skladů je potřebné rozložit tyto údaje na atomické hodnoty. Při transformaci údajů se zpravidla řeší nejednoznačnost údajů, kdy stejná hodnota příslušných údajů je uložena různě, dále se řeší chybějící hodnoty, případně otevřená či skrytá duplicita záznamů. Řeší také otázku sloučení terminologie a vytvoření jednotné konvence názvů v případě, že slučujeme údaje z různých zdrojů, které v podstatě popisují stejný jev, ale mají jednotlivé entity vedeny pod různými názvy. V neposlední řadě se nutně zabývá řešením referenční integrity a chybějícího datumu viz [1].

Kromě hodnot jsou v údajích skryty i různé vztahy, ty se mohou změnit, po takové změně zůstávají v databázi záznamy tzv. “údajové sirotkové”, které mohou nepříz-

nivě ovlivnit kvalitu údajů. Čas plní v datovém skladu významnou úlohu. Většina analytických databází má časovou dimenzi. Časový údaj musí být přítomný v datech před jejich zavedením do datového skladu, nebo se musí určit a přidat při zavádění dat. Transformaci je možné vykonávat sériově či paralelně se zaváděním údajů. U sériového způsobu se transformace vykoná před zavedením dat do datového skladu. U paralelní metody se tento proces vykonává souběžně se zavedením viz [1].

3. Přenos

Završením etapy ETL je přenos údajů z paměti zdrojových dat nebo přechodné vynášecí oblasti do datového skladu. Přenos spočívá v přesunu údajů a jejich uložení do databázových tabulek datového skladu. Přenos by měl být plánovaný a automatizovaný. Při prvotním naplnění datového skladu může jít o obrovské množství údajů. Potom se již údaje zavádějí v pravidelných časových obdobích, například každý den, a to v takových objemech, kolik údajů za dané období v databázových OLTP vznikne. Po zavedení údajů zpravidla dochází k jejich indexování, aby byl přístup k nim optimalizovaný. Pro jednoznačnou identifikaci údajů se používají i uměle vytvářené klíče, s jejichž pomocí zajistíme jednoznačnost každého řádku v tabulce. Je to z toho důvodu, že jsou data datového skladu často kombinací mnoha transformovaných záznamů, které nemají žádné přirozené klíče, které by se daly použít pro jednoznačnou identifikaci viz [1].

Při návrhu nástroje pro OLAP analýzu provozních dat knihovního systému MU jsem použila model vzdáleného vynášení. Knihovní systém Aleph–MU je sám o sobě značně vytížený systém, proto jsem volila zmíněný model, aby nezatěžoval operační paměť víc než je nutné. Příslušné tabulky jsou z databáze spravované databázovým serverem ORACLE exportovány do flat souborů, ty jsou transformovány do flat souborů představující sql tabulky datového skladu knihovního systému MU. Tabulky jsou kopírovány do vynášecí oblasti /opt/ETL/data_aleph_mu.

Vzhledem k tomu, že data jsou extrahována z homogenního systému Aleph–MU, kde jsou uložena do tabulek jedné struktury pro stejná data a celá databáze je spravována jedním databázovým serverem ORACLE a to vše v jednotném prostředí operačního systému UNIX, fázi extrakce představuje vlastně v jistém smyslu model vzdáleného vynášení. Pro zbylé dvě fáze ETL byly navrženy skripty (uložené v adresáři /opt/ETL), dle nich probíhá transformace dat a zavedení dat do datového skladu.

2.7 DWH systému

Datový sklad (*Data Warehouse*) je podnikově strukturovaný depozitář subjektivě orientovaných, integrovaných, časově proměnlivých, historických dat, použitých na získávání informací a podporu rozhodování viz [12].

Subjektově orientovaný znamená, že údaje se do datového skladu vkládají podle předmětu zájmu (výpůjčky, jednotky, události v systému z pohledu knihovníka, čtenáře, fakulty, typu publikace apod.).

Integrovaný znamená, že údaje týkající se konkrétního předmětu se do datového skladu ukládají jen jednou. Proto se též zavádí jednotná terminologie, jednotné a konzistentní jednotky veličiny. Před zavedením do datového skladu jsou data nejdříve upravena, vyčištěna a sjednocena, provádí se etapa ETL (*Extraction, Transformation, Loading*).

Časově proměnlivý znamená, že se údaje ukládají do datového skladu jako série snímků, z nichž každý reprezentuje určitý časový úsek. Údaje jsou tedy platné pro určitý časový moment, časový snímek viz [12].

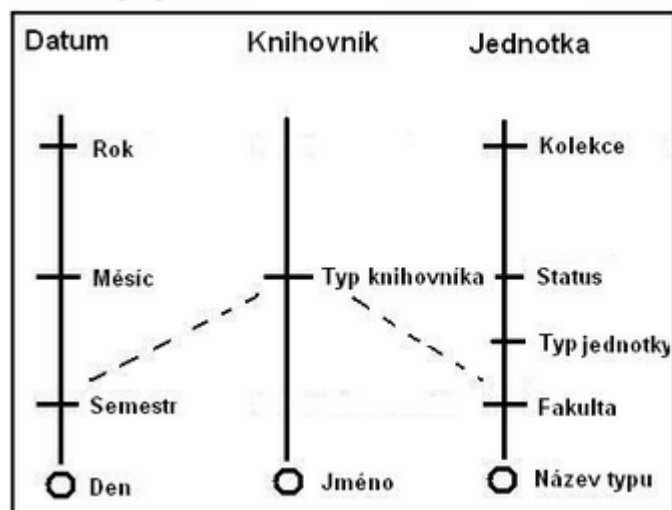
Neměnný znamená, že údaje v datovém skladě se obvykle nemění ani neodstraňují, jen se v pravidelných intervalech přidávají údaje nové. V zásadě existují v datových skladech jen dva typy operací, zavedení údajů do datového skladu a přístup k těmto údajům. V datovém skladu jsou uložena atomická a sumární data. Je to v podstatě databáze jejíž tabulky nemusí splňovat pravidla normálních forem.

Velmi významnou součástí datového skladu jsou agregace a sumace. Možnost používat agregace velmi zvyšuje výkon datového skladu z hlediska rychlosti zpracování dotazů, a proto také zvyšuje celkový výkon a použitelnost skladu.

Agregace je předpřipravená sumace uložená v rámci skladu, obvykle v samostatném schématu. Agregace jsou zpravidla vypočítávány na základě záznamů na nejpodrobnější úrovni a jsou využívány při zpracování dotazů, které vyžadují pouze data na nejvyšší úrovni nebo sumarizovaná data. Jsou to tedy sumace dat na základní úrovni pro vyšší úrovně v hierarchiích dimenzí viz [1].

Místo dotazu na vysoké úrovni nad daty na základní úrovni tedy mohou uživatelé provést dotaz nad agregovanými daty, čímž se významně redukuje počet zpracovávaných záznamů a tedy zvyšuje výkon viz [12].

Příklad agregace demonstrováné na dimenzích datového skladu knihoven MU



K údajům v datovém skladě a výsledkům analýz nad těmito údaji se zpravidla přistupuje prostřednictvím webu, přidělením přístupových práv. Datový sklad v malém je též možné simulovat i na trochu lépe vybaveném desktopovém počítači.

Prostředí datových skladů jsou obvykle oddělena od prostředí provozních systémů, především z důvodu zabránění možným „bojům“ mezi provozními a rozhodovacími procesy. Jasně oddělení výpočetních platforem pro provozní systémy a systémy podporující rozhodování poskytuje podniku příležitost využít odlišné platformy pro datový sklad od platforem pro provozní systémy viz [12].

V průběhu projektu datového skladu se používá několik typů odlišných softwarových nástrojů. Tyto softwarové produkty lze obvykle rozdělit do těchto tří následujících kategorií:

1. **Extrakce a transformace:** pro extrakční a transformační procesy potřebuje realizační tým nástroje, které mohou získávat, přenášet, integrovat a čistit data ze zdrojových systémů a těmito daty plnit databáze datového skladu. Mohou být využívány brány a další middleware získávající data z centrálně řízených zdrojových systémů viz [12].
2. **Ukládací prostor:** jsou potřebné také produkty pro uložení dat a odpovídajících metadat viz [12].
3. **Přístup k datům:** jsou potřebné různé typy programů pro přístup k datům a k jejich distribuci a poskytnutí koncovým uživatelům viz [12].

2.7.1 Metadata – logický model datového skladu

Metadata vytváří jakýsi logický model datového skladu. Jsou určitou formou abstrakce, která popisuje strukturu a obsah datového skladu. Abstrakce položky skladu je vlastně popis na základě jejich jednotlivých vlastností (atributů). Například osoba může být popsána svým jménem, věkem, pohlavím, povoláním apod viz [12].

V technologiích datových skladů je abstrakce aplikována na datové zdroje, extrakční a transformační pravidla a programy, datové struktury a také na obsah datového skladu viz [12].

Metadata stanovují obsah dat ve skladu. Pomáhají administrátorům a uživatelům určit a pochopit datové položky, a to jak ve zdrojových systémech, tak i v datových strukturách skladu. Metadata popisují přesný formát položky, čímž je jasně určen význam položky viz [12].

Metadata usnadňují provádění analýz. Umožňují uživatelům rychle lokalizovat požadovaná data. Analytikům umožňují správně interpretovat data, a to pomocí informací o datových formátech a o definici dat viz [12].

Metadata popisují transformaci zdrojových dat do datového skladu. Na základě metadat musí být jasné, jaký konkrétní obsah datového skladu byl odvozen z provozních systémů. Pomocí metadat jsou popsána také všechna pravidla, která regulují transformaci dat

do nových hodnot či nových formátů. Používají se též na generování extrakčních a transformačních skriptů, které je pak možné využít pomocí back-end komponent datového skladu (komponent pro zpětný zápis získaných informací z analýzy do skladu) viz [12].

Metadata mohou zvyšovat či udržovat kvalitu dat v datovém skladu a to pomocí definování přípustných hodnot pro jednotlivé datové položky skladu. Před vlastním nahráním do datového skladu mohou být nahrávací obrazy zkontrolovány nástrojem pro zajištění kvality dat, který ověří, že hodnoty jsou v souladu s přípustnými hodnotami pro jednotlivé datové položky. Obsahují též v takovém případě popis pravidel pro opravu chyb viz [12].

Obecně se rozlišují dva typy metadat:

1. Administrativní metadata

Popisují zdrojové databáze a jejich obsah, objekty datového skladu a pravidla určující transformaci dat ze zdrojových systémů do datového skladu viz [12].

Těmito metadaty jsou popsány všechny zdroje dat použité datovým skladem, včetně informací o vlastnictví dat, dále je definován každý záznam a každá datová položka, čímž je zajištěno jejich jednotné chápání a v neposlední řadě vazby mezi různými zdroji dat viz [12].

Jsou jimi přiřazeny zdrojové atributy (nacházející se v provozních systémech) cílovým atributům (v datovém skladu) a stanoveny transformace a změny formátů aplikované při získávání dat pro datové sklady viz [12].

Tato kategorie metadat popisuje servery, databáze, databázové tabulky, atributy a libovolné hierarchie, které mohou v datech existovat. Popis se týká také všech referenčních tabulek, systémových kódů apod viz [12].

Jsou jimi definovány všechny extrakční a transformační nástroje včetně programů pro zajištění kvality dat, které jsou určeny pro vybudování či obnovování datového skladu. Součástí jsou také informace o tom, v jakém pořadí a jak často jsou tyto programy spuštěny a popsány podpůrné tabulky, tabulky nahrávacího obsahu a libovolných dalších dočasných datových struktur používaných během transformace dat viz [12].

Zpravidla jsou jimi definována pravidla určující přístupová oprávnění a zabezpečení viz [12].

2. Metadata koncových uživatelů

Pomáhají uživatelům ve vytváření dotazů a se správnou interpretací výsledků. Pro uživatele může být také užitečná znalost definic a popis dat a hierarchie, které mohou existovat v rámci různých dimenzí viz [12].

Tento typ metadat popisuje datové struktury a obsah datového skladu v uživatelsky přívětivých pojmech a pravidla pro vytváření sumací a dalších předpřipravovaných výpočtů viz [12].

Jsou jimi dokumentovány dotazy a výstupy, které byly předdefinovány a již jsou dostupné uživatelům viz [12].

Tato kategorie metadat popisuje i hierarchii jednotlivých dimenzí. Popis hierarchií je důležitý na podporu zjemňování a zobecňování prostřednictvím dimenzí datového skladu viz [12].

Jsou jimi stanovena pravidla určující, kdy je možné data z datového skladu odstranit. Uživatelé potřebují tento typ informace k tomu, aby se dověděli, do kdy budou data v datovém skladu dostupná viz [12].

Metada použitá při návrhu nástroje pro OLAP analýzu provozních dat knihoven MU jsou blíže popsána v kapitole 4.3.

2.7.2 Budování datového skladu

Datový sklad lze budovat jako celek nebo po částech, což znamená, že pro některé důležité organizační složky se vytvoří podmnožiny datového skladu – datové trhy. Takové datové trhy mohou existovat jako subsystémy datového skladu, v tomto případě je nejdříve vytvořen centrální integrovaný datový sklad a z něho se potom vytvoří několik datových trhů, nebo mohou existovat samostatně.

Nejdůležitějším krokem při budování datového skladu je výběr nejvhodnější metody. Nejznámější a nejčastěji používané metody jsou metoda „velkého třesku“ a přírůstková metoda.

1. Metoda „velkého třesku“

Předpokládá budování datového skladu pomocí jediného projektu.

Metoda se skládá ze tří etap:

- **Analýza požadavků**
- **Vytvoření podnikového datového skladu**
- **Vytvoření přístupu buď přímo, nebo přes datové trhy**

Ale vývoj datového skladu je náročná záležitost a pravděpodobně se jí nepodaří vyřešit najednou a to v rozumném čase, což znamená, že se mezitím mohou změnit nejen technologie, ale i požadavky uživatelů, na což se vybudování skladu touto metodou nedokáže adaptovat. A nezbývá než předělat návrh tohoto skladu tak, že se začne od začátku vypracováním nového návrhu. Jedinou výhodou této metody je skutečnost, že můžeme celý projekt kompletně vypracovat ještě před začátkem jeho realizace viz [1].

2. Přírůstková metoda

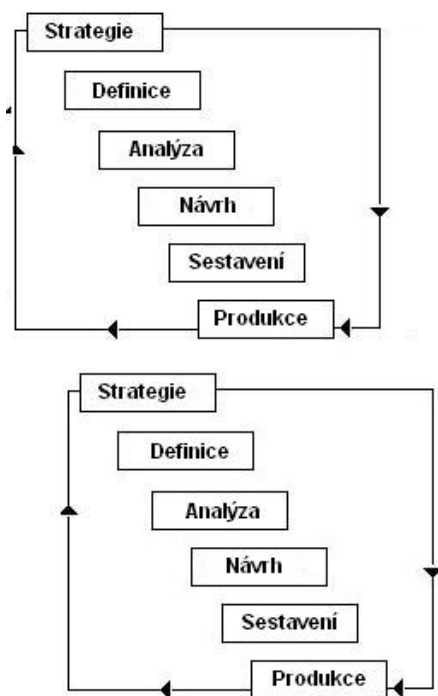
Předpokládá budování datového skladu po jednotlivých etapách, tedy místo vybudování celého datového skladu, postupně přibývají přírůstková řešení, která postupně zapadají do celkové architektury datového skladu. Začne se budováním několika předmětných oblastí. Zmíněné částečné řešení se implementuje jako škálovatelný datový trh a poskytne se uživatelům. Poté, co se částečné řešení osvědčí, je možné do systému přidat další předmětnou oblast, čímž získáme novou funkcionalitu. A tak se stále pokračuje až do úplného vytvoření datového skladu.

Jedná se tedy o iterativní proces, který udržuje neustálou spojitost mezi datovým skladem a uživateli viz [1].

K hlavním výhodám této metody patří:

- Zachovává kontinuitu budovaného projektu s požadavky a potřebami uživatelů viz [1].
- Umožňuje implementovat škálovatelnou, tedy rozšiřitelnou architekturu viz [1].

Schéma přírůstkové metody:



•
•
•

1. **Strategie:** v této fázi se definují cíle. Představují účel řešení datového skladu viz [1].
2. **Definice:** zde se navrhuje architektura datového skladu. Zaměřuje se na pochopení struktury operačních a externích zdrojů údajů viz [1].
3. **Analýza:** cílem je zaměřit se na informace o předmětech zájmu, na získávání dat a požadavků, na přístup k datům pro účely rozhodování. Vytváří se zde relační a multidimenzionální modely pro datový sklad, definují se metadata a požadavky na ně viz [1].
4. **Návrh:** v této fázi se transformují požadavky získané během fáze analýzy do detailních podmínek návrhu a dokončí se instalace technické architektury viz [1].
5. **Sestavení:** zde se sestaví a testují navržené databázové struktury, moduly získávání dat, moduly správy datového skladu, moduly metadat, moduly přístupu k datům a sestavy viz [1].
6. **Produkce:** závěrečná fáze, ve které se nainstaluje datový sklad, začne se používat a zároveň se začne řídit růst a údržba datového skladu viz [1].

Existují následující varianty přírůstkové metody:

- **Přírůstková metoda směrem „shora dolů“**
U této metody se nejdříve na základě požadavků uživatelů vytvoří konceptuální model datového skladu, přičemž se stanoví hierarchie předmětných oblastí. Následně se sestaví konceptuální modely jednotlivých předmětných oblastí.
- **Přírůstková metoda směrem „zdola nahoru“**
Princip budování datového skladu jde naopak od vytvoření předmětných oblastí k vytvoření konceptuálního modelu.

Při návrhu datového skladu pro OLAP analýzu provozních dat knihovního systému MU jsem použila princip přírůstkové metody, konkrétně „shora dolů“. Vymezila jsem několik předmětných oblastí, zhodnotila, které faktory při analýze těchto oblastí sledovat, jak se mohou předmětné oblasti vzájemně prolínat a na základě toho, vytvořila datový model skladu. Bližší informace lze nalézt v popise návrhu v kapitole 3.2.

Tato metoda byla zvolena především z důvodu potřeb dalšího případného rozšíření datového skladu na základě nových požadavků uživatelů systému pro OLAP analýzu provozních dat Aleph–MU.

Aby mohli uživatelé, v tomto případě knihovníci nebo analytici systému Aleph–MU, získat ve stanoveném čase kvalitní podklady pro podporu rozhodování, musí proběhnout příprava údajů pro analýzu, jejich zavedení do datového skladu a následná analýza. Asi nejvíce používaná varianta provozu datového skladu je začít etapu ETL ihned po půlnoci, po jejím dokončení začne přepočít krychlí a ostatních agregační a analytické výpočty. Na tento proces naváže generování sestav a reportů.

Kapitola 3

Návrh OLAP systému v prostředí knihoven MU

Pro analýzu provozních dat v prostředí knihoven byl navržen třívrstvý OLAP systém, jehož první vrstvu tvoří vybrané tabulky databáze knihovního systému Aleph-MU a datový sklad navržený nad těmito tabulkami. Druhou vrstvu tvoří ROLAP server Beard blíže popsany v kapitole 3.1. a funkci třetí vrstvy plní Osito (součást BEE Projectu), vystupující v OLAP systému jako „tenký“ webový client a zároveň webová aplikace, poskytující uživatelsky příjemné prostředí pro OLAP analýzu provozních dat knihovního systému MU.

3.1 Popis systému Aleph-MU

Aleph-MU je systém používaný Masarykovou univerzitou pro podporu knihovnických činností – katalogizaci, akvizici, půjčování, vyhledávání v katalozích apod. Do tohoto centrálního univerzitního systému byly v uplynulých letech postupně převedeny dílčí fakultní katalogy provozované původně v izolovaných systémech TINLIB resp. KP-SYS (na LF). V současnosti jsou v rámci jednotného centrálně provozovaného knihovního systému Aleph-MU spravovány již všechny fakultní ústřední knihovny univerzity viz [4].

Prostřednictvím webového rozhraní systému Aleph-MU na <http://aleph.muni.cz/> jsou uživatelům dostupné základní automatizované služby knihoven – prohledávání katalogů (souborného katalogu celé univerzity i dílčích katalogů fakult), zjišťování stavu výpůjček a finančních transakcí s knihovnou, prodlužování výpůjček, rezervace dokumentů, zasílání informací o novinkách v knihovně apod. Některé funkce systému Aleph-MU jsou dostupné komukoliv, pro jiné se musí uživatel autentizovat – přihlásit se pod svým číslem UČO a zadat své sekundární heslo IS MU (tj. heslo používané typicky pro výběr pošty přes POP3 nebo pro přístup do Celouniverzitní počítačové studovny) viz [4].

Aleph-MU je propojen na další automatizované systémy na MU. Z personalistiky jsou přebírány aktuální informace o uživateli z MU, v informačním systému MU (IS MU) jsou ověřována přístupová (sekundární) hesla a zpětně jsou předávána data pro publikační aplikace. V Aleph-MU byly propojeny bibliografické záznamy kvalifikačních prací na jejich plné texty, uložené v IS MU viz [4].

ALEPH je komerční systém. Sada konfiguračních tabulek umožňuje knihovníkům ovládat a nastavovat v prostředí systému téměř cokoliv. Jedná se o systém se skutečnou vícevrstvou architekturou klient-server. Funkce a služby spojené se správou dat, s knihovnickými aplikacemi a s prezentací dat jsou umístěny do odpovídajících vrstev této architektury viz [5].

Tabulky systému ALEPH-MU jsou spravovány prostřednictvím databázového systému ORACLE, přístup k tabulkám systému je možný prostřednictvím přehledného a uživatelsky příjemného webového rozhraní. Knihovní struktura MU je tvořena devíti ústředními knihovnami fakult. Pro každou fakultu zvlášť existuje soubor tabulek nutných pro provoz knihovního systému. Pro potřeby OLAP analýzy byly jako vhodné tabulky, nad kterými byl vybudován datový sklad, zvoleny tabulky Z00 Documents, Z35 Events, Z36 Loans, Z36H Loans History, Z30 Items, Z30H Items History. Význam a funkce jmenovaných tabulek jsou následující:

1. **Z00 Documents** – tabulka obsahuje bibliografické záznamy, jako jsou například název publikace, vydavatelství, autor či jazyk v jakém je publikace napsaná.
2. **Z35 Events** – zachycuje veškeré informace o událostech, provedených v systému uživatelem systému. Jde o informace typu o jaký typ události se jednalo, kdo provedl příslušnou událost, kdy ji provedl, které fakulty, publikace či kterého čtenáře se událost týkala, z jaké IP adresy byla událost provedena apod.
3. **Z36 Loans** – v této tabulce jsou uloženy veškeré informace týkající se výpůjček provedených v systému. Jde o informace typu kdy byla výpůjčka vytvořena, expirace výpůjčky, do kdy je výpůjčka prodloužena, pokud byla prodloužena, kdy byla výpůjčka vrácena, informace o knihovníkovi, který provedl výpůjčku nebo vyřizoval vrácení publikací na výpůjčce, informace o čtenáři, který se váže k příslušné výpůjčce, informace o vypůjčené publikaci, o fakultě na které byla výpůjčka provedena apod.
4. **Z36H Loans History** – v této tabulce se archivují veškeré informace týkající se výpůjček provedených v systému. Jde o stejné informace, které byly vyjmenované u tabulky Z36 Loans.
5. **Z30 Items** – obsahuje informace o jednotkách, které se nachází v knihovnách MU a je možné si je prostřednictvím systému vypůjčit. Jde o informace typu fakulta, kde je jednotka fyzicky umístěna, typ jednotky podle materiálu, status jednotky, kolekce, do které jednotka patří, datum založení jednotky, datum změny jednotky, rok vydání, jazyk ve kterém je jednotka napsána apod.
6. **Z30 Items History** – v této tabulce se archivují veškeré informace týkající se jednotek, které se nachází v knihovnách MU a je možné si je prostřednictvím systému vypůjčit. Navíc obsahuje informace typu datum poslední změny provedené na jednotce, informace o knihovníkovi, který změnu provedl, datum posledního vypůjčení jednotky.

3.2 Návrh datového skladu systému Aleph-MU

Datový sklad, který byl vytvořen pro OLAP analýzu provozních dat knihovního systému Aleph-MU, byl navržen přírůstkovou metodou „shora dolů“. Byly stanoveny tři předmětné oblasti, které mají statistickou výtežnost, tzn. lze v nich „vydolovat“ důležité informace, jejichž zhodnocením lze zjistit trendy provozu knihovního systému, jeho vytížení, efektivnost využití či řízení a to z různých úhlů pohledu. Postupně byla vytvořena příslušná datová tržiště dle předmětných oblastí.

První předmětnou oblastí se stalo „sledování výpůjček“. Bázi pro příslušné tabulky datového skladu vytvořily tabulky Z36 Loans a Z36H Loans History, které jsou součástí databáze knihovního systému Aleph.

Bylo vytvořeno datové tržiště označené jako „sledování výpůjček“, jehož schéma datového modelu je k vidění v příloze A.

Pro tržiště „sledování výpůjček“ bylo vytvořeno 5 dimenzí (Fakulta, Jednotka, Čtenář, Čas, Knihovník)

1. Dimenzi Fakulta reprezentuje tabulka lu_fakulta.

Sloupec	Typ	Nulový	Výchozí
<u>id</u>	tinyint(4)	Ne	0
nazev	char(30)	Ne	
nazev_zkratka	varchar(5)	Ne	

Je tvořena dvěma hierarchiemi (název fakulty, zkratka fakulty) o jedné úrovni.

2. Dimenzi Jednotka reprezentují tabulky – lu_status_jednotky a lu_material.

Sloupec	Typ	Nulový	Výchozí
<u>id</u>	smallint(4)	Ne	
status	char(2)	Ne	

Sloupec	Typ	Nulový	Výchozí
<u>id</u>	smallint(4)	Ne	
nazev	char(5)	Ne	

Je tvořena dvěma úrovněmi – status jednotky a typ jednotky podle materiálu jednotky.

3. Dimenzi Čtenář reprezentují tabulky – lu_status_ctenare a lu_typ_ctenare.

Sloupec	Typ	Nulový	Výchozí
<u>id</u>	smallint(4)	Ne	
status	char(2)	Ne	

tabulka lu_typ_ctenare

Sloupec	Typ	Nulový	Výchozí
<u>id</u>	tinyint(3)	Ne	
nazev	char(25)	Ne	
nazev_zkratka	varchar(10)	Ne	

Je tvořena dvěma úrovněmi – status čtenáře a typ čtenáře (student, zaměstnanec fakulty, ostatní).

4. Dimenzi Čas reprezentuje tabulka – lu_datum.

Sloupec	Typ	Nulový	Výchozí
<u>id</u>	date	Ne	0000-00-00
typ_dne_id	tinyint(3)	Ne	0
mesic_id	mediumint(6)	Ne	0
semestr_id	mediumint(6)	Ne	0
rok_id	smallint(4)	Ne	0

Je tvořena čtyřmi úrovněmi – rok, měsíc, den, semestr.

5. Dimenzi Knihovnick reprezentuje tabulka – lu_knihovnik.

Sloupec	Typ	Nulový	Výchozí
<u>id</u>	mediumint(20)	Ne	
jmeno	char(50)	Ne	
typ_knihovnika_id	tinyint(3)	Ne	0

Je tvořena dvěma úrovněmi – jméno knihovníka a typ knihovníka.

6. Faktová tabulka pro tržiště „sledování výpůjček“ byla navržena následovně:

Sloupec	Typ	Nulový	Výchozí
<u>id</u>	smallint(5)	Ne	0
status_ctenare_id	tinyint(3)	Ne	0
status_jednotky_id	tinyint(4)	Ne	0
typ_ctenare_id	tinyint(4)	Ne	0
knihovnik_id_pujcujici	mediumint(8)	Ne	0
knihovnik_id_prijimajici	mediumint(8)	Ne	0
fakulta_id	tinyint(4)	Ne	0
material_id	tinyint(4)	Ne	0
datum_id_pujceni	date	Ne	0000-00-00
datum_id_vraceni	date	Ne	0000-00-00
datum_id_expirace	date	Ne	0000-00-00
datum_id_prodlouzeni	date	Ne	0000-00-00
pocet	smallint(5)	Ne	

OLAP analýzou nad navrženým tržištěm lze sledovat počty výpůjček z pohledu fakulty, na které byla výpůjčka provedena, z pohledu statutu či názvu jednotky, která byla vypůjčena, z pohledu čtenáře na kterého byla výpůjčka napsána, z pohledu data půjčení, data vrácení, data prodloužení, data expirace výpůjčky a z pohledu knihovníka, který provedl příslušnou výpůjčku.

Druhou předmětnou oblastí se stalo „vyhodnocení jednotek“. Bázi pro příslušné tabulky datového skladu vytvořily tabulky Z30 Items, Z30H Items History, které jsou součástí databáze knihovního systému Aleph-MU.

Bylo vytvořeno datové tržiště označené jako „vyhodnocení jednotek“, jehož schéma datového modelu je k vidění v příloze B.

Pro tržiště „vyhodnocení jednotek“ bylo vytvořeno 7 dimenzí (Fakulta, Materiál, Status, Kolekce, Jazyk, Knihovník, Čas). Dimenze Fakulta, Materiál, Status jednotky, Knihovník a Čas již byly popsány výše u datového tržiště „sledování výpůjček“ a nijak se od nich neliší.

1. Dimenzi Kolekce reprezentuje tabulka lu_kolekce.

Sloupec	Typ	Nulový	Výchozí
<u>id</u>	smallint(4)	Ne	
nazev	char(5)	Ne	

Je tvořena jednou hierarchií (název kolekce) o jedné úrovni.

2. Dimenzi Jazyk reprezentuje tabulka – lu_jazyk.

Sloupec	Typ	Nulový	Výchozí
id	smallint(4)	Ne	
nazev	char(10)	Ne	
zkratka	char(2)	Ne	

Je tvořena dvěma hierarchiemi (název jazyka a zkratka názvu jazyka) o jedné úrovni.

3. Faktová tabulka pro tržiště „vyhodnocení jednotek“ byla navržena následovně:

Sloupec	Typ	Nulový	Výchozí
id	smallint(4)	Ne	
status_jednotky_id	tinyint(4)	Ne	0
kolekce_id	tinyint(4)	Ne	0
fakulta_id	tinyint(4)	Ne	0
datum_id_zalozeni_jednotky	date	Ne	0000-00-00
datum_id_posledni_zmeny	date	Ne	0000-00-00
datum_id_posledniho_vraceni	date	Ne	0000-00-00
rok_vydani_id	tinyint(4)	Ne	0
knihovnik_zalozil_id	tinyint(4)	Ne	0
knihovnik_provedl_posl_zmeny_id	tinyint(4)	Ne	0
material_id	tinyint(4)	Ne	0
jazyk_id	tinyint(4)	Ne	0
abecedni_kod_id	tinyint(4)	Ne	0
cena	double(10,2)	Ne	0.00
pocet_stran	int(4)	Ne	0
pocet_pujceni	int(11)	Ne	0
pocet_jednotek	int(1)	Ne	1

OLAP analýzou nad navrženým tržištěm lze sledovat počty vypůjčení příslušné jednotky, (celkový, minimální, maximální, průměrný) počet jednotek, počet stran jednotek nebo (celkovou, minimální, maximální, průměrnou) cenu jednotek z pohledu fakulty, na které je jednotka evidována, z pohledu statutu či názvu jednotky, z pohledu kolekce do které byla jednotka zařazena, z pohledu jazyka, ve kterém je jednotka napsána, z pohledu knihovníka, který jednotku zakládal či ji naposledy změnil a z pohledu data založení jednotky, data poslední změny jednotky, roku vydání jednotky.

Třetí předmětnou oblastí se stalo „sledování událostí v systému“. Bází pro příslušné tabulky datového skladu vytvořila tabulka Z35 Events, která je součástí databáze knihovního systému Aleph-MU.

Bylo vytvořeno datové tržiště označené jako „sledování událostí v systému“, jehož schéma datového modelu je k vidění v příloze C.

3.2. NÁVRH DATOVÉHO SKLADU SYSTÉMU ALEPH-MU

Pro tržiště „sledování událostí v systému“ bylo vytvořeno 7 dimenzí (Fakulta, Jednotka, Čtenář, Knihovník, Typ události, IP adresa, Čas). Dimenze Fakulta, Jednotka, Čtenář, Knihovník a Čas již byly popsány výše u datového tržiště „sledování výpůjček“ a nijak se od nich neliší.

1. Dimenzi IP adresa reprezentuje tabulka lu_ip_adresa.

Sloupec	Typ	Nulový	Výchozí
id	int(11)	Ne	
cislo	varchar(20)	Ne	
poskytovatel	varchar(20)	Ne	

Je tvořena dvěma hierarchiemi (číslo IP adresy, poskytovatel) o jedné úrovni.

2. Dimenzi Typ události reprezentuje tabulka – lu_typ_udalosti.

Sloupec	Typ	Nulový	Výchozí
id	smallint(4)	Ne	
nazev	varchar(20)	Ne	
cislo	tinyint(4)	Ne	0

Je tvořena dvěma hierarchiemi (název události a číslo události) o jedné úrovni.

3. Faktová tabulka pro tržiště „sledování událostí v systému“ byla navržena následovně:

Sloupec	Typ	Nulový	Výchozí
id	smallint(4)	Ne	
ip_adresa_id	tinyint(4)	Ne	0
knihovnik_id	tinyint(4)	Ne	0
fakulta_id	tinyint(4)	Ne	0
datum_id_udalosti	date	Ne	0000-00-00
status_ctenare_id	tinyint(4)	Ne	0
typ_ctenare_id	tinyint(4)	Ne	0
status_jednotky_id	tinyint(4)	Ne	0
typ_udalosti_id	tinyint(4)	Ne	0
pocet	int(4)	Ne	

OLAP analýzou nad navrženým tržištěm lze sledovat počty událostí provedených v systému (rezervace jednotky, půjčení jednotky, prodloužení jednotky, update jednotky apod.) z pohledu fakulty, které se týká příslušná událost, z pohledu knihovníka či čtenáře, který provedl příslušnou událost, z pohledu typu události, která byla provedena, z pohledu IP adresy, ze které byla událost provedena a z pohledu jednotky, které se událost týkala.

3.3 Výběr nástroje pro OLAP analýzu provozních dat Aleph-MU

Lze říci, že pro implementaci OLAP analýzy na bázi "open source" software jsou již ve vy-zrálé podobě k dispozici všechny technologické komponenty se zřejmými výhodami "open source" konceptu, jak pro kvalitu software a jeho bezpečnost, tak pro ekonomiku řešení a ochranu investice zákazníka (otevřený systém snižuje jeho závislost na jednom dodavateli, zákazník má plná práva k disponování zdrojovými kódy systému) viz [1].

Pro OLAP analýzu provozních dat systému knihoven MU lze například použít sadu nástrojů JasperSoft Business Intelligence Suite nebo sadu nástrojů BEE Project Insight Strategy.

3.3.1 JasperSoft Business Intelligence Suite

Sada nástrojů pomocí nichž lze vytvořit úplný OLAP systém, tedy vícevrstvou aplikaci slo-ženou z vrstvy plnící funkci ETL, vrstvy pro samotnou OLAP analýzu a vrstvy pro pre-zentaci výsledků uživateli v přehledné a srozumitelné formě. Funkce jednotlivých vrstev plní moduly JasperETL, JasperAnalysis, iReport a řídicí funkci, neboli „mozek“ systému plní model JasperServer. Jednotlivé nástroje jsou vyvíjeny v jazyce JAVA s podporou XML, nejsou tak závislé na použitém operačním systému ani případné aplikaci na kterou navazují, jejíž nastavbu tvoří.

1. JasperETL

Disponuje aparátem pro předem naplánovanou extrakci, transformaci a přenos dat z operačního prostředí do hvězdicového schématu datového skladu, kde jsou data připravena pro bezpečnou a okamžitou analýzu a reportování. Vytváří úplné, tzv. „ready-to-go“ grafické prostředí pro sjednocení datového formátu údajů, pro od-stranění duplikací v datech, čištění dat, zajištění kvality dat a pro přenos dat. Díky uživatelsky příjemnému grafickému prostředí lze poměrně jednoduše definovat a také sledovat, krokovat, odladit etapu ETL, bez znalosti skriptovacích jazyků. Do-káže zpracovávat data z různých datových zdrojů jako jsou například flat soubory, xml soubory, všechny druhy databází, POP a FTP servery a další viz [8].

2. JasperAnalysis

Velice výkonná webová aplikace poskytující aparát pro všechny operace OLAP ana-lýzy (Drill-down, Roll-up, Pivoting, Slicing, Diviny). OLAP krychle je možné zpra-covávat na bázi jazyka MDX. Metadata OLAP analýzy jsou popsána v jazyce XML. Podporuje celou řadu komerčních RDBMS systémů, aplikačních serverů a operač-ních systémů. Krom základních operací poskytuje i další užitečné analytické ope-race jako třídění, výpis prázdných řádků či sloupců, přehození os, výpis diagramů, výpis zdrojových dat, či dat vzniklých drillováním, změnu OLAP kostek, filtrování, výpis dotazů jazyka MDX, exportu výsledků analýzy do tabulek Excel nebo jako se-stav v pdf formátu a operace pro administraci ROLAP serveru, konfiguraci přístupů uživatelů a uživatelských rolí a konfiguraci OLAP cache viz [9].

3.3. VÝBĚR NÁSTROJE PRO OLAP ANALÝZU PROVOZNÍCH DAT ALEPH-MU

3. iReport

Nástroj pro interpretaci výsledků analýzy ve formě grafů všech možných typů, kontingenčních tabulek, různých přehledových sestav a hierarchických struktur. Umožňuje uživateli vytvořit vzhled stránky, včetně nadpisů, čísla stránek, pozadí na které se zobrazí v nějaké formě výstup OLAP analýzy, taková grafická podoba reportu může být dále exportována do formátu pdf nebo html stránky viz [10].

3.3.2 BEE Project Insight Strategy

V rámci projektu je řešena metodologie optimálního uložení dat s ohledem na jejich analytickou výtečnost, je vyvíjena infrastruktura pro ETL (extrakce a transformace dat, plnění datového skladu) a vícevrstvá aplikace pro analytické reportování (využívající „tenkého“ klienta a WEB technologie). Architektura řešení je založena na metodologii ROLAP s cílem ekonomicky efektivně pokrýt projekty o objemu dat do 50 GB s využitím "open source" technologií na platformě Linux/Intel. Produkty vytvořené v rámci projektu BEE jsou publikovány paralelně pod GPL licencí a komerční licencí společnosti Insight Strategy.

V rámci projektu BEE je vyvíjena vícevrstvá aplikace pro analytické reportování. Přístup prostřednictvím „tenkého“ klienta (*WEB browser*) je vhodný zejména pro tzv. "business" uživatele, kterým lze připravit potřebné reporty a zajistit k nim snadný přístup (včetně například vzdáleného přístupu pro mobilní uživatele). Jako primární uživatelské rozhraní je navržen WEB portál umožňující definování analytických reportů, jejich formátování a prezentaci viz [18].

Využití statistického software projektu "R" umožňuje analytikům použít širokou škálu pokročilých statistických metod a metod pro dolování dat (*data mining*). Součástí projektu "R" jsou vyvíjeny na nejvýznamnějších světových akademických pracovištích zabývajících se uvedenou problematikou a jsou publikovány pod GPL licencí. Grafické knihovny projektu "R" jsou také využity pro generování kvalitní tzv. "statistické" i "obchodní" grafiky.

Použité technologie (XML, HTML, CSS a JavaScript) mají minimální nároky na síťovou infrastrukturu a zaručují snadnou administraci celého prostředí. Celý systém je důsledně internacionalizován (interně využívá kódování UTF-8) a umožňuje každému uživateli nezávislou volbu lokalizace jak uživatelského rozhraní, tak i pojmenování objektů a vlastních dat. ROLAP server zpřístupňuje své služby přes standardní aplikační rozhraní SOAP pro potenciálně širokou škálu dalších klientských aplikací viz [18].

Užitečnou pomocnou metodou pro analýzu vícerozměrných dat je interaktivní zobrazení dat v třírozměrném prostoru (tyto metody bývají zahrnovány do oblasti vizuálního data miningu). Možnost interaktivní manipulace s 3D prezentací usnadňuje uživateli pochopení struktury dat, popřípadě trendů jejich vývoje. Pro popis 3D scény je využito jazyka VRML, který je vhodný pro distribuci zobrazení v síťovém prostředí. Jazyk VRML jako uznávaný standard organizace W3C zaručuje možnost zobrazení scény na nejrůznějších platformách prostřednictvím různých typů VRML prohlížečů. Scénu je možno také prohlížet přímo z prostředí WEB prohlížeče doplněného o vhodný VRML plug-in modul viz [18].

3.3. VÝBĚR NÁSTROJE PRO OLAP ANALÝZU PROVOZNÍCH DAT ALEPH-MU

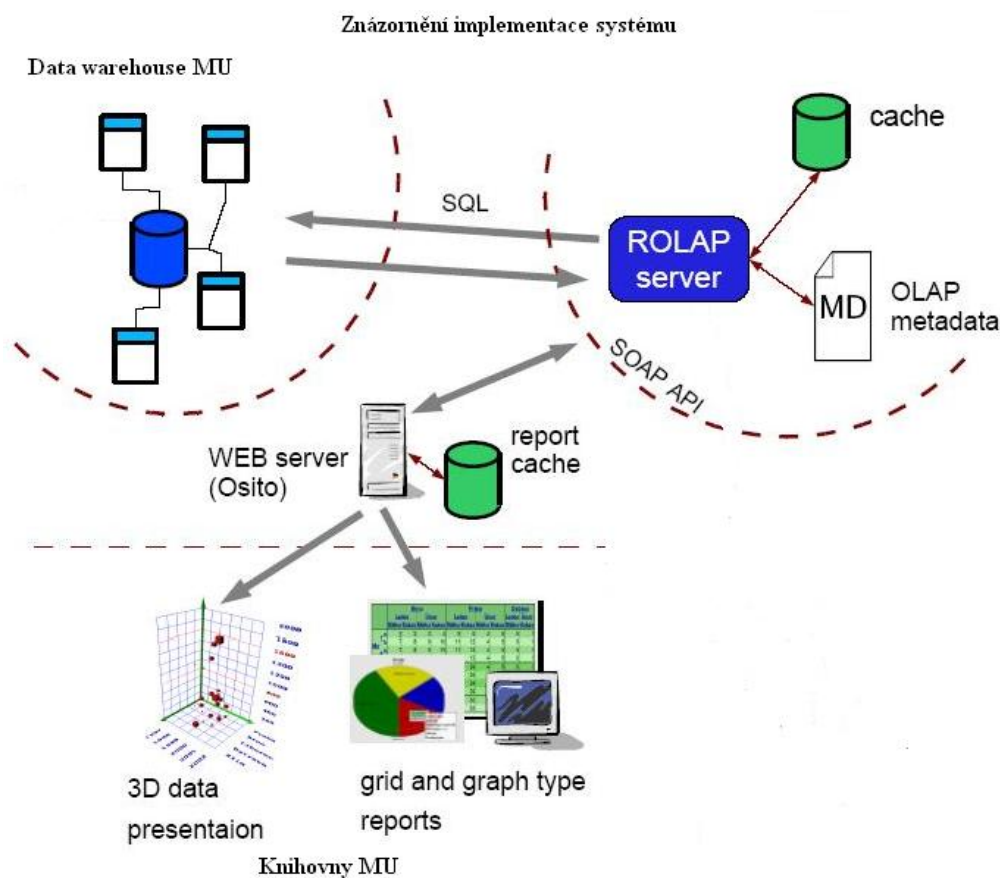
Technologie, zvolená pro projekt BEE během důkladného testování typovými úlohami, prokázala dostatečný výkon a spolehlivost pro nasazení v prostředí středně velkých organizací, při zachování dobrých ekonomických parametrů řešení. Výhodou je možnost v maximální možné míře využít stávající technické infrastruktury a díky důsledné optimalizaci RDBMS minimalizovat náklady na hardware datového skladu.

Použitá platforma díky své modulárnosti a škálovatelnosti umožňuje efektivní rozvržení projektu do jednotlivých etap a tak optimálně rozložit a zhodnotit nutné investice i zapojení vnitřních zdrojů odběratele. Otevřenost systému a využití standardních technologií (XML, SOAP API) zaručuje jeho snadnou integraci s technologiemi třetích stran viz [18].

Pro analýzu provozních dat knihovního systému MU jsem zvolila sadu nástrojů projektu BEE, který jsem upravila tak, aby se dal použít pro OLAP analýzu dat v prostředí knihoven MU. Oba nástroje Jasper i BEE, velmi dobře pokrývají potřeby OLAP analýzy dat a dají se použít. BEE Project byl upřednostněn, protože je při jeho použití celková režie menší (nepoužívá grafická rozhraní pro konfiguraci, spuštění ROLAP serveru a ETL procesů), jeho vlastní provoz podstatně méně snižuje výkon celého systému, vývoj tohoto nástroje byl delší a narozdíl od Jasper je ceněn jako "Profesionální software" a stal se již součástí několika úspěšných projektů v oblasti business intelligence. Jasper byl dokončen a je dispozici v open source podobě k testování použití teprve od listopadu letošního roku, což do jisté míry zpochybňuje jeho úplnou spolehlivost.

Kapitola 4

Implementace OLAP systému v prostředí knihoven MU



Implementovat OLAP systém v prostředí knihoven znamenalo zrealizovat v tomto prostředí ETL fázi systému, upravit použitý OLAP analyzátor tak, aby byl schopný analyzovat data nově vytvořeného datového skladu knihovního systému Aleph-MU a nakonfigurovat použitý ROLAP server a webového klienta tak, aby byla analýza dat možná. Samotná implementace neprobíhala přímo ve skutečném prostředí knihovního systému. Byly vyexportovány tabulky databáze systému Aleph-MU, jejichž data se bude plnit datový sklad, do podoby flat souborů. Celý systém jsem implementovala a ladila na lokálním webovém serveru ozn. „localhost“ na vlastním počítači. Nicméně dle návodu v kapitole 4.5, lze tento systém implementovat ve skutečném prostředí knihoven MU pouze drobnými změnami

v konfiguračních souborech ROLAP serveru *Beard*), webového serveru (*Osito*) a ETL fáze v sekci připojení. Blíže jsou změny popsány v kapitole 4.5. OLAP analyzátor pracuje a řídí se metadaty, uloženými v adresáři `/opt/beard/mdata`, konkrétně jde o soubor `DWHMU.xml`). Tato metadata popisují strukturu datového skladu (dimenze, fakta) tak, aby jim OLAP analyzátor rozuměl.

Příprava dat, pro OLAP analýzu navrženého systému probíhá tak, že na základě metadata se naplní cache paměť, která odráží strukturu datového skladu a nad ní ve skutečnosti pracuje příslušný OLAP analyzátor. Data se před nahráním do cache paměti čistí a přidělují se v případě potřeby nové cizí a primární klíče. Každá změna cache paměti je logována a logy (záznamy o změnách) se též ukládají do datového skladu. Tímto způsobem jsou tedy dokumentovány transformace a zavedení dat do datového skladu, které tvoří fázi ETL.

4.1 Popis použitého ROLAP serveru Beard

Poté co jsou data zavedena do datového skladu, nastupuje nástroj pro OLAP analýzu, který dokáže vytěžit z těchto dat informace, které koncového uživatele zajímají. ROLAP server Beard (součást projektu BEE) se stal řídicí jednotkou OLAP analyzátoru, použitého pro analýzu dat v prostředí knihoven MU. Plní funkci tzv. analytické vrstvy OLAP. Jeho úkolem je zajistit sestavení dat pro prezentační vrstvu OLAP systému. Tento server zpřístupňuje své služby přes standardní aplikační rozhraní SOAP pro potenciálně širokou škálu klientských aplikací. Na základě požadavku klientské aplikace, který je definován pomocí analytických objektů (jako jsou např. atributy, metriky, filtry), sestaví dotaz v jazyce SQL (dle potřeby i víceprůchodový), pomocí kterého jsou získána data ze SŘBD datového skladu. Na úrovni ROLAP server jsou data uspořádána do požadované podoby a atributy jsou opatřeny zvolenou formou zobrazení a lokalizace. Pro zajištění vyšší propustnosti systému jako celku je využito několika úrovní vyrovnávacích pamětí viz [11].

Podporuje víceuživatelské přístupy a běh více odlišných projektů, provádějících OLAP analýzu nad různými datovými sklady u různých klientů.

Definice uživatelských účtů, jejich práv a propojení s příslušným projektem jsou snadno pochopitelné a také rozšířitelné. Jsou obsaženy v souborech `Acl.xml` a `root.xml`, které lze nalézt v adresářích

`/opt/beard/mdata/nazev_projektu` a `/opt/beard/mdata/nazev_projektu/Roles`.

Například definice uživatelských účtů (`DWHMU_User1` a `DWHMU_User2`), přidělení práv a jejich přiřazení k příslušnému projektu (`DWHMU`) vypadá následovně:

1. Definice v souboru `Acl.xml`

```
<?xml version='1.0' standalone="yes"?>
  <project>
    <pacl>
      <deny all="RWED"/>
      <allow all="----">
    <puser>
```

```
<DWHMU User_1 ac="R---"/>
<DWHMU User_2 ac="RW--"/>
</puser>
<pgroup>
  <DWHMU ac="RWED"/>
</pgroup>
</allow>
</pacl>
</project>
```

puser – zadefinuje příslušného uživatele

ac – definuje práva (R – čtení, W – zápis)

pgroup – definuje projekt a jeho práva přístupu

2. Definice v souboru root.xml

```
<?xml version='1.0' encoding="UTF-8" standalone="yes"?>
<project>
  <urole id="DWHMU_root" acl="DWHMU_all"
    change="on" sql="edit" tabs="">
    <description cz="Správce systému"
      en="System administrator"/>
    <name cz="Root" en="Root">
    <rights allow="RWEDC" deny=""/>
    <uuser>DWHMU User_1</uuser>
    <uuser>DWHMU User_2</uuser>
    <meta></meta>
  </urole>
</project>
```

urole – zadefinuje příslušný projekt

uuser – zadefinuje uživatele projektu

4.2 Popis použitého nástroje Osito

Je také součástí BEE projektu a byl použit, aby plnil funkci prezentační vrstvy OLAP systému navrženého pro OLAP analýzu provozních dat knihovního systému Aleph-MU. Tedy poskytuje uživatelsky příjemné webové rozhraní pro přístup k výsledkům OLAP analýzy nebo pro provádění OLAP analýzy v prostředí knihoven MU. Plní zároveň funkci „tenkého“ klienta.

Webový portál je navržen jako primární uživatelské rozhraní pro tvorbu, údržbu a přístup k reportům.

Byl vyvíjen a optimalizován dle standardů organizace W3C a funkčnost byla ověřena na web prohlížečích typu MS, IE a Mozilla. Tím je také zajištěn kvalitní přístup k portálu

v rámci heterogenních prostředí. Aplikace nevyžaduje na straně „tenkého“ klienta instalaci žádných dodatečných komponent a díky minimálním nárokům na přenos dat se dobře hodí i k vzdálenému přístupu mobilních uživatelů (dial-up, Wi-fi, GPRS) viz [11].

Reporty jsou zobrazovány ve formě tabulek s možností definování stylů zobrazení a možností zobrazení kontextových menu v reportu (pro drillování, pilotování a snadnou integraci s dalšími web aplikacemi.) Reporty je možné zobrazovat i ve formě grafů, včetně možností generování kontextových menu vázaných na grafické objekty (pro reporty typu bar, point, line a pie) viz [11].

Pro analýzu vícerozměrných dat poskytuje aplikace interaktivní zobrazení dat v třírozměrném prostoru. Pro popis 3D scény je využito jazyka VRML, který je vhodný pro distribuci zobrazení v síťovém prostředí. Jazyk VRML je uznávaný standard organizace W3C zaručuje možnost zobrazení scény na nejrozličnějších platformách prostřednictvím různých typů VRML prohlížečů viz [11].

Aplikace využívá statistického software projektu „R“, což umožňuje analytikům použít širokou škálu pokročilých statistických metod a metod pro dolování dat (*data mining*) viz [11].

4.3 Popis DWH metadat OLAP systému

Soubor metadat se nachází v příloze D. Jedná se o XML soubor. Pomocí XML lze v dokumentu vyznačit význam jednotlivých částí a vlastností příslušného objektu, který je předmětem našeho zájmu, v případě OLAP systému je tímto objektem strom reprezentující hierarchii a strukturu datového skladu MU. Tento strom tvoří tabulky dimenzí a faktové tabulky.

Význam metadat budu dále prezentovat na části již zmíněného datového tržiště „sledování výpůjček“. Jak už bylo řečeno jednou z dimenzí výpůjček je Čas, ten vystupuje v tržišti v několika svých rolích a dále se dělí do čtyř úrovní. Nadefinování dimenze příslušnými metadaty probíhá v následujících krocích:

1. Vytvoří se časová dimenze (group), přidělí se přístupová práva k této dimenzi (acl), její příslušné role (role) a zvolí se její id:

```
<group id="9990">
  <name en="Time" cz="Čas" />
  <acl id="allow" DWHMU_all="R-E--"/>
  <acl id="deny" DWHMU_all="-W-DC"/>
  <description en="Time Dimension" cz="Časová dimenze" />
  <role id="pujzeni">
    <name en="Loan of item" cz="Půjčení jednotky" />
  </role>
  <role id="vraceni">
    <name en="Return of item" cz="Vrácení jednotky" />
  </role>
  <role id="expirace">
    <name en="Due of loan" cz="Expirace výpůjčky" />
  </role>
</group>
```

```

</role>
<role id="prodlouzeni">
  <name en="Renewal of loan" cz="Prodloužení výpůjčky" />
</role>
</group>

```

2. Dimenzi a rolím dimenze se přiřadí příslušné tabulky datového skladu, primární (pk) a cizí klíče (fk) a hierarchie (display) a úrovně (attr), které ji vytváří a přiřadí se opět přístupová práva (acl). Pomocí (def) se definují možné modifikace výpisu údajů v grafickém prostředí analyzátoru. Vysvětlující titulek, který se objeví při přejetí položky myši, se nadefinuje pomocí (description). Přiřazení je realizováno pomocí group = „9990“, kde 9990 odpovídá id již nadefinované group časové dimenze.

```

<attr group="9990" id="datum">
  <pk>demohost/demo_bee/lu_datum/id</pk>
  <name en="Day" cz="Datum" />
  <acl id="allow" DWHMU_all="R-E--"/>
  <acl id="deny" DWHMU_all="-W-DC"/>
  <description></description>
  <display id="lname">
    <def en="demohost/demo_bee/lu_datum/DATE_FORMAT(id,'%M %D, %Y') "
      cz="demohost/demo_bee/lu_datum/CASE MONTH(id)
        WHEN 1 THEN DATE_FORMAT(id,'%e. leden %Y')
        WHEN 2 THEN DATE_FORMAT(id,'%e. únor %Y')
        WHEN 3 THEN DATE_FORMAT(id,'%e. březen %Y')
        WHEN 4 THEN DATE_FORMAT(id,'%e. duben %Y')
        WHEN 5 THEN DATE_FORMAT(id,'%e. květen %Y')
        WHEN 6 THEN DATE_FORMAT(id,'%e. červen %Y')
        WHEN 7 THEN DATE_FORMAT(id,'%e. červenec %Y')
        WHEN 8 THEN DATE_FORMAT(id,'%e. srpen %Y')
        WHEN 9 THEN DATE_FORMAT(id,'%e. září %Y')
        WHEN 10 THEN DATE_FORMAT(id,'%e. říjen %Y')
        WHEN 11 THEN DATE_FORMAT(id,'%e. listopad %Y')
        WHEN 12 THEN DATE_FORMAT(id,'%e. prosinec %Y') END" />
    <name en="Date Long" cz="Datum dl." />
    <acl id="allow" DWHMU_all="R-E--"/>
    <acl id="deny" DWHMU_all="-W-DC"/>
    <description en="Long description" cz="Dlouhý popis" />
  </display>
  <display id="name">
    <def en="demohost/demo_bee/lu_datum/DATE_FORMAT(id,'%m/%d/%Y') "
      cz="demohost/demo_bee/lu_datum/DATE_FORMAT(id,'%d.%m.%Y') " />
    <name en="Date Short" cz="Datum kr." />
    <acl id="allow" DWHMU_all="R-E--"/>
    <acl id="deny" DWHMU_all="-W-DC"/>
    <description en="Short description" cz="Kratký popis" />
  </display>
  <fk>demohost/demo_bee/r_datum_typ_dne/datum_id</fk>

```

```

<role id="pujzeni">
  <fk>demohost/demo_bee/lu_vypujcka/datum_id_pujzeni</fk>
</role>
<role id="vraceni">
  <fk>demohost/demo_bee/lu_vypujcka/datum_id_vraceni</fk>
</role>
<role id="expirace">
  <fk>demohost/demo_bee/lu_vypujcka/datum_id_expirace</fk>
</role>
<role id="prodlouzeni">
  <fk>demohost/demo_bee/lu_vypujcka/datum_id_prodlouzeni</fk>
</role>
</attr>

```

V případě, že datový sklad obsahuje tabulky předpřipravených agregací, je nutné je též nadefinovat příslušnými metadaty. Například u časové dimenze a agregace semestr v roce bude popis agregace metadaty vypadat tak, že se nadefinuje vazba 1:N (from cardinality : to cardinality), kde názvy (semestr, rok) odpovídají příslušným id již nadefinovaných atributů (attr) typu semestr a rok.

```

<hrelation name="Rok Semestr" id="rok_semestr" description="">
  <to cardinality="N" optional="false">semestr</to>
  <from cardinality="1" optional="false">rok</from>
</hrelation>

```

Faktové tabulky se nadefinují pomocí (fact), veličina kterou sledujeme se definuje pomocí (expr), práva přístupu k faktům se nadefinují pomocí (acl). Pro fakt počet výpůjček by definice vypadala následovně:

```

<fact id="pocet_vypujcek">
  <expr>demohost/demo_bee/lu_vypujcka/pocet</expr>
  <expr func="SUM">
    demohost/demo_bee/agr_semestr_fakulta/pocet_vypujcek
  </expr>
  <name en="No. of loans" cz="Počet výpůjček" />
  <acl id="allow" DWHMU_all="R-E--"/>
  <acl id="deny" DWHMU_all="-W-DC"/>
  <description></description>
</fact>

```

4.4 Popis ETL skriptů

Jak již bylo řečeno, vzhledem k tomu, že data jsou extrahována z homogenního systému Aleph-MU, kde jsou uložena do tabulek jedné struktury pro stejná data a celá databáze je spravována jedním databázovým serverem ORACLE a to vše v jednotném prostředí operačního systému UNIX, fázi extrakce představuje vlastně v jistém smyslu model vzdáleného vynášení.

Tabulky databáze Aleph, spravované databázovým serverem ORACLE, které budou použity pro naplnění datového skladu, je nutné exportovat do adresáře /opt/ETL/data_aleph_mu) v podobě flat souborů.

V projektu byl vytvořen spustitelný soubor ETL.sh, uložený v adresáři /opt/ETL, který ETL fázi provádí v následujících krocích:

1. Exportuje flat soubory uložené v adresáři /opt/ETL/data_aleph_mu do odpovídajících tabulek datového skladu spravovaného databázovým serverem MySQL. Tyto tabulky tvoří tzv. „staging area“, bázi datového skladu.
2. Z takto načtených tabulek vybere a vytvoří příslušným SQL dotazem postupně data pro tabulky (např. lu_ip_adresa, lu_status_ctenare, lu_status_jednotky apod.). Tabulky se nachází na nejnižší úrovni hierarchie každé dimenze a obsahují pouze atomické hodnoty, dále nedělitelné. Příslušná data exportuje do flat souborů, jejichž jméno odpovídá jménu příslušné tabulky skladu. Z flat souborů jsou potom nahrány do odpovídajících tabulek datového skladu.
3. Stejným způsobem jako v kroku 2 se vytvoří flat soubory tabulek na o jedno vyšším stupni v hierarchii tabulek dimenzí. Tento soubor se tvoří exportem dat odpovídajícím příslušnému SQL dotazu, který vznikne spojením tabulek tvořících bázi datového skladu a tabulek na nejnižší úrovni v hierarchii tabulek tvořících příslušné dimenze. Z flat souborů jsou potom údaje nahrány do odpovídajících tabulek datového skladu.
4. Pokračuje se v podobném duchu tímto iteračním procesem plnění tabulek skladu po úrovních, až jsou data naplněny všechny tabulky, na všech úrovních hierarchie tabulek tvořících příslušné dimenze dat. skladu.
5. Spuštěním perlovského skriptu create_date_dimension.pl následuje vytvoření časové dimenze. Pro časovou dimenzi již existují tabulky dnů, měsíců a semestrů. Protože jsou neměnné, načtou se rovnou při prvotním vytvoření tabulek datového skladu, které proběhne pouze jednou, na začátku celého procesu implementace datového skladu v prostředí knihoven MU. Perlovským skriptem se „vytáhnou“ z databázových tabulek všechny možné, navzájem různé hodnoty roků a uloží se do tabulky lu_rok. Naplní se jimi pole roky_poradi[], jehož asociativní klíče nabývají hodnot jednotlivých roků a v polích jsou uložena pořadí roků (počítají se od 0) tak, jak následují po seřídění vzestupně. Tabulky lu_mesic_v_roce a lu_semestr_v_roce mají svoje jednoznačné identifikátory (id) odvozena pomocí vzorce $(24000 + \text{id měsíce} + \text{poradí roku} \cdot \text{celkový počet měsíců})$ a $(8000 + \text{id semestru} + \text{poradí roku} \cdot \text{celkový počet semestrů})$. Jednotlivým dnům byla jako jednoznačné identifikátory (id) přidělena pořadová čísla dnů v týdnu. (id pondělí = 1). Jednotlivým měsícům byla jako jednoznačné identifikátory (id) přidělena pořadová čísla měsíců v roce. (id leden = 1). Tabulka lu_datum se naplní všemi možnými daty, které se vyskytují v příslušném roce a to u všech roků z tabulky lu_rok. Podobně se tabulka lu_mesic a lu_semestr

naplní roky v tomto rozmezí a všemi možnými variantami kombinace roku a měsíce či roku a semestru (např. leden 2000, leden 2001 nebo jaro 2000, jaro 2001 apod.). Jednoznačné identifikátory roků odpovídají samotnému číslu roku. Hodnoty cizích klíčů `mesice_v_roce_id`, `semestr_v_roce_id` a `semestr_id`, `mesic_id` se poté doplní výpočtem těchto klíčů dle předešlého popisu.

Exporty do flat souborů byly použity proto, že v případě manipulace s velkým množstvím dat jsou ceněny jako nejrychlejší řešení. Fáze transformace dat různých formátů pro stejný atribut do jednotného formátu nebyla potřebná, vzhledem k tomu, že data jsou načítána z databáze Aleph, která má pro všechny atributy, které si v principu odpovídají, stejný způsob zápisu a ten je použit i v případě tabulek datového skladu. Fáze čištění byla vyřešena tak, že všechny atributy tabulky datového skladu jsou nadefinovány jako nenulové a s výchozí hodnotou. Při pokusu o vložení hodnoty NULL či prázdné hodnoty se vloží a použije výchozí hodnota. Výchozí hodnoty jsou zvoleny tak, aby neměly vliv na výsledky OLAP analýzy (u čísel hodnota 0, u řetězců prázdná hodnota, u datumu 0000-00-00 apod.). Nakládání s hodnotami stejnými jako výchozí je ošetřeno v metadatech – v souboru DWHMU.xml a to pomocí atributu `def`.

Importy a exporty dat jsou provedeny pomocí SQL příkazů `LOAD DATA INFILE` a `LOAD DATA OUTFILE` viz [16].

4.5 Instalace OLAP analyzátoru a spuštění systému

Instalace nástroje se v jednotlivých unixových prostředích může lišit malou změnou v názvech balíčků či způsobu jejich instalace. Uvádím způsob jakým se analyzátor instaluje v unixové distribuci Ubuntu.

1. Instalace prostředí pro běh analyzátoru BEE (<http> server a databázový server)

Nutné stáhnout a nainstalovat balíčky: `apache2`, `mysql-client-5.0`, `mysql-common`, `mysql-server-5.0`, (`phpmyadmin` – grafické prostředí pro správu databází). Doporučuje se vytvořit nový uživatelský účet např. `bear` a jemu předat vlastnictví ke všem souborům, nutných pro běh analyzátoru a analyzátor spouštět pod tímto nově vytvořeným účtem a ne pod uživatelským účtem `root`.

2. Instalace podpůrných prostředků pro ETL fázi a CPAN (*Comprehensive Perl Archive Network*)

Nutné stáhnout a nainstalovat balíčky: `openssl`, `libssl`, `libssl-dev`, `libexpat1-dev`, `perl-base`, `perl-modules`, `libperl-dev`, `lynx`, `ncftp2`, `libmysqlclient15-dev`, `libgnademySQL-dev`, `libdbd-anydata-perl`, `libdbd-csv-perl`, `libdbd-excel-perl`, `libdbd-odbc-perl`, `libdbd-sqlrelay-perl`, `libtk-objscanner-perl`, `perl-tk`, `libdbd-mysql`

Dále se konfiguruje CPAN příkazem `perl -MCPAN -e shell`.

Spočívá v nastavení několika parametrů, odpovědí na otázky, které klade instalační prostředí. Odpovědi se zapisují do [] a potvrdí se tlačítkem `enter`.

4.5. INSTALACE OLAP ANALYZÁTORU A SPUŠTĚNÍ SYSTÉMU

Po konfiguraci CPAN je nutné nainstalovat následující balíčky: XML::Parser, XML::Simple, FreezeThaw, DBI, DBD::mysql, DBD::Ingres, DBD::ODBC, Tk, Tk::FileDialog, DBD::XMLViewer.

Balíčky se nainstalují následovně: nejdříve se zadá v terminálové konzole příkaz `cpan`, čímž se přepneme do prostředí CPAN a poté `install název_modulu`.

3. Instalace Common části analyzátoru BEE

Je nutné vytvořit adresář `bee` v adresářové struktuře `/usr/local/share/` a přidělit jeho vlastnictví účtu `bear`.

Z přiloženého CD z adresáře `download` je nutné zkopírovat obsah složky `download/beeCommon` do `/usr/local/share/bee` a přidělit jeho vlastnictví účtu `bear`. U souboru `/usr/local/share/bee/util/checkenv.pl` a `/usr/local/share/bee/test.sh` je nutné nastavit právo spuštění pro uživatele `bear`.

Po spuštění `checkenv.pl` (zadá se `cd /usr/local/share/bee` a potom `./util/checkenv.pl`) se objeví tabulka, která vypíše moduly prostředí, které je nutné nainstalovat pro běh části `Common`. Moduly u kterých se objeví `miss`, je nutné nainstalovat ručně. Instalace se provede následovně: nejdříve se zadá v terminálové konzole příkaz `cpan`, čímž se přepneme do prostředí CPAN a poté `install název_modulu`. (Například `install Heap::Elem`). Poté, co se již v tabulce nevyskytuje `miss` u žádného modulu, je nutné spustit `test.sh` a zjistit zda je vše v pořádku (All test are succesfull).

4. Instalace Beard části analyzátoru BEE (ROLAP server analyzátoru)

Je nutné vytvořit adresář `beard` v adresářové struktuře `/opt/` a přidělit jeho vlastnictví účtu `bear`.

Z přiloženého CD z adresáře `download` je nutné zkopírovat do `/opt/beard/` obsah složky `download/beeBeard` a přidělit jeho vlastnictví účtu `bear`. Zkontroluje se jestli se v `beard` adresáři nachází adresář `metadata` a `mdata/Prefer`, pokud ne, vytvoří se a přidělí se jim právo spuštění pro uživatele `bear`.

Po spuštění `checkenv.pl` (zadá se `cd /opt/beard`

a potom `/usr/local/share/bee/util/checkenv.pl`) se objeví tabulka, která vypíše moduly prostředí, které je nutné nainstalovat pro běh části `Beard`. Moduly u kterých se objeví `miss`, je nutné nainstalovat ručně. Instalace se provede následovně: nejdříve se zadá v terminálové konzole příkaz `cpan`, čímž se přepneme do prostředí CPAN a poté `install název-modulu`. (Například `install Algorithm::Permute`). Poté, co se již v tabulce nevyskytuje `miss` u žádného modulu, spustí se soubor `create_databases.sh`, který je umístěn v adresáři `/opt/beard` čímž se vytvoří databázový účet `bear` s heslem `honey` a příslušné databáze datového skladu knihoven `MU`, jde o databáze `cache`, `share` a `DWHMU`. Konfigurační soubor `ROLAP server Beard` je uložen v adresářové struktuře `opt/beard/const/Const.pl`. Je nakonfigurován jako `localhost server`,

4.5. INSTALACE OLAP ANALYZÁTORU A SPUŠTĚNÍ SYSTÉMU

na kterém běží datový sklad knihoven MU. Je nutné změnit parametry \$DispatchHost a \$DispatchPort a \$preforkedFirstPort dle serveru na kterém aplikace poběží a nahradit jím tak localhost, podobně je nutné změnit parametry dle databázového serveru, který bude spravovat data aplikace a změnit parametry \$CacheDB, \$ShareDB.

5. Instalace Osito části analyzátoru BEE (webová aplikace pro tvorbu reportů ze strany klienta)

Je nutné vytvořit adresář html a v něm adresář osito, umístit do adresářové struktury /var/www/ a přidělit jeho vlastnictví účtu bear.

Z příloženého CD z adresáře download je nutné zkopírovat do /var/www/html/osito obsah složky download/beeOsito a přidělit jeho vlastnictví celého obsahu účtu bear. Je nutné změnit skupinu vlastníků všech složek a souborů nacházejících se ve /var/www/html/osito na skupinu http serveru, na kterém aplikace poběží (například u Apache je to skupina www-data) a nastavit pro tuto skupinu právo čís, psát i spouštět pro všechny složky a soubory v adresáři osito. Především se toto nastavení práv vyžaduje u adresářů html, CSS a const.

Je nutné stáhnout a nainstalovat balíčky: libapache2-mod-perl2, libapache2-mod-perl2-dev, libpng12-0, libpng12-dev, gda2-mysql, gda2-odbc, htmldoc, htmldoc-common, libgda2-common, libgd-gd2-perl, libgd2-xpm-dev

Po spuštění checkenv.pl (zadá se cd /var/www/html/osito

a potom /usr/local/share/bee/util/checkenv.pl) se objeví tabulka, která vypíše moduly prostředí, které je nutné nainstalovat pro běh části Osito. Moduly u kterých se objeví miss, je nutné nainstalovat ručně. Instalace se provede následovně: nejdříve se zadá v terminálové konzole příkaz cpan, čímž se přepneme do prostředí CPAN a poté install nazev_modulu. (Například install GD::Graph).

Poté, co se již v tabulce nevyskytuje miss u žádného modulu, je nutné z příloženého CD zkopírovat soubor httpasswd do složky obsahující soubory pro běh http serveru (např. u Apache je to /etc/apache2) a přidat do konfiguračního souboru příslušného http serveru (např. u Apache je to soubor /etc/apache2/apache2.conf) následující řádky:

```
<Directory "/var/www/html/osito">
    SetEnv XML_SIMPLE_PREFERRED_PARSER XML::Parser
    Order deny,allow
    deny from all
    AllowOverride None
</Directory>

<Directory "/var/www/html">
    Order deny,allow
    deny from all
    AllowOverride None
```

```
</Directory>

AddHandler cgi-script .cgi
Alias /DWHMU /var/www/html/osito/
<Directory "/var/www/html/osito">
    Options Indexes ExecCGI
    Order deny,allow
    allow from localhost .localdomain host .domain.com
    deny from all
    AuthType Basic
    AuthName Bear
    AuthUserFile /etc/apache2/htpasswd
    require valid-user
    DirectoryIndex index.cgi
</Directory>
```

AuthUserFile a allow from je nutné změnit podle příslušného http serveru na kterém aplikace skutečně poběží.

Bližší informace k instalaci jednotlivých částí BEE analyzátoru lze nalézt v příloženém CD ve složce download /beeBeard/doc/Readme a

download /beeOsito/doc/Readme a download /beeCommon/doc/Readme. Konfigurační soubor Osito se nachází v adresáři /var/www/html/osito/const/Const.pm. Dle serveru na kterém aplikace poběží je nutné změnit parametr \$bearURL.

6. VI. Spuštění systému:

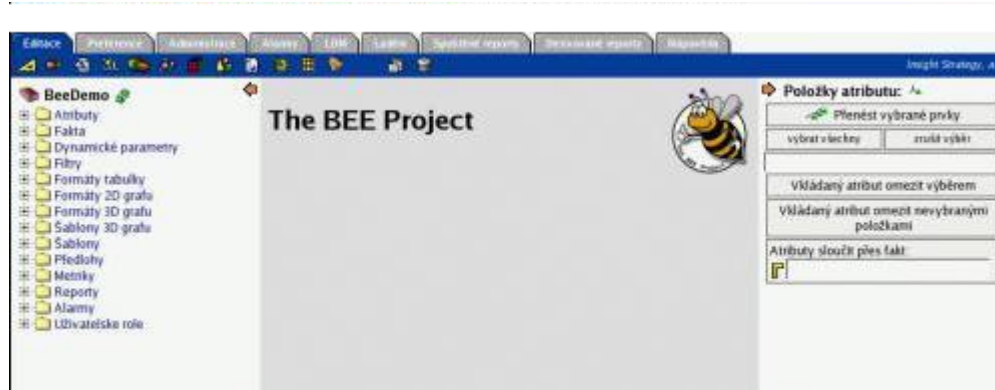
Pokud je nutné provést ETL fázi, spustí se postupným zadáním příkazů `cd /opt/ETL/` a `./ETL.sh` a poté je nutné spustit ROLAP server podle následujícího návodu. Pokud se nemá provádět ETL fáze, tak se ROLAP server spustí hned jako první. ROLAP server se spustí postupným zadáním příkazů `cd /opt/beard/` a `./beard.pl`. Jednotliví klienti se připojují pomocí www rozhraní na adrese `http://nazev_serveru/DWHMU` a pro přihlášení lze použít jako přihl. jméno: `DWHMU_User` a heslo: `DWHMU`.

4.6 Popis uživatelského rozhraní

Při vytváření reportu (určité zobrazení požadovaných dat ve vhodném formátu – tabulka, 2D graf a 3D graf) je uživatelský přístup k datovému skladu knihoven MU umožněn prostřednictvím WWW prohlížeče (Mozilla, MS Internet Explorer aj.).

4.6. POPIS UŽIVATELSKÉHO ROZHRAŇÍ

Po zadání odpovídající adresy serveru, kde je použitý OLAP analyzátor nainstalován, se objeví v okně prohlížeče základní rozvržení podle následujícího obrázku.



Práce uživatele v projektu BEE při vytváření reportů vychází ze systému vzájemně se překrývajících karet s jejich viditelným popisem (záložek) v záhlaví obsahu okna. Obsah jednotlivých karet je následující:

- **„Editace“** slouží pro práci s objekty pro definici reportu (metrika, šablona, formát, filtr). V rámci této karty je možné objekty vytvářet, editovat, kopírovat a případně i odstraňovat. Obsah vnořených částí stromu objektů umístěného v levé části okna je přitom závislý na používaném datovém modelu vytvořeného datového skladu knihoven MU.
- **„Preference“** je karta pro nastavení uživatelského prostředí (cookies, nastavení reportů, šířka postranních panelů apod.) a pro výběr z dostupných projektů.
- **„Alarmy“** je karta, která obsahuje seznam existujících alarmů. Kliknutím na vybraný alarm můžeme přímo editovat jeho definici.
- **„LDM“** (*Logical Data Model*) obsahuje grafické schéma logického datového modelu datového skladu aktuálního projektu. Struktura datového modelu vzniká při logickém návrhu datového skladu.
- **„Spuštěné reporty“** je karta, ve které se podle nastavení parametrů v záložce „Preference“ (nastavení reportu a zobrazení výsledku) zobrazí vytvořené reporty v samostatné záložce nebo v záložce společné. V této kartě můžeme:
 - **smazat** - kliknutím na ikonu popelnice . Report bude vymazán pouze ze seznamu spuštěných reportů, jeho definice zůstane dostupná pod kartou „Editace“.
 - **vybrat vše** - umožní kliknutím na ikonu  označit všechny reporty v této kartě a poté je můžeme např. všechny smazat.

- *invertovat výběr* - umožňuje kliknutím na ikonu  invertovat označení vybraných reportů.
 - *editace* - označením reportu a kliknutím na ikonu  se nám otevře editace vybraného reportu, kde můžeme provádět dané změny.
- „**Derivované reporty**“ je záložka pro zobrazení reportů, které vznikly jako odvozené v důsledku manipulace s reportem (drillování, pivotování a filtrování). Tato karta nám umožňuje vrátit se k jednotlivým krokům reportů, které byly vytvořeny. Kliknutím na ikonu získáme historii jednotlivých modifikací. Ovládání funkcí smazat, vybrat vše a invertovat výběr je totožné s kartou „Spuštěné reporty“.

Report začínáme vytvářet v kartě „Editace“. V levé části této karty je zobrazován strom objektů používaný při tvorbě reportu tak, že při rozvinutí jeho jednotlivých částí (kliknutím na znaménko „+“ před názvem objektu) se zobrazí seznam konkrétních využitelných objektů. Ve stromu objektů umístěném v levé části okna definujeme report na základě již existujících nebo nově vytvářených šablon, formátů a filtrů. Šablonu pak vytváříme na základě atributů a faktů, filtr pracuje s atributy, fakty a metrikami.

Při vytváření nových objektů lze použít ikony, které jsou uvedeny v záhlaví karty „Editace“ na panelu (liště) nástrojů. Při umístění myši nad tyto ikony se zobrazí odpovídající nápověda. Kliknutím na jednotlivé ikony se spouští práce s příslušnými objekty.

Ikony v panelu nástrojů:



„**Nová metrika**“ - vytvoří novou metriku.



„**Nový filtr**“ - vytvoří nový filtr.



„**Nový formát**“ - vytvoří nový formát tabulky.



„**Nový formát 2D**“ - vytvoří nový formát 2D grafu.



„**Nový formát mapa 2d**“ - vytvoří nový formát mapy 2D.



„**Nový formát 3D**“ - vytvoří nový formát 3D grafu.



„**Nová šablona**“ - vytvoří novou šablonu tabulky nebo 2D grafu.



„**Nová šablona 3D**“ - vytvoří novou šablonu 3D grafu.



„**Nová předloha**“ - vytvoří novou předlohu.

-  „Nový dynamický parametr“ - vytvoří nový dynamický parametr.
-  „Nový report“ - vytvoří nový report.
-  „Nový alarm“ - vytvoří nový alarm.
-  „Kopie objektu“ - vytvoří kopii aktuálního objektu.
-  „Smazat objekt“ - umožňuje smazání aktuálního objektu.

Detailní popis uživatelského rozhraní samotného systému lze nalézt prostřednictvím uživatelského prostředí systému pod záložkou Nápověda.

- **Kliknutí levým tlačítkem myši jednou** používáme pro otevření ikon, složek a karet, dále pro uchopení a vložení objektu v rámci metody drag and drop.
- **Kliknutí levým tlačítkem myši dvakrát** používáme pro výběr objektu (kromě atributu a faktu, které jsou definovány systémově) pro další práci, tj. otevření formuláře objektu pro definici či úpravy.
- Metodou **drag and drop** rozumíme přenos objektu (jeho uchopení, přenos a vložení do požadované cílové oblasti). V projektu BEE to provedeme jedním kliknutím levým tlačítkem myši nad určitým grafickým prvkem (např. nad názvem faktu), který se takto dočasně spojí s kurzorem myši. Přemístěním kurzoru s připojeným objektem nad požadované cílové místo a jedním kliknutím opět levým tlačítkem myši se pak uchopený prvek vloží na požadované místo.
- **Kliknutí pravým tlačítkem myši jednou** používáme pro práci s kontextovým menu.
- **Ctrl+levé tlačítko myši** používáme pro výběr více elementů atributů současně.

4.7 Ukázka tvorby reportu

Na následujícím příkladu budu sledovat celkovou cenu, která byla zaplacená za jednotky podle data založení jednotky v knihovně a statutu jednotky.

Příslušný report se vytvoří dle následujících kroků:

1. Je nutné definovat metriku.

Po kliknutí na ikonu „Nová metrika“, umístěnou v horní části karty „Editace“ v panelu nástrojů, se ve stromu objektů vytvoří nový objekt metriky a v pravé části okna se zobrazí formulář pro definování jejích parametrů. V příkladu sledujeme celkovou cenu jednotek. Proto definujeme metriku, která je sumou faktu [Cena jednotky]. Požadovaný fakt, v našem případě [Cena jednotky], najdeme ve stromu objektů ve

složce „Fakta“. Metodou drag and drop uchopíme vybraný fakt [Cena jednotky] a přeneseme jej do karty „Základní definice“ metriky na tlačítko funkce „SUM“. Tlačítko funkce „SUM“ se prosvětlí. V poli „Definice metriky“ se objeví výraz „SUM[Cena jednotky]“. Vytvořenou metriku je potřeba pojmenovat. Kliknutím se přepneme do karty „Popis“. V poli „Jméno“ zapíšeme požadované jméno metriky Celková cena jednotek. Do pole „Popis“ můžeme zapsat další podrobné informace týkající se definovaného objektu, tj. metriky.

Nadefinovanou metriku, uložíme kliknutím na tlačítko „Uložit“.

2. Vytvoříme nový report.

Po kliknutí na ikonu „Nový report“, umístěnou v horní části karty „Editace“ v panelu nástrojů, se v pravé části okna otevře formulář pro definování reportu. Po vytvoření nového reportu je nám ve formuláři pro definování reportu nabídnuta karta „Popis“. Report můžeme v kartě „Popis“ pojmenovat. V poli „Jméno“ zapíšeme požadované jméno reportu: „Cena za jednotky jednotlivých fakult v příslušném roce“. V poli „Popis“ lze zapsat další podrobné informace týkající se definovaného reportu. Zadané informace uložíme kliknutím na tlačítko „Uložit“.

3. Definujeme šablonu, tj. definici rozložení objektů ve výsledném reportu.

V kartě „Editace“ v okně pro definici reportu klikneme na kartu „Šablona“. Zobrazí se nám okno pro definici šablony. V okně šablony máme na výběr z následujících možností: „Vytvořit a editovat: Šablona“ a „Vytvořit a editovat: Vnořená 3D šablona“. Z nabízených možností si vybereme odkaz „Vytvořit a editovat: Šablona“ a otevřeme formulář pro definici šablony.

Metriku, kterou chceme sledovat, si vybereme ve stromu objektů a metodou „drag and drop“ ji přeneseme do karty „Šablona“ do pole „Metriky“ (umístit na konec). Dle zadání chceme sledovat celkovou cenu jednotek v členění podle roku založení jednotek a fakulty jednotky. Zvolíme, že fakulty chceme zobrazit v řádcích a jednotlivé roky založení ve sloupcích tabulky.

Atribut „Rok“ nalezneme ve stromu objektů ve složce „Atributy -> Čas Založení jednotky“.

Každý atribut nabízí několik forem zobrazení. V případě atributu „Rok“ máme k dispozici formy zobrazení „Dl.název“ a „Kr. název“. Ve stromu objektů klikneme levým tlačítkem myši na „Dl.název“ a přeneseme jej metodou drag and drop. Podobným způsobem si pro zobrazení údajů v řádcích reportu vybereme atribut „Fakulta (Jednotky)“ ze složky, ve stromu objektů „Atributy -> Fakulta -> Fakulta (Jednotky)“ ve formě zobrazení „Zkratka názvu fakulty“.

Po provedení popsaných kroků je vše připraveno pro zobrazení reportu, takže můžeme kliknout na tlačítko „Uložit a spustit“. Otevře se obrazovka výsledného reportu. Pro další spuštění vytvořeného reportu již stačí kliknout na tlačítko „Spustit report“.

4.7. UKÁZKA TVORBY REPORTU

LDM	Ladění	Spuštěné reporty	Derivované reporty	Nápověda	Bee Info					
Fakulta (Jednotky)	Rok (Založení jednotky)	1999	2000	2001	2002	2003	2004	2005	2006	2007
ESF							7929081,68	977310,90	659397,00	1408485,00
FF						16662558,50	4543017,54	4151069,16	5355789,14	5479808,87
FF-K						8836333,96	1818369,62	1621660,85	2701664,49	2897603,56
FF-S						3381876,97	467076,11	381978,53	114901,62	32969,70
FFHUD						658366,94	148527,41	92329,16	694503,89	1146528,45
FI							9223606,82	945538,80	1110681,30	805117,90
KUK		4198,00	41129,00	10132,00	23936,00	17876,00	27077,00	37600,30	87649,20	346348,35
LF		41760,00	161032,30	287990,40	460371,30	350086,60	537013,50	1332383,10	202618570,58	738390,90
PEDF							3363064,77	987060,00	890699,00	659859,00
PRAF							20313984,72	3346076,00	2165933,50	2595007,00
PRIF							4976140,01	1785473,00	779063,14	700,00
PRIMA							130203,04	211732,30	75334,20	
TELC							52860,00	36071,00	34486,00	6888,00

Konkrétní příklady reportů (výsledků možných analýz), týkajících se všech tří výše popsaných předmětných oblastí, byly na ukázkou vytvořeny a jsou součástí navrženého systému pod uživatelským účtem DWHMU_User a heslem DWHMU.

Kapitola 5

Závěr

Výsledkem diplomové práce je OLAP systém umožňující OLAP analýzu provozních dat v prostředí knihoven MU. Při návrhu a implementaci zmíněného OLAP systému byl kladen důraz na to, zpřístupnit analytickou informaci uživatelům knihovního systému tak, aby byla efektivně vyhodnotitelná, snadno a včas dostupná a důvěryhodná. Navíc byly při návrhu systému zvoleny takové technologie, které jsou optimální z hlediska celkových nákladů na vlastnictví systému – náklady jsou nulové.

Pro splnění výše uvedených požadavků byly zjišťovány všechny možné technologie, postupy a metodiky, obecně používané při návrhu jednotlivých vrstev OLAP systému. S ohledem na prostředí ve kterém bude OLAP systém implementován, byly zvoleny ty, které jsem shledala jako nejvhodnější řešení dané problematiky.

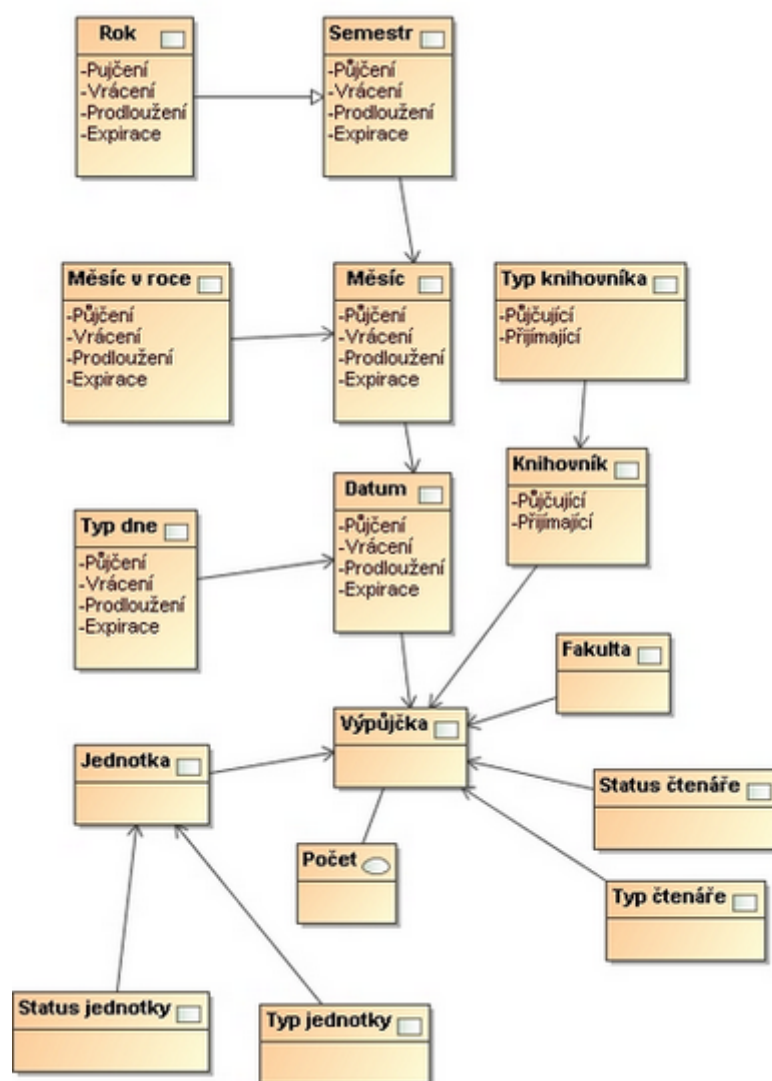
Navržený OLAP systém poskytuje dle mého názoru silný aparát pro OLAP analýzu provozních dat týkajících se sledování počtů všech výpůjček, počtů jednotek v knihovně, jejich ceny, počtu vypůjčení konkrétní jednotky, počtu stran jednotek a počtu událostí provedených v systému a to z mnoha možných úhlů pohledu jako jsou například fakulta, knihovník, který vypůjčil jednotku nebo přijal jednotku, status jednotky, status čtenáře, datum půjčení, datum vrácení výpůjčky, datum prodloužení výpůjčky, datum jednotky, typu události, provedené systémem apod. Na základě takto provedených analýz lze prostřednictvím systému zjistit a poskytnout knihovníkům pomocí předpřipravených reportů, informace typu, v jakém období je systém nejvíce vytížen, kteří čtenáři v jakém období nejvíce nevrací knížky, které jednotky se ve kterém období nejvíce půjčují, která fakulta provedla v daném období nejvíce výpůjček, jaká je celková cena zaplacená za příslušné jednotky v knihovně v daném období a spoustu další užitečných informací.

Velice příjemnou vlastností systému je také možnost exportovat výsledky analýz v podobě tabulky, grafu či jiné hierarchické struktury formou pdf či xls souboru.

Co se týče uživatelského prostředí systému, je dle mého názoru uživatelsky příjemné a to, jak po stránce intuitivního pochopení použití jednotlivých prvků systému, tak i dle míry rychlosti vyhodnocení příslušného požadavku.

Příloha A

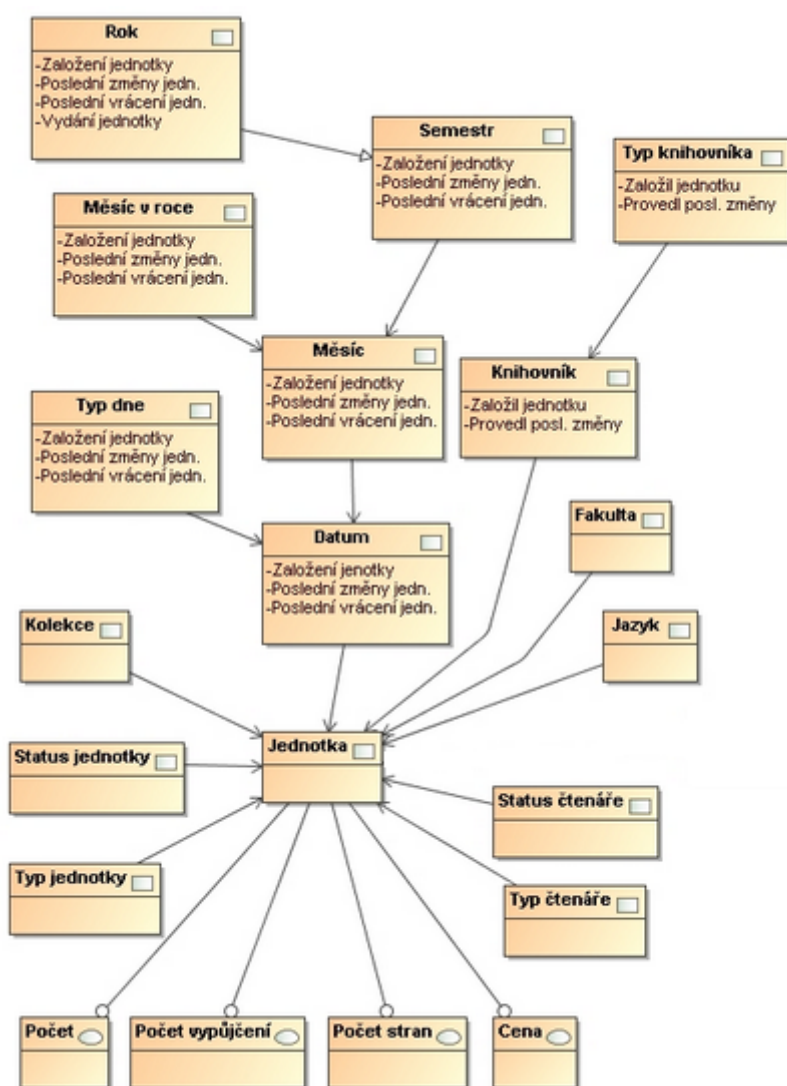
Schéma datového tržiště „sledování výpůjček“



Obrázek A.1:

Příloha B

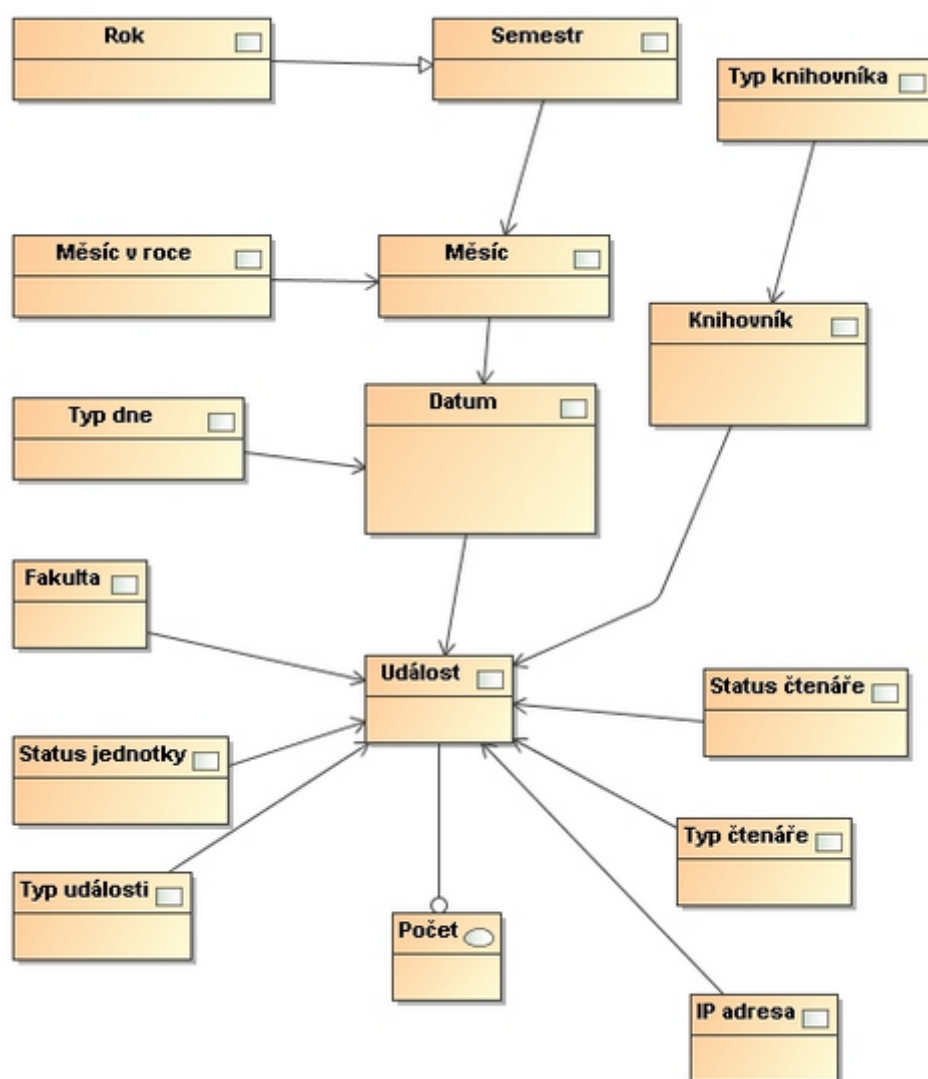
Schéma datového tržiště „vyhodnocení jednotek“



Obrázek B.1:

Příloha C

Schéma datového tržiště „sledování událostí v systému“



Obrázek C.1:

Příloha D

metadata DWH MU

```
<?xml version='1.0' encoding="UTF-8" standalone='yes'?>
<project name="BEE DWHMU" description="The BEE DWHMU
  project" mdversion="2.0">
  <hrelation name="Datum TypDne" id="datum_typ_dne" description="">
    <to cardinality="N" optional="false">datum</to>
    <from cardinality="N" optional="false">typ_dne</from>
    <join table="demohost/demo_bee/r_datum_typ_dne"/>
  </hrelation>
  <hrelation name="Den v týdnu - Den" id="wday_datum" description="">
    <to cardinality="N" optional="false">datum</to>
    <from cardinality="1" optional="false">wday</from>
  </hrelation>
  <hrelation name="Rok Semestr" id="rok_semestr" description="">
    <to cardinality="N" optional="false">semestr</to>
    <from cardinality="1" optional="false">rok</from>
  </hrelation>
  <hrelation name="Rok - Semestr v roce" id="rok_semestr_v_roce"
description="">
    <to cardinality="N" optional="false">semestr_v_roce</to>
    <from cardinality="1" optional="false">rok</from>
  </hrelation>
  <fact id="pocet_vypujcek">
    <gui-presentation coord="549 424" id="5" />
    <expr>demohost/demo_bee/lu_vypujcka/pocet</expr>
    <expr func="SUM">
demohost/demo_bee/agr_semestr_fakulta/pocet_vypujcek</expr>
    <name en="No. of loans" cz="Počet výpůjček" />
    <acl id="allow" DWHMU_all="R-E--"/>
    <acl id="deny" DWHMU_all="-W-DC"/>
    <description></description>
  </fact>
  <fact id="pocet_stran">
    <gui-presentation coord="661 424" id="5" />
    <expr>demohost/demo_bee/lu_jednotka/pocet_stran</expr>
    <name en="Count of Pages" cz="Počet stran" />
    <acl id="allow" DWHMU_all="R-E--"/>
    <acl id="deny" DWHMU_all="-W-DC"/>
    <description></description>
```

```

</fact>
<fact id="pocet_jednotek">
  <gui-presentation coord="661 424" id="5" />
  <expr>demohost/demo_bee/lu_jednotka/pocet_jednotek</expr>
  <expr func="SUM">
    demohost/demo_bee/agr_semestr_fakulta/pocet_vypujcek</expr>
  <name en="Count of Items" cz="Počet jednotek" />
  <acl id="allow" DWHMU_all="R-E--"/>
  <acl id="deny" DWHMU_all="-W-DC"/>
  <description></description>
</fact>
<fact id="pocet_vypujceni_jednotky">
  <gui-presentation coord="661 424" id="5" />
  <expr>demohost/demo_bee/lu_jednotka/pocet_pujceni</expr>
  <name en="Count Loan of Items" cz="Počet vypůjčení jednotky" />
  <acl id="allow" DWHMU_all="R-E--"/>
  <acl id="deny" DWHMU_all="-W-DC"/>
  <description></description>
</fact>
<fact id="pocet_udalosti_v_systemu">
  <gui-presentation coord="661 424" id="5" />
  <expr>demohost/demo_bee/lu_udalost/pocet</expr>
  <expr func="SUM">
    demohost/demo_bee/agr_semestr_fakulta/pocet_vypujcek
  </expr>
  <name en="Count of Events in the
System" cz="Počet událostí v systému" />
  <acl id="allow" DWHMU_all="R-E--"/>
  <acl id="deny" DWHMU_all="-W-DC"/>
  <description></description>
</fact>
<fact id="doba_pujceni">
  <gui-presentation coord="661 424" id="5" />
  <expr>
    demohost/demo_bee/lu_vypujcka/datum_id_vraceni-datum_id_pujceni
  </expr>
  <name en="Time of Loan" cz="Doba půjčení" />
  <acl id="allow" DWHMU_all="R-E--"/>
  <acl id="deny" DWHMU_all="-W-DC"/>
  <description></description>
</fact>
<fact id="cena_polozky">
  <gui-presentation coord="544 399" id="5" />
  <expr>demohost/demo_bee/lu_jednotka/cena</expr>
  <expr func="SUM">
    demohost/demo_bee/agr_kvartal_pobocka/cena
  </expr>
  <name en="Price of the Item" cz="Cena jednotky" />

```

```

    <acl id="allow" DWHMU_all="R-E--"/>
    <acl id="deny" DWHMU_all="-W-DC"/>
    <description></description>
</fact>
<attr group="9990" id="typ_dne">
    <gui-presentation coord="207 151" id="5" />
    <pk>demohost/demo_bee/lu_typ_dne/id</pk>
    <name en="Type of day" cz="Typ dne" />
    <acl id="allow" DWHMU_all="R-E--"/>
    <acl id="deny" DWHMU_all="-W-DC"/>
    <description></description>
    <display id="name">
        <def en="demohost/demo_bee/lu_typ_dne/popis"
            cz="demohost/demo_bee/lu_typ_dne/popis" />
        <name en="Name of the type of day" cz="Název typu dne" />
    </display>
    <fk>demohost/demo_bee/r_datum_typ_dne/typ_dne_id</fk>
</attr>
<attr group="9990" id="datum">
<link id="kal">
<def en="http://demohost/bear/cal.cgi?id=$ID&l=en&usr=$USER"
    cz="http://demohost/bear/cal.cgi?id=$ID&l=cz&usr=$USER" />
<name en="Calendar" cz="Kalendář" />
    </link>
    <link id="sum">
    <def en="http://demohost/bear/dnote.cgi?atr=$ATR&
    id=$ID&l=en&usr=$USER"
    cz="http://demohost/bear/dnote.cgi?atr=$ATR&
    id=$ID&l=cz&usr=$USER" />
        <name en="Note" cz="Poznámka" />
    </link>
    <gui-presentation coord="379 131" id="9" />
    <gui-presentation coord="355 111" id="5" />
    <pk>demohost/demo_bee/lu_datum/id</pk>
    <name en="Day" cz="Datum" />
    <acl id="allow" DWHMU_all="R-E--"/>
    <acl id="deny" DWHMU_all="-W-DC"/>
    <description></description>
    <display id="lname">
<def
    en="demohost/demo_bee/lu_datum/DATE_FORMAT(id,'%M %D, %Y') "
    cz="demohost/demo_bee/lu_datum/CASE MONTH(id)
    WHEN 1 THEN DATE_FORMAT(id,'%e. leden %Y')
    WHEN 2 THEN DATE_FORMAT(id,'%e. únor %Y')
    WHEN 3 THEN DATE_FORMAT(id,'%e. březn %Y')
    WHEN 4 THEN DATE_FORMAT(id,'%e. duben %Y')
    WHEN 5 THEN DATE_FORMAT(id,'%e. květen %Y')
    WHEN 6 THEN DATE_FORMAT(id,'%e. červen %Y')

```

```

WHEN 7 THEN DATE_FORMAT(id,'%e. červenec %Y')
WHEN 8 THEN DATE_FORMAT(id,'%e. srpen %Y')
WHEN 9 THEN DATE_FORMAT(id,'%e. září %Y')
.
.
.
.
.
.

<role id="zakladajici_jednotku">
  <name en="Constituent of the item"
  cz="Zakládající jednotku" />
</role>
<role id="provedl_zmeny_jednotky">
  <name en="Updating the item"
  cz="Měnicí jednotku" />
</role>
<role id="provedl_prisl_udalost">
  <name en="Do the event"
  cz="Provedl příslušnou událost"/>
</role>
<acl id="allow" DWHMU_all="R-E--"/>
<acl id="deny" DWHMU_all="-W-DC"/>
<description en="Cataloger Dimension"
cz="Dimenze knihovníka" />
</group>
<const></const>
<gui-layer width="300" visible="YES" height="300" id="5">
  <name cz="LDM" en="LDM" />
</gui-layer>
</project>

```

Celý soubor DWHMU.xml metadat lze nalézt na příloženém CD v adresáři download /BeeBeard/beard/mdata/

Příloha E

Obsah CD

Součástí práce je i přiložené CD, které obsahuje:

- zdrojový kód této práce ve formátu XML
- zdrojový kód této práce ve formátu pdf
- zdrojové kódy jednotlivých částí navrženého OLAP analyzátoru
- dokumentace k jednotlivým částem navrženého OLAP analyzátoru

Literatura

- [1] Lacko, L.: *Datové sklady, analýza OLAP a dolování dat*, Computer Press, 2002, 80–7226–969–0, <<http://www.cpress.cz>> . 1, 2.2, 2.3, 1, 2, 2.4, 1, 2, 2.5, 2.5.1, 1, 2, 2.5.2.1, 2.5.2.2, 2.6, 1, 2, 3, 2.7, 1, 2, 1, 2, 3, 4, 5, 6, 3.3
- [2] *Aleph–MU administrace*, <<http://kic.ics.muni.cz>>.
- [3] *Aleph v knihovnách MU*, <<http://www.ics.muni.cz/zpravodaj/articles/303.html>>.
- [4] *Knihovny MU jako jednotný systém*, <<http://library.muni.cz/system--knihoven.html>>. 3.1
- [5] *O systému Aleph*, <http://library.cuni.cz/system--knihoven.html> <<http://aleph.cuni.cz/Aleph--32.html>>. 3.1
- [6] *Perl Programming*, <<http://forums.devshed.com/perl--programming--6/flat--file--database--to--mysql--7994.html>>.
- [7] *Jasper Soft*, <<http://www.jaspersoft.com/>>.
- [8] *Jasper ETL*, <http://www.jaspersoft.com/JasperSoft_JasperETL.html>. 1
- [9] *Jasper Analysis*, <http://www.jaspersoft.com/JasperSoft_JasperAnalysis.html>. 2
- [10] *Jasper Reports*, <http://www.jaspersoft.com/JasperSoft_JasperReports.html>. 3
- [11] *Bee_white_paper.pdf*, I.: *Řešení datového skladu a manažerského informačního systému*. 4.1, 4.2
- [12] Humphries, M.: *Data Warehousing, návrh a implementace*, Computer Press, 2002, 80–7226–560–1, <<http://www.cpress.cz>> . 2.7, 1, 2, 3, 2.7.1, 1, 2
- [13] *Ovládání a autorizace APACHE*, <<http://www.root.cz/clanky/ovladani--apache--a--autorizace>>. 2.2
- [14] *Eporty a importy dat z příkazové řádky*, <<http://www.interval.cz/clanky/exporty--a--importy--dat--v--sql--serveru--z--prikazove--radky>>.
- [15] Kejkula, M.: *Získávání znalostí z databází*, 2004. 1, 4
- [16] *Eporty a importy dat v SQL serveru*, <<http://dev.mysql.com/doc/refman/5.0/en/load--data.html>>. 4.4

-
- [17] Kosek, J.: *XML pro každého*, Grada Publishing, 2000, 80–7169–860–2, <<http://www.grada.cz>> .
- [18] *Příloha* 6/2003, <<http://www.systemonline.cz/site/data--warehousing/6insight.htm>>. 2.1, 2.6, 3.3.2
- [19] *Datové sklady a OLAP*, <<http://datamining.xf.cz/view.php?cisloclanku=2002102808>>. 1, 2, 3
- [20] *Typy datových skladů*, <<http://datamining.xf.cz/view.php?cisloclanku=2002102809>>.
- [21] *OLAP uložení dat*, <<http://datamining.xf.cz/view.php?cisloclanku=2002102810>>. 1, 2, 3, 4, 5, 1, 2, 3
- [22] *Datové kostky*, <<http://datamining.xf.cz/view.php?cisloclanku=2002102811>>.
- [23] Dejmek, P.: *Multidimenzionální kostka nad relační databází Kosu*, 2002. 2.3