VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

NÁVRH INTERAKTIVNÍHO WWW OLAP ROZHRANÍ PRO ANALÝZU PRODUKCE VÝROBNÍCH ZÁVODŮ

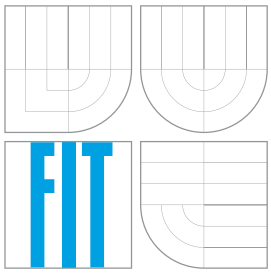
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. PAVEL MAZÁČ

BRNO 2008

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

NÁVRH INTERAKTIVNÍHO WWW OLAP ROZHRANÍ PRO ANALÝZU PRODUKCE VÝROBNÍCH ZÁVODŮ

DESIGN OF INTERACTIVE WWW OLAP INTERFACE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. PAVEL MAZÁČ

VEDOUCÍ PRÁCE
SUPERVISOR

Prof. Ing. TOMÁŠ HRUŠKA, CSc.

BRNO 2008

Abstrakt

Tato práce se zabývá problematikou OLAP analýzy. Uvádí důležité teoretické poznatky a srovnává vybrané existující produkty. Hlavním cílem bylo vytvořit vlastní implementaci systému OLAP. Práce popisuje způsob návrhu a implementace tohoto systému.

Klíčová slova

OLAP, OLAP analýza, Business Intelligence, BI, Microsoft SQL Server, Mondrian, Firebird, databáze

Abstract

This work is focused on OLAP analysis. The work presents important theoretic facts and compares availible OLAP systems in different ways. The main goal was to create own OLAP system. Design and implementation of this system is described in the project.

Keywords

OLAP, OLAP analysis, Business Intelligence, BI, MicroSoft SQL Server, Mondrian, Firebird, database

Citace

Pavel Mazáč: Návrh interaktivního WWW OLAP rozhraní pro analýzu produkce výrobních závodů, diplomová práce, Brno, FIT VUT v Brně, 2008

Návrh interaktivního WWW OLAP rozhraní pro analýzu produkce výrobních závodů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Tomáše Hrušky. Další technické informace mi poskytl pan Slavomír Skopalík. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Pavel Mazáč
31. července 2008

Poděkování

Děkuji panu Tomáši Hruškovi za vedení mého diplomového projektu a především panu Slavomíru Skopalíkovi z firmy Elektlabs za odborné rady v oblasti databází a zpracování dat.

Obsah

Kapitola 1

Úvod

V dnešní době téměř každá organizace produkuje a sbírá obrovské množství dat při svém procesu podnikání. Tyto data dnes umíme zpracovat a uložit bez větších problémů. Mezi nejpoužívanější úložiště perzistentních dat patří databáze. Existuje velké množství dodavatelů těchto řešení od největších firem jako Oracle po mnohem menší projekty jako PostgreSQL, které jsou často dostupné zdarma. Převážná většina techto databázových systémů je založena na *relačním modelu*, který definoval v roce 1970 pan Edgar F. Codd.

Problém, kterému dnes čelíme, je získání užitečných informací z těchto rozsáhlých databází. Informací v tomto významu myslíme netriviální poznatek o datech, který lze získat jen těžko pomocí jednoduchých technik[1].

Tyto poznatky jsou výsledkem složité analýzy velkého množství dat pomocí sofistikovaných postupů a technik. Velmi často bývá hodnotnou informací znalost vazeb, vztahů a závislostí v analyzovaných datech, nebo objevení trendu, který se v časově uspořádaných údajích vyskytuje.

Získané informace jsou velice důležitým prvkem při plánování řízení, obecně při rozhodování o budoucích krocích organizace. Informace jsou typicky určeny pro manažery a musí být pro tyto uživatele upraveny do formy, která bude lehce pochopitelná a použitelná. Obecně se tyto techniky a postupy souhrně označují jako *Business intelligence (BI)*. BI charakterizuje proces sběru, uložení a analýzy dat, získání a prezentace informací pro převážně obchodně zaměřené subjekty.

Tato práce se zabývá jednou z technik BI, konkrétně technologí *OLAP (Online Analytical Processing)*. Tato volně definovaná sada nástrojů a postupů umožňuje konečnému uživateli souhrný pohled na analyzovaná data. Výsledkem jsou agregované údaje podle různých vlastností, které jsou charakteristické pro danou oblast podnikatelského zaměření organizace.

Cílem projektu je navrhnout vlastní systém OLAP analýzy, který by zahrnoval všechny její stupně a tvořil komplexní soubor aplikací a nástrojů pro datovou analýzu.

V první části práce se zabývám současným stavem znalostí v oblasti OLAP technologie, ve druhé části práce popisuji vlastní návrh a implementaci OLAP systému pro analýzu dat v průmyslových podnicích. Práce byla konzultována a navrhována ve spolupráci se společností *Elektlabs s.r.o.* tak, aby bylo v budoucnu možný systém úspěšně nasadit, používat a v neposlední řadě dále rozvíjet pro analýzu dat získaných při výrobě v nejrůznějších odvětvích strojírenství.

Kapitola 2 — obecně definuje systémy pro podporu rozhodování, jejich vlastnosti a po-

[1] např. zodpovězení dotazu některého dotazovacího jazyka

žadavky, které jsou na ně kladeny. Uvádí základní definici a zařazení systémů OLAP analýzy. Porovnává vlastnosti klasických a analytických databází.

Kapitola 3 — je zaměřena na definici datového skladu, jeho vlastnostmi a způsobu implementace pomocí dostupných systémů pro správu dat.

Kapitola 4 — se zabývá OLAP analýzou. Popisuje jednotlivé operace, jejich význam a použití. Dále naznačuje význam OLAP serveru při výpočtu výsledků analýzy a diskutuje způsoby vizualizace konečných výsledků ve formě tabulek a grafů.

Kapitola 5 — ukazuje existující řešení OLAP analýzy. Porovnává dva odlišné produkty – Microsoft SQL Server 2005 a systém Mondrian.

Kapitola 6 — definuje požadavky na vyvíjený systém OLAP a obsahuje návrh implementace celého systému.

Kapitola 7 — obsahuje detaily implementace jednotlivých částí výsledného systému OLAP analýzy.

Při studiu technologie OLAP bylo použito hlavně knihy [2], jejíchž autoři jsou pan Han a Kamber, také studijní opory předmětu ZZN od pana Zendulky [1] a materálů k přednáškám z předmětu IIS od pana Hrušky [3]. Další informace jsem čerpal z knihy od pana Inmona [4] zaměřenou na detailní problematiku datových skladů a knihy [5] od pana Lacka, která je však zaměřena spíše na koncové uživatele než designéry samotných systémů.

V diplomové práci se objevují myšlenky, nápady, názory, předpoklady a zkušenosti, které autor získal při studiu na FIT, účastí na konferencích zaměřených zejména na databázové aplikace a během své dosavadní programátorské praxe.

Práce navazuje na Semestrální projekt vypracovaný v zimním semestru akademického roku 2007/2008. Tento projekt dále rozvíjí, obohacuje o teoretické i praktické poznatky nabyté při implementaci vlastního OLAP systému a popisuje postup návrhu a implementace této sady aplikací.

Kapitola 2

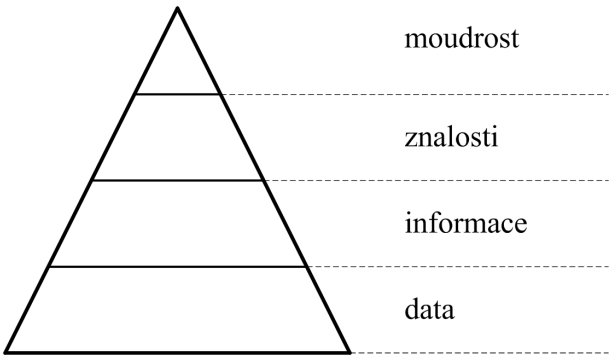
Systémy pro podporu rozhodování, OLAP analýza

Systémy pro podporu rozhodování jsou v dnešní době čím dál více diskutované. Jedná se o techniky schopné hlubší analýzy velkého množství údajů produkující užitečné informace, nezbytné pro strategické rozhodování při řízení organizace. Souhrně lze tyto techniky označit názvem *Business intelligence*, jejichž součástí je také *OLAP analýza*.

Volná definice BI od paní *Shaku Atre* (prezidentka Atre Group, experta na DWH/BI[1]): "*DWH/BI je soubor integrovaných strategických a operačních aplikací, databází a doporučených postupů, které by měly zajistit podnikatelské komunitě snadný přístup k obchodním datům.*"

2.1 Požadavky na systémy Business intelligence

Soubor nástrojů a technik Business intelligence (BI) by měl být schopen analyzovat data na vyšší úrovni abstrakce, nežli je tomu u jednoduchých analýz či reportů postavených nad primárními datovými zdroji. Analýza by naopak často měla probíhat nad více integrovanými zdroji dat.



Obrázek 2.1: Úrovně informací a znalostí

V celém procesu BI nám jde o získání užitečných informací. Na vstupu procesu je velké množství "surových" dat pocházejících z nejrůznějších datových zdrojů organizace. Tyto

[1] zkratkou DWH se myslí Data Ware House – datový sklad

data často obsahují chyby a nepřesnosti, proto se musí čistit, transformovat a integrovat do společného úložiště, které poskytne kvalitní zdroj dat pro následné analýzy. Na výstupu je očekávána užitečn8 informace, získaná analýzou údajů.

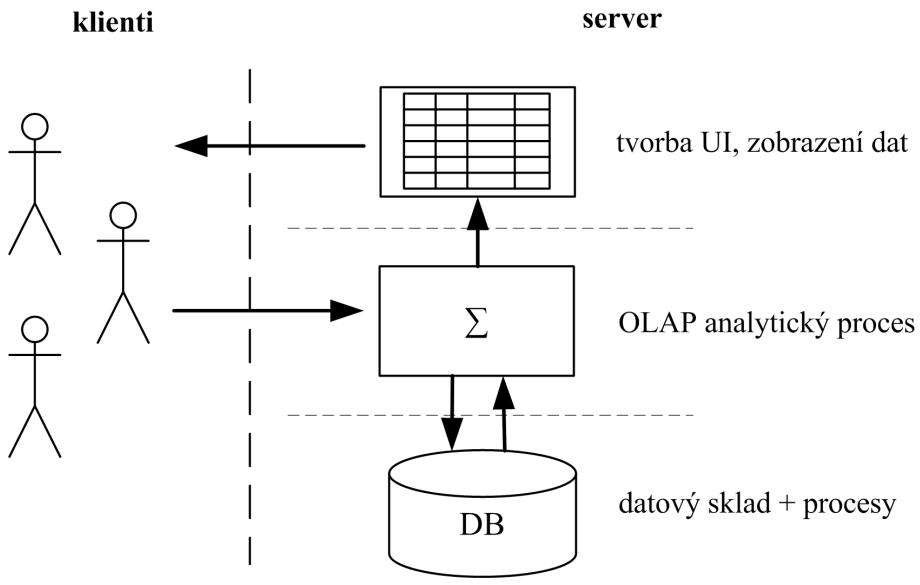
Obrázek 2.1 znázorňuje jednotlivé úrovně znalostí a nastiňuje i kvantitativní poměr mezi jednotlivými stupni.

Není nezbytně nutné pracovat s absolutně všemi údaji, výsledek je mnohdy dostatečný, i když je v něm zanesena malá chyba. Například strategii velké organizace působící na evropském trhu nemůže ovlivnit nezařazení (či chybné zpracování) malé části dat pocházející z jedné pobočky v České republice. Tyto údaje jsou naopak zcela klíčové v organizační strukture této pobočky pro evidenci obchodní činnosti.

2.2 Základní definice OLAP systému

Úkolem OLAP systému je zpracování velkého množství převážně kvantitativních dat pro získání agregovaných informací podle různých kritérií – parametrů. Těmito parametry je definován kontext získaných agregovaných údajů.

Systém OLAP tedy lze definovat jako službu, která umožňuje analýzu nad množinou dat. Požadavky na parametry této analýzy definují uživatelé – analytici, kteří údaje zkoumají. Jedná se tedy o architekturu *klient-server* (obrázek 2.2), kde se server spravuje data, poskytuje prostředky a klienti se na tyto data dotazují.



Obrázek 2.2: Architektura klient-server a jednotlivé části systému

Systém OLAP se skládá z několika částí znázorněných na obrázku 2.2, které jsou na sobě závislé a tvoří jeden celek. Jsou to:

- datový sklad a procesy ho spravující
- OLAP analytický proces
- proces pro zobrazení dat uživateli

Hranice jednotlivých částí se mohou v různých případech lišit. Jde například o přesunutí části aplikační logiky na klientské aplikace. V tomto případě by se diagram 2.2 pozměnil. Současné aplikace se však ubírají opačnou cestou a snaží se co nejvíce logiky implementovat na straně serveru. Nejčastějším klientem bývá internetový prohlížeč. Tento přístup je preferován pro centralizovnou správu serveru a snadnou použitelnost klientských aplikací ve formě internetových prohlížečů.

Všechny zmíněné části OLAP systému jsou dále v textu práce podrobněji popsány.

2.2.1 Datový sklad a multidimenzionální pohled na data

Datovým skladem rozumíme zdroj analyzovaných dat, *databázi*,[2] často integrující údaje celé organizace. Těmito zdroji mohou být operační databáze, datové sešity, textové soubory či jiné dokumenty. Nejrozšířenějším typem databází jsou databáze založené na relačním modelu. Proto dále v textu budeme předpokládat tento typ všude tam, kde zmíníme pojem databáze.

Datový sklad je přizpůsoben potřebám OLAP analýzy, údaje jsou většinou předzpracovány. Jelikož se data sdružují z více zdrojů, musí se provést jejich čištění a trasformace. Tento proces se nazývá *ETL (Extraction, Transformation, Loading)* – extrakce, transformace a zavedení. Detailní popis se nachází v sekci 3.1.

Data v datovém skladu slouží jako zdroj údajů, které vstupují do samotné OLAP analýzy. Datový sklad obsahuje dva typy tabulek:

- tabulky faktů
- tabulky dimenzí

Tabulky faktů jsou zpravidla v databázích největší a obsahují numerická data. Tabulky dimenzí obsahují textové popisky a definují kontext kvantitativních údajů.

Fakta a dimenze jsou základními pojmy a budeme se s nimi setkávat v celém dalším textu. Datový sklad se modeluje jako multidimenzionální struktura – datová kostka či krychle. Hrany krychle tvoří jednotlivé dimenze a jejich průsečíky obsahují kvantitativní údaje – fakta. *Hodnoty dimenzí tedy definují kontext faktů.*

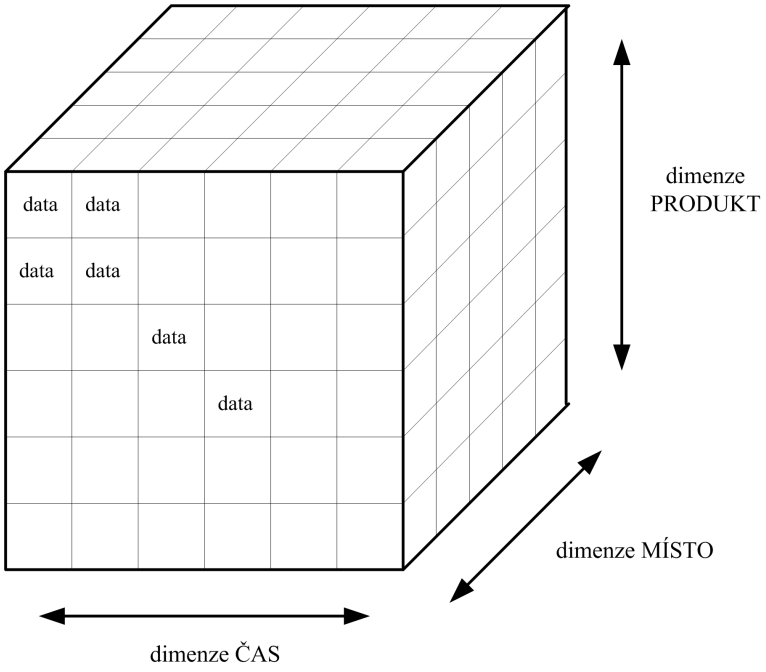
Na obrázku 2.3 je zobrazena jednoduchá tří-dimenzionální datová kostka. Je složena z dimenzí času, místa a produktu. Průsečíky těchto dimenzí obsahují údaje o prodejích v kontextu jednotlivých dimenzí. Detailní popis datových skladů je obsažen v kapitole 3.

2.2.2 Základy OLAP analýzy

Smyslem OLAP analýzy je tedy umožnit efektivní, interaktivní práci s daty, které jsou uloženy v datovém skladu. Tyto data se agregují, filtrují a řadí podle vybraných kritérií. Jde tedy o vnitřní (a pro mnoho uživatelů skrytou) manipulaci s datovou multidimenzionální kostkou.

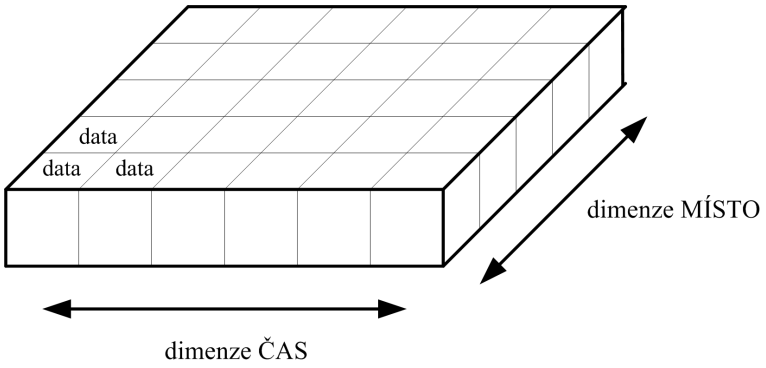
Kostka se může natáčet, převracet, transformovat, některé dimenze se mohou redukovat či úplně odstranit. Tím že do výsledné analýzy nezahrneme některé dimenze, agregované údaje se zvýší. Naopak když dimenzi přídáme, hodnoty agregovaných údajů se sníží a rozdrobí do více menších částí.

[2] nejčastěji se setkáváme s databázemi relačními, avšak používají se i databáze či objektově-relační či objektové



Obrázek 2.3: Multidimenzionální datová kostka

Na obrázku 2.4 je zobrazena datová kostka, která vznikla redukcí datové krychle z obrázku 2.3. Dimenze *produkt* byla odstraněna, zůstaly pouze *místo a čas*. Jako výsledek analýzy bychom získali agregované údaje vztažené k místu a času prodeje. Typ produktu by do analýzy nebyl zahrnut.



Obrázek 2.4: Redukovaná multidimenzionální datová kostka

Jednotlivé operace, které lze realizovat s datovou kostkou jsou přesně definovány. Získaná data je dále potřeba zobrazit uživateli a to v co možná nepřirozenějším formátu. Multidimenzionální kostka má zpravidla vyšší počet dimenzí než zde uvedené tři, my však pro zobrazení používáme prostředky s 2D průmětnou (monitor, tiskárna, ...) a musíme pomocí nich data vizualizovat. To samozdřejmě přináší určité problémy. Detailní popis OLAP operací, jejich realizace a zobrazení dat je obsažen v kapitole 4.

2.3 Rozdíl OLAP systémů od tradičních OLTP

Zkratkou OLTP rozumíme systémy *on-line transaction processing*, tedy systémy založené na transakčním zpracování. Slovo *analytické* tu narozdíl od OLAP systémů chybí.

Jedná se tedy o klasické operační databázové systémy shromažďující data, která jsou produkována organizací. Předpokládá se velké množství zápisů a aktualizací databáze. Naprosto klíčové je řízení transkací a uzamykání databázových objektů. Jde o model zachycení událostí reálného světa – ty nastanou, jsou zaznamenány (perzistentně uloženy) a končí. Klíčový je výkon systému – propustnost transakcí, jaký počet je systém schopen realizovat. Každý záznam v databázi musí být přesně identifikovatelný a přesně dohledatelný (například číslo daňového dokladu).

Naopak u systémů OLAP se předpokládá dlouhodobá analýza dat. Je realizováno jen velmi málo (nebo žádné) operace zápisu a modifikace. Výjimku tvoří dávkové nahrávání nových dat. Klíčový je dotazovací výkon systému, kolik je schopen analyzovat dat. Jednoznačná identifikace každého záznamu z primárních zdrojů není důležitá, zpravidla nebude potřeba analyzovat na takové úrovni detailu (například jeden daňový doklad). Proto jsou často data v datových skladech již předpřipravena za účelem snížit náročnost analýzy. Může jít o částečnou agregaci dat na úrovni nejnižšího detailu, který bude ve výsledné analýze dostupný.

Tabulka 2.1 ukazuje hlavní rozdíly těchto dvou systémů.

Vlastnost	**OLTP**	**OLAP**
Orientace	transakce	analýza
Data	aktuální	historická, integrovaná
Detail	detailní data	agregovaná data
Zaměření	data	informace
Velikost databází	GB	TB
Jednotka operace	jednoduchý dotaz, modifikace	komplexní dotazy
Přístup	čtení/zápis	hodně čtení
Metriky výkonu	průchodnost transakcí	odezva dotazů

Tabulka 2.1: Porovnání hlavních rozdílů systémů OLTP a OLAP (částěčně převzato z [2])

Velikost databází je velice individuální podle konkrétní organizace. Datový sklad může zabírat několik TB, může ale také mít jen několik stovek MB. Většinou však jsou datové sklady velice rozsáhlé, obsahují historická data z mnoha zdrojů. Naopak operační databáze spravují jen “aktuální” data – ta, která jsou potřeba pro momentální chod organizace.

Kapitola 3

Datový sklad pro systém OLAP

William H. Inmon, jeden z významných architektů datových skladů, definoval ve své knize [4] datový sklad takto:

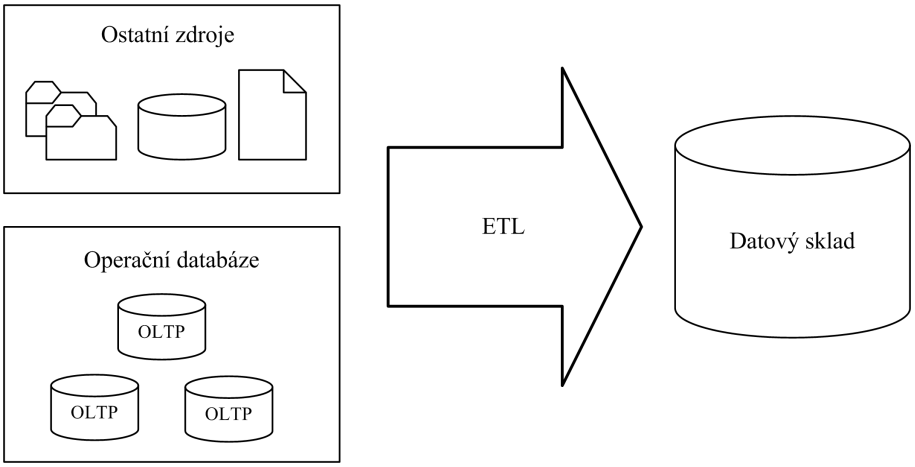
"A data warehouse is a subject-oriented, integrated, nonvolatile, and time-variant collection of data in support of management's decisions."

Tedy: *Datový sklad je soubor subjektově orientovaných, integrovaných, přetrvávajících a časově proměnlivých dat sloužících k podpoře rozhodování.*

Subjektovou orientací je myšlen důraz na význam hlavních předmětů (subjektů) podnikání – jako produkt, zákazník, místo, prodej. Tento fakt se odrazí v návrhu datového skladu.

Datový sklad bývá většinou udržován odděleně od operačních databází. Konsoliduje data z těchto zdrojů do jednoho úložiště a umožňuje komplexní, dlouhodobou analýzu nad daty celé organizace.

Etapa či proces, který se stará o přenesení dat z primárních zdrojů do datového skladu se označuje *ETL – Extrtaction Transformation Loading*, tedy extrakce, transformace a zavedení.



Obrázek 3.1: Datový sklad, proces ETL

3.1 Fáze ETL

Jak již bylo uvedeno v předchozím textu, ETL je fáze, při které se data přesunou z primárních zdrojů do datového skladu. Data v datovém skladu by měla dosahovat vysoké kvalitní a být přichystána pro analýzu. V průběhu tohoto procesu by měly projít jednotlivými, dále uvedenými fázemi, které data připraví a přenesou do finálního úložiště.

Fáze extrakce — Tento proces je zodpovědný za načtení dat z různých datových zdrojů. Těmito zdroji mohou být relační databáze, aplikace poskytující data, datové sešity, data ve formátu XML či textové soubory. Nejčastěji se ale jedná právě o databáze. Protože ne všechny údaje, potřebné v OLTP databázích, je nezbytné uchovávat v datovém skladu, proces extrakce provádí selekci vhodných údajů pro transformaci.

Fáze transformace — V průběhu této fáze se musí data transformovat ze svých vstupních tvarů na formát, který je definován v datovém skladu. Důležitou součástí transformace je čištění dat. V každém jednom zdroji dat mohou být jiné konvence pro uchování hodnot, jejich jednotky, rozsahy, formáty uložení, nejednoznačnosti hodnot. Je potřeba sjednotit textové informace jako jména, adresy, telefoní kontakty, ale také názvy číselníků databáze. Data se musí transformovat na databázové struktury vytvořené v datovém skladu, které jsou přizpůsobeny účelu OLAP analýzy.

Data, která se pokoušíme transformovat mohou být poškozená, mohou obsahovat nesmyslné hodnoty či hodnoty mohou úplně chybět. To může být způsobeno špatným návrhem datových zdrojů, lidskou chybou apod. Tyto případy je nutno detekovat a rozhodnout, zda a jak budou údaje zpracovány. V některých případech lze data dopočítat či odhadnout, jindy je lépe data do finální množiny vůbec nezařadit. Vždy ale záleží na konkrétním případu a je nutno s tímto faktem dopředu počítat.

Často se data ukládají již částečně agregovaná, což snižuje konečné množství údajů a může výrazně urychlit následnou analýzu. V datovém skladu není potřeba z dlouhodobé analýzy uchovávat přílišné, pro analytické zpracování nevýznamné detaily.

Fáze zavedení dat — Tato fáze je poslední fází ETL etapy, kde se data, která jsou vyčištěna, transformována a připravena pro uložení, nahrána do datového skladu.

Celá etapa ETL může být velice časově náročná. Je proto důležité dobře naplánovat a automatizovat jednotlivé kroky a určit periodu, kdy se budou data nahrávat do datového skladu. Tato perioda musí být samozdřejmě delší, než je délka celá fáze ETL a to i s výhledem na budoucí růst objemu dat.

3.2 Datová kostka

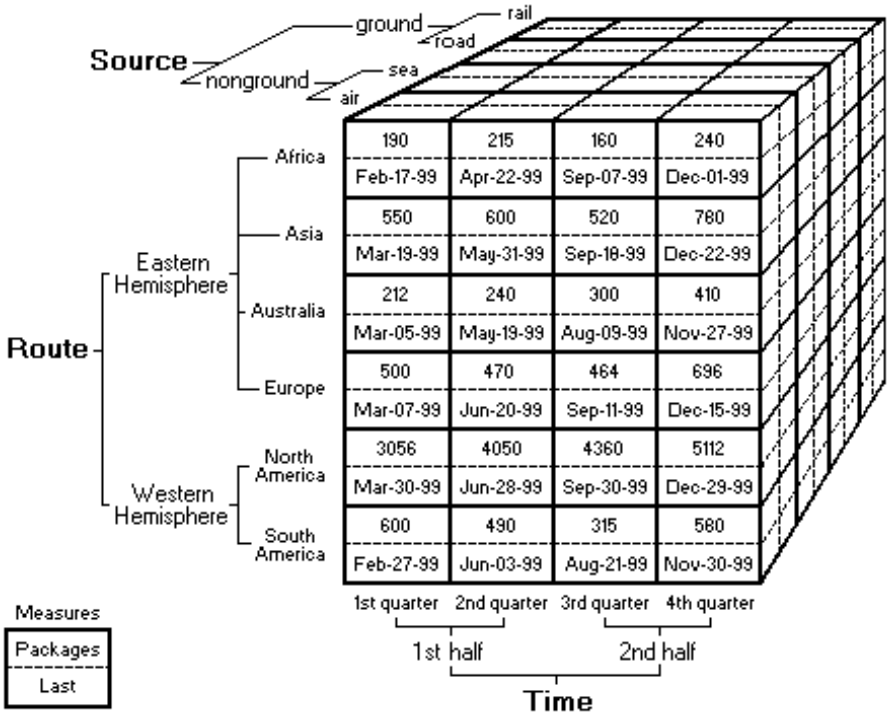
Jak již bylo naznačeno v sekci 2.2, struktura datového skladu se modeluje jako multidimenzionální krychle. Rozlišujeme 2 typy údajů – dimenze a fakta.

Fakta — Fakta jsou numerické údaje, které odpovídají hodnotám sledované veličiny či jevu. Právě agregovaná fakta jsou výsledkem konečné analýzy. Každý jeden údaj (fakt) se vztahuje k určitým hodnotám dimenzí. Dimenze tvoří kontext tohoto kvantitativního údaje – např. prodej železa, 21.2.2008 v Brně je reprezentován hodnotou 30t. Vztaženo k datové kostce, na prusečíku dimenzí produktu, času a místa s uvedenými hodnotami by byl uložen údaj 30t.

Dimenze — Dimenze tedy definují kontext dat. Jsou to většinou textové popisy měrných jednotek. Dimenze může být složena hierarchicky, typickým příkladem je dimenze *místa*, kterou lze členit na státy, územní celky, města, pobočky a podobně.

Počet dimenzí, které se v praxi v datových skladech vyskytují je vyšší než tři (jak uvádí předchozí příklad). Počet se pohybuje přibližně v desítkách.

Na obrázku 3.2 je zobrazena datová kostka se třemi dimenzemi, které všechny jsou hierrchické. Kostka představuje model datového skladu. Manipulací s ní dosáhneme jiného pohledu na data. Můžeme snížit úrověň detailu hierarchických dimenzí nebo dimenze úplně odstranit, pořadí dimenzí lze mezi s sebou měnit, možná je aplikace filtrů. Datová kostka představuje abstrakci, se kterou OLAP analýza pracuje. Slovo abstrakce je na místě, protože data jsou většinou uložena v klasické relační databázi.



Obrázek 3.2: Datová kostka (převzato z `http://biolap.sourceforge.net/`)

3.3 Model datového skladu

Jak bylo uvedeno, datový sklad modelujeme jako multidimenzionální krychli. Data v tomto skladu je však zajisté potřeba uložit.Nejelegantnějším řešením by asi bylo uchovávat data také v multidimenzionálních strukturách a poté takto data i analyzovat. Bohužel tato možnost s sebou nese nevýhody v podobě značné paměťové náročnosti, hlavně ale nejsou podobné systémy dostupné.

Proto se pro uložení dat v datovém skladu nejčastěji používá desítkami let prověřená koncepce relační databáze. Dále budeme diskutovat tuto variantu, její možnosti a přizpůsobení požadavkům datových skladů a OLAP analýzy.

Analytický databázový server bude zpracovávat především velké množství komplexních dotazů nad velkým množstvím dat. Naopak operace modifikace či zápisu budou velmi ojedinělé či nemožné. Předpokládá se, že datový sklad bude provozován mimo operační databáze a jeho modifikace – zápis nových dat bude probíhat v přesně definovaných okamžicích.

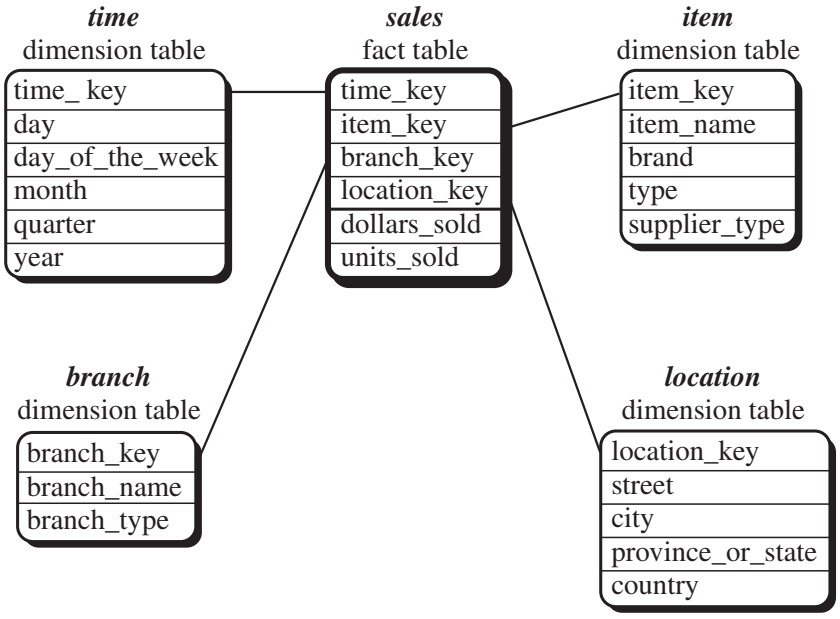
Důraz tedy musí být kladen na výkon při zpracování dotazů. Proto jsou některé postupy dodržované při návrhu schématu relační databáze opomíjené. Jde především o ne úplně striktní *normalizaci tabulek*. Datový sklad obsahuje historická data, modifikace záznamů proto bude spíše vyjímečná. Striktní normalizace umožňující flexibilnější návrh databáze proto není nutná, tabulky jsou často jen v první nebo druhé normální formě. Tento přístup umožňuje rychlejší analýzu uložených dat, protože databázový server bude nucen provádět mnohem menší počet spojení tabulek. Vše je ale potřeba promyslet, vždy je řešení závislé na konkrétní implementaci datového skladu.

Už bylo zmíněno, že rozlišujeme dva typy tabulek – tabulky faktů a tabulky dimenzí. Tyto tabulky spolu tvoří v databázi různá schémata, rozlišjeme tři základní:

- hvězda
- sněhová vločka
- souhvězdí

3.3.1 Schéma hvězda

Toto schéma tvoří jedna tabulka faktů, která je navázána na okolní tabulky dimenzí (obrázek 3.3).



Obrázek 3.3: Databázové schéma hvězda (převzato z [2])

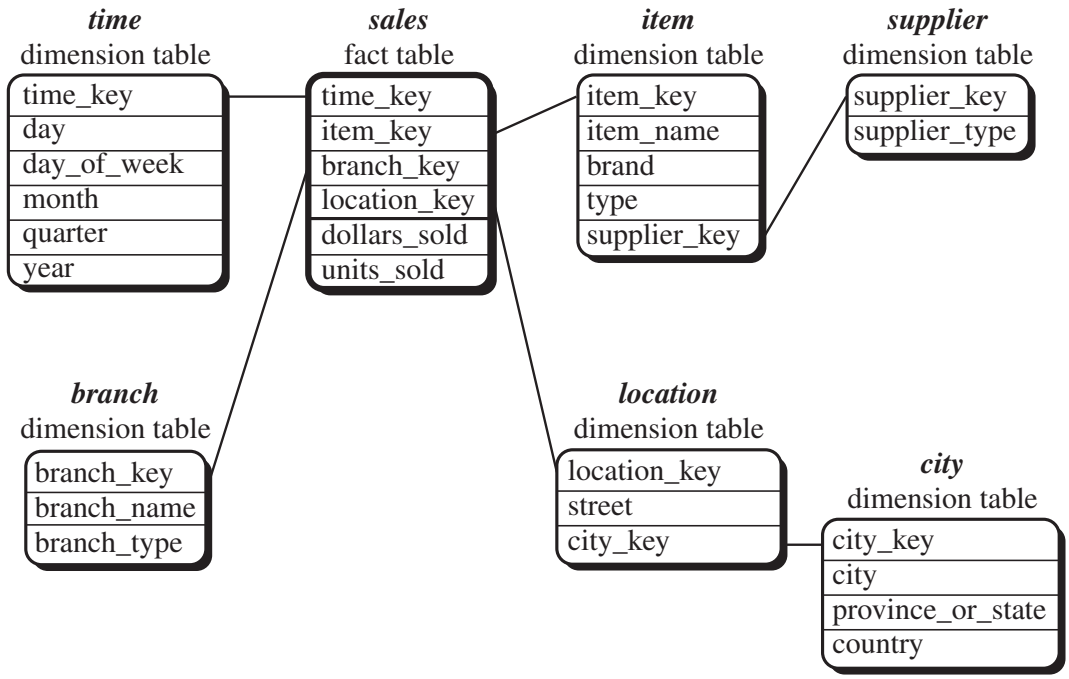
Tyto tabulky dimenzí nejsou normalizované. To má za následek jednodušší návrh a větší dotazovací výkon. Nevýhodou je, že některé informace budou v tabulkách duplicitní a

integrita hodnot nebude vždy automaticky hlídána databázovým serverem. O tuto operaci se musí postarat samotná aplikace, která provádí nahrávání a správu dat, popřípadě je nutno definovat další integritní omezení typu unikátních indexů.

Tento jednoduchý návrh však umožňuje efektivní analýzu dat.

3.3.2 Schéma sněhová vločka

Schéma sněhové vločky se liší od schématu hvězdy tím, že některé nebo všechny tabulky dimenzí jsou rozděleny na více menších procesem normalizace. Obsahuje ale stejně jako předchozí model jednu tabulku faktů navázanou na okolní dimenze (obrázek 3.4). Rozdělení tabulek dimenzí vlastně naznačuje možnou hierarchii, která lze u těchto dimenzí definovat.



Obrázek 3.4: Databázové schéma sněhová vločka (převzato z [2])

Výhodou je lepší správa dat, protože informace je uložena v databázi pouze jednou. Při analýze bude ale potřeba vykonávat větší množství spojení tabulek, které může snížit výkon systému.

Schéma souhvězdí je nejsložitější a jde o kombinací obou předchozích. Model databáze obsahuje více tabulek faktů, které sdílí některé dimenze. Tyto tabulky dimenzí mohou být normalizované a spolu s tabulky faktů tvoří tvar připomínající souhvězdí.

Při návrhu databáze je nutno posuzovat klady a zápory jednotlivých řešení individuálně. Schéma hvězdy a vločky jsou mezi sebou vzájemně poměrně lehce převoditelné. Společným rysem všech modelů je to, že tabulky faktů obsahují cizí klíče, které tvoří odkazy do tabulek dimenzí. Takto se definuje důležitý kontext těchto faktů – numerických jednotek, které se budou při analýze agregovat.

Tabulky faktů jsou v datových skladech zpravidla největší a to mnohonásobně oproti tabulkám dimenzí. Tabulky dimenzí slouží jako číselníky, které známe z tradičního návrhu relačních databází.

Kapitola 4

OLAP analýza a prezentace dat

Tato kapitola se zabývá samotnou datovou analýzou. Ta se skládá z manipulace s datovým modelem – multidimenzionální kostkou, jejího výpočtu a zobrazením. Pro tuto manupulaci jsou definovány čtyři základní operace – *roll-up, drill-down, slice&dice, pivoting*. Ty umožňují dosáhnout jakéhokoliv možného pohledu na data uchovávaná v datovém skladu.

4.1 Olap operace

Konečný uživatel, aniž by si to často uvědomoval, provádí v průběhu analýzy manipulaci s datovou kostkou. Při tomto procesu používá přirozené pojmy jako přidání či odebrání dimenze (kontextu podle kterého se data agregují), snížení či zvýšení detailu nebo filtrování dat podle zvolené podmínky.

Účelem OLAP analýzy je dát uživateli volnost v možnostech, jak se na data dívat. Je to opačný přístup než u klasických reportů či sestav generovaných z databází, kde je většina parametrů pevně dána a mění se například jen čas.

Každá jedna operace je transformací mezi dvěma tvary datové kostky. Operace se mohou provádět opakovaně, tím lze docílit všech možných tvarů datové krychle.

4.1.1 Roll-up

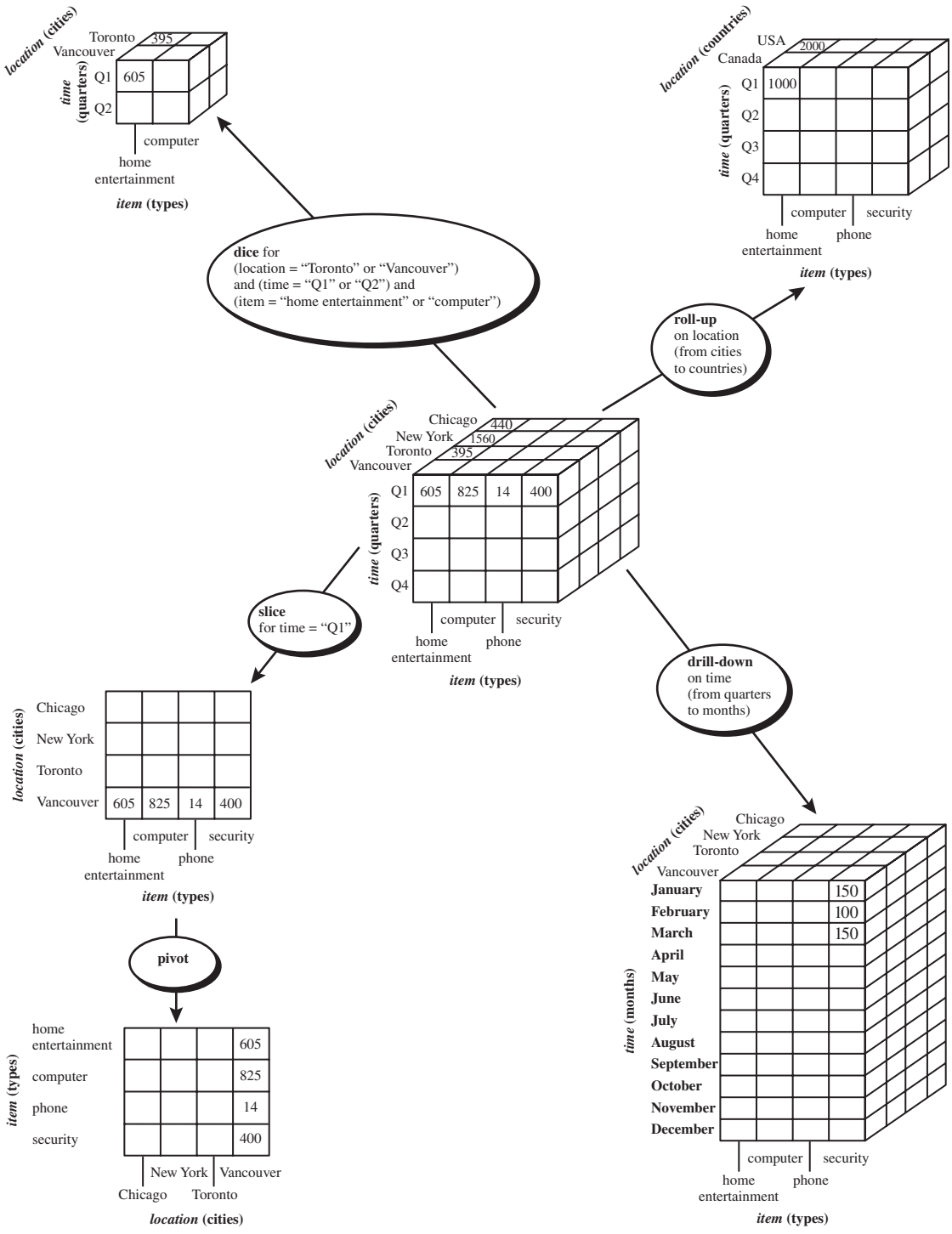
Operací roll-up neboli vyrolování rozumíme snížení detailu a tím souvsející zvýšení agregace hodnot faktů. Za toto snížení detailu lze považovat posun v hierarchii dimenze směrem výše (k nadřazeným stupňům) či úplné vyloučení dimenze z analýzy.

Lze dokonce docílit i faktu, že do výsledné analýzy nebude zahrnuta žádná z dimenzí definovaných v datovém skladu. V tom případě je výsledkem jedna agregovaná hodnota (případně hodnot více, pokud je jich více v tabulce faktů), která se vypočtena bez ohledu na hodnoty příslušných dimenzí.

Na obrázku 4.1 je vidět příklad operace roll-up, kdy se redukuje hierarchie dimenze *location* z hodnot měst na státy, které jsou nadřazeným stupněm. Datová krychle se zmenší, obsahuje nižší počet agregovaných údajů, které však nabývají vyšších hodnot – hodnoty pro města se sloučily do hodnot pro státy.

4.1.2 Drill-down

Operace drill-down je inverzní operací k roll-up. Provádí zvýšení detailu a snížení agregace výsledných hodnot faktů. Ty se rozdělí na více menších, protože do analýzy vstoupí další



Obrázek 4.1: Přehled OLAP operací (převzato z [2])

dimenze – kontext, který je nutno začlenit.

Může jít o přidání dimenze či posun v hierarchii některé dimenze směrem dolů (k podřazeným stupňům).

Na obrázku 4.1 je operace drill-down demonstrována u dimenze *time*, kdy se zvýší detail

z hodnot čtvrtletí na měsíce, které jsou podřazeným stupněm. Datová krychle se zvětší, obsahuje vyšší počet agregovaných údajů, které však nabývají nižších hodnot – hodnoty čtvrtletí se rozpadly na více měsíců.

4.1.3 Slice & Dice

Operace slice & dice neboli selekce a projekce znamená výběr určité části analyzované datové kostky. Jinými slovy jde o aplikaci filtrů, které jsou složeny z podmínek kladených na hodnoty vybraných dimenzí.

Na obrázku 4.1 jsou uvedeny dva příklady. První ukazuje výběr dat podle hodnoty dimenze času, konkrétně hodnoty prvního čtvrtletí. Ostatní části roku do výsledné krychle nejsou zařazeny. Druhým příkladem je aplikace složitějšího filtru na základě hodnot dimenzí místa, času i položky. Výsledkem je podkrychle, splňující uvedený filter.

4.1.4 Pivoting

Operací pivoting lze měnit natočení krychle, jejích hran. Tyto přetočení ovlivní pořadí dimenzí, podle kterých se počítají agregační honoty. Jednou lze mít agregované výpočty podle místa a poté podle času, po přetočení (čas, místo) budou napřed data agregována podle času a teprve poté se rozdělí podle místa.

Na obrázku 4.1 je ukázáno přetočení os (dimenzí) místa a položky (druhu zboží).

4.2 Olap server

Olap server je samotné jádro systému, které vykonává datovou analýzu. Server komunikuje s datovým skladem, získává data, ty zpracovává a poskytuje prezentační vrstvě k vizualizaci. Server zpracovává data na základě požadavků od uživatelů, kteří definují, jaká data si přejí a jakým způsobem je chtějí zobrazit. Uživatel definuje tvary datových kostek, které OLAP server počítá.

Na obrázku 4.2 je zobrazen OLAP server ve struktuře celého systému a je naznačen jeden možný způsob, jak rozdělit systém podle jednotlivých vrstev:

- datová vrstva
- aplikační vrstva
- prezentační vrstva

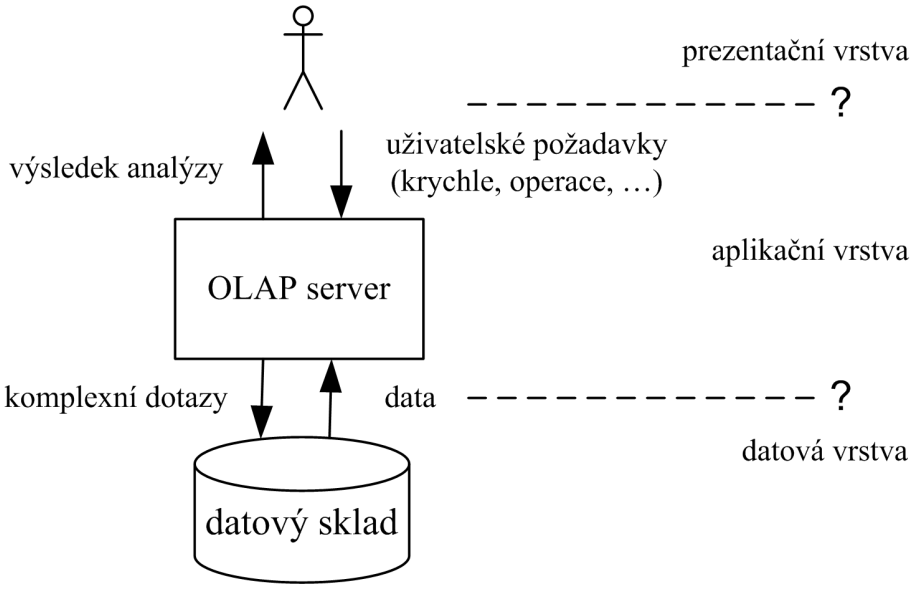
Naznačené členění jednotlivých vrstev na obrázku 4.2 se může v mnoha případech lišit. Rozsah prezentační vrstvy velice záleží na možnostech klientské aplikace, která zobrazuje finální podobu analýzy. V případě tzv. “tlustých klinetů”, mohou tyto aplikace obsahovat také velké množství aplikační logiky. Mohou také generovat náročnější grafické výstupy. Nevýhodou ovšem je decentralizovaná správa takového systému.

Čím dál tím více se prosazuje řešení s použitím “tenkých klientů” typu internetového prohlížeče. V tomto případě odpadá náročná správa klientských stanic a aplikační logika může být udržována na serveru. Nevýhodou je omezení dané možnostmi zobrazovacích schopností těchto aplikací. Výhodou je možnost přesunutí části logiky prezentační vrstvy na server, kde se generuje tvar výsledných webových stránek.

Stejně je tomu i u části datové. Často nemusí být všechny výpočty realizovány pouze aplikačním serverem, jako aplikací nadřazenou datovému skladu. Velkou část analýzy může

provádět již samotný databázový stroj, např. zpracování komplexních dotazů nebo výpočty realizované pomocí úložných procedur či programů integrovaných do tohoto serveru.

Hranice mezi jednotlivými částmi je tedy značně volná, v konečném řešení se ale všechny tři vrstvy vyskytují.



Obrázek 4.2: OLAP server

4.2.1 Typy OLAP serverů

V praxi se můžeme setkat s několika odlišnými přístupy k realizaci OLAP systémů. Rozdíl spočívá ve strategii manipulace s daty. Rozlišujeme:

- ROLAP – relační OLAP
- MOLAP – multidimenzionální OLAP
- HOLAP – hybridní OLAP (kombinace obou předchozích)

ROLAP architektura uchovává data v relačních databázích a OLAP server tvoří vrstvu mezi uživatelem a touto databází. Data se zpracovávají pomocí dotazů na databázový server.

MOLAP systém je založený na uchovávání dat v multidimenzionálních datových strukturách. Datové kostky se snaží uchovávat v paměti. Výhodou je rychlost, nevýhodou náročnost na uložení dat.

HOLAP architektura kombinuje výhody obou předchozích. Data spravuje ve formě relační databáze, avšak OLAP server se snaží uchovávat agregovaná data také v přechodných struktutách v paměti. Tyto již vypočítané tvary datových kostek je poté schopen využít pro analýzu dalších požadavků.

V důsledku snadné dostupnosti relačních databázových serverů, nárokům a požadavkům na vlastní implementaci OLAP sytému, tak se dále zmiňuji jen o možnostech této realizace.

4.3 Vizualizace dat uživateli

Data získaná během analýzy je nutno zobrazit uživateli a to pokud možno v co nejintuitivnější podobě. Mezi nějčastěji používané možnosti patří tabulky a grafy. Obě tyto možnosti jsou dále popsány.

Při vizualizaci dat musíme mít na mysli, že provádíme projekci multidimenzionální datové struktury na nejčstěji 2D průmětnu (zařízení jako monitor, tiskárna, projektor, ...). Musíme tedy zařídit tuto transformaci bez přílišné ztráty přehlednosti.

4.3.1 Pivot tabulky

Pivot tabulky umožňují zobrazit agregovaná data jako průsečíky dimenzí, které jsou umístěny na svislou i vodorovnou osu. Protože potřebujeme zobrazit více dimenzí než dvě (a máme jen dvě osy tabulky), dají se jednotlivé dimenze strukturovat do hierarchií. Dimenze lze přesouvat mezi osami, vzájemně zaměňovat, přesouvat, přidávat či odebírat.

Příkladem této tabulky je implementace ve formě *kontingenční tabulky* v aplikaci Microsoft Excel. Příklad je uveden na obrázku 4.3. Tabulka zobrazuje údaje o průměrném počtu dětí v závislosti na rodinném stavu, pohlaví a věku. Jako agregační funkce byl použit průměr. Svislou osu tvoří dvě dimenze (stav a pohlaví). Osu vodorovnou dimenze věková kategorie. Na průsečíku jednotlivých hodnot můžeme vidět agregované hodnoty.

Průměr z počet		věk.kategorie			
stav	pohlaví	30-50 let	do 30 let	nad 50 let	Celkový průměr
rozvedená(ý)	muž	2,0		2,0	2,0
	žena	2,0	2,0	3,0	2,3
Celkem z rozvedená(ý)		2,0	2,0	2,5	2,2
svobodná(ý)	muž	0,5	0,3		0,4
	žena	1,5	0,0		1,2
Celkem z svobodná(ý)		1,2	0,3		0,7
vdaná-ženatý	muž		1,0	2,4	2,0
	žena	2,5	2,0		2,4
Celkem z vdaná-ženatý		2,5	1,3	2,4	2,2
vdova(ec)	muž	2,0		1,5	1,7
	žena			2,0	2,0
Celkem z vdova(ec)		2,0		1,8	1,8
Celkový průměr		1,8	0,7	2,2	1,6

Obrázek 4.3: Kontingenční tabulka (Microsoft Excel)

Volitelně mohou být uvedené průběžné i konečné agregace za jednotlivé dimenze a celou tabulku. Tyto údaje rozšiřují celkový pohled na data, lze je ovšem i vynechat – uživatel má vždy možnost se k těmto údajům pomocí OLAP operací dostat.

Jinou implementací je tabulka získaná jako výstup ze systému Mondrian, což je OLAP server, který je dále popsán v sekci 5.2.

Výhodou pivot tabulek je poměrně intuitivní zobrazení hodnot dat, nevýhodou možnost zobrazení pouze jednoho faktu na průsečíku hodnot dimenzí[1].

[1] Při více faktech by se musela tabulka modifikovat, průsečíky by se museli rozdělit pro zobrazení více hodnot. Jinou možností by byl např. explicitní výběr jednoho faktu ze seznamu všech možných .

		Measures		
		Profit		
		Education Level		
Product	Customers	Bachelors Degree	Graduate Degree	High School Degree
−All Products	+Long Beach	$1 348,03	$231,14	$800,48
	+Los Angeles	$552,77	$151,71	$701,43
	+Oregon City	$1 202,01	$216,38	$1 876,41
	+Portland	$1 176,49	$211,84	$1 350,88
+Drink	+Long Beach	$124,36	$34,24	$56,75
	+Los Angeles	$49,37	$6,07	$58,07
	+Oregon City	$80,52	$6,99	$172,03
	+Portland	$101,32	$13,84	$205,77
+Food	+Long Beach	$947,30	$145,02	$602,53
	+Los Angeles	$371,09	$110,25	$529,13
	+Oregon City	$885,53	$189,01	$1 332,18
	+Portland	$834,90	$159,26	$921,92
+Non-Consumable	+Long Beach	$276,37	$51,88	$141,20
	+Los Angeles	$132,31	$35,40	$114,23
	+Oregon City	$235,96	$20,39	$372,21
	+Portland	$240,27	$38,74	$223,20

Obrázek 4.4: Pivot tabulka - webové rozhraní OLAP serveru Mondrian [6]

4.3.2 Relační tabulky

Relační tabulka je dalším způsobem vizualizace multidimenzionálních dat. Obsahuje dva typy sloupců – sloupce hodnot dimení a sloupce hodnot faktů. Dohromady tvoří uspořádanou n-tici. Právě na pořadí sloupců dimenzí záleží – definují pořadí agregace faktů. Tabulka může obsahovat také částečné součty, které dodávají komplexnější informaci. Není to ale nezbytně nutné, tyto údaje lze zobrazit příslušnou změnou tvaru datové kostky. Jednoduchým příkladem může být obrázek 4.5.

Za výhodu můžeme považovat možnost zařadit do tabulky více sloupců s hodnotami faktů. Nevýhodou může být relativně velký počet řádků, které mohou vzniknout při analýze, do které je zahrnuto mnoho dimenzí.

4.3.3 Grafy

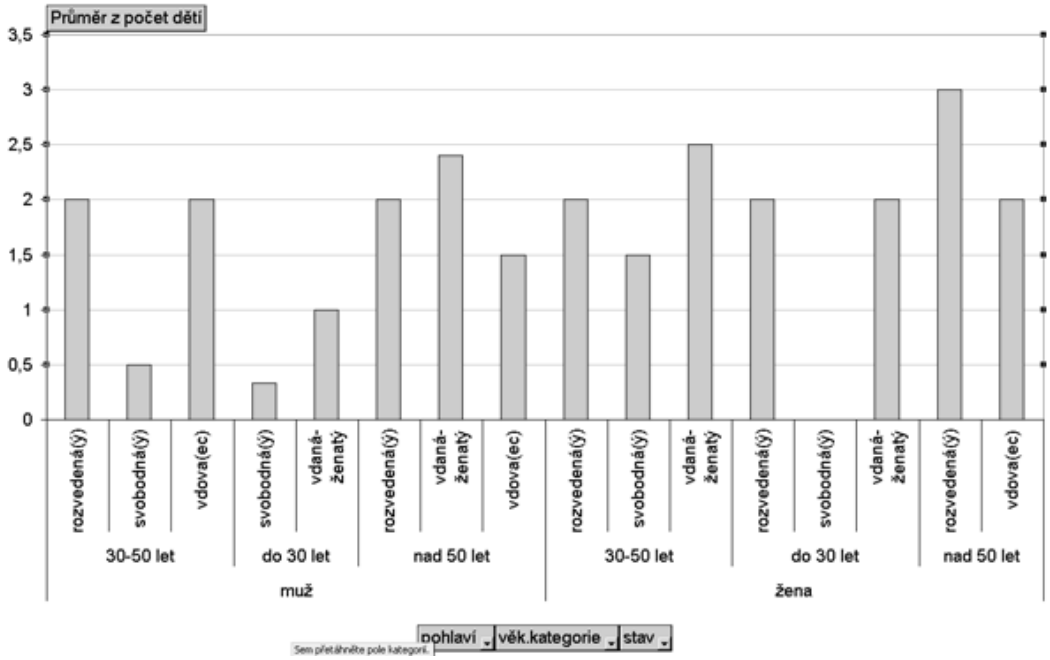
Prezentace OLAP analýzy uživateli formou grafu může být v určitých situacích vhodnější pro celkové pochopení rozložení dat v analyzovaném souboru – zejména při první vizualizaci výsledků. Poté je vhodné zobrazit i detail analýzy, např. jako nějakou formu tabulky s konkrétními údaji.

V grafu jsme ale limitování jen jednou osou pro zobrazení dimenzí, druhá slouží jako osa pro vynášení hodnot. Proto musíme volit různé filtry či hierarchii dimenzí. Lze použít i například sloupcové grafy složené z více částí, které odpovídají podřízené dimenzi. Obecně lze říci, že použití grafů při OLAP analýze má své limity a je vhodné je použít jen pro vizualizace analýz s malým množstvím dimenzí, jinak se mohou stát grafy hůře čitelné.

Jako příklad je zobrazen kontingenční graf 4.6, kde je použita hierarchická dimenze na ose x.

čas	město	produkt	celkem Kč
2005	Brno	software	205 456
2005	Brno	hardware	100 875
2005	Brno	služby	50 900
2005	**Brno**		**357 231**
2005	Olomouc	software	300 500
2005	Olomouc	hardware	150 654
2005	Olomouc	služby	20 800
2005	**Olomouc**		**471 954**
2005			**829 185**
2006	Brno	software	400 867
2006	Brno	hardware	300 948
2006	Brno	služby	100 900
2006	**Brno**		**802 715**
2006	Olomouc	software	350 006
2006	Olomouc	hardware	200 980
2006	Olomouc	služby	89 050
2006	Olomouc		640 036
2006			**1 442 751**
celkem			**2 271 936**

Obrázek 4.5: Relační tabulka



Obrázek 4.6: Kontingenční graf (Microsoft Excel)

Kapitola 5

Existující OLAP řešení

Trh s technologiemi pro podporu rozhodování se neustále rozvíjí a stále více firem se snaží své produkty o tyto technologie obohatit. Mnoho výrobců databázových serverů jako Oracle, IBM nebo Microsoft se snaží vyvíjet příbuzné aplikace spolupracující s databázovým serverem či do něj přímo zabudované, které by umožnily hlubší analýzu dat. *Business intelligence* se stal velice rozšířeným (marketingovým) pojmem, který v sobě skrývá mnoho technologií. Nikdo z velkých firem nechce být pozadu a s každou novou verzí přichází výrazná vylepšení.

Pro detailnější pohled a porovnání byly vybrány dva na první pohled velice odlišné produkty. Prvním se balík aplikací společnosti Microsoft postavený na databázovém serveru *Microsoft SQL Server 2005*. Druhým produktem je zdarma dostupný *OLAP server Mondrian*, který je co do své velikosti mnohem menší, specifičtěji zaměřený projekt.

5.1 Microsoft OLAP technologie

Microsoft se v posledních letech zařadil mezi zavedené firmy na trhu databázových technologií a snaží se nabízet ucelené řešení v oblasti BI. Tato platforma je velkým balíkem nástrojů. Základ tvoří databázový server Microsoft SQL Server ve verzi 2005. A není to jen klasický relační databázový server. Microsoft klade velký důraz na doplňující analytické služby, na efektivní lehce použitelné nástroje pro administraci samotného SQL serveru a v neposlední řadě na nástroje určené pro vývoj aplikací na této platformě.

5.1.1 Architektura

Tento server poskytuje také podporu pro OLAP analýzu. Podmínkou je instalace analytických služeb společně s databázovým serverem. Poté lze definovat zdroje dat, tyto data transformovat a nad nimi budovat datové kostky.

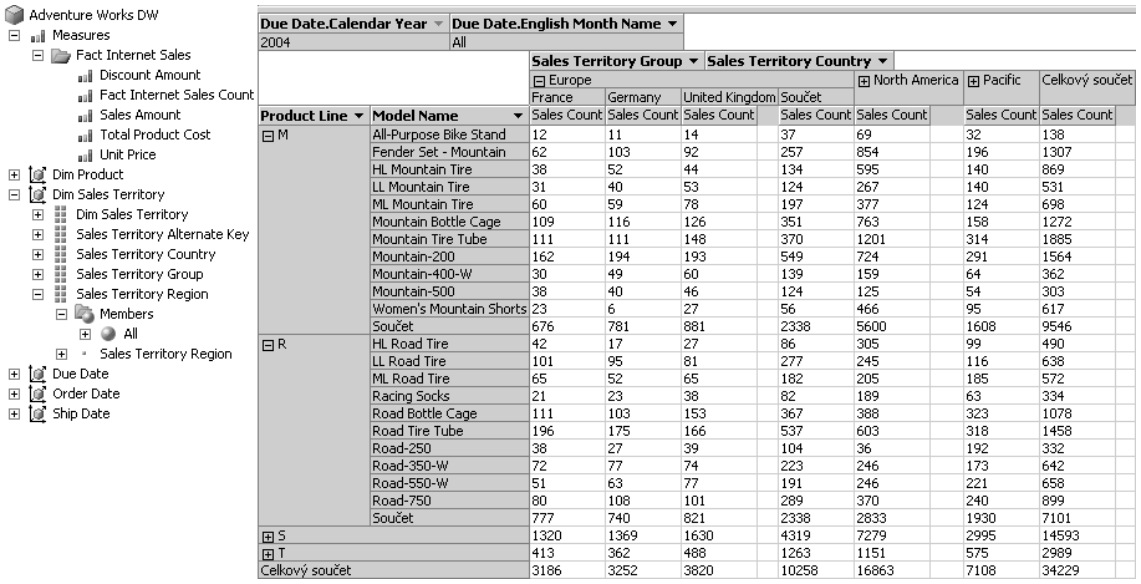
SQL Server 2005 využívá hybridní technologii HOLAP pro uložení a analýzu dat. Jedná se o model založený na relačním ROLAPu, který si ukládá vhodné agregace pro zrychlení výpočtů. Analyzovat lze data, která spravuje samotný MS SQL Server nebo se lze připojit k externím zdrojům pomocí nativních či ODBC rozhraní.

5.1.2 Práce s datovou kostkou

Součástí instalace je i nástroj *SQL Server Business Intelligence Development Studio*. Ovládání je velmi podobné klasickému programu pro vývoj Visual Studio (na platformě Windows,

.NET). Po založení nového analytického projektu máme k dispozici nástroje pro všechny fáze analýzy OLAP.

Definujeme zdroj dat, model na úrovni relačním tabulek, samotnou datovou kostku. Všechny fáze jsou usnadněny pomocí intuitivních průvodců, pomocí kterých provedeme krok za krokem definici datového modelu – kostky. Lehce můžeme definovat jednotlivé dimenze, jejich hierarchie a strukturu. Výsledkem je schéma hvězdy či sněhové vločky.



Due Date.Calendar Year	Due Date.English Month Name							
2004	All							
		Sales Territory Group	Europe				North America	Pacific	Celkový součet
		Sales Territory Country	France	Germany	United Kingdom	Součet			
Product Line	Model Name		Sales Count	Sales Count	Sales Count	Sales Count	Sales Count	Sales Count	Sales Count
M	All-Purpose Bike Stand		12	11	14	37	69	32	138
	Fender Set - Mountain		62	103	92	257	854	196	1307
	HL Mountain Tire		38	52	44	134	595	140	869
	LL Mountain Tire		31	40	53	124	267	140	531
	ML Mountain Tire		60	59	78	197	377	124	698
	Mountain Bottle Cage		109	116	126	351	763	158	1272
	Mountain Tire Tube		111	111	148	370	1201	314	1885
	Mountain-200		162	194	193	549	724	291	1564
	Mountain-400-W		30	49	60	139	159	64	362
	Mountain-500		38	40	46	124	125	54	303
	Women's Mountain Shorts		23	6	27	56	466	95	617
	Součet		676	781	881	2338	5600	1608	9546
R	HL Road Tire		42	17	27	86	305	99	490
	LL Road Tire		101	95	81	277	245	116	638
	ML Road Tire		65	52	65	182	205	185	572
	Racing Socks		21	23	38	82	189	63	334
	Road Bottle Cage		111	103	153	367	388	323	1078
	Road Tire Tube		196	175	166	537	603	318	1458
	Road-250		38	27	39	104	36	192	332
	Road-350-W		72	77	74	223	246	173	642
	Road-550-W		51	63	77	191	246	221	658
	Road-750		80	108	101	289	370	240	899
	Součet		777	740	821	2338	2833	1930	7101
S			1320	1369	1630	4319	7279	2995	14593
T			413	362	488	1263	1151	575	2989
Celkový součet			3186	3252	3820	10258	16863	7108	34229

Obrázek 5.1: Microsoft BI Development Studio - OLAP analýza

Pro samotnou analýzu je dostupný nástroj Browser (obrázek 5.1), který umožní zobrazit vybranou (část) kostky a umožní provádět typické OLAP operace. Zobrazení informace je velmi podobné kontingenční tabulce z programu Microsoft Excel. Do jednotlivých částí této tabulky můžeme pouhým přetažením vkládat jednotlivé dimenze a fakta, která nás zajímají. Tyto dimenze lze řadit a tvořit hierarchie. Lze aplikovat filtry s podmínkami, které musí zobrazená data splňovat.

Na databázové a analytické služby navazuje mnoho dalších nástrojů. Výhodou je snadná spolupráce a komunikace, na kterou jsou aplikace připraveny. Jedná se např. o reportovací služby s možností tvorby graficky bohatých sestav nebo pokročilé vývojové nástroje (rodina aplikací Microsoft Visual Studio), které umožňují efektivní vývoj moderních aplikací především na platformě .NET. Jde o obrovskou paletu nástrojů, které lze využít pro realizaci kompletního systému pro analýzu dat.

Datové zdroje poskytované MS SQL Serverem lze však využívat v mnoha jiných nástrojích (především) firmy Microsoft, pro běžné uživatele budou zajímavé především aplikace kancelářského balíku Office. Microsoft SQL Server (a další aplikace) je kompletní platforma pro analýzu multidimenzionálních dat s propracovaným uživatelským rozhraním a možností spolupráce mnoha jiných aplikací (vývojová prostředí, kancelářské programy, . . .).

5.2 Mondrian

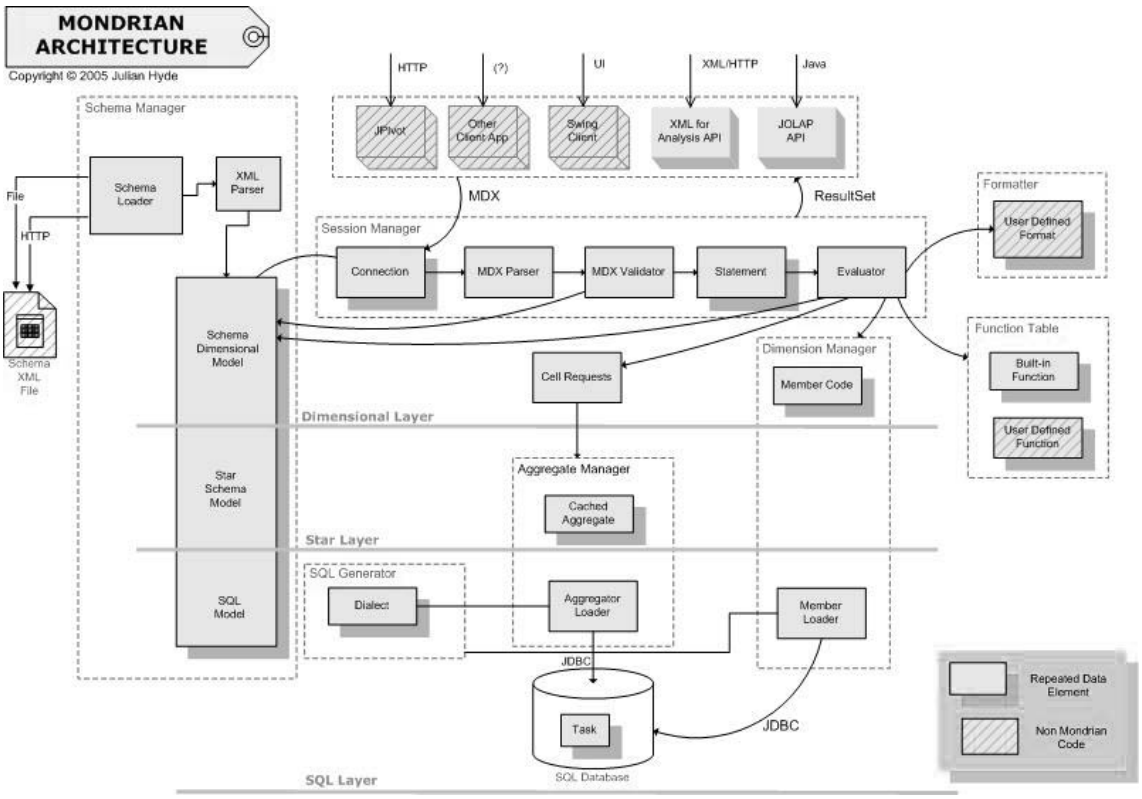
Mondrian projekt je výsledkem aktivity organizace *Pentaho* [6] zavývající se oblastí business intelligence. Jde o sadu aplikací imožňující provádět OLAP analýzu nad rozsáhlými kolekcemi dat. Systém je dostupný zdarma jako open source.

5.2.1 Architektura

Celý projekt Mondrian je postaven na technologii Java. Běh tohoto systému je tedy umožněn na všech strojích, které podporují tuto technologii.

Server je koncipován pro práci nad relačními databázemi. Není omezen pro jednu konkrétní, naopak nativně podporuje velký počet databázových serverů. Kdyby přesto nebyl požadovaný server dostupný přes nativní datové rozhraní, lze jen připojit přes ODBC. Protože ODBC rozhraní podporuje naprostá většina různorodých datových zdrojů, je řešení dostatečně flexibilní.

Řešení projektu Mondrian je podstatně jednodušší než je tomu u MS SQL 2005. Nenajdeme zde žádné průvodce pro import a čistění dat, pro definici zdrojů atd. Všechny nezbytné kroky a procesy přípravy dat je nutné provést vlastní režii. Práce není tak intuitivní, předpokládá se určitá technická znalost. Všechna nastavení serveru je nutné definovat v konfiguračním souborech XML (přesnou definici zdrojů, operací, chování systému, ...).



Obrázek 5.2: Architektura systému Mondrian (získáno z [6])

Na obrázku (5.2) jsou zobrazeny jednotlivé části projektu Mondrian. Jsou definovány jednotlivé vrstvy a moduly aplikace, zodpovědné za konkrétní činnosti. Obecně je definován

také způsob komunikace s tímto serverem a prezentace dat (HTTP - JPivot, Swing client, ...).

5.2.2 Práce s datovou kostkou

Definice datových kostek jsou realizovány v podobě konfiguračních souborů. Je nutno definovat dimenze, jejich hierarchie a fakta. Tato struktura představuje datovou kostku reprezentující část reality, kterou modelují analyzovaná data. Nad touto kostkou se poté provádí OLAP analýza. Definice nastavení v konfiguračních souborech a absence průvodců může být ze začátku překážkou, ale po jejím zvládnutí lze plně využít tohoto flexibilního řešení.

Samotná analýza (dostupná například přes webové rozhraní) je ale naopak velice jednoduchá a intuitivní. Provádět analýzu může uživatel jen s pomocí klasického webového prohlížeče.

Server obsluhuje požadavky klientů v podobě dotazů v jazyku MDX (viz obrázek 5.2). MDX (Multidimenzional Expresions) je jazyk dotazování nad multidimenzionálními datovými strukturami. Jde o ekvivalent jazyka SQL pro relační databáze. Je mu velice podobný a obsahuje rozšíření právě o práci s multidimenzionálním daty. Tímto se definuje standardní rozhraní pro komunikaci (podobně jako SQL u rel. databází), které lze využít pro přístup k datům ze všech druhů aplikací.

Na základě zpracování dotazu v jazyce MDX je generován kód SQL, pomocí kterého jsou získána data z datových zdrojů (které leží mimo samotný server). Tyto data jsou poté zpracována do výsedné podoby a vrácena klientovi ve formátu XML.

Pro vizualizaci analýzy bylo vybráno WWW uživatelské rozhraní. To slouží pro manipulaci s daty a zobrazení výsledků analýzy. Instalace je poměrně jednoduchá, jako webový a aplikační server byl použit *Tomcat*[1]. Pro zobrazení výsledků analýzy se využívá komponenta JPivot, která umožňuje provádět OLAP operace a zobrazit výsledek.

Ukázka webového rozhraní pro přístup k datům použité u projektu Mondrian je na obrázku 5.3.

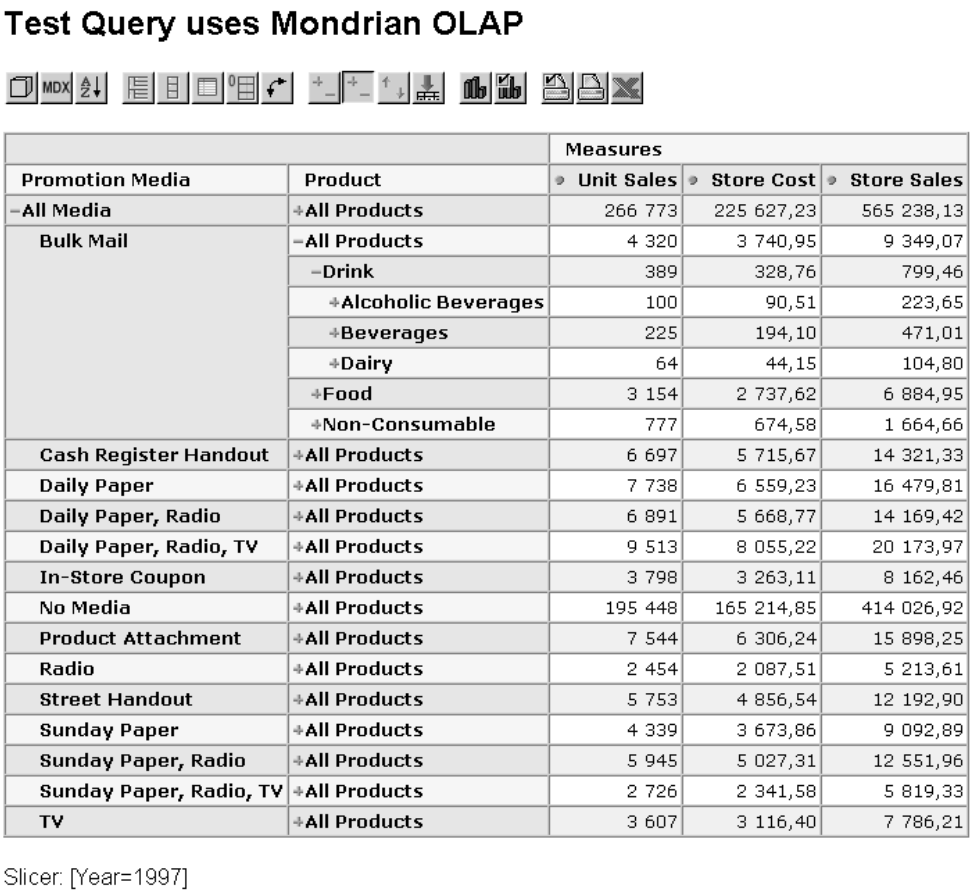
5.3 Shrnutí

Každý z porovnávaných nástrojů si klade jiné cíle především v rozsahu poskytovaných služeb a komplexnosti. Platforma MS SQL Server je propracované komplexní řešení, soubor aplikací pro uložení, správu a analýzu dat. Mondrian je mnohem menší specifičtější projekt, který umožňuje OLAP analýzu nad již exitujícími relačními databázemi.

U produktu MS SQL Server lze očekávat další vylepšení v následujících verzích. Microsoft se poslední dobou stále více prosazuje na poli databázových serverů, z čehož bude i řešení BI postavené na této platformě profitovat.

MS SQL Server je řešení postavené na principu "vše v jednom". Umožňuje i začátečníkům snadnou práci pomocí průvodců a propracovaného uživatelského rozhraní. Navíc nabízí vynikající podporu, dokumentaci a vývojové nástroje, které spolu dokáží úzce spolupracovat a tudíž není potřeba řešit různé problémy v kompatibilitě. Licence na tento produkt jsou relativně drahé, ale odpovídají kvalitě dodávaného systému. Řešení OLAP provozované na MS SQL Serveru zvolí pravděpodobně větší firmy či instituce, které chtějí svá data nejen spravovat, ale získat z nich i vzácné informace a znalosti. Nevýhodou je svázanost s platformou Windows.

[1] Tomcat je aplikační server dostupný zdarma, který podporuje technologie aplikací Java EE.



Obrázek 5.3: WWW rozhraní – Mondrian, JPivot

Naopak Mondrian může být použit pro menší projekty či řešení, kde je potřeba OLAP analýza dat nad již existujícími databázemi. Umožňuje snadnou instalaci a správu. Samotná velikost instalace je cca 20 MB, aplikace je distribuována jako webový archiv (war), který je nahrán na aplikační server.

Uživatelský přístup je možno implementovat mnoha způsoby, implicitní možností je webové rozhraní, které poskytuje dostatečně intuitivní ovládání i prezentaci dat OLAP analýzy. Server však podporuje mnoho dalších a nic nebrání i implementaci vlastní prezentační vrstvy, která bude komunikovat se samotným jádrem OLAP serveru zpracovávajícího dotazy.

Mondrian server je dostupný zdarma i se zdrojovými kódy a je možné se podílet na jeho dalším vývoji. Není ani vázán na konkrétní operační systém, jen potřebuje ke svému běhu podporu běhového prostředí Java.

Kapitola 6

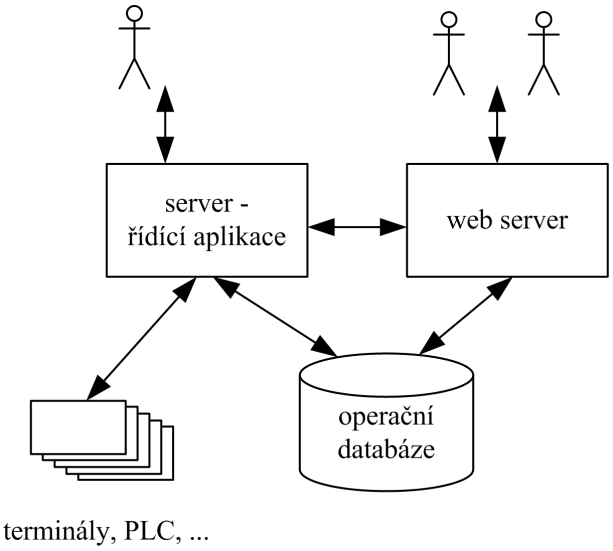
Návrh vlastního OLAP řešení

Cílem projektu bylo vyvinout vlastní řešení systému, který by sloužil pro OLAP analýzu dat. Projekt byl vypracován ve spolupráci se společností Elektlabs s.r.o., která se věnuje podnikání v oblasti sběru dat a řízení provozu v průmyslových podnicích.

Projekt tedy rozšiřuje tento stávající systém a slouží pro analýzu dat uložených v operační databázi.

6.1 Stávající řešení

Na obrázku 6.1 je zjednodušená architektura systému obsahující důležité části pro analázu OLAP.



Obrázek 6.1: Zjednodušená architektura stávajícího řešení sběru dat a řízení provozu

Základem celého systému je relační operační databáze. Řídící aplikace komunikuje s terminály a moduly PLC, monitoruje a řídí jejich provoz a zaznamenává naměřěná data do databáze. Webový server je používán k realizaci uživatelského přístupu k datům – webové formuláře, statistické přehledy, grafy, reporty, tiskové sestavy, ...

6.1.1 Technologie

Tato část obsahuje základní informace o technologii, na které je celý systém postaven. Tyto informace jsou důležité pro návrh a realizaci vlastního řešení.

Nejčastěji využívanou platformou je Windows, některé části systému však mohou alternativně běžět také pod systémy Linux.

Jako databázový server je použit *Firebird*, aktuálně ve verzi 2.1. Jde o moderní relační databázový server vyvinutý z projektu Interbabase jako vývojový klon, který se šíří jako open source. Vyniká hlavně jednoduchou správou a škálovatelností. Je vhodný jak pro malé databázové aplikace, tak pro správu databází o velikosti stovek GB. Složitější funkcionalitu je možno realizovat prostředncvím úložných procedur (rychlost, procedurální SQL), popřípadě externích knihoven (UDF – obecné algoritmy). Databázový server využívá vestavěné moduly (UDF knihovny – nativní win32 kód), které částečně omezují možnost portace na jiné systémy. Tyto knihovny však obsahují funkcionalitu, která bez které se řešení v současnosti neobejde.

Jako webový server je používán *Apache* (verze 2.2). WWW rozhraní je realizováno pomocí dvou technologií – *PHP* a *ASP.NET*. Je tedy zajištěna podpora pro běh aplikací postavených na platformě .NET. V budoucnu se počítá s nahrazením PHP stránek novějšími, postavenými na technologii ASP.NET. Tento přístup přináší výhody ve snadnějším vývoji, ladění i správě stránek, umožňuje vyšší rychlost výpočtů (kompilovaný kód) i snadnější použití nástrojů třetích stran (grafy, knihovny, ...).

Řídící aplikace je napsaná v jazyce Delphi, s moduly PLC komunikuje nejčastěji prostřednictvím IP protokolu (LAN) nebo sériového portu (RS 232) či jejich kombinací.

6.2 Požadavky na systém OLAP

Požadovanou funkcí systému je OLAP analýza dat získaných při řízení výroby v průmyslových podnicích. Tyto data jsou uložena v relační databázi Firebird. Hlavním předmětem analýzy je produkce výrobků v závislosti na parametrech (základní výčet):

- čas
- typ výrobku
- zodpovědná osoba (směna)
- výrobní stroj
- zakázka

Systém musí být navržen flexibilně tak, aby umožňoval analýzu i nad jinými produkčními daty, bez nutnosti zásadní změny celého systému. Definice dat určených k analýze by se měla uchovávat v konfiguračních souborech, které lze snadno měnit.

Systém musí efektivní filtrování dat podle času, který hraje v analýze významnou úlohu.

Samotná analýza by měla být prováděna mimo operační databázi a výsledný systém by neměl být závislý na původních externích modulech, které využívá databáze operační. To umožní umístit analytickou databázi na jiný server či jinak škálovat výkon celého systému. Pro účely analýzy není potřeba rozlišovat jednotlivé produkty na úrovni databázových řádků, pro urychlení výpočtů je možno (pokud bude potřeba) sumarizovat data již v datovém úložišti (stanovit rozumně zvolený časový interval).

Uživatelské rozhraní bude realizováno pomocí WWW stránek. Tento přístup umožní snadné rozšíření, správu i používání produktu. Klíčovým požadavkem je také rychlá odezva systému – to znamená efektivní práci s velkým množstvím dat.

Souhrn klíčových požadavků heslovitě:

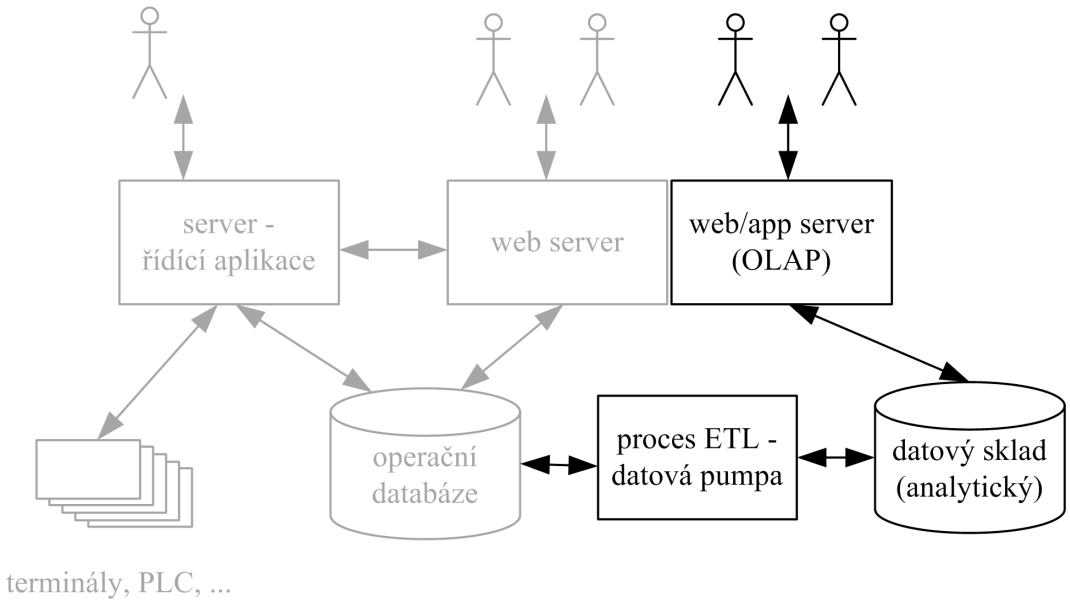
- analýza produkce podle parametrů
- WWW rozhraní
- obecný systém, možno měnit definici analyzovaných dat
- efektivní práce s daty

6.3 Návrh systému

Tato sekce obsahuje detaily návrhu vycházející z požadavků a navazující na stávající řešení systému.

6.3.1 Architektura

Na obrázku 6.2 jsou ukázány jednotlivé části výsledného systému a jejich vzájemné vazby. Jednotlivé části a jejich detaily jsou popsány v dalších částech této kapitoly.



Obrázek 6.2: Návrh OLAP systému navazující na stávající architekturu

Návrh OLAP systému počítá s rozšířením stávajícího o jednu databázi – datový sklad. Ten bude obsahovat upravená data pro analýzu. Transformaci a přesun dat do tohoto skladu bude mít na starosti aplikace *datová pumpa*. Samotné jádro OLAP systému bude realizováno ve formě knihovny.

Webový server může být společným se současným systémem nebo se může použít dedikovaný. Oddělení jednotlivých webových serverů znamená rozložení zátěže na více uzlů

systému i možnost použít jiných technologií. Jako alternativní možnost k serveru Apache může být použit server IIS společnosti Microsoft. Ten má výhodu v nativní podpoře technologií .NET a není nutné instalovat žádné dodatečné moduly či knihovny.

6.3.2 Datový sklad a proces ETL

Asi ideálním datovým skladem sloužící pouze pro OLAP analýzu by byla nějaká implementace databáze, která by byla pro tento přístup k datů přizpůsobena. Nejlépe nějakou databázi obsaženou v paměti (po načtení), kde by data byla uložena ve speciálních strukturách. Vše by bylo velmi rychlé. Taková technologie však dostupná není a vývoj takového řešení stojí daleho za rozsahem tohoto projektu.

Další nevýhody klasických relačních databázových serverů můžou například být (pro účely tohoto projektu):

- transakční zpracování
- řízení přístupu
- verzování a zamykání řádků

Všechny tyto vlastnosti nejsou potřeba pro implemetaci tohoto datového skladu.

Po této kratší analýze jsme však nuceni se vrátit ke klasickému řešení, v tomto případě použít databázový server Firebird a to s ohledem na reálné zkušenosti se správou, provozem a využíváním tohoto serveru v dalších reálných projektech.

Tato databáze bude oddělená od operační a plně na ní nezávislá. Datový sklad bude sloužit jako zdroj dat pro analytické zpracování. Není navržen pro uchování stavu celé operační databáze (případně i dat z jiných zdrojů). Předpokládá se použití pouze pro analytické zpracování.

Databáze může být provozována na dedikovaném serveru, který poskytne dostatečný dotazovací výkon bez omezení výkonosti stávajícího systému[1].

Jako prostředník mezi těmito dvěma databázemi bude aplikace datová pumpa, která implementuje proces ETL. Provádí načtení dat z operační databáze, jejich čištění, transformaci a nahrání do datového skladu. Datová pumpa bude implementována jako aplikace s uživatelským rozhraním založená na platformě .NET (napsaná v jazyce C#).

Výhodou bude bezesporu flexibilita řešení implementovaného prostřednictvím vyššího programovacího jazyka i možnosti ladění, které klasická aplikace (v porovnání například se službou) nabízí. V aplikaci bude v budoucnu možné aplikovat libovolný transformační a výpočetní algoritmus, kterým lze data upravit a přenést do datového skladu. Datová pumpa bude realizovat transformaci dat ze struktur operační databáze na struktury implementované v datovém skladu. Zároveň bude provádět kontrolu integrity dat – detekci chybějících hodnot a jejich nahrazení.

Ukládání dat do datového skladu bude částečně realizováno úložnou procedurou. Tato procedura bude při každém spuštení aplikace datová pumpa znovu vytvořena, což umožní jednodušší správu ETL procesu při nasazení nové verze datové pumpy. Procedura zajistí plnění tabulek dimenzí i faktů a provede sumarizaci dat. Nedělitelný časový interval pro agregaci hodnot je zvolen na 2 hod. U řešení používající úložnou proceduru se předpokládá

[1] Lze také škálovat výkon na stejném databázovém serveru obsahující více procesorů či jáder, kdy se práce rozdělí mezi jednotlivé procesy. Vhodné je umístit databáze na jiné fyzické diskové úložiště. Je možno použít také Firebird server ve verzi Embedded, kdy je celý server zapouzdřen do uživatelské knihovny

dostatečný výkon, protože téměř veškeré operace týkající se databáze budou prováděny v jejím kontextu.

Jenotlivé kroky realizované aplikací datová pumpa (proces ETL):

- načtení dat z obecně implementovaných struktur operační databáze do aplikace
- čištění a transformace těchto hodnot
- uložení dat do datového skladu pomocí úložné procedury

V budoucnu se předpokládá napojení aplikace datová pumpa na systém zamykání záznamů – znemožnění editace historických dat (uzavření období výroby vedením podniku). Po ukončení určitého výrobního období by aplikace aktualizovala data v datovém skladu. Pro účely tohoto projektu bude aplikace akceptovat pouze zadání počátečního a koncového data, které charakterizují úsek dat pro nahrání do datového skladu. Aplikace však bude připravena na toto rozšíření.

Jelikož bude údaj o datumu a čase velice často používán k filtrování dat, bude umístěn přímo v tabulce faktů. Bude ale sloužit i jako dimenze, která se z něj bude počítat pomocí funkcí zjišťující např. rok či měsíc z údaje data. Datový typ `TimeStamp` je realizován jako celé číslo, proto operace extrakce budou velice rychlé a neměly by mít větší vliv na rychlost systému.

6.3.3 Jádro OLAP systému

Systém bude pracovat v režimu požadavek-odpověď. Klient požádá server o zpracování dotazu prostřednictvím datové kostky a server tuto informaci spočítá a poskytne. Reprezentace datové kostky bude definována pomocí tříd, které budou definovat jednotlivé dimenze, fakta, způsob mapování na databázové tabulky, definici operací, textových popisků a dalších informací, které budou ovlivňovat výpočet a zobrazení datové kostky.

Tvar datové kostky bude možné definovat pomocí jazyka XML. Jádro systému bude moci umět takto definovanou kostku načíst z konfiguračních souborů a používat. Bude možné i aktuálně analyzovanou datovou kostku zobrazit jako XML, které lze uložit a poté znovu načíst. Tímto mechanismem umožníme uživateli jednoduše definovat různé tvary datových kostek založených na jednom datovém zdroji. Další výhodou je uložení tvaru datové kostky pro budoucí analýzu (uživatelský pohled), kdy si bude moci uživatel vytvářet a ukládat své vlastní datové pohledy. Projekt bude implementovat tuto fukčnost, složitější správu těchto uživatelských kostek však bude řešit až případné rozšíření.

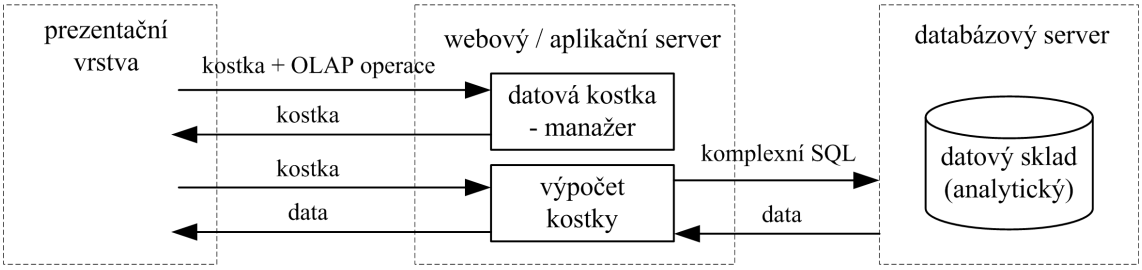
OLAP jádro bude uchovávat několik předdefinovaných tvarů datové kostky, které budou dostupné pro inicializaci prezentační vrstvy. Dále se o uchování tohoto tvaru bude starat vrstva prezenční (udržování kontextu) a jádro bude jen zpracovávat požadavky jednotlivých klientů ve formě definic datových kostek.

Jádro bude implementováno jako .NET knihovna, která bude volaná z prostředí webových stránek a bude implementovat následující hlavní funkčnost:

- definice struktury datové kostky
- aplikace OLAP operace na tvar datové kostky
- výpočet datové kostky

- načtení a uložení tvaru datové kostky v jazyku XML

Na obrázku 6.3 je zobrazena struktura a operace jádra OLAP systému.



Obrázek 6.3: Základní operace jádra OLAP systému

Na základě požadovaného tvaru datové kostky, OLAP jádro načte data z databáze a prezentační vrstva tyto data zobrazí. Definice mapování dimenzí a faktů datové kostky na databázové tabulky umožní dynamické generování SQL příkazů pro načítání dat. Nejvíce výpočetního výkonu při získání a výpočtu dat bude přesunuto na samotný databázový server. Výsledné dotazy zpracovávané databázovým serverem budou generované sofistikovaně a s ohledem na co možná největší výkon celého systému.

6.3.4 WWW uživatelské rozhraní

Uživatelské rozhraní bude realizováno pomocí WWW stránek. Systém generování stránek bude postaven na technologii ASP.NET a nejnovější verzi .NET frameworku 3.5. Tato platforma umožňuje efektivní a moderní vývoje webových aplikací a mnoho dostupných návazných technologií – například AJAX. Samotné jádro systému bude zapouzdřeno v .NET knihovnách a funkce budou dostupné z tříd jednotlivých stránek.

Každá stránka bude zodpovědná za udržování kontextu. Kontextem rozumíme tvar datové kostky a data, která jsou aktuálně zobrazena. Tyto informace musí být perzistentní v rámci stránky a musí se zachovat přes případné odeslání stránky na server. Pro serializaci tvaru datové kostky nebude použit její tvar ve formě XML dokumentu (ačkoliv by to bylo možné), ale využije se možnosti serializovat objekty v prostředí .NET. Tento přístup bude efektivnější z hlediska výsledné velikosti i rychlosti zpracování, načtení z formátu XML by vyžadovalo složitější analýzu.

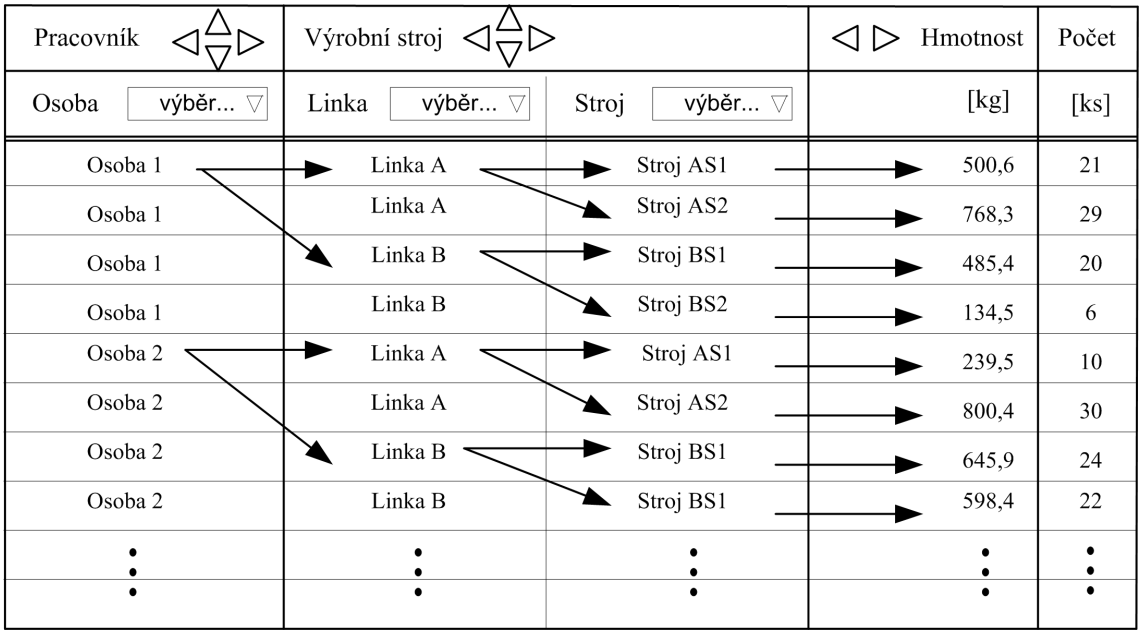
Data OLAP analýzy budou zobrazena uživateli ve formě relační tabulky (viz sekce 4.3.2). Bude obsahovat dva typy sloupců – sloupce dimenzí a faktů.

Záhlaví sloupců budou obsahovat ovládací prvky umožňující aplikací operací OLAP analýzy. Pomocí těchto prvků se bude dát provádět jednotlivé OLAP operace – měnit tvar datové kostky.

Slouce dimenzí – Jednotlivé řádky budou tvořeny hodnotami dimenzí. Každý sloupec dimenze bude tvořen množinou sub-dimenzí, které tvoří hierarchii dimenze. Pokud konkrétní dimenze hierarchická není, bude obsahovat jen jednu sub-dimenzi. V záhlaví sloupců subdimenzí budou obsaženy ovládací prvky pro filtraci dat – seznamy s možností výběru jedné hodnoty.

Slouce dimenzí – Řádky budou tvořit hodnoty agregovaných faktů, které budou výsledkem analýzy.

Filtrace (operace slice&dice) budou realizovány pomocí výběru hodnot dimenzí ze všech možných. V budoucích verzích rozšiřujících tento projekt se počítá s implementací mnohem složitějšího filtru dat, který by umožňoval zadávat komplexní podmínky.



Obrázek 6.4: Návrh tabulky OLAP pro vizualizaci analýzy

Na obrázku 6.4 je zobrazen návrh uživatelského rozhraní datové tabulky, která:

- vizualizuje hodnoty OLAP analýzy
- umožňuje provádět operace OLAP analýzy

Jako příklad byla zvolena tabulka obsahující dvě dimenze – pracovník a výrobní stroj. Druhá dimenze je hierarchická a skládá se z linky a stroje. Tabulka dále obsahuje dvě fakta – hmotnost a počet, což jsou agregované hodnoty podle hodnot dimenzí – výsledky analýzy. Šipky v těle tabulky naznačují postupné dělení hodnot ve směru seřazených dimenzí.

V záhlaví každého sloupce *dimenze* jsou čtyři šipky:

šipka vlevo – Tento prvek umožňuje provést operaci *pivoting*. Zamění pořadí dvou dimenzí, aktuální s levou v zobrazeném pořadí. Pokud se žádná dimenze nalevo nenachází, operace není dostupná.

šipka vpravo – Inverzní operace *pivoting* k předchozí možnosti.

šipka dolů – Tento prvek umožňuje provést operaci *drill-down*. Zvýší urovněň detailu hierarchických dimenzí. Pokud není dimenze hierarchická nebo již není větší detail k dispozici, je operace nedostupná.

šipka nahoru – Tento prvek umožňuje provést operaci *roll-up*. Sníží urovněň detailu hierarchických dimenzí. Pokud není dimenze hierarchická nebo je již aktuální sub-dimenze tou nejvíce nadřazenou, operace je nedostupná.

V záhlaví každého sloupce *faktů* jsou dvě šipky:

šipka vlevo – Tento prvek umožňuje provést operaci *roll-up*. Z následné analýzy se vyloučí dimenze, která je v seřazené posloupnosti nejvíce vpravo. Tím se počet dimenzí o jednu zredukuje a sníží se detail analýzy. Pokud již není do analýzy zahrnuta jediná dimenze, operace je nedostupná.

šipka vpravo – Tento prvek umožňuje provést operaci *drill-down*. Do následné analýzu se zahrne o jednu dimenzi více. Tím se zvýší detail analýzy. Jde o inverzní operaci k předchozí možnosti. Pokud již není žádná dimenze k dispozici, je tato operace nedostupná.

Opearce *slice&dice* je realizována jako filter v záhlaví sloupců sub-dimenzí. Seznam dostupných hodnot umožňuje vybrat jednu, kterou se data budou filtrovat. Jednotlivé podmínky filtrů se mohou libovolně kombinovat.

Tímto je definováno úplné mapování OLAP operací na relační tabulku, která bude implementována pro zobrazování a manipulaci s daty v rámci tohoto projektu.

Jelikož je datum významnou dimenzí, která má v praxi větší důležitost než ostatní a je potřeba přesně definovat časový úsek analýzy dat, bude uživatelské rozhraní obsahovat kalendář, který bude umožňovat vybrat jakýkoliv časový interval. Interval poté bude omezovat data, která budou zařazena do analýzy. Filtrování podle data bude moci vypnout a analyzovat celou databázi bez ohledu na tuto informaci.

Kapitola 7

Implementace systému OLAP

Kapitola popisuje základní postupy použité při implementaci vlastního řešení. Implementace vychází z návrhu systému z kapitoly 6. V jednotlivých sekcích jsou uvedeny jen nejdůležitější modely a způsob implementace jednotlivých částí systému. Není mým cílem zde rozebírat všechny technické detaily, např. způsob tvorby webových stránek na platformě ASP.NET.

Kapitola je rozdělena na sekce, které odpovídají jednotlivým vrstvám výsledného systému – vrstva datová, aplikační a prezentační.

7.1 Datová vrstva

Datová vrstva definuje základ systému, na kterém jsou další části postaveny. Do datové části zahrnujeme datový sklad a procesy, které se starají o přenos dat z operační databáze do tohoto datového skladu.

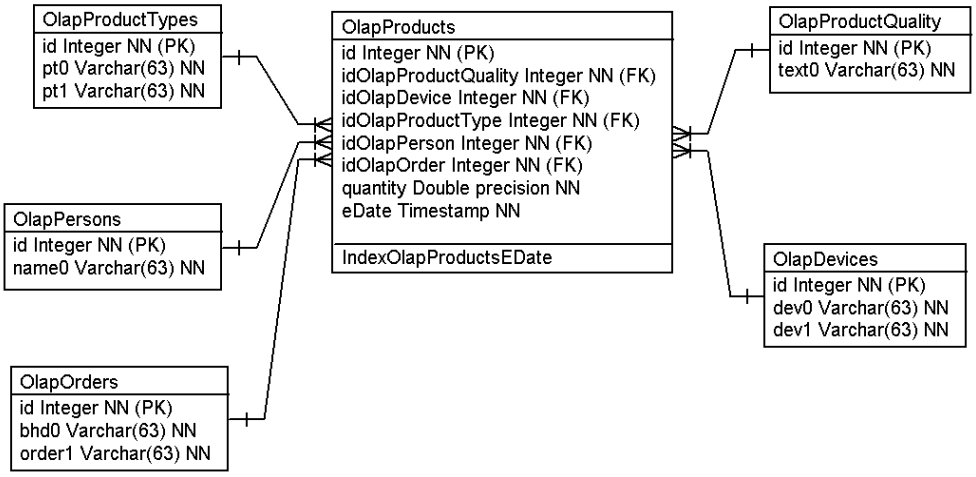
7.1.1 Datový sklad

Model datového skladu je postaven na schématu hvězdy, popsané v sekci 3.3.1. Tento přístup přináší výhody ve své jednoduchosti a poskytuje vyšší dotazovací výkon. Tabulky faktů nejsou normalizované, ačkoliv některé tvoří hierarchii. Model je zobrazen na obrázku 7.1.

Model datového skladu obsahuje pět tabulek dimenzí (šestá časová dimenze může být počítána ze sloupce `eDate`):

- `OlapProductTypes` – Dimenze typ výrobku, může být hierarchická – sdružovat typy výrobků do skupin.
- `OlapPersons` – Dimenze pracovník, osoba či skupina osob zodpovědná za výrobu produktu.
- `OlapOrders` – Dimenze zakázka, sdružuje výrobky do skupin (hierarchická).
- `OlapProductQuality` – Dimenze jakost, definuje stav výrobků a typy zmetků.
- `OlapDevices` – Dimenze stroj, definuje stroj či linku, kde byl produkt vyroben (hierarchická).

Názvy jednodlivých sloupců tabulek dimenzí jsou definovány velice obecně a je možno kdykoliv přidat další potřebné sloupce – hierarchie dimenze. Hierarchie jednotlivých dimenzí



Obrázek 7.1: Datový sklad pro systém OLAP

je v databázi jen naznačena, jak by mohla následně v analýze vypadat. Vše je však závislé na vyšší aplikační vrstvě, která buduje datovou kostku nad schématem databáze.

Tento přístup umožňuje velmi obecně realizovat datový sklad, nad kterým lze provádět různé analýzy – podle definice datové kostky.[1] Další výhodou je, že stejný model datového skladu lze použít pro více implementací v různých produkčních prostředích (různí zákazníci). Může se tedy model spravovat a vyvíjet jen v jedné verzi.

Transformaci dat do těchto tabulek datového skladu zajistí proces ETL – datová pumpa. Tato aplikace bude pro každou implementaci datového sklad jedinečná a zohlední ve svých transformačních algoritmech změny, které se vyskytují u různých zákazníků. Také proto je možné definovat jednodušší model datového skladu (nenormalizované tabulky), příslušné transformace dat zajistí logika v datové pumpě.

V databázi nejsou definovány žádné unikátní indexy nad skupinami sloupců v tabulkách dimenzí. Tato integrita se zatím nebude kontrolovat na úrovni databáze. Model je obecný, aby se dal použít pro různé definice datových kostek.

Jediný index[2] je `IndexOlapProductsEDate` nad sloupcem `eDate` v tabulce faktů (`OlapProducts`). Tento slouží pro rychlé filtrování časového intervalu, který bude dále analyzován.

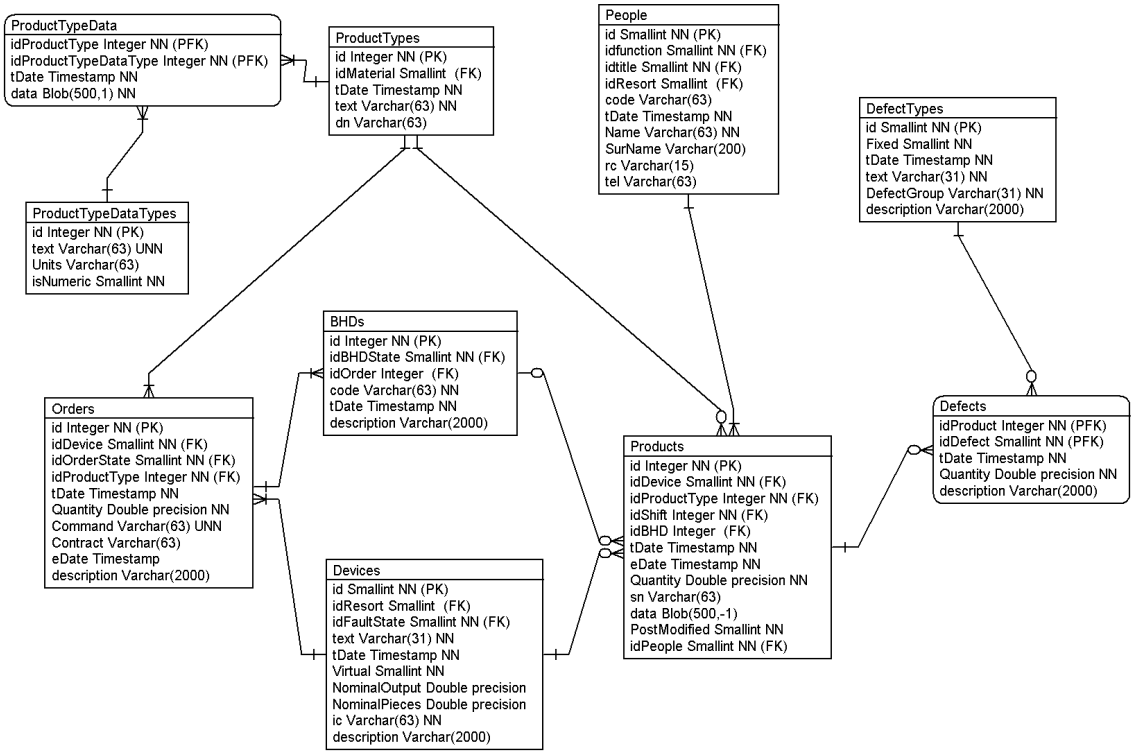
7.1.2 Proces ETL – datová pumpa

Na obrázku 7.2 je část stávající modelu databáze, která popisuje strukturu tabulek pro uložení informací o výrobcích. Uvedený model je zjednodušený, avšak obsahuje všechny části potřebné pro přenos dat do datového skladu.

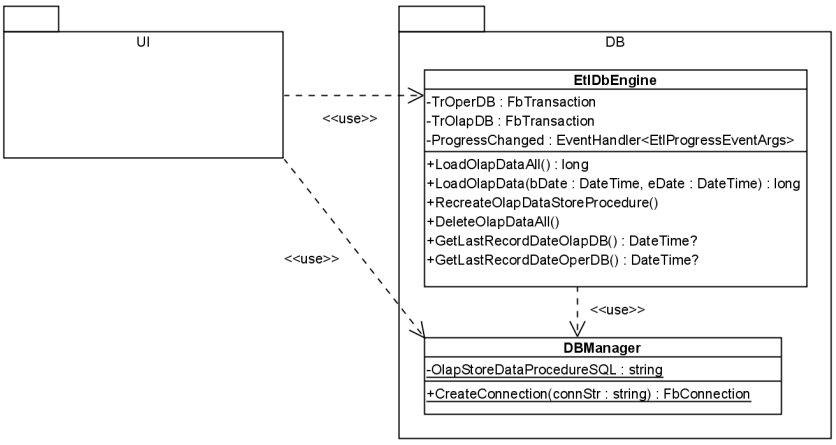
Úkolem aplikace datová pumpa je selekce dat z operační databáze a jejich transformace do struktur datového skladu (viz 7.1). Zároveň je nutné provést čištění, opravu a agregace těchto dat.

[1] Datová kostka nemusí obsahovat vždy všechny dimenze, nebo všechny hierarchie. Jednou může být postavená spíše pro obecnější analýzu, jindy bude obsahovat i detailnější stupně hierarchii dimenzí a bude sloužit pro analýzu například v menším časovém intervalu.

[2] Jediný explicitně definovaný, u cizích klíčů se indexy vytváří automaticky



Obrázek 7.2: Část modelu databáze (výrobky) – stávající systém



Obrázek 7.3: UML diagram tříd aplikace datová pumpa

Na obrázku 7.3 je diagram tříd, reprezentující hlavní části aplikace datová pumpa. Oddělené je uživatelské rozhraní od tříd, které zapouzdřují metody pro práci s databázemi a samotné transformační algoritmy. V balíku `UI` nejsou znázorněny žádné třídy, předpokládá se použití existujících (implementovaných v .NET framework) tříd a komponent jako `Form`, pomocí kterých se vytvoří uživatelské rozhraní.

Spojení k oběma databázím si aplikace otevře při svém startu a pak udržuje po celou

dobu běhu (voláním statické metody `CreateConnection` třídy `DBManager`). Hodnoty definující spojení k databázím jsou uloženy v konfiguračním souboru aplikace. Typický sled operací, které mění stav databáze vypadá následovně:

1. vytvoření transakce
2. vytvoření a inicializace objektu třídy `EtlDbEngine`
3. zavolání příslušné metody objektu třídy `EtlDbEngine`
4. commit transakcí (při jakékoliv chybě rollback)

Uživatelské rozhraní aplikace umožňuje spouštět požadované operace, zadat jejich parametry a vizualizovat jejich průběh.

7.2 Aplikační vrstva

Aplikační vrstva se stará o zpracování požadavků, které generují uživatelé systému. Provádí výpočet datové kostky – generuje dotazy SQL, které jsou vykonávány v databázovém serveru. Dále provádí vzájemné transformace datových kostek pomocí OLAP operací.

7.2.1 Implementace datové kostky

Datová kostka poskytuje způsob pohledu na data obsažená v datovém skladu. Zároveň popisuje jejich uspořádání, textové popisky a hlavně způsob uložení a mapování faktů a dimenzí na databázové tabulky.

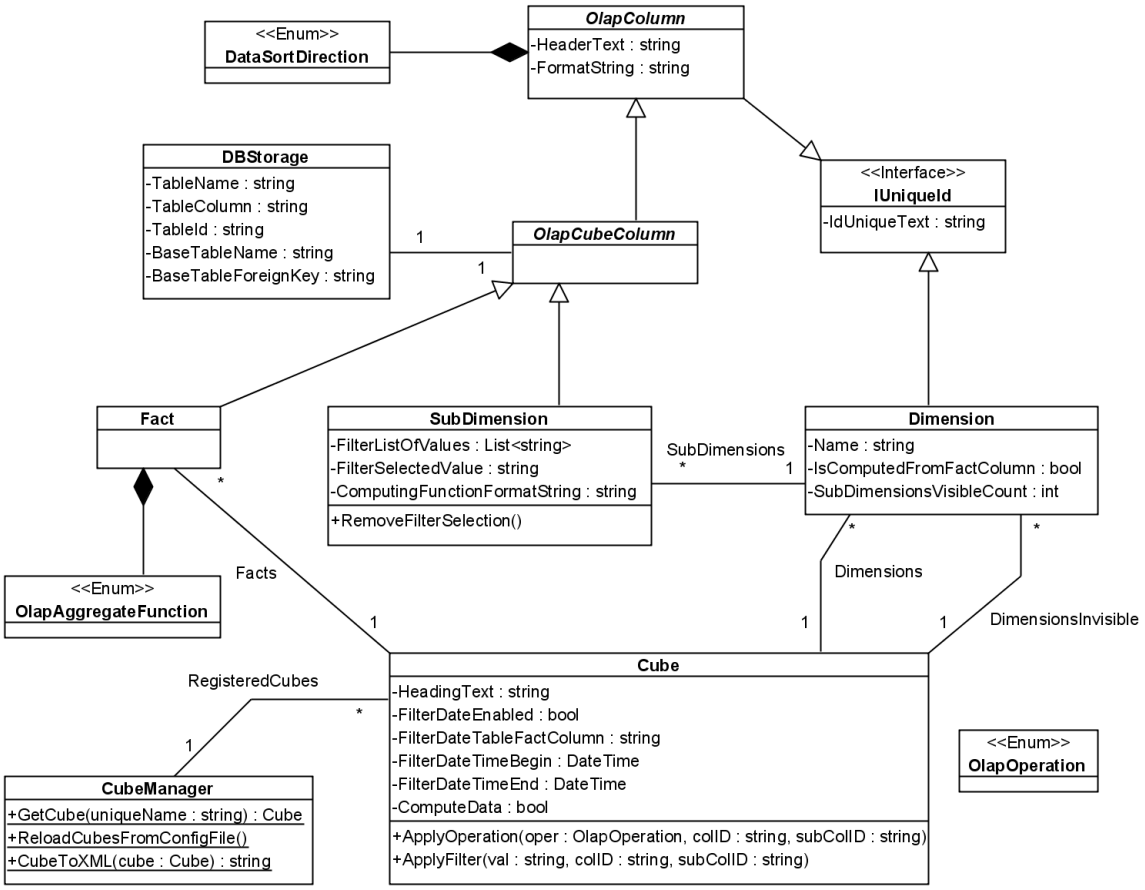
Na obrázku 7.4 je diagram tříd ukazující výslednou implementaci reprezentující datovou kostky. Celá kostka je reprezentována třídou `Cube`. Asociace naznačují, že obsahuje seznam dimenzí a faktů. Celá struktura tříd je realizována relativně složitěji, avšak abstraktní předci `OlapColumn` a `OlapCubeColumn` se s výhodou používají k unifikovanému přístupu ke konkrétním instancím faktů a subdimenzí. Tyto třídy se také dále využívají pro definici datové kostky, která slouží prezentační objektům prezentační vrstvy.

Důležitou metodou třídy `Cube` je `ApplyOperation()`. Ta aplikuje OLAP operaci na datovou kostku, dalšími parametry jsou řetězce identifikující fakt nebo dimenzi a ještě případně subdimenzi. *Výsledkem této operace je změna tvaru datové kostky.* Vizualizací této kostky získáme nový požadovaný pohled na data. Mezi operace definované výčtovým typem `OlapOperation` patří všechny operace OLAP analýzy (drill-down, roll-up, slice&dice a pivoting) a operace řazení podle faktu či subdimenze.

Třída `DBStorage` definuje způsob mapování faktů a dimenzí na tabulky uložené v relační databázi. Podle těchto informací je poté možné generovat dotazy SQL, pomocí kterých se získávají data.

Dimenze jsou většinou realizovány jako samostatné tabulky v databázi. Ale například u dimenze času v naší realizaci to tak neplatí. Tato dimenze je virtuální a je počítána ze sloupce obsaženého v tabulce faktů. Proto třídy `Dimension` a `Subdimension` obsahují atributy `IsComputedFromFactColumn` a `ComputingFunctionFormatString`, které definují použití libovolné funkce (dostupné z databázového serveru) sloužící pro výpočet hodnot této dimenze.

Třída `CubeManager` slouží jako správce datových kostek. Kostky lze registrovat v konfiguračním souboru pomocí jejich tvaru ve formátu XML. Tyto kostky jsou poté načteny a mohou být přístupné pomocí statické metody `GetCube` z prezentační vrstvy systému. Třída



Obrázek 7.4: UML diagram tříd reprezentující datovou kostku

také zapouzdřuje statické metody pro vytvoření objektu `Cube` z definice ve formátu XML a serializaci do formátu XML.

Filtrování hodnot je definováno na dvou místech. Atributy potřebné pro filtrování dat podle času (který je uložen v tabulce faktů) jsou obsaženy ve třídě `Cube`. Pro definici filtru podle hodnot jednotlivých dimenzí (v diagramu tříd subdimenzí) je určena operace `ApplyFilter()` třídy `Cube`. Ta nastaví hodnotu atributu `FilterSelectedValue` ze třídy `SubDimension`. Jednotlivé možné hodnoty jednotlivých subdimenzí jsou získány z databáze, při prvním výpočtu datové kostky. Hodnoty jsou uloženy v kolekci `FilterListOfValues`.

7.2.2 Generování dotazů na databázi

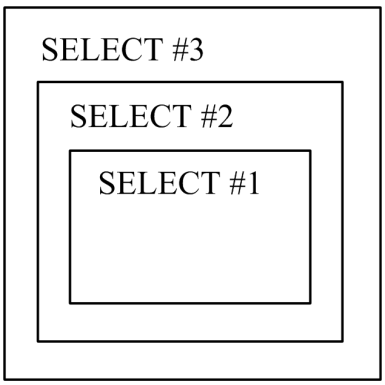
Dynamické generování SQL dotazů je možné provádět díky přesné informaci o mapování jednotlivých dimenzí a faktů na tabulky uložené v relační databázi. Tyto informace obsahují objekty třídy `DBStorage` uvedené na obrázku 7.4.

Strategie generování dotazů spočívá ve spojení tabulky faktů s tabulkami dimenzí, které jsou zahrnuty do analýzy. Vybrat příslušné sloupce z těchto tabulek, aplikovat podmínky na hodnoty sloupců dimenzí – filtry, aplikovat agregační funkce na sloupce tabulky faktů a provést databázovou operaci `group by` v takovém pořadí, jak jsou dimenze definovány v datové kostce.

Tento přístup však není příliš efektivní, protože se provede mnoho databázových operací `join` a teprve poté se na celý blok dat aplikují podmínky filtrů a operace `group by` nad textovými sloupci dimenzí.

Základní strategie by se měla snažit redukovat počet zpracovávaných databázových řádků co nejdříve aplikací filtrů (hodnoty dimenzí a čas), provést operace `group by` ještě dříve než operace `join` a to nejlépe na cizích klíčích ukazujících do tabulek dimenzí. Tyto klíče mají hodnoty celočíselných datových typů a tím operace `group by` budou mnohem rychlejší než u textových řetězců.

Dále je popsán způsob generování SQL dotazů použitý v implementovaném systému OLAP. Výsledný tvar SQL příkazu se skládá ze tří dotazů, jak naznačuje obrázek 7.5.



Obrázek 7.5: Tvar SQL dotazu pro získání dat

Funkce a význam jednotlivých dotazů:

SELECT #1 – Select číslo jedna zajistí základní výběr dat a co největší redukci počtu záznamů. Provede selekci pouze z tabulky faktů a to ty cizí klíče dimenzí, které jsou do analýzy obsaženy. Spočítá agregace a aplikuje operaci `group by` na všechny vybrané klíče dimenzí. V případě dimenzí počítaných ze sloupce tabulky faktů, provede tuto operaci. Aplikovány jsou také všechny podmínky (včetně omezení dat časem), které odpovídají největšímu detailu dimenze a odpovídají tedy cizímu klíči v tabulce faktů. Tento dotaz pracuje jen s jednou tabulkou a sloupci, které obsahují numerické hodnoty. Výpočet je proto velice rychlý a je důležité aplikovat na data co nejvíce operací, které jdou v této fázi vykonat.

SELECT #2 – Select číslo dva provede redukci hierarchických dimenzí (sníží detail) pomocí operace `group by` a aplikuje podmínky na hodnoty dimenzí, které nejsou nejnižším detailem – nelze je tedy aplikovat již na hodnoty cizích klíčů v tabulce faktů. V těchto případech se provede již finální přiřazení textových popisek z tabulek dimenzí.

SELECT #3 – Finální select provede přiřazení textových popisek, které nebyly přiřazeny ve vnitřním selectu a provede řazení podle hodnot dimenzí. Toto řazení uspořádá data do výsledné požadované formy. Žádné operace `group by` ani agregace již neprovádí.

Příklad dotazu generovaného systémem (tabulky odpovídají schématu uvedeného na obrázku 7.1):

```
SELECT OlapDevices.dev0 AS dimDevice0, TMPTABLE0.dimTime1 AS dimTime1,
       TMPTABLE0.dimTime0 AS dimTime0, TMPTABLE0.dimProductType1 AS dimProductType1,
       OlapPersons.name0 AS dimPerson0, TMPTABLE0.factQuantity
FROM
(
  SELECT TMPTABLE1.dimDevice, TMPTABLE1.dimTime1, TMPTABLE1.dimTime0,
         OlapProductTypes.pt1 AS dimProductType1, TMPTABLE1.dimPerson,
         SUM(TMPTABLE1.factQuantity) AS factQuantity
  FROM
  (
    SELECT idOlapDevice AS dimDevice, EXTRACT(YEAR FROM eDate)AS dimTime1,
           EXTRACT(MONTH FROM eDate)AS dimTime0, idOlapProductType AS dimProductType,
           idOlapPerson AS dimPerson, SUM(quantity) AS factQuantity
    FROM OlapProducts
    WHERE idOlapDevice = (SELECT FIRST 1 id FROM OlapDevices WHERE dev0 = 'Linka L1')
    AND eDate >= '01-12-2006 06:00:00' AND eDate < '03-22-2008 06:00:00'
    GROUP BY 1, 2, 3, 4, 5
  )
  TMPTABLE1
  JOIN OlapProductTypes ON OlapProductTypes.id = TMPTABLE1.dimProductType
  GROUP BY TMPTABLE1.dimDevice, TMPTABLE1.dimTime1, TMPTABLE1.dimTime0,
           dimProductType1, TMPTABLE1.dimPerson
)
TMPTABLE0
JOIN OlapDevices ON OlapDevices.id = TMPTABLE0.dimDevice
JOIN OlapPersons ON OlapPersons.id = TMPTABLE0.dimPerson
ORDER BY dimDevice0 DESC, dimTime1 ASC, dimTime0 ASC, dimProductType1 ASC, factQuantity DESC
```

Více informací o dotazech na databázi lze nalézt v příloze B, kde je uvedeno několik příkladů, porovnání mezi jednotlivými strategiemi a jsou uvedeny počty záznamů, naměřené časy a detailní technické informace o plánu zpracování dotazů.

7.3 Prezentační vrstva

Prezentační vrstva se stará o generování uživatelského rozhraní, vizualizaci výsledků analýzy a umožnění komunikace uživatele se systémem. Uživatelské rozhraní je realizováno pomocí WWW stránek, na serveru je použitá technologie ASP.NET.

Programování webových aplikací na platformě ASP.NET se do jisté míry podobá klasickému programování formulářů pro Windows. Stránka je reprezentována třídou `Page`, která obsahuje již mnoho implementovaných vlastností. Kód se na serveru neinterpretuje, ale vykonává se v kompilované podobě.

Důležitým požadavkem je zachování kontextu probíhané analýzy přes jednotlivá odeslání stránky na server, kde se generuje HTML kód pro zobrazení na prohlížeči. Je více způsobů, jak tento kontext ukládat. Prvním může být možnost uložení těchto dat na serveru (session) s unikátním klíčem, kterým se bude přistupovat k těmto datům. Klíč je unikátní a každý klient si jej přenáší v těle stránky. Nevýhodou může být přístup z více panelů či oken aplikace, kde se tento kontext sdílí.

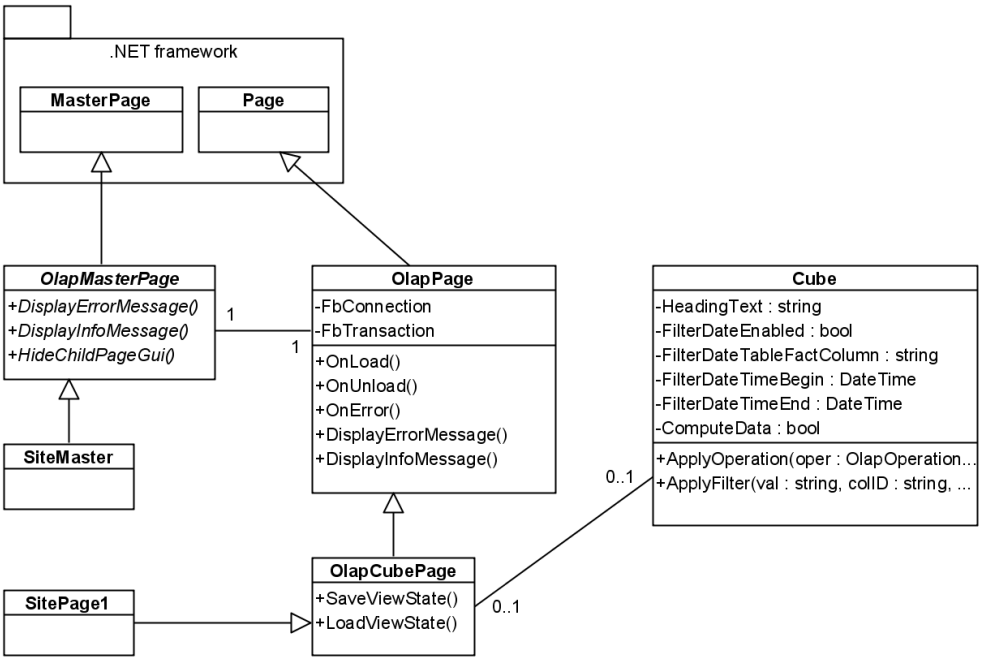
Druhou možností a v projektu implementovanou je serializace těchto dat a přenos v samotném těle stránky na klienta a poté na server, kde se zpět získá jejich objektová reprezentace. Tyto data jsou zakódována a přenášena ve skrytých polích formulářů. Nárust

velikosti dat, které je nutno přenášet nijak dramaticky neroste a každá jedna stránka si uchovává svůj kontext nezávisle na všech ostatních.

Jako prostředek k této serializaci byla použita standardní možnost serializovat objekty do textové podoby, kterou nabízí .NET framework. Tento způsob je efektivnější, než serializace ve tvaru XML, který v projektu využíváme pro uchování tvaru kostek v konfiguračních souborech aplikace.

7.3.1 Implementace generování uživatelského rozhraní

Každá stránka si při své inicializaci na serveru otevírá spojení k databázi a to zavírá při ukončení svého běhu. Dodržuje se striktní princip jedné transkace na stránku. Na obrázku 7.6 je diagram tříd zobrazující strukturu implementace základních tříd uživatelského rozhraní na straně serveru.



Obrázek 7.6: UML diagram tříd uživatelského rozhraní

Třída `OlapPage` překrývá metody `OnLoad()`, `OnUnload` a `OnError()` svého předka. V těchto metodách vytváří a otevírá spojení s databází, při ukončení běhu jej zavírá a provádí commit. Jestliže nastane nějaká chyba na stránce, je zobrazena chybová hláška a provádí se rollback transkace a zavření spojení k databázi.

Třída OlapCubePage rozšiřuje `OlapPage` o metody, které zajistí uložení a načtení stavu asociované datové kostky. Tímto automaticky zajistí perzistenci tohoto objektu v rámci stránky.

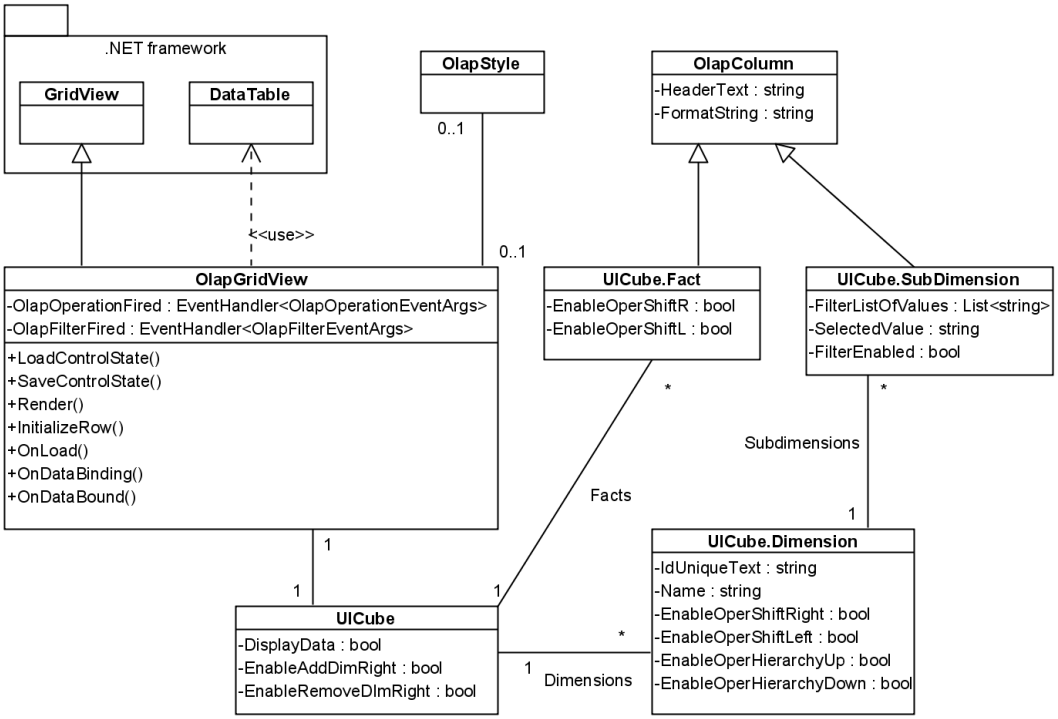
Třídy `SiteMaster` a `SitePage1` jsou již konkrétními třídami, které implementují uživatelské rozhraní webové aplikace. Třída SiteMaster definuje společné uživatelské rozhraní všem stránkám webové aplikace – nadpisy, menu, rozložení, ... Tato implementace je tedy uložena jen na jednom místě a jednotlivé stránky již implementují jen svůj obsah.

Výsledné implementace stránek mají své uživatelské rozhraní, které obsahuje kalendář pro zvolení časového intervalu analýzy. Stránka pro OLAP analýzu je implementována jako

parametrická, očekává parametr (metoda GET) jména datové kostky, která je registrována ve třídě `CubeManager` (viz 7.4). Tvar této kostky poté získá a podle něj generuje uživatelské rozhraní. Tuto třídu proto stačí implementovat jen jednou, i když různých analýz může vizualizovat mnoho.

7.3.2 Zobrazení datové kostky

Samotná datová tabulka – vizualizace datové kostky (viz návrh 6.4) je implementována rozšířením komponenty `GridView`, která je dostupná v .NET frameworku a používá se pro zobrazení dat z různých zdrojů. Diagram tříd je uveden na obrázku 7.7.



Obrázek 7.7: UML diagram tříd – uživatelské rozhraní datová kostka

Třída `OlapGridView` překrývá některé metody svého předka a implementuje tím své uživatelské rozhraní a rozšiřuje funkčnost. Tato třída tvoří komponentu, která je schopná zobrazit jakýkoliv tvar datové kostky.

Pro definici tvaru datové kostky pro uživatelské rozhraní slouží třída `UICube` a třídy v ní vnořené. Třída `OlapGridView` obdrží data z aplikační vrstvy ve formě objektu třídy `DataTable` a právě třída `UICude definuje`, jak se mají tyto data zobrazit.

Funkce komponenty `OlapGridView`:

- zobrazit data OLAP analýzy
- umožnit provádět operace OLAP analýzy
- udržovat kontext – uložení zobrazených dat a třídy `UICube` serializací do těla stránky

Samotný vzhled datové tabulky je definován pomocí třídy OlapStype. Tato třída obsahuje poměrně mnoho atributů, které defnují vzhled výsledné tabulky – jsou to definice kaskádových stylů, odkazy na obrázky znázorňující jednotlivé OLAP operace, ...

Při implementaci uživatelkého rozhraní byl kladen velký důraz na možnost implementovat jednotlivé části jako komponenty, které se dají znovu použít nejen v této webové aplikaci, ale i pro další použití. V jiném projektu proto stačí jen naimportovat potřebné knihovny a lze použít a popřípadě rozšířit všechny části projektu.

Vzhled celé webové aplikace a jednotlivých komponent je defunován pomocí externích souborů – kaskádových stylů (css) nebo stylových souborů pro komponenty .NET. Tyto soubory lze lehce přičazovat jednotlivým stránkám či komponentám a snadno měnit vzhled bez nutnosti zásahu do programového kódu.

Ukázky uživatelského rozhraní systému OLAP analýzu jsou uvedeny v příloze A.

Kapitola 8

Závěr

Technologie se souhrnným názvem Business intelligence jsou velmi populární a žádané. Jde o soubor technik a postupů, které slouží pro podporu rozhodování především v obchodní sféře.

Tato práce se zabývala jednou z jejích částí – OLAP analýzou. Jde o komplexní datovou analýzu zaměřenou na základní subjekty, které se vyskytují v obchodních činnostech organizace. OLAP analýza poskytuje komplexní pohled na data na vyšší úrovní abstrakce. Umožní agregovat a zobrazit obchodní výsledky společnosti za určité časové období a analyzovat přednosti i nedostatky.

Hlavním cílem práce bylo navrhnout a implementovat vlastní systém, který by umožňoval provádět OLAP analýzu. Takovýto systém aplikací byl vyvinut a bude následně používán a rozšiřován pro praktické využití při analýze rozsáhlých datových souborů.

Výsledný produkt tvoří sada spolupracujících aplikací, které implementují jednotlivé části OLAP analýzy. Každá část byla navrhnuta s ohledem na budoucí rozšíření a doplnění o nové funkce. Mezi náměty na budoucí zlepšení patří implementace komplexního systému pro zadávání filtrů nebo obohacení uživatelského rozhraní o generování grafů.

S technickou úrovní výsledné aplikace jsem spokojen, domnívám se, že má velký potenciál pro využití v reálném provozu. Celá problematika bussines intelligence a OLAP analýzy mě velice zaujala a rád bych se tomuto oboru věnoval i v mé budoucí programátorké kariéře.

Literatura

[1] Jaroslav Zendulka a kol. *Získávání znalostí z databází - studijní opora.* Fakulta informačních technologií, VUT Brno, 2006.

[2] Jiawei Han and Micheline Kamber. *Data Mining Concepts and Techniques.* Diane Cerra, 2006. ISBN 1-55860-901-6.

[3] Tomáš Hruška. *Datová skladiště a technologie OLAP - slajdy k přednáškám.* Fakulta informačních technologií, VUT Brno, 2004.

[4] W. H. Inmon. *Building the Data Warehouse.* JohnWiley & Sons, 2002. ISBN 0-471-08130-2.

[5] Lacko L. *Datové sklady analýza OLAP a dolování dat.* Computer Press, 2003. ISBN 80-7226-969-0.

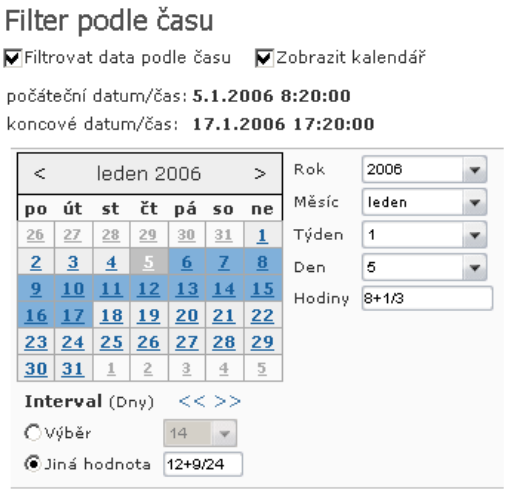
[6] WWW stránky. Mondrian project. `http://mondrian.pentaho.org/`.

Dodatek A

Ukázka uživatelské rozhraní OLAP systému

Příloha obsahuje ukázky snímků obrazovek uživatelského rozhraní. Následuje popis jednotlivých částí.

A.1 Kompletní uživatelské rozhraní



Obrázek A.1: Uživatelské rozhraní - časový filtr

Uživatelské rozhraní se skládá ze dvou hlavních částí. První slouží pro definici časového filtru (obrázek A.1). Obsahuje rozšířený kalendář, který umožňuje zvolit libovolný časový interval. Do analýzy jsou zahrnuty pouze ta data, která do něj patří.

Druhou částí je samotná relační tabulka (obrázek A.2), která vizualizuje data a umožňuje provádět operace OLAP analýzy. Detaily umístění ovládacích prvků OLAP operací jsou popsány v následující sekci.

Zobrazení dat

Počítat a zobrazovat data

VÝROBEK ↑ Průměr	STROJ ↓ Linka	ČAS • Rok	↑ Množství [m]
35	Linka L3	2006	114254390,3
35	Linka L3	2007	151954733,1
35	Linka L1	2006	89301688,9
35	Linka L1	2007	120966503,2
38	Linka L1	2006	3480921,4
38	Linka L1	2007	4311848,6
45	Linka L6	2007	4138993,5
45	Linka L5	2006	12110391,7
45	Linka L5	2007	197997325,7
45	Linka L4	2006	141441210,1
45	Linka L4	2007	197179686,7
45	Linka L2	2006	57230716,2
45	Linka L2	2007	74819787,4
53	Linka L5	2007	13832010,1
53	Linka L5	2006	52579401,8
53	Linka L2	2006	73624491,7
53	Linka L2	2007	96315334,8

Obrázek A.2: Uživatelské rozhraní - relační tabulka

A.2 Ovládací prvky OLAP operací

Na obrázku A.3 jsou označeny čísly jednotlivé ovládací prvky, pomocí kterých lze řídit analýzu. Význam jednotlivých operací je uveden v následující tabulce.

KVALITA ↑ Kvalita	ČAS 1 2 3 4 ↑ Rok	↑ Měsíc	6 ↑ Den	STROJ ↑ Linka	7 8 9 • Množství [m] 10 11
Dobré	2006	3	30	Linka L1	36226,2
Dobré	2006	3	30	Linka L2	46456,3
Dobré	2006	3	30	Linka L3	23917,4
Dobré	2006	3	30	Linka L4	56934,5

Obrázek A.3: Uživatelské rozhraní - OLAP operace (ovládací prvky)

Číslo	Význam ovládacího prvku
1	operace pivoting, změna pořadí sousedních dimenzí (posun doleva, pokud je možný)
2	operace roll-up, snížení detailu dimenze (posun v hierarchii směrem nahoru, pokud je možný)
3	operace drill-down, zvýšení detailu dimenze (posun v hierarchii směrem dolů, pokud je možný)
4	operace pivoting, změna pořadí sousedních dimenzí (posun doprava, pokud je možný)
5	operace slice&dice, filtr definovaný hodnotou dimenze
6	řazení podle hodnot dimenzí
7	operace roll-up, snížení detailu, redukce dimenze (v pořadí nejvíce vpravo, pokud existuje)
8	operace drill-down, zvýšení detailu, přidání dimenze (pokud je k dispozici)
9	řazení podle hodnot faktů
10	posun sloupce doleva v posloupnosti faktů (pokud je faktů více a je operace možná)
11	posun sloupce doprava v posloupnosti faktů (pokud je faktů více a je operace možná)

Dodatek B

Porovnání SQL dotazů

Dodatek porovnává výkon dotazů, které jsou použity pro získání dat z datového skladu. Porovnává dvě odlišné strategie. První neaplikuje žádné speciální techniky a klade dotaz v nejjednosušší formě. Druhá se snaží o co největší efektivitu dotazu za cenu vyšší složitosti.

Efektivnější způsob je implementován ve vyvinutém systému, kde jádro OLAP sytému generuje dotazy v této formě. Oba způsoby dotazů se provádí nad celou databází (227999 záznamů v tabulce faktů), nejsou omezeny časem. Oba produkují totožný výsledek, statistické hodnoty jsou získány při opakovaném provedení dotazu – data se nemusí číst z disku, jsou v cache serveru.

Podmínky při provedení testu:

Vlastnost	**Hodnota**
pc	notebook Compaq nc8430
cpu	Intel Centrino Duo T2500 @2.00GHz
paměť	3 GB
databázový server	Firebird 2.1
počet záznamů tabulky faktů	227999

B.1 Jednoduchý dotaz

Použitý dotaz je napsán velice jednoduše. Hlavní nevýhodou je práce s velkou datovou množinou vzniklou spojením databázových tabulek a poté aplikací operace `group by` na textové sloupce.

```
SELECT D.dev0 AS dimDevice0, EXTRACT(YEAR FROM P.eDate) AS dimTime1,
       EXTRACT(MONTH FROM P.eDate) AS dimTime0, PT.pt1 AS dimProductType1,
       PR.name0 AS dimPerson0, SUM(P.quantity) AS factQuantity
FROM
OlapProducts P
JOIN OlapDevices D ON D.id = P.idOlapDevice
JOIN OlapProductTypes PT ON PT.id = P.idOlapProductType
JOIN OlapPersons PR ON PR.id = P.idOlapPerson
WHERE D.dev0 = 'Linka L1'
GROUP BY 1, 2, 3, 4, 5
ORDER BY dimDevice0 DESC, dimTime1 ASC, dimTime0 ASC, dimProductType1 ASC,
         factQuantity DESC
```

plán dotazu:

```
PLAN SORT (SORT (JOIN (PR NATURAL, P INDEX (RDB$FOREIGN9), D INDEX (RDB$PRIMARY3),
     PT INDEX (RDB$PRIMARY2))))
```

statistické informace:

```
Records affected: 119
Elapsed time   = 1.59 sec
Reads  (physical)  = 0
Writes (physical)  = 0
Fetches(in-memory) = 1542916
Marks  (in-memory) = 0
```

B.2 Složený dotaz

Dotaz se skládá ze tří částí, popis strategie vytváření lze nalézt v sekci 7.2.2.

```
SELECT OlapDevices.dev0 AS dimDevice0, TMPTABLE0.dimTime1 AS dimTime1,
       TMPTABLE0.dimTime0 AS dimTime0, TMPTABLE0.dimProductType1 AS dimProductType1,
       OlapPersons.name0 AS dimPerson0, TMPTABLE0.factQuantity
FROM
(
  SELECT TMPTABLE1.dimDevice, TMPTABLE1.dimTime1, TMPTABLE1.dimTime0,
         OlapProductTypes.pt1 AS dimProductType1, TMPTABLE1.dimPerson,
         SUM(TMPTABLE1.factQuantity) AS factQuantity
  FROM
  (
    SELECT idOlapDevice AS dimDevice, EXTRACT(YEAR FROM eDate)AS dimTime1,
           EXTRACT(MONTH FROM eDate)AS dimTime0, idOlapProductType AS dimProductType,
           idOlapPerson AS dimPerson, SUM(quantity) AS factQuantity
    FROM OlapProducts
    WHERE idOlapDevice = (SELECT FIRST 1 id FROM OlapDevices WHERE dev0 = 'Linka L1')
    GROUP BY 1, 2, 3, 4, 5
  )
  TMPTABLE1
  JOIN OlapProductTypes ON OlapProductTypes.id = TMPTABLE1.dimProductType
  GROUP BY TMPTABLE1.dimDevice, TMPTABLE1.dimTime1, TMPTABLE1.dimTime0,
           dimProductType1, TMPTABLE1.dimPerson
)
TMPTABLE0
JOIN OlapDevices ON OlapDevices.id = TMPTABLE0.dimDevice
JOIN OlapPersons ON OlapPersons.id = TMPTABLE0.dimPerson
ORDER BY dimDevice0 DESC, dimTime1 ASC, dimTime0 ASC, dimProductType1 ASC, factQuantity DESC
```

plán dotazu:

```
PLAN (TMPTABLE0 TMPTABLE1 OLAPDEVICES NATURAL)
PLAN SORT (JOIN (SORT (JOIN (SORT (TMPTABLE0 TMPTABLE1 OLAPPRODUCTS INDEX (RDB$FOREIGN8)),
     TMPTABLE0 OLAPPRODUCTTYPES INDEX (RDB$PRIMARY2))), OLAPPERSONS INDEX (RDB$PRIMARY4),
     OLAPDEVICES INDEX (RDB$PRIMARY3)))
```

statistické informace:

```
Records affected: 119
Elapsed time   = 0.35 sec
Reads  (physical)  = 0
Writes (physical)  = 0
Fetches(in-memory) = 95098
```

B.3 Shrnutí

Složený dotaz je proveden asi 5x rychleji, než je tomu u jednoduchého. Je to způsobeno strategií redukce zpracovávaných záznamů a snahou provést co nejvíce operací nad celočíselnými datovými typy.

V praxi se rozdíl nejvíce projeví u analýz s vysokým počtem dimenzí. Z dalšího testování vyplynulo, že zvýšení vykonu systému díky složeným dotazům vzrostlo asi o jeden řád.

Dodatek C

Obsah přiloženého datového nosiče

Příloha obsahuje popis jednotlivých částí práce uložených na datovém nosiči (DVD), který je součástí práce. Jde o databáze, dokumentaci a zdrojové soubory celého projektu. Součástí je také demo aplikace, kterou je možné po krátké inicializaci spustit a vyzkoušet vytvořený systém OLAP.

C.1 Databáze

Adresář `data` na DVD obsahuje dvě databáze firebird:

- `db_oper.fdb` – Operační databáze, která slouží jako primární zdroj zdroj dat. Tyto data se transformují do datového skladu pomocí nástroje datová pumpa.
- `db_olap.fdb` – Analytický datový sklad, slouží jako zdroj dat pro analýzu OLAP.

C.2 Zdrojové kódy

Adresář `src` obsahuje archiv se zdrojovými kódy systému. K otevření projektů je možné použít produktů *Microsoft Visual C# 2008 Express Edition* a *Microsoft Visual Web Developer 2008 Express Edition*, které jsou dostupné zdarma k volnému použití. Stáhout se dají ze stránek firmy Microsoft.

C.3 Dokumentace

Adresář `doc` obsahuje diplomovou práci ve formátu pdf a archiv se zdrojovými texty dokumentu napsané v jazyce LaTeX.

C.4 Demo systému OLAP

Adresář `demo` obsahuje archiv, ve kterém je zabaleno několik aplikací:

- `etl_process` – aplikace datová pumpa
- `fb_server` – databázový server Firebird
- `net_framework` – instalační soubor .NET framework 3.5

- `web_server` – jednoduchý webový server Cassini
- `www` – webová aplikace – systém OLAP

C.4.1 Instalace demo systému

Instalace demo systému spočívá v pouhém nakopírování souborů a spuštění aplikací. Vše je tedy jednoduché a nic se neintegruje do systému.

1. V případě, že systém Windows neobsahuje .NET framework 3.5 je nutné ho nainstalovat. Instalační soubor je dostupný ve složce `net_framework` – nutný je přístup k internetu.
2. Na disku `c:` vytvořit složku `olap_demo`. Konfogirační soubory aplikací jsou nastaveny na tuto cestu, jinak by se musely opravit.
3. Do této složky nakopírovat složky z archivu demo aplikace a složku `data`[1] z kořene DVD. Složka `c:/olap_demo/` by měla obsahovat následující adresáře:
   - `data`
   - `etl_process`
   - `fb_server`
   - `web_server`
   - `www`
4. Spustit dávku `start.bat` z adresáře `fb_server`, která spustí databázový server Firebird jako aplikaci. V tray liště systému se objeví ikona, pomocí které lze pak aplikaci pohodlně ukončit.
5. Spustit dávku `start.bat` z adresáře `web_server`, která spustí webový server Cassini jako aplikaci (na portu 8081). Aplikaci lze poté jednoduše ukončit.
6. Při dodržení postupu je webová aplikace systému OLAP přístupná na adrese: `http://localhost:8081/Web/`
7. (Aplikace datová pumpa je dostupná v adresáři `etl_process`.)

[1] soubor `db_oper.fdb` je nutný jen pro aplikaci datová pumpa, pro OLAP analýzu jej není třeba kopírovat