VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

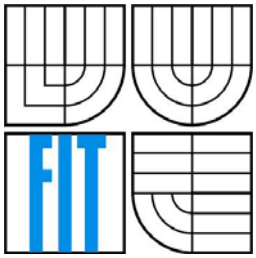
MODERNÉ TECHNOLÓGIE V OLAP

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. DANIEL JANOŠKA
AUTHOR

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

MODERNÍ TECHNOLÓGIE V OLAP

MODERN TECHNOLOGY IN OLAP SYSTEM

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. DANIEL JANOŠKA

VEDOUCÍ PRÁCE
SUPERVISOR

Prof. Ing. TOMÁŠ HRUŠKA CSc.

BRNO 2008

Abstrakt

Nástroje OLAP sa v stále väčšej miere uplatňujú v podnikoch a inštitúciach po celom svete. Sú špeciálne zamierené na podporu potrieb riadiacich pracovníkov. Hlavnou témou diplomovej práce je OLAP analýza s jej možnosťami. Diplomová práca sa tiež zaoberá technológiami FLEX a AIR. V diplomovej práci sú prebrané základné princípy fungovania týchto technológií.

Klíčová slova

OLAP, dátový sklad, databáza, agregácia, FLEX, AIR, tlustý klient, platforma, runtime.

Abstract

OLAP systems are sought-after tool which are deployed in companies and industry environment. The fundamental task of these systems are support of executive management. This work is dealing with OLAP analysis and its capabilities. It also discussed FLEX and AIR technology, their base principles and functions.

Keywords

OLAP, DataWarehouse, database, aggregation, FLEX, AIR, thick client, platform, runtime.

Citace

Daniel Janoška: Moderné technológie v OLAP. Brno, 2008, diplomová práce, FIT VUT v Brně.

Moderné technológie v OLAP

Prohlášení

Prehlasujem, že som túto diplomovú prácu vypracoval samostatne pod vedením Prof. Ing. Tomáše Hrušky, CSc.
Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

……
Daniel Janoška
19.4.2008

Poděkování

Na tomto mieste by som rád poďakoval Prof. Ing. Tomášovi Hruškovi, CSc za odbornú pomoc.

Obsah

1 Úvod

Popri procese zhromažďovania veľkého množstva údajov, je potreba tieto údaje analyzovať, získavať z nich informácie v požadovanom čase a v správnej forme, dostupnej nielen pre analytikov a ľudí s informatickým vzdelaním, ale aj pre užívateľov z oblasti riadenia a manažmentu. Získanie pravdivých a relevantných informácií v správnom čase sa stáva nevyhnutnou požiadavkou pri strategickom rozhodovaní a podpore rozhodovania. Pod názvom OLAP sú zahrnuté technológie, metódy a prostriedky, ktoré umožňujú ad-hoc analýzu údajov multidimenzionálneho charakteru. OLAP umožňuje užívateľovi pracovať s údajmi veľmi flexibilne a analyzuje dáta z mnohých hľadísk. OLAP je rozšírenou súčasťou Business Intelligence a oblasť aplikácie nájde v obchodnom prostredí, manažérskom rozhodovaní, finančnom sektore, pri zostavovaní rozpočtu, odhadovaní a analýzy trendov rôznych veličín a podobných oblastí. Databáza určená pre služby OLAP využíva multidimenzionálny databázový model, ktorý svojou štruktúrou dovoľuje komplexné analýzy a ad-hoc dotazovanie s veľmi rýchlou odozvou. V prvých kapitolách prinášam ucelený súbor definícii a prehľad o problematike. Mapujem rozdiely v oblasti využitia transakčných databáz a dátových skladov. Definujem logický multidimenzionálny databázový model a porovnávam ho s relačným. Taktiež definujem OLAP, oblasti použitia a porovnávam implementačné varianty OLAP. V druhej časti práce sa sústredím na moderné implementačné technológie FLEX a AIR ich podrobný popis a ich využitie v OLAP produktoch. Porovnávam ich architektúru, funkcionalitu, využitie. Taktiež rozoberám detaily navrhovanej aplikácie.

1.1 Cieľ práce

Cieľom práce je zoznámiť sa s problematikou OLAP, zhodnotiť súčastné možnosti a prístupy pri návrhu a tvorbe OLAP technológie a desktopových aplikácií. Súčasťou práce je špecifikácia požiadavkov a návrh prototypu a jeho implementácia s využitím technológií AIR a FLEX. Výsledná OLAP aplikácia je zameraná na využitie tlustého klienta s možnosťou ukladania agregačných hodnôt na klientovi.

2 Podpora rozhodovania v IS

2.1 Klasifikácia informačných potrieb

2.1.1 Operatívna úroveň

Na tejto úrovni sa nachádzajú všetky potreby každodennej prevádzky systému ako napríklad príjem objednávok,vedenie skladových zásob, výroba, účtovníctvo, personálne a mzdové agendy atď. Tieto aplikácie pracujú v princípe pomocou ad-hoc transakcií a preto spracovanie na tejto úrovni tiež nazývame OLTP (On-Line Transaction Processing). Databázy, v ktorých sa realizuje táto úroveň nazývame operatívne, transakčné, alebo prevádzkové databázy.

2.1.2 Analytická úroveň

Analytická úroveň je nadradená operatívnej úrovni z hľadiska dlhodobého charakteru riadenia organizácie. K tomu je potrebné mať databázu obsahujúcu istý časový horizont a rôzne agregácie. Rozhodovanie na tejto úrovni je vykonávané využitím rôznych modelov, analýz na základe ktorých je možné vykonať správne rozhodnutia. Spracovanie na tejto úrovni nazývame OLAP (On-Line Analytical Processing). Dôležitým znakom tejto úrovne je charakter prístupu užívateľov – iba čítanie.

2.2 Klasifikácia databázových systémov

Nasleduje stručný prehľad transakčných systémov a informačné systémy pre podporu rozhodovania vrcholového riadenia.[2]

2.2.1 Systémy OLTP (Online Transaction Processing)

Sú to relačné databázy, v ktorých vykonávame veľké množstvo transakcií v reálnom čase. Nazývame ich tiež operatívne, produkčné alebo transakčné databázy. Transakčné databázy, do ktorých sa ukladajú aktuálne operatívne údaje, sú organizované ako relačné, čo znamená, že údaje sú uložené v databázových tabuľkách, medzi ktorými sú relačné vzťahy vyplývajúce z aplikačnej logiky.

Primárnym cieľom transakčných databázových systémov je umožniť užívateľom databázového serveru vykonávanie veľkého množstva transakcií online (napríklad obchodných, bankových a pod.) Cieľom transakčných databázových systémov je automatizácia činností, ktoré sa opakujú. Patrí sem zautomatizovanie bežných úloh ako napríklad vedenie účtovníctva, spracovanie miezd, evidenčné systémy atď. Typickou vlastnosťou týchto systémov je, že veľká časť celkového spracovania je vykonávaná už pri vkladaní dát resp. tesne po ňom. K zdroju údajov v rovnakom čase pristupujú

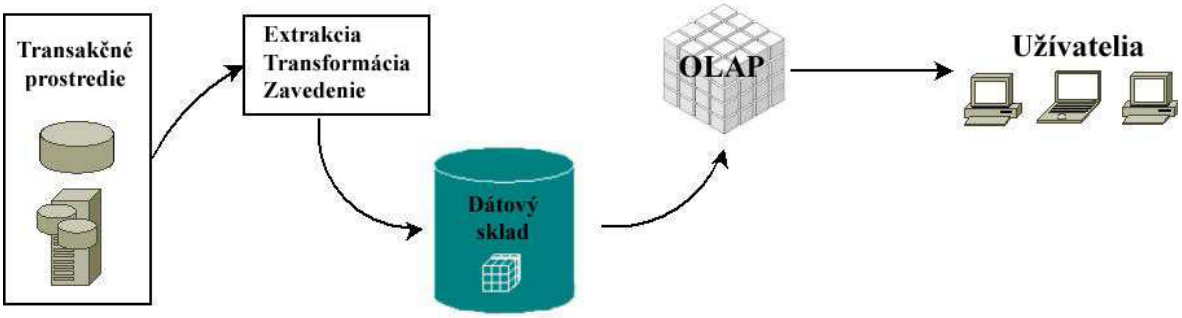
užívatelia, ktorí údaje z databázy čítajú, zapisujú, prípadne vykonávajú jednoduchšie analýzy. Údaje v transakčných databázach by mali byť uložené v normalizovaných tabuľkách, ktoré vyhovujú aspoň podmienkam 2NF alebo 3NF. To má za dôsledok veľa atomických, relačne zviazaných tabuliek. Práve preto je analýza veľkého množstva takto uložených údajov často neefektívna, pomalá a tiež ťažšie vytvoriteľná. V transakčných systémoch je využívaný princíp relačných databáz. Medzi hlavné obmedzenia relačných databáz patrí absencia komplexných analytických nástrojov a potenciálne obmedzenie údajov, ku ktorým je možné v rozumnom čase pristupovať.

2.2.2 OLAP (Online Analytical Processing)

Ide o informačné systémy pre analýzu veľkého množstva údajov. Výsledkom analýzy sú súhrny a reporty slúžiace ako podklad pre rozhodovanie a riadenie procesov. OLAP systémom sa budem viac venovať v ďalších kapitolách

2.3 Dátový sklad (Data Warehouse)

Data Warehouse možno označiť ako centrálny podporný systém, ktorý obsahuje údaje z rôznych interných a externých zdrojov, zhromažďuje ich, vytvára medzi nimi vzťahy, a tým pôsobí ako databanka pre ostatné systémy riadenia.[6] DWH môže poskytnúť len také informácie, ktoré získal zo svojich zdrojov a závisí od kvality jednotlivých údajov a ich zdrojov, a nie od použitých prostriedkov. Vytvára sa individuálne na báze existujúcich informačných systémov a dôležitých informácií.



Obrázek 1:Proces získavania analýz

Údaje sa získavajú a ukladajú do transakčných databáz, ktoré môžu byť v rôznych oddeleniach firiem, pripadne rozličných lokalitách. Tieto údaje sa v pravidelných intervaloch zozbierajú, predspracujú a zavedú do dátového skladu. Proces znázorňuje obrázok 1.

2.3.1 Dátové Tržnice (Data mart)

Sú to presne špecifikované oddelené podmnožiny dátového skladu, určené pre istú skupinu ľudí . Dáta pre dátové tržnice sú vyberané s cieľom vyhovieť špecifickým požiadavkám častí organizácie. Dátové tržnice sú často preferované podnikom ako prvý krok k vybudovaniu dátového skladu. Dátová tržnica je špeciálna verzia dátového skladu, ktorá tiež obsahuje snímok operatívnych dát. Základný rozdiel je v tom, že pri vytváraní dátovej tržnice sa zameriava na špecifické, preddefinované potreby konkrétnych užívateľov a konfiguráciu dát.[3] Dáta uložené v dátových tržniciach je možné používať predovšetkým ako podklad pre cielené analýzy.

Výhoda tohto usporiadania na viac samostatných dátových skladíšť je jednoduchšia a rýchlejšia implementácia a z toho vyplývajúce rýchlejšie prínosy pre užívateľa. Na druhú stranu, medzi nevýhody patrí, že môže dochádzať k nekonzistencii medzi jednotlivými tržnicami. Táto schéma je tiež náročnejšia na údržbu.

2.3.2 Dátová pumpa

Informácie sú do DWH prenášané z transakčných systémov pomocou softwarových komponentov – **dátových púmp**. Úlohy ktoré dátové pumpy plnia sú:

- selekcia a extrakcia dát z produkčného systému,
- transformácia extrahovaných dát,
- reštrukturalizácia podľa potrieb užívateľov DWH,
- agregácia dát podľa vybraných kritérií,
- konsolidácia dát z rôznych dátových zdrojov,
- vytváranie časových radov.

Dátová pumpa je proces napĺňania dátového skladu dátami z dátových zdrojov. Napĺňanie z transakčných systémov väčšinou prebieha v čase, keď je predpoklad nízkeho zaťaženia, aby sa nepredlžovala doba odozvy pre užívateľov týchto systémov (nočné hodiny, víkendy). Tento proces môžeme rozdeliť do troch fáz: výber, transformácia a prenos dát. Tieto nástroje a postupy sa tiež označujú ETL.[4]

2.3.2.1 Výber

V tejto fáze dochádza k napájaniu na rôzne dátové zdroje a získavaniu požadovaných informácií na ďalšie spracovanie. Pre samotný výber dát sa používajú rôzne nástroje a prístupy. Môžeme použiť nástroje, ktoré generujú kód pre výber a majú univerzálne použitie. Alebo vlastné aplikácie a externé programy vo vyšších procedurálnych programovacích jazykoch C++.

2.3.2.2 Transformácia

Táto fáza zahŕňa procesy slúžiace k premene extrahovaných dát do podoby prijateľnej pre navrhnutý dátový sklad. Ide o operácie:

- **Validácia -** Overovanie správnosti dát z dátového zdroja. Je neprípustné mať nekvalitné, zle štruktúrované dáta. Použitie nekvalitných dát vedie k chybným alebo minimálne nepresným zostavám.
- **Prečisťovanie -** Korekcie, prípadne odstránenie nesprávnych dát, ktoré sú nepostačujúce. Táto fáza sa tiež zvykne nazývať cleasing, scrubbing. Niekedy vzhľadom na nízky prínos a vzhľadom k nákladom a náročnosti nemá zmysel čistiť údaje.
- **Integrácia –** zjednotenie dát z viacerých dátových zdrojov do konzistentného formátu (dátové typy, formáty). Sem patrí napríklad problém nejednoznačnosti údajov, keď jeden typ údaju môže byť uložený v rôznych zdrojoch v rozličnom formáte (napríklad pohlavie vo formáte Male/Female, M/F, man/woman a podobne).
- **Derivácia –** výpočet odvodených dát, problém s chýbajúcimi údajmi, duplicitné dáta, zjednotenie hodnôt dát.
- **Denormalizácia –** združovanie dát z viacerých normalizovaných tabuliek do jednej denormalizovanej tabuľky (faktov) za účelom zníženia počtu spojovania tabuliek.
- **Agregácia –** vytvorenie, výpočet požadovaných súhrnov z detailných dát.

2.3.2.3 Prenos dát

Zavŕšením etapy ETL je prenos, ukladanie, alebo zavedenie do dátového skladu. Prenos spočíva v presune údajov a ich uložení do databázových tabuliek. Je to netriviálny proces a musí byť plánovaný a automatizovaný v najvyššej možnej miere. Po zavedení spravidla prebieha indexovanie, aby bol prístup k dátam optimalizovaný. Pri prvotnom naplnení dátového skladu môže ísť o veľké množstvo údajov. Následne sa dáta obnovujú v pravidelných cykloch spúšťania dátovej pumpy. Existujú tri základne scenáre pre ukladanie dát do dátového skladu:

- **Celková obnova** – celý dátový sklad bude nahradený novými dátami.
- **Prírastková obnova** – ukladajú sa len pridané dáta väčšinou za nejaké časové obdobie.
- **Synchronizácia** – ukladajú sa len zmenené záznamy.

2.3.2.4 Synchronizácia

Statické snímanie dát – nezaznamenávajú sa zmeny dát prevedené medzi dvoma prevodmi, snímkami. Definuje dva prístupy k synchronizácii. [5]

- Jednoduché statické snímanie – vytvorí sa periodický snímok zdrojových dát a tieto sa načítajú do dátového skladu.
- Snímanie podľa časových pečiatok – s využitím časovej pečiatky zmeny na detekciu zmeny záznamu. Výhodou je zníženie počtu prenášaných dát. Nevýhodou je nutnosť evidencie časových pečiatok na strane zdrojového systému.
- Snímanie porovnávaním súborov – zmeny sa identifikujú porovnávaním rozdielov pred a po snímaní, porovnávaním celých záznamov.

Inkrementálne snímanie dát. **–** zaznamenáva sa každá zmena v dátach

- Snímanie s prispením aplikácie – operatívny systém zapisuje všetky zmenené záznamy do osobitného súboru, tabuľky.
- Snímanie založené na triggroch – menené záznamy sú uschovávané do vyhradenej tabuľky pomocou databázových triggrov.
- Snímanie transakčného žurnálu – zmeny záznamov sú zaznamenávané do transakčného žurnálu.

2.3.3 Porovnanie transakčných databáz a dátových skladov

Hlavným rozdielom medzi transakčnými databázami a dátovými skladmi je, že transakčné databázy (OLTP) sú určené na ukladanie operatívnych údajov a dátový sklad je navrhnutý a optimalizovaný na rozsiahle analýzy.

Výsledkom dotazov OLTP sú tabuľky, zostavy a súhrny získané agregačnými funkciami. OLTP sú kvôli jednoduchému dotazovaniu a vďaka vylúčeniu prebytočnosti spravidla normalizované a teda operatívne údaje sú komplexné a vysoko štruktúrované. Dosahujú vysokých výkonov skôr pri transakciách on-line ako pri zložitých analýzach. Sú write-optimized, teda optimalizované na zápis.

Dátový sklad je databáza, ktorá je navrhnutá ako prostriedok na dotazovanie a analýzu. Obsahuje read-only dáta, ktoré sú vyhodnocované a analyzované omnoho efektívnejšie ako regulárne OLTP transakčné databázy. Sú teda optimalizované na čítanie, read-optimized. Vytvorenie DWH vedie priamo k zvýšeniu kvality analýzy, štruktúra tabuliek je jednoduchšia (udržuje iba potrebné informácie v jednoduchších tabuľkách), je štandardizovaná (používa dobre zdokumentované tabuľkové štruktúry) a je denormalizovaná, (znižuje viazanosť tabuliek medzi sebou a odozvu pri zložitých dotazoch).

Decentralizovanosť systémov OLTP je ďalšou prekážkou pri použití pre analýzy. OLTP nemajú k dispozícii integrovaný zdroj údajov zo všetkých operačných systémov. Údaje na základe ktorých sa tvorí analýza sú roztrúsené v rôznych, spravidla heterogénnych, systémoch OLTP. To sťažuje tvorbu komplexných analýz, keďže sa tieto údaje musia integrovať skôr, ako je možné z nich získať potrebné informácie. Problémy pri použití transakčných databáz pre analýzy nastávajú napríklad pri integrácii dát z jednotlivých systémov. Integrácia sa nemusí podariť, časová náročnosť prípadných analýz (aj u nie príliš zložitej analýzy) je vysoká alebo u veľkého objemu nahromadených údajov je vyťaženosť databáz príliš veľká.

2.3.3.1 Porovnanie OLTP a OLAP

Systémy OLTP a systémy OLAP sa dajú porovnať podľa troch hlavných kritérií:
účelu, koncepčnej schémy.

2.3.3.2 Porovnanie podľa účelu

Tabulka 1:Porovnanie OLTP a OLAP

	OLTP	**OLAP**
Hlavná funkcia	Automatizácia operácii alebo procesov	Poskytovanie optimálnych informácií pre rozhodovanie
Orientácia	Customer-oriented	Market-oriented
Užívatelia	IT profesionál	Znalostný analytik
Účel dát	Kontrola a chod základných úloh	Napomáhať s plánovaním, riešením problémov a podpory rozhodovania

2.3.3.3 Porovnanie podľa koncepčnej schémy

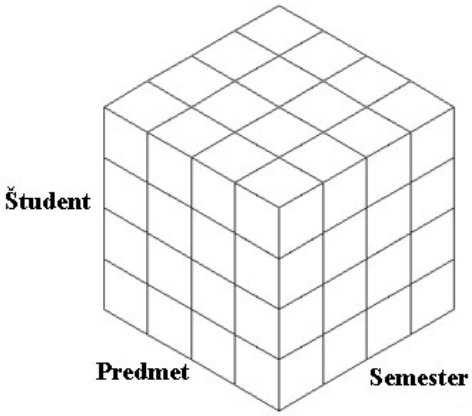
Tabulka 2:Porovnanie na základe koncepčnej schémy

	OLTP	OLAP
Hlavná funkcia	Vkladá dáta do systému	Získava informácie zo systému
Zdroj dát	Operatívne dáta: OLTP systémy sú vlastné zdroje dát	Konsolidované dáta: zdroje dát pochádzajú z externého prostredia, z viacerých OLTP databáz
Dotazy	Relatívne štandardizované a jednoduché dotazy. Návratová hodnota je niekoľko záznamov	Komplexné dotazy umocňované agregáciami
Reprezentácia dát	Snímok aktuálnych dát, obmedzené historické dáta, izolované	Historické dáta, konsolidované, rozdielna organizácia, viacrozmerný pohľad
Pohľad na dáta	Detailný, plocho relačný	Sumarizovaný, viacrozmerný
Aktivity	Podporujú každodenné firemné aktivity	Podporujú dlhodobé stratégie firmy
Veľkosť Databázy	100 MB-GB	100 GB-TB

3 Multidimenzionálny databázový model

Multidimenzionálna databáza je typ databázy, ktorý je optimalizovaný pre data warehouse a OLAP aplikácie. Multidimenzionálne databázy často využívajú zdroj z existujúcich relačných databáz. Používajú princíp dátovej kocky resp. hyperkocky, tiež nazývanej kocka (data cube, hypercube) na reprezentáciu dimenzií dát dostupných pre užívateľa. V tejto kapitole sa budem venovať logickému multidimenzionálnemu modelu.

Prevažná väčšina údajov je organizovaná v relačnej databáze v dvojrozmerných relačných tabuľkách. Výsledkom agregácie a analýzy býva obvykle multidimenzionálna dátová štruktúra – hyperkocka. Multidimenzionálny databázový model si môžeme najjednoduchšie predstaviť ako priestorovú kocku, obrázok 2. Každá kocka môže mať niekoľko dimenzií. Príkladom trojdimenzionálneho modelu môže byť hyperkocka s dimenziami Študent, Predmet, Semester:



Obrázek 2:Dátová kocka

Údaje sa nachádzajú v prienikoch jednotlivých dimenzií. Môžeme napríklad analyzovať údaje len za určité časové obdobie - semester. Alebo aby sme vyhodnotili výsledky konkrétneho študenta ktoré dosahoval počas piatich semestrov v predmete IIS.

3.1 Fakty a Dimenzie

Do multidimenzionálnych databáz sa ukladajú upravené dáta, ktoré sú podkladom pre získanie sumarizovaných a agregovaných údajov. Na rozdiel od relačných databáz sa používajú prevažne nenormalizované tabuľky, ktoré rozdeľujeme na dva druhy: na tabuľky faktov a tabuľky dimenzií. Každá kocka OLAP je teda vytvorená na základe týchto dvoch údajov:

- Fakty – numerické merné jednotky obchodovania. Prvotné fakty sa môžu kombinovať alebo vypočítať pomocou iných faktov a vytvoriť tak merné jednotky.
- Tabuľky faktov – je hlavná tabuľka, na ktorú sú viazané tabuľky dimenzií. Uchováva veľké množstvo dát. Dáta sa nemenia často. Spravidla je len jedna tabuľka faktov pre jednu kocku. Tabuľka faktov býva najväčšia tabuľka v databáze. Môže vytvárať rôzne schémy.
- Dimenzie – obsahujú logicky alebo organizačne hierarchicky usporiadané údaje. Možno povedať, že to sú textové popisy obchodovania, teda že charakterizujú dáta. Elementy sú členovia(members) niektorej dimenzie.
- Tabuľky dimenzií – obsahujú usporiadané údaje. Sú naviazané na tabuľku faktov, alebo na inú tabuľku dimenzií. Sú spravidla menšie ako tabuľky faktov a dáta sa v nich nemenia tak často. Veľmi často sa používajú časové, produktové a geografické dimenzie. Obsahujú atribúty popisujúce fakty. Tabuľky dimenzií obvykle používajú stromovú (hierarchickú) štruktúru napr.:
    - Čas: ◦ rok, ◦◦ kvartál, ◦◦◦ mesiac
    - Škola: ◦ vysoká škola ◦◦ fakulta ◦◦◦ odbor ◦◦◦◦ ročník ◦◦◦◦◦ krúžok

Máme možnosť zjemňovať - *drill-down* a zovšeobecňovať - *roll-up* hierarchickú úroveň dimenzie na nižší, alebo vyššiu úroveň (momentálna pozícia v hierarchii). Priestor pre celú kocku je dopredu určený. Jednotlivé záznamy sa nachádzajú na priesečníkoch dimenzií. S rastúcim počtom rozmerov multidimenzionálnej databázy veľmi rýchlo rastú aj požiadavky na úložnú kapacitu. V prípade, že sa na všetkých priesečníkoch dimenzií nenachádzajú údaje, kocku nazývame aj riedka kocka.

4 OLAP (On-Line Analytical Processing)

Pod názvom OLAP sú zahrnuté technológie, metódy a prostriedky, ktoré umožňujú ad-hoc analýzu multidimenzionálnych informácií. OLAP dovoľuje užívateľovi pracovať s údajmi veľmi flexibilne a analyzuje dáta podľa mnohých hľadísk.

4.1 Definícia OLAP

OLAP (On-Line Analytical Processing) je druh softwarovej technológie, ktorá slúži ku spracovaniu údajov (ich transformácii) uložených v dátovom sklade do podoby pre koncových užívateľov, teda manažérov a analytikov. Umožňuje konzistentný a interaktívny prístup k širokému spektru možných pohľadov na informácie.

4.1.1 Funkcionalita OLAP

Funkcionalitu OLAP charakterizuje dynamická multidimenzionálna analýza dát za analytickej a navigačnej podpory pre užívateľa. Implementácia OLAP je v prostredí klient/server, za poskytovania sústavnej rýchlej odozvy na dotazy, bez ohľadu na veľkosť databázy a jej zložitosť.

Funkcionalita je najčastejšie implementovaná pomocou osobitného OLAP serveru. OLAP server má buď vlastnú multidimenzionálnu databázu alebo v reálnom čase plní dátové štruktúry z inej databázy (väčšinou relačnej). Funkcionalita umožňuje:

- výpočty a modelovanie naprieč dimenziami, skrz hierarchie, naprieč členmi,
- analýza trendov v rozličných časových periódach,
- rozdeľovanie podmnožín pre zobrazovanie,
- zostup a vzostup do nižších a vyšších úrovní konsolidácie (drill-down/ drill-up),
- prienik do príslušnej detailnej úrovne dát,
- rotácie pre porovnania v nových dimenziách príslušnej oblasti,
- sústavne rýchlu odozvu na dotazy, bez ohľadu na veľkosť databázy a jej zložitosť.

4.1.2 Pravidlá pre OLAP

Existuje 12 základných pravidiel OLAP, ktoré sformuloval Dr. E. F. Codd [6].
Tieto pravidlá boli napísané pre architektúru produktu dodávateľa Arbor Software (Hyperion Solutions).

1. Multidimenzionálny konceptuálny model: OLAP by mal poskytovať užívateľovi multidimenzionálny model tak, aby zodpovedal jeho potrebám a aby tento model mohol využívať pre analýzu zhromaždených údajov.
2. Transparentnosť: To, aby užívateľ mohol naplno využívať svoju produktivitu, odbornosť a prostredie docielime tým, že technológia systému OLAP, jej databáza a architektúra výpočtu bude transparentná. Dôležitá je heterogénnosť vstupných dát, ktorú zaistíme v procese ETL.
3. Dostupnosť: Systém OLAP by mal pristupovať len k údajom, ktoré sú potrebné pre analýzu. Systém by mal navyše byť schopný pristupovať ku všetkým takýmto údajom, nezávisle na tom, z ktorého heterogénneho podnikového zdroja pochádzajú a ako často sú obnovované.
4. Stabilná výkonnosť: Užívateľ nesmie pocítiť žiadne podstatné zníženie výkonu, aj keď veľkosť databáz postupom času rastie.
5. Architektúra klient/server: Systém OLAP musí fungovať na základe architektúry klient- server. Dôležitá je cena, výkon, flexibilita, interoperatibilita.
6. Generická dimenzionalita: Každá dimenzia údajov musí byť ekvivalentná v štruktúre aj operačných schopnostiach.
7. Dynamická manipulácia s riedkymi maticami: Systém OLAP musí byť schopný prispôsobiť svoju fyzickú schému na konkrétny analytický model, ktorý optimálne ošetrí riedke matice za udržania požadovanej úrovne výkonu.
8. Podpora viacerých užívateľov: Systém OLAP musí byť schopný podporovať viac užívateľov alebo skupiny užívateľov pracujúcich súčasne na konkrétnom modeli.
9. Neobmedzené operácie naprieč dimenziami: Systém OLAP musí rozoznať dimenzionálne hierarchie a automaticky vykonávať výpočty v rámci dimenzií a medzi dimenziami.
10. Intuitívna manipulácia s dátami: Užívateľské rozhranie musí umožňovať všetky manipulácie s údajmi v pre neho prístupnom (user-friendly) prostredí. Napríklad pre operácie ako drill down a drill up.
11. Flexibilné výstupy: Schopnosť usporiadať riadky, stĺpce a bunky spôsobom, ktorý umožní analýzu a intuitívnu prezentáciu analytických zostáv.

12. Neobmedzené dimenzie a úrovne agregácií: V závislosti na požiadavkách podnikania môže mať analytický model viac dimenzií, pričom každá z nich môže mať viacnásobné hierarchie. Analytický model by nemal byť umelo obmedzovaný počtom dimenzií alebo úrovňou agregácií.

4.2 Implementačné varianty OLAP

Z hľadiska vlastného uloženia dát môžeme databázové systémy rozdeliť do troch skupín, na základe implementačných variantov, MOLAP, ROLAP a HOLAP.

4.2.1 MOLAP

Multidimenzionálny OLAP je technológia, ktorá na implementáciu multidimenzionálneho modelu využíva pre tento účel špeciálne vyvinutý OLAP server s vnútornou architektúrou databázy optimalizovanou pre multidimenzionálne dáta.

Využíva dvojvrstvovú architektúru klient/server. Je to model, v ktorom prebieha spracovanie všetkých funkcií aplikácií na dvoch komponentoch – klient a databázový server. Údaje sú ukladané do špecializovanej multidimenzionálnej databázy, do n–rozmerného priestoru. Počet dimenzií (n) zodpovedá počtu pohľadov na údaje. Uloženie údajov je závislé na ich predkompilácii a obmedzených možnostiach dynamicky pridávať nové pohľady. Samozrejmosťou je alokácia diskového priestoru v závislosti na počte dimenzií.

Výhodou je veľmi vysoká rýchlosť spracovania údajov s priamym prístupom Užívateľov. Technológia je vhodná pre menšie aplikácie s obmedzenou multidimenzionalitou. Databáza musí byť periodicky kompilovaná. Dáta sa získavajú buď z dátového skladu, alebo operatívnych zdrojov. Údaje ukladáme vo vlastných dátových štruktúrach.

4.2.1.1 Charakteristika MOLAP

- dvojvrstvová architektúra klient/server,
- pred uložením dát na disk potreba alokácie priestoru,
- veľká rýchlosť spracovania dotazov,
- potrebná stála rekompilácia.

4.2.1.2 Porovnanie MOLAP a ROLAP

MOLAP sa odlišuje hlavne tým, že potrebuje predpočítané dáta a ich uloženie v kocke. Ukladá ich optimalizované vo viacrozmernej polovej úložnej štruktúre a nie do relačnej databázy.

Výhody MOLAP

- rýchle vyhodnotenie dotazov, vďaka optimalizovanému uloženiu, viacrozmernému indexovaniu a caching,
- vyžaduje menší úložný priestor (oproti relačnému modelu) vďaka kompresívnym technikám,
- automatický výpočet agregovaných dát,
- umožnenie použitia indexovacích techník na poli (ako natural indexing).

Nevýhody MOLAP

- načítavanie dát je zdĺhavé, hlavne pri veľkých množstvách, využíva sa inkrementálny prístup, prípadne načítavanie len zmenených dát,
- zložitejšie dotazovacie modely pri dimenziách s veľkou mohutnosťou.

4.2.2 ROLAP (Relačný OLAP)

Vznikli snahou prispôsobiť relačný model DWH modelu. Je založený na relačných databázach a trojvrstvovej architektúre klient/server. Vykonávanie aplikačného programu je rozdelené medzi tri komponenty: klient (prezentačná funkcia), aplikačný server (obchodná logika) a databázový server (vlastná manipulácia s údajmi), čím je dosiahnutá väčšia flexibilita v prípade zmien.

Nevýhodou sú vyššie náklady na zavedenie a zabezpečenie systému. Výhodou je otvorenosť prostredia, využitie SQL. Ako ďalšia výhoda sa uvádza, že sa nevyžaduje duplicita dát (transakčné dáta sú uložené v relačnom systéme), avšak z dôvodu zvýšenia výkonu, agregácie a časových rezov je potrebné dáta aj tak duplikovať. Popis architektúry je zjednodušený. Stupeň kompilácie dát závisí na voľbe administrátora. Pre modelovanie štruktúry dátového skladu nad relačnou databázou sa používajú logické schémy snow-flake (snehová vločka). U databáz dochádza k redundanciám, ktoré sú potrebné na niekoľkonásobné skrátenie doby odozvy. Relačný model je dvojrozmerný. Viacrozmerné pohľady sa musia vyriešiť pomocou dôkladnej indexácie a duplikácie tabuliek.

Najjednoduchší model vytvorený pomocou dimenzionálneho modelovania sa skladá z tzv. faktovej tabuľky, ktorej primárny kľúč je zložený z rôznych dimenzií. Najčastejšie používaný dimenzionálny model sa nazýva hviezdicová schéma – každá dimenzia faktovej tabuľky je nahradená cudzím kľúčom, zodpovedajúcim dimenzii tabuľky. Vrstva metadát organizuje údaje podľa tematických okruhov.

ROLAP je flexibilná technológia pri práci nad veľkým rozsahom dát a väčším množstvom multidimenzionálnych pohľadov. Databáza sa nemusí neustále rekompilovať a neexistuje problém riedkeho zaplnenia priestorových dát. Rýchlosť spracovania je však oveľa nižšia oproti multidimenzionálnej tabuľke.

4.2.2.1 Charakteristika ROLAP

- snaha o prispôsobenie relačnej DB ,
- trojvrstvová architektúra klient/server,
- nad relačnou DB sa využíva schéma snehovej vločky,
- databáza nie je normalizovaná,
- viacrozmerný pohľad riešený indexáciou a duplikáciou tabuliek,
- čas vedený len ako pevný dátum,
- existuje možnosť použitia kombinácie MOLAP/ROLAP.

4.2.2.2 Porovnanie ROLAP a MOLAP

Výhody ROLAP

- ROLAP je považovaný za lepšie škálovateľný, hlavne pri modeloch s dimenziami s veľkou mohutnosťou (rádovo miliónmi členov),
- Načítavanie dát je rýchlejšie vďaka rozmanitosti nástrojov a možnosti prispôsobenia dátového modelu,
- Dáta sú uložené v štandardnej relačnej databáze a môžu byť prístupne aj pre SQL reportovacie nástroje,
- ROLAP nástroje efektívnejšie pracujú s neagregovanými údajmi (textovými údajmi) ako MOLAP.

Nevýhody ROLAP

- ROLAP nástroje vykazujú nižšiu výkonnosť ako MOLAP nástroje,
- Načítavanie agregačných tabuliek musí byť riadené pomocou samostatného ETL Kódu,
- Niektoré špeciálne techniky MOLAP nie sú dostupné, napríklad ako hierarchické indexovanie. ROLAP využíva najnovšie vylepšenia jazyka SQL, operácii ako CUBE, ROLLUP.

4.2.3 HOLAP (Hybridný OLAP)

Hybridný OLAP kombinuje výhody MOLAP a ROLAP. Princíp činnosti je v možnosti voľby užívateľa, ktorá časť údajov ostane v relačnej forme (informácie s vyššou mierou detailov) a ktorá bude agregovaná do multidimenzionálnej databázy (informácie s vyššou mierou agregácie).

Základnou podmienkou je transparentné použitie MOLAP pre dáta s vyšším stupňom agregácie a ROLAP pre prácu s dátami na detailnejšej úrovni.

5 Moderné technológie

5.1 Adobe FLEX

Adobe FLEX je skupina produktov, ktoré ponúkajú najkomplexnejšie riešenia vo vývoji RIA aplikácií. Sú navrhnuté tak, aby uspokojili všetky potreby vyplývajúce z tvorby enterprise a webových riešení. FLEX sa skladá z programového modelu FLEX framework, integrovaného prostredia Flex Builder a robustných služieb pre integráciu dát FLEX Data Services(FDS), ktoré umožňujú developerom dramaticky zvýšiť produktivitu práce, zvýšiť konkurencie schopnosť výrobku a hlavne integrovať aplikácie do existujúcich aplikácii, frameworkov a webových stránok. Aplikácie vyvíjané vo FLEXe využívajú to najlepšie, čo ponúka internetový prehliadač a Flash Player. Ten je naištalovaný na 98% osobných počítačoch.

Flash Player je runtime nezávislý na platforme, ktorý ponúka výkonný virtuálny stroj integrujúci prostriedky pre vizualizáciu textu, podporu pre tlač, manipuláciu s dátami, animáciami a multimedálnym obsahom. FLEX poskytuje komponenty na strane klienta, ktoré umožňujú aplikáciam komunikovať so vzdialeným serverom prostredníctvom SOAP webových služieb, REST a protokolu HTTP alebo protokolu, ktorý je založený na soketoch. Pre sofistikovanejšie riešenia sú k dispozícií Flex Data Services, ktoré zahrňujú podporu pre posielanie správ tzv. *„publish/ subscribe messaging“*. Okrem toho je možný real-time streaming dát a priama integrácia s existujúcimi Java objektami na strane servera.

FLEX ponúka veľmi efektívny vývojársky model založený na existujúcich procesoch a štandardoch a bere si z nich to najlepšie, čo priniesol vývoj internetových aplikácií. Poskytuje prostriedok na tvorbu uživatelského rozhrania založeného na XML tzv. MXML a na tvorbu logiky na strane klienta implemetáciu ECMAScriptu tzv. ActionScript. FLEX taktiež poskytuje nástroje na testovanie. Vyššie popísané vlastnosti budú diskutované v nasledujúcich podkapitolách.

5.1.1 Platforma FLEX

Ako vidieť na obrázku č.3[1] , tak FLEX platforma pozostáva zo 4 nasledujúcich častí.

[1] Prevzaté z http://www.adobe.com/devnet/flex/

Obrázek 3: Platforma FLEX

- Flex Development Kit (SDK) - pozostáva zo základnej knižnice komponent, vývojového jazyka a kompileru pre FLEX aplikácie.
- Flex Builder IDE – Vývojové prostredie založené na prostredí Eclipse s integrovaným debuggerom a vizuálnych nástrojov pre tvorbu GUI.
- Flex Data Services – serverové riešenie založené na technológií Java. Poskytuje prostriedky pre synchronizáciu dát riešenie konfliktov a zasielanie správ.
- Flex Charting – knižnica pre rozšírenú tvorbu grafových komponent.

Podrobný popis bude uvedený neskôr.

5.1.2 Flash Player

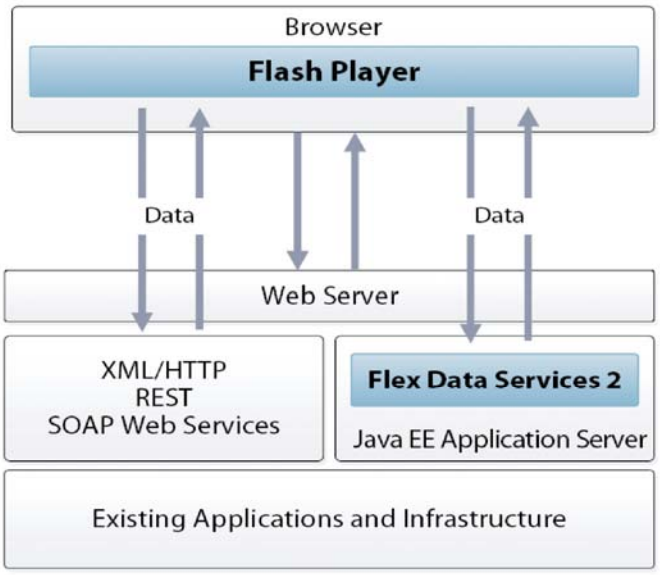
Ďalšie diskusie by nemali význam bez toho, aby sme spomenuli runtime pre FLEX aplikácie. Je ním práve Flash Player.

Aplikácie vyvinuté vo FLEXe sú distribuované v podobe skompilovaného bytekódu, ktorý interpretuje Flash Player. Flash Player poskytuje konzistentné prostredie nezávislé na internetovom prehliadači či operačnom systéme. Možnosti Flash Playera sú zhrnuté v nasledujúcich bodoch:

- Rozsiahle možnosti pre renderovanie textu,
- Silné grafické API,
- Integrované audio-video kodeky,
- Výkonný virtuálny stroj,
- Účinná správa pamäti tzv. Garbage Collector.

5.1.3 Flex architektúra

Architektúra FLEX aplikácií bola navrhnutá tak, aby využívala stávajúce možnosti modelu webových aplikácií. Klientská časť aplikácie sa distribuuje v sieti ako binárny súbor, ktorý obsahuje skompilovaný bytekód. Súbor je umiestený na web server podobne ako HTML súbor či iný dokument. Ak internetový prehliadač pošle požiadavok na daný súbor, súbor je stiahnutý a následne spustený v prostredí Flash Playeru. Na obrázku č.4[2] je ilustrovaný príklad, kedy aplikácia požaduje dodatočné dáta prostredníctvom HTTP protokolu, webovej služby (SOAP). Klient založený na FLEXe nie je závislý na špecifickom serverovom riešení.



Obrázek 4: FLEX architektúra

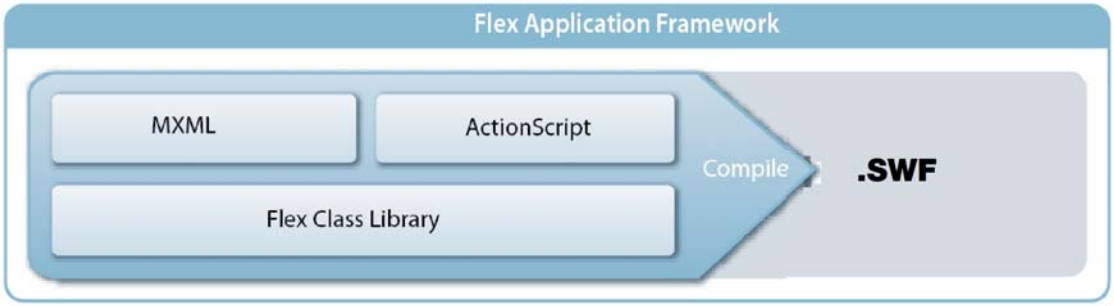
V prípade, že FLEX klient využíva služieb Flex Data Services, je možné využívať i rozšírené služby a možnosti. Klient môže vzdialene volať Java objekty rovnako, ako požadovať real-time dáta, posielať správy ostatným klientom, či zapojiť do hry existujúci JMS systém. FDS aplikácia beží na servery.

[2] Prevzaté z http://www.adobe.com/devnet/flex/

5.1.4 Vývojový model a aplikačný framework v prostredí FLEX (SDK)

Vývojový proces pre FLEX aplikácie je inšpirovaný tým najlepším, čo prináša vývoj aplikácií v Jave, C#, C++ alebo iných tradičných nástrojoch. Vývojár napíše zdrojový kód v MXML a ActionScripte, ten je následne skompilovaný do bytekódu, ktorého výsledkom je binárny súbor s koncovkou *.swf.

Ako ukazuje obrázok č.5[3] aplikačný framework pozostáva z MXML, ActionScriptu a Flex knižníc.



Obrázek 5: Aplikačný framework

MXML je dialekt XML a je použitý na deklaratívne definovanie prvkov uživateľského rozhrania. Actionscript vykonáva logiku na strane klienta. Flex class library je súbor komponent, manažerov a služieb. Medzi najväčšie benefity modelu založeného na komponentách je možnosť používať pri implemetácií vstavané komponenty, či tieto komponenty rozširovať podľa potreby.

5.1.4.1 ActionScript 3.0

ActionScript je objektovo-orientovaný programovací jazyk, ktorý vychádza z implementácie ECMAScriptu rovnako ako napríklad JavaScript. Avšak ActionScript obsahuje možnosti, ktoré presahujú možnosti JavaScriptu. V čase vývoja môže programátor využívať programatorské techniky ako je silné typovanie, interface, delegovanie či namespace, alebo zachytávanie výnimiek. Počas behu aplikácie sa prejaví výrazné rozdiely medzi JavaScriptom a ActionScriptom a to, že ActionScript sa za chodu skompiluje do natívneho kódu Flash Playeru. Ako výsledok potom dostaneme oveľa väčšiu výkonnosť aplikácie a tiež efektívnejšiu správu pamäti, ako pri interpretovanom JavaScripte.

[3] Prevzaté z http://www.adobe.com/devnet/flex/

5.1.4.2 MXML – značkovací jazyk pre FLEX aplikácie

Rovnako ako HTML i MXML je značkovací jazyk, ktorým definujeme uživateľské rozhranie. Avšak na rozdiel od HTML, MXML poskytuje okrem abstrakcie na strane klientskej vrstvy aj väzby medzi GUI a aplikačnými dátami.

5.1.4.3 FLEX class library

FLEX obsahuje rozsiahlu knižnicu tried pre prácu s komponentami, dátami a definuje správanie FLEX komponent. V nasledujúcich odstavcoch popíšem základne piliere tejto knižnice.

Vizuálne komponenty

Vývojárom je ponúknutá sada vstavaných komponent, čo do veľkej miery uľahčuje vývojárom prácu. Vývojár tak má možnosť implementovať profesionálne vypadajúce aplikácie vo veľmi rýchlom tempe. Komponenty je možné rozširovať pridávaním nových metód, vlastností, nového vzhľadu. Charakter komponent sa v priebehu vývoja dá definovať pomocou CSS, v dobe behu potom pomocou ActionScript API.

Low-Level komponenty

Tento druh komponent sa vo FLEX aplikáciach stará o pristupovanie k dátam zo širokého spektra možných zdrojov. Patria sem napríklad komponenty pre volanie webových služieb SOAP, prácu s XML alebo inými dátami cez HTTP protokol. Flash Player podporuje prácu s binárnymi soketmi, čo dáva nespočetné možnosti v návrhu vlastných protokolov a prácou s nimi.

Po prenose dát je nutnosť dáta serializovať a ukladať ich do presne definovaných dátových štruktúr. K tomu je k dispozícii napríklad natívna podpora XML, či rôzne druhy kolekcií. FLEX automaticky naviaže dáta na vizuálne prvky, ktoré ich zobrazujú bez nutnosti explicitného príkazu po zmene dát.

Komponety pre definovanie chovania aplikácie

Chovanie vo FLEX aplikáciach je prednastavené, ale je možné ho taktiež meniť. Je to vlastne kombinácia spúšťača, akcie a požadovaného efektu. Je možné si pod tým predstaviť napríklad udalosť stlačenia tlačítka myši a následnej zmeny stavu vizuálnej komponenty napr. tlačítka. Efekty je možné taktiež definovať a upravovať podľa potreby.

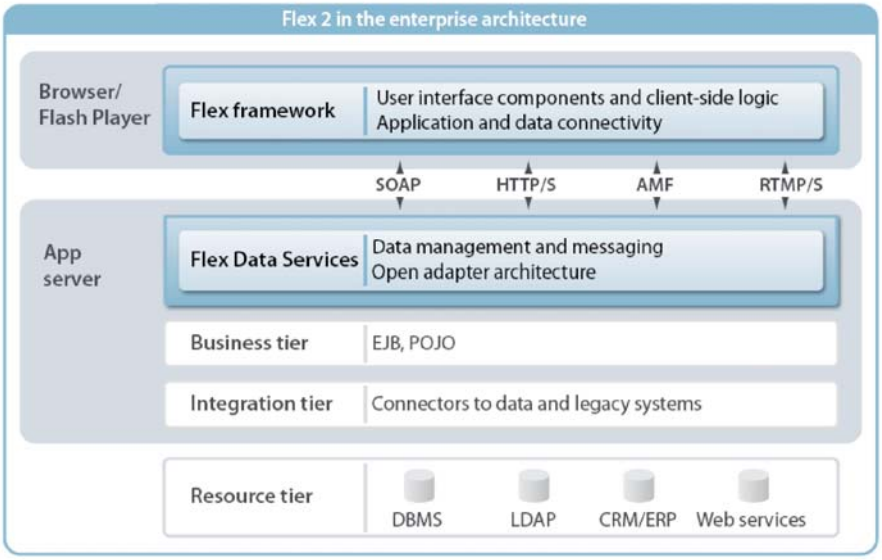
5.1.5 Flex Charting

Flex Charting poskytuje rozsiahlu knižnicu interaktívných diagramov a grafov, ktorá umožňuje bohatú prezentáciu a interaktívnu analýzu dát. Pomocou zakreslovacích komponent, ktoré sa dynamicky renderujú u klienta a ponúkajú plnú podporu pre model dátových väzieb a udalostí softwaru FLEX, je jednoduché pridať možnosť prechodu na podrobnejšiu úroveň, zobrazenia

prekrývaných vrstiev a ďalšie interaktívne prvky, takže grafy majú väčšiu informačnú hodnotu. Všetky grafy je možné rozšíriť o ďalšiu funkcionalitu, prípadne môžu vývojári využiť základné triedy k vytváraniu vlastných typov grafov.

5.1.6 Flex Data Services(FDS)

Aplikácie Flex Data Services ponúkajú sadu výkonných funkcií pre správu dát na servery, ktorá vývojárom umožňuje rýchlo realizovať dátovo náročné aplikácie. FDS sa inštaluje ako štandardná webová aplikácia na platforme J2EE a obohacuje klientský framework FLEX o výkonné dátové prepojenie s existujúcimi serverovými dátami a business logikou. Aplikácia FDS, založená na robustnej architektúre predávania správ, sa integruje so stávajúcim štandardizovaným middleware a ponúka služby, ktoré automatizujú synchronizáciu dát medzi klientom a serverom, pridávajú podporu pre zasielanie dát v reálnom čase (data push) a komunikáciu typu pub/sub a umožňujú tvorbu aplikácií pre podporu spolupráce i aplikácií založených na občasnom pripojení. Nasledujúci obrázok č. 6[4] vykresľuje integráciu FDS do stávajúcej architektúry.



Obrázek 6: Integrácie FDS

5.1.7 OLAP a FLEX

V tejto podkapitole opíšem, ako do celej koncepcie OLAP aplikácií zasahuje FLEX technológia.

S príchodom novej verzie frameworku FLEX verzie 3.0, dostali vývojari do rúk nástroj s funkcionalitou OLAP. Tá je vo frameworku prístupna prostredníctvom OLAP API. Celý koncept je postavený na reprezentácií kocky - OLAPCube tzv.*„in-memory“*, t.z je v RAM pamäti. Z toho vyplývajú nasledujúce skutočnosti:

[4] Prevzaté z http://www.adobe.com/devnet/flex/

- Riešenie pre rozumné množstvo dát, do 50000 záznamov - konzumuje značné systemové prostriedky,
- Rýchle dotazovanie,
- Schopnosť analyzovať dáta offline, bez prítomnosti OLAP serveru,
- OLAP API je implementovné tak, aby ho bolo možné rozšíriť na komunikáciu, so skutočným OLAP serverom.

5.1.7.1 Postup pri vytvárani analýz vo FLEXe

Definovanie schémy kocky- ako prvé pri vytváraní kocky je nutné určiť dimenzie kocky. Dimenzie dovolujú kategorizovať dátové polia do nezávislých entít ako napríklad čas, poloha, produkt a podobne. Každá dimenzia obsahuje kolekciu atribútov, kde každý atribút korešponduje s jedným stĺpcom v tabuľke *flat-data*[5].

Schéma v aplikácií FLEX môže byť reprezentované staticky prostredníctvom MXML(viď vyššie) alebo dynamicky prostredníctvom jazyka ActionScript.

Ukážka statickej definicie schémy: Táto schéma pozostáva z troch dimenzií *Zákazník, Štvrťrok,* a *Produkt*.

```
<mx:OLAPCube  >
      <mx:OLAPDimension name="ZakaznikDim">
            <mx:OLAPAttribute name="Zakaznik" dataField="zakaznik"/>
            <mx:OLAPHierarchy name="ZakaznikHier" hasAll="true">
                  <mx:OLAPLevel attributeName="zakaznik"/>
            </mx:OLAPHierarchy>
      </mx:OLAPDimension>
      <mx:OLAPDimension name="ProduktDim">
            <mx:OLAPAttribute name="Produkt" dataField="produkt"/>
            <mx:OLAPHierarchy name="ProduktHier" hasAll="true">
                  <mx:OLAPLevel attributeName="Produkt"/>
            </mx:OLAPHierarchy>
      </mx:OLAPDimension>
      <mx:OLAPDimension name="StvrtrokDim">
            <mx:OLAPAttribute name="Stvrtrok" dataField="stvrtrok"/>
            <mx:OLAPHierarchy name="StvrtrokHier" hasAll="true">
                  <mx:OLAPLevel attributeName="Stvrtrok"/>
            </mx:OLAPHierarchy>
      </mx:OLAPDimension>
      <mx:OLAPMeasure name="Trzba" dataField="trzba"
aggregator="SUM"/>
</mx:OLAPCube>
```

Ako prvé definujeme dimenzie, a následnie miery. Každá dimenzia obsahuje

- Atribút *OLAPAttribute* asociuje dátove pole v zdroji dát s danou dimenziou

[5] flat-data je formát súboru, ktorý obsahuje záznamy, kde každý záznam obsahuje množinu dvojíc meno:hodnota napr. {name:”Daniel”,surname:”Janoska”}

- Hierarchiu *OLAPHierarchy* a uroveň v hierarchií *OLAPLevel*. Dimenzia obsahuje hierarchiu s aspoň jednou úrovňou

Miera OLAPMeasure - definuje dátove pole zo zdroja, ktoré má byť použité na vstupe agregačnej funkcie.

Zostavenie kocky (začiatok)- v prípade, že je shéma definované, priradíme kocke datový zdroj XML, kolekciu, výsledok SQL dotazu a zavoláme metódu objektu OLAPCube *refresh()*.

Zostavenie kocky (koniec)- vybudovanie kocky prebieha asynchrónne. Počas tohto procesu kocka vysiela udalosti *CUBE_PROGRESS* a *CUBE_COMPLETE*, ktoré je možno odchytiť a patrične na ne reagovať

Dotazovanie – OLAP dotaz je zložený z troch častí stĺpcovej, riadkovej a rezovej osi. Každú z týchto osí je potreba naplniť množninou datového typu *OLAPSet*. Pre komplexnejšie dotazy je možné zaviesť hierarchiu, spojenie alebo crossjoin.

Zobrazenie výsledkov - v okamžiku, kedy je výsledok(*OLAPResult*) k dispozicií je možné určiť komponentu na vizualizáciu výsledku, ako je napríklad graf, či tabuľka.

OLAP a FLEX podrobnejšie

Komplexné dotazy- ako už bolo spomenuté vyššie, podporuje spojenie, crossjoin a tvorbu hierarchií, operácie slice a filter.
Implementácia vlastných agregačných funkcií – prednastavená agregačná funkcia je sumarizácia. OLAP kocka vo FLEXE podporuje okrem iného počet, priemer, minimum, maximum a je možné implementovať vlastnú agregačnú funkciu.

5.2 Adobe AIR

AIR je označenie pre nový runtime nezávislý na operačnom systéme, ktorý dovoľuje webovým vývojárom využiť ich programátorské schopnosti (Flash, Flex, HTML, JavaScript, and PDF) nato, aby vyvinuli aplikácie určené pre desktop.[7]

V podstate je to platforma medzi desktopom a internetovým prehliadačom, ktorá kombinuje bohatosť a jednoduchosť prostriedkov pri tvorbe webových stránok a funkcionalitu z desktopového modelu. Pre kompletnosť dodám, že AIR nie je nízko úrovňový runtime. To znamená, že zrejme na ňom nepôjde vybudovať operačný systém ako taký. AIR je primárne určený na zabudovanie webových aplikácií na desktop.

5.2.1 Primárne AIR technológie

Rozlišujeme dva typy technológii, na ktoré sa AIR zameriava: aplikačné a dokumentárne technológie.

5.2.1.1 Aplikačné Technológie

Aplikačné technológie tvoria základ pre aplikácie postavené na technológií AIR. Sú nimi Flash a HTML.

FLASH

Jednou z kľúčových technológií , na ktorých je AIR vystavaný je Flash Player. Špeciálne ide o Flash Player 9, čo znamená, že zahrňuje ActionScript 3 rovnako ako virtuálny stroj Tamarin, ktorý bude v ďalšej verzií prehľiadača FireFoxu interpretovať JavaScript. Niektoré existujúce Flash Player API bolo pre AIR upravené, vylepšené alebo doplnené. Medzi niektoré funkcie môžme uviesť:

- JIT interpreter ActionScriptu pre rýchly beh aplikácií,
- Plná podpora pre sieťové spojenie, vrátane HTTP, RTMP, binárne a XML sokety,
- Kompletný engine pre renderovanie založené na vektoroch,
- Rozsiahla podpora pre multimédia, bitmapy, audio, video.

HTML

Druhá technológia podporovaná v AIR je HTML. Je tu implementovaný plný HTML renderovací engine, ktorý podporuje:

- HTML,
- JavaScript,
- CSS,
- XHTML,
- Document Object Model (DOM).

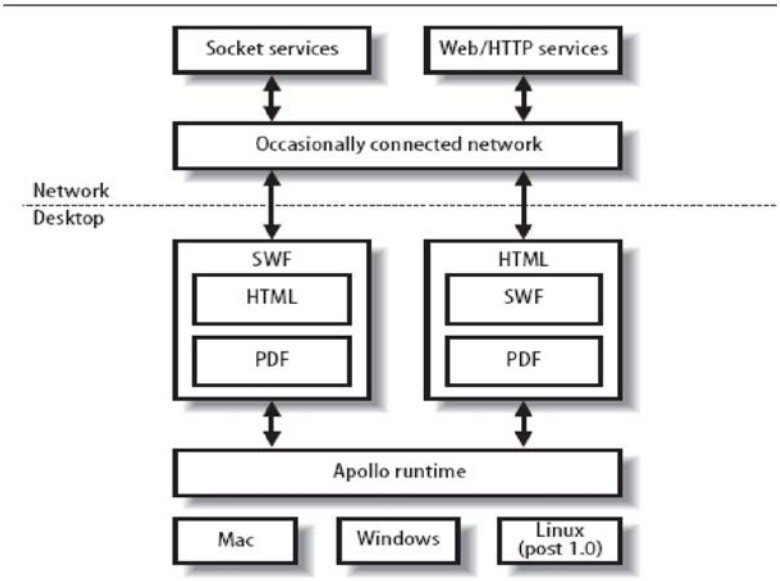
To znamená, že sa vývojár zaobíde i bez technológie Flash. Na vybudovanie plnohodnotnej aplikácie stačí HTML a JavaScript. HTML engine, ktorý používa AIR je open source project WebKit. Využíva sa napríklad ako engine do Safari prehliadača.

5.2.1.2 Dokumentačné technológie

Týmto druhom technológie sa myslia aplikácie, kde primárnou funkciou je zobrazovanie a interakcia s elektronickými dokumentami. Do tejto aplikačnej domény zaraďujeme PDF a HTML.

5.2.1.3 Aplikácie založené na technológií AIR

Na obrázku č.7[6] je znázornená integrácie jednotlivých dostupných technológií.



Obrázek 7: AIR platforma

[6] Prevzaté z http://www.adobe.com/devnet/air/

Aplikácia teda môže pozostávať s kombinácií nasledujúcich technológií.

- Flash, Flex,
- Flash s HTML obsahom,
- HTML/JavaScript ,
- HTML/JavaScript s Flash obsahom,
- Kombinácia všetkých predcházajúcich + PDF.

5.2.2 AIR API

AIR poskytuje bohatú sadu programového API a taktiež prostriedky na úzku spoluprácu s desktopovým zariadením. Medzi hlavné programové API okrem toho, čo poskytuje Flash Player a WebKit patrí:

- Úplné súborové I/O API,
- Kompletné natívne API pre prácu s oknami,
- Kompletné natívne API pre prácu s menu ponukami,
- Online/Offline API na detekciu sieťovej konektivity,
- API pre prácu s dátovou cache a synchronizačné,
- API pre prácu s dátovou cache a synchronizačné,
- Podpora pre editáciu vzhľadu aplikácie,
- API pre lokálne úložisko,
- API pre systémovú notifikáciu,
- API pre aktualizáciu aplikácie,
- SQLite API.

Vlastnosti

Ako bolo zmienené v predchádzajúcom texte aplikácie bežiace v prostredí prehliadača nie vždy dokážu uspokojiť potreby z hľadiska uživateľskej interakcie. To vedie k tomu, že ovládanie aplikácie je ťažkopádne a uživateľ má problém si zvyknúť na ovládanie.

AIR a aplikácie postavené na tejto technológií tieto prekážky odburávajú vďaka nasledovnej funkcionalite, ktorú prinášajú.

- Inštalačná rutina je súčasťou aplikácie,
- Podpora pre ukladanie do schránky,
- Podpora pre systémovú notifikáciu,
- Ikony,
- Schopnosť aplikácie bežať na pozadí.

- Podpora drag-and-drop, a to
    - medzi OS a AIR aplikáciou,
    - medzi AIR aplikáciami,
    - medzi natívnymi a AIR aplikáciami.

Zhrnutie AIR

V nasledujúcej časti sa zameriam na výhody a nevýhody spojené s technológiou AIR.

Výhody

- Rýchle spracovanie - porovnatelný výkon s platformou Java resp. .NET.
- Platformovo nezávislý - aplikácie sú spustitelné na systémoch Windows, Linux, Mac OS.
- Jednoduchá konverzia existujúcich HTML aplikácií.
- Jednoduchá inštalácia.
- Integrácia SQLite - možnosť využívať rýchlej lokálnej databázy.
- Podpora pre tzv. „bohatý“ obsah a multimédia.
- Deklaratívne programovanie - je momentálne najlepšou a najrýchlejšou cestou tvorby GUI.

Nevýhody

- Limitovaná rozšíritelnosť - aplikácie postavené na AIR majú prístup k súborovému systému a schránkam tzv.clipboard. Podpora natívnych okien, podpora drag and drop, notifikácie.
- Databázová podpora - priamy prístup je obmezený na SQLite a webové služby.
- Properietálna technológia - v buducnosti sa môžu zmeniť licenčné podmienky.
- Podpora vláken - zatiaľ žiadna, je horúcim kandidátom pre nasledujúcu verziu 2.0.
- Bezpečnostné riziká - z vývojárskej perspektívy sú tieto riziká značné oklieštené sandboxom, ale zo strany uživateľov to môže byť problém. Používajú sa certifikáty.

5.2.3 Bezpečnosť

Čo sa týka bezpečnosti, tak na tú sa kladie mimoriadna pozornosť vzhľadom na lokálny charakter bežiacej aplikácie. AIR aplikácie môžu byť podpísané digitálnym podpisom a kontolované za behu. Administátor môže prostredníctvom napríklad registrového klúča zistiť, ktorú aplikáciu je možno

nainštalovať a či je možná aktualizácia tejto aplikácie. A pretože sú aplikácie postavené na AIR natívne, firewall môže zvážiť jej spustenie či blokovanie na základe individuálnych potrieb.

6 Analýza

6.1 Súčastný stav riešenej problematiky

Kapitola popisuje podrobnú analýzu riešenia problému. Prvá podkapitola je venovaná charakteristike problémovej oblasti dátových skladov, ich prínos. Druhá podkapitola sa venuje prieskumu existujúcich systémov na trhu OLAP aplikácií. Ďalšia podkapitola sa venuje očakávaniam požadovanej aplikácie OLAP a stručnej špecifikácie riešenia. Východiskom pre túto analýzu sú skúsenosti z renovovaných nástrojov a prác v oblasti OLAP technológie. Cieľom je pokryť všetky bežné požiadavky kladené na aplikáciu.

6.1.1 Prínos dátového skladu

K vybudovaniu dátového skladu vedú obyčajne požiadavky spoločnosti, ktoré potrebujú pre svoju činnosť najrôznejšie analýzy, ktoré bohužial nie je možné implementovať v prevádzkovom systéme. Pretože so zväčšovaným počtom zákazníkov a s tým spojeným nárastom objemu dát nie je možné prevádkovať analýzy nad OLTP systémy. Odozvy sú neúnosne pomalé a niektoré požiadavky nie je možné vôbec riešiť.

Obchodný potenciál, ktorý vytvárajú dáta koncentrované v dátovom sklade, je skutočne vysoký. Na týchto základoch je možné totiž vytvoriť rozličné reporty a analýzy, ktoré slúžia predovšetkým obchodnému a marketingovému oddeleniu k meraniu úspěšnosti súčasných služieb a príprave nových ponúk. Ako príklad môžem uviesť marketingové oddelenie, ktoré pomocou analýz skúma, čo by sa stalo, keby sa zmenily ceny za nejakú službu. Alebo obchodníkov, ktorí majú na starosti velkých zákazníkov, zaújma, aký je profil zákazníka, aby mu mohli ponúknuť dalšie zaujmavé služby alebo produkty.

6.1.2 Priezkum existujúcich systémov

V tejto podkapitole uvádzam rešerše problematiky Business Intelligence v databázových systémoch. Systémy, ktoré som vyhodnocoval, slúžia ako námet vlastností a funkcí pre implementovanú aplikáciu. Hodnotenie sa zameriava predovšetkym na analytické a reportovacie schopnosti týchto aplikácií.

MS SQL Server

Microsoft SQL Server je DBMS vyvinutý spoločnosťou Microsoft. Je najčastejšie používaný pre malé a stredné databázy. Implementácia Business Intelligence na platforme MS SQL Serveru sa rozdeľuje na tri časti podľa zamerania a funkcionality[8]:

- **Integration Services** – získavanie dát z externých aj interných nehomogénnych zdrojov, ich transformácia, integrácia, syntéza (ETL funkcionalita).
- **Analysis Services** – analýza dát, obohatenie dát, hierarchizácia dát, hľadanie závislostí. Umožňuje organizovať dáta do intuitívnych štruktúr pre podporu preddefinovaných aj jednorazových dotazov, ktoré dokážu identifikovať pravidlá, vzťahy a trendy.
- **Reporting Services** – prezentácia a distribúcia dát, forma a rozsah výstupov. Platforma pre generovanie zostáv umožňuje vytvárať zostavy (reporty) v reálnom čase aj podľa definovaných časových plánov. Zostavy môžu byť prístupné z webového prehliadača, kancelárskych aplikácií, špecializovaných obchodnýc nástrojov.

Reportovacie služby

Reportovacie služby plnia funkciu podpory rozhodovania, slúžia na návrh reportu a starajú sa o generovanie výstupu v elektronickej alebo papierovej forme. Reporty môžu byť statické alebo interaktívne (pomocou rôznych ovládacích prvkov môžeme report prispôsobovať).

Podobne aj Reporting Services od Microsoftu poskytuje vstupy SQL tabuľky, Analysis Services kocky, XML, textové súbory. Výstupy podporuje tie isté PDF, MS Excel, XML, HTML. Možnosti doručovania a formátovania reportu sú tiež porovnateľné.

Analytické služby

Pri analýze máme nasledujúce možnosti:

- prehliadanie a úprava modelu OLAP kocky. Relačné vzťahy medzi tabuľkami faktov a dimenzií.
- definícia použitia dimenzií v OLAP kockách, editácia vzťahov medzi tabuľkami faktov a dimenzií.
- návrh zdrojového kódu definície kalkulácie.
- kľúčové indikátory, kľúčová metrika vyžadujúca cieľ, ktorý by sa mal splniť. Určia sa hodnoty, ktoré prevyšujú doporučenú toleranciu a ktoré sa blížia očakávanej hodnote. KPI je definovaný pomocou MDX výrazu: hodnota, cieľová hodnota, stav, trend, typ vizualizácie.
- prehliadnutie a kontrola výsledkov analýzy.

Oracle Business Intelligence 10g

Oracle je DBMS, moderný multiplatformový databázový systém s možnosťami spracovávania dát, vysokým výkonom a jednoduchou škálovateľnosťou. Databázový systém Oracle je vyvíjaný firmou Oracle Corporation[9].

Analytické služby

Poskytuje samostatné riešenie pre potreby základných analýz ako reportovanie, ad-hoc dotazovanie, pokročilých MOLAP aj ROLAP analýz. Prinášajú možnosť vývoja vlastných aplikácií, správy metadát riešenia, návrh, tvorbu a monitorovanie dátových skladov.

- Umožňuje užívateľom kombinovať výstupy analýz dát s informáciami z iných externých zdrojov.
- Súčasťou sú nástroje pre distribúciu pripravených reportov a dotazovania. Umožňujú prístup k metadátam dátového skladu. Doručovanie reportov uskutočňuje cez Web Prehliadač.
- Poskytuje prístup do relačných, aj OLAP dát.
- Poskytujú nástroj pre tvorbu základných a pokročilých analýz. Nie je potrebná znalosť SQL.
- Zabezpečuje funkcionalitu ako vytváranie a ukladanie pracovných zošitov (workbooks), drill to detail, rotacie, drill to related.
- Umožňuje prácu s grafmi, určovanie podmienok, triedenie, kalkulácie a parametrizovanie, dolovanie dát z OLTP do OLAP.

Reportovacie služby

Prehliadanie už pripravených reportov, analýz. Jedná sa o tenký klient, DHTML prehliadanie workbookov. Pokrýva OLTP aj OLAP, má možnosť ukladať vykonané zmeny. Poskytuje funkcie ako pivoting, rezy, triedenia, zmena hodnôt atribútov, formátovanie vzhľadu reportov. Tlačenie reportov je možné vo výstupe pdf, xls, html, csv, rtf.

Oracle reportovací nástroj je Oracle Reports. Umožňuje prístup k rôznym zdrojom ako SQL tabuľky, OLAP, XML súbory, textové súbory. Poskytuje výstup reportu vo formátoch PDF, XML, HTML, RTF, MS Excel. Taktiež máme možnosť formátovania reportu využitím tabuliek, matíc, mailových návestí a ďalšie.

Cognos 8 Business Intelligence

Spoločnosť Cognos patrí medzi najväčších svetových producentov firemných informačných systémov (BI) a plánovacieho softvéru pre veľké firmy. Je založený na webovom užívateľskom a administrátorskom rozhraní. Architektúra webových služieb (web services) minimalizuje požadované zdroje pre vývoj, prácu a údržbu. Využíva otvorené štandardy ako XML, SOAP a WSDL. V nasledujúcom texte som niektoré informácie čerpal z [10].

Analytické služby

Obsahuje užívateľské a administrátorské rozhranie dostupné cez webové rozhranie. Nachádzajú sa tu aj nástroje pre tvorbu komplexných zostáv alebo analýzu dát. Cognos 8 BI poskytuje nástroje slúžiace na analýzu dát, ad-hoc dotazovanie, reportovanie, vytváranie komplexných reportov nezávisle od zdroja dát relačných alebo multidimenzionálnych dát. Medzi analytické možnosti patrí

- rozšírené triedenie a zoraďovanie,
- možnosť vytvárania filtrov na kategórie a kombinovania filtrov, rozšírené možnosti načítavania kategórií,
- funkcionalitu reportovania nad OLAP dátami.

Reportovacie služby

Poskytuje nástroj na ad-hoc dotazovanie a jednoduché reportovanie. Na vytvárané dotazy je možnosť použiť štandardnú šablónu, predpripravenú. Ďalšie možnosti sú podmienené formátovanie hodnôt v stĺpcoch dotazu, reťazenie filtrov pomocou logických operátorov AND, OR, vnáranie a vynáranie pre zobrazenie detailnejších dát nižšej alebo vyššej hierarchickej úrovni (drill down, drill up).

Slúži na vytváranie komplexných reportov nad dátami z relačných databáz, multidimenzionálnymi dátami a kockami. Ponúka dynamické reporty. Je to aj dashboardingovým nástrojom. Dostupné sú nové typy grafov, diagramov. Ponúka možnosť vytvárať krížové tabuľky s uzlami, definovanie ľubovoľných lokálnych väzieb a prepájanie dotazov.

Pri zdroji dát Cognos môže využiť najväčšiu variabilitu. Ako relačné zdroje Oracle, SQL, IBM, Teradata, Sybase, ODBC. Ako OLAP zdroje Cognos OLAP, SAP BW, Microsoft SSAS, Essbase, Oracle 10G, IBM DB2 CubeViews. Ďalej napríklad XML, Java beans, JDBX, LDAP, WSDL, textové súbory, Excel súbory , Access súbory. Variabilita možných vstupov je teda vyššia. Formáty výstupu a možnosti doručovania sú porovnateľné s SQL Server RS a Olap Reports. Možnosti formátovania sú väčšie, keď môžeme využiť dynamické reporty, grafy, schémy a veľa ďalších prvkov.

Architektúra dátových skladov

Súčasťou architektúry vyššie spomenutých datových skladov sú následujúce tri vrstvy

- Spodnou vrstvou je *server datového skladu*, väčšinou relačný databázový server. Data sú získané z operačnej databáze pomocou tzv. brány, ktorá umožnuje spustiť SQL dotaz na danom servery. Patrí sem napr. rozhranie ODBC alebo JDBC.
- Prostrednou vrstvou je OLAP server, ktorý je implementovaný ako ROLAP nebo MOLAP.
- Najvyššou vrstvou je klient, ktorý obsahuje dotazovacie a reportovacie nástroje, nástroje pre analýzu a data mining.

6.2 Špecifikácia riešenia

Na základe poznatkov získaných z prieskumu existujúcich systémov a očakávaní plynúcich zo zadania, s prihliadnutím na rozsah práce budú v nasledujúcich podkapitolách vyslovené požiadavky na aplikáciu.

6.2.1 Funkcionálne požiadavky

6.2.1.1 Práca s údajmi OLAP v režime on-line

Program bude využívať vlastný OLAP engine popísaný vyššie rozšírený o službu tzv. pivot tabuľky a olap grafu. Pri klientskom prístupe je potrebná konektivita na databázový server. Ak uložíme pracovný súbor s výsledkom analýzy a neskôr ho otvoríme a budeme potrebovať napríklad meniť rozsah, alebo presnosť dimenzií, podarí sa nám to len za predpokladu, že budeme mať spojenie na databázový server. Po pripojení sa k databázovému serveru pomocou klienta a stiahnutí aktuálnych dát, sa agregáty vypočítajú na klientovy, následne tieto údaje zobrazuje, prípadne ďalej spracováva, alebo napríklad ukladá. Pri zobrazení, alebo zmene zostavy sa preto medzi databázovým serverom a klientskou aplikáciou neposielajú žiadne dáta. Pre podporu analýz je klientská aplikácia rozšírená o kontingenčnú tabuľku.

6.2.1.2 Modelovanie multidimenzioných databáz

Pre modelovanie a návrh multidimenzionálnych databáz sa využívajú grafické návrhové nástroje integrované s databázovými servermi, nakoľko túto činnosť vykonávajú hlavne databázoví administrátoria analytici. Je treba podotknúť, že táto časť nie je súčasťou riešeného projektu, rovnako ako samotný ETL proces.

6.2.1.3 Vizualizácia

Z pohľadu používateľa je najlepším riešním klientská aplikácia taká, ktorú tento dôverne pozná a obsahuje štandartné prístupy v zobrazovaní výsledkov analýz. Takýmito prostriedkami sú grafy a kontingenčné tabuľky. K dispozicií by malo byť niekoľko typov grafov ako napríklad stĺpcový, koláčový, či líniový.

Pivot Table

Kontingenčná tabuľka (anglicky Pivot Table) má na rozdiel od klasickej tabuľky niekoľko špeciálnych vlastností. Napríklad umožňuje určitú rotáciu, teda výmenu riadkov a stĺpcov, kombináciu a hierarchickú štruktúru riadkov a stĺpcov.

Vzhľadom k týmto vlastnostiam sa mimoriadne hodí pre zobrazenie údajov z multidimenzionálnych databáz. Do obdĺžnika pre polia údajov umiestnime fakty a do obdĺžnikov pre polia riadkov a stĺpcov umiestnime príslušné dimenzie. V zostave kontingenčnej tabuľky alebo kontingenčného grafu sa z každej dimenzie stáva množina polí, v ktorých možno rozbaliť a zbaliť podrobnosti na jednotlivých úrovniach hierarchií. Údajové polia sú polia zo zdrojového zoznamu, tabuľky alebo databázy, ktoré obsahujú údaje zosumarizované v zostave kontingenčnej tabuľky alebo kontingenčného grafu. Údajové polia zvyčajne obsahuje číselné údaje, napríklad štatistické údaje alebo čiastky predaja. Na obrázku č.8 je ilustrovaná pivot tabuľka z aplikácie MS Excel.

	A	B	C
1	Customer	(All)	
2			
3	Sum of Sold		
4	Salesperson	Category	Total
5	Boston	Bars	330,646.80
6		Cookies	252,625.99
7		Crackers	595,109.97
8	Boston Total		1,178,382.76
9	Lee	Bars	225,745.17
10		Cookies	167,482.21
11		Crackers	392,037.47
12	Lee Total		785,264.85
13	Parent	Bars	328,327.02
14		Cookies	255,501.08
15		Crackers	578,699.22
16	Parent Total		1,162,527.32
17	Grand Total		3,126,174.93

Obrázek 8: Kontingenčná tabulka MS Excel [11]

6.2.1.4 Export

Grafový výstup je možné exportovať vo formáte PDF a výsledky analýz vo formáte XML.

6.2.1.5 Navigácia

Uživateľ sa naviguje prostredníctvom myši alebo klávesnice.

6.2.1.6 Práca s údajmi OLAP v režime off-line

Analýzy OLAP kocky uložené v súboroch s príponou XML na pevnom disku počítača nám umožnia pracovať v režime off-line, teda aj po odpojení od databázového servera. Na základe týchto údajov môžeme vytvoriť zostavu kontingenčnej tabuľky alebo kontingenčného grafu.

6.3 Nefunkcionálne požiadavky

Základné rozdelenie klientských aplikácií je podľa kritéria, či jadro aplikácie beží na serveri, alebo na klientskom počítači rozdeľujeme tieto aplikácie na takzvaného *„tenkého“* a *„bohatého“* klienta.

Pod pojmom „*OLAP tlustý klien*“ (v anglickej terminológii rich client) budeme rozumieť klientskú aplikáciu, ktorá beží na lokálnom počítači, a teda môže naplno využívať výkon jeho hardvéru a možnosti operačného systému. Pre svoju činnosť, ale využíva údaje a služby rôznych serverov.

Výsledný systém by mal byt' navrhnutý a implementovaný, ako skupina spolupracujúcich komponent, pričom uživatel'ovi by sa mal javit' ako jednotný transparentný systém. Jednotlivé časti aplikácie by mali byt' prehl'adné a zdokumentované, aby bolo možné v práci na systéme v budúcnosti pokračovat'. I napriek tomu, že vyvýjaný systém je koncipovaný ako prototyp, sa kladie dôraz na spolahlivosť a funkčnosť s intuitívnym ovládaním a prehľadným uživateľským rozhraním.

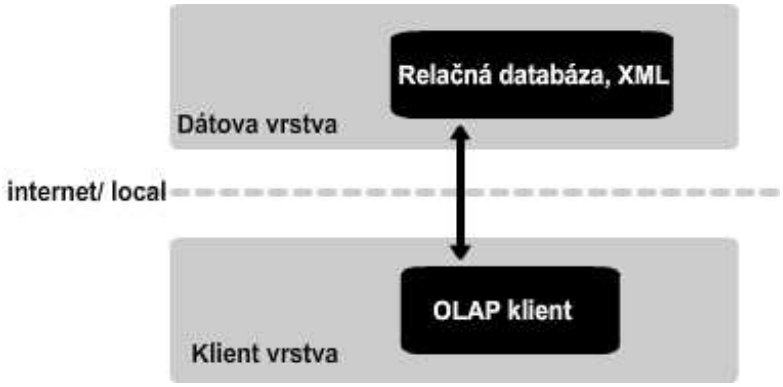
7 Návrh

Kapitola obsahuje návrh systému v podobe modelov návrhu. Prvá časť návrhu je venovaná návrhu architektúry aplikácie. V druhej časti som sa zameral na návrh prostriedkov pre vizualizáciu.

7.1 Architektúra aplikácie

Klientské aplikácie v prostredí internetu sa dajú rozdeliť na dve skupiny: ***tenký klient*** a ***tlustý klient***. Tenký klient je označenie takých aplikácí, u nich sa na strane klienta vykonává minimum aplikačnej logiky a väčšina je vykonávana na strane serveru. U tlustého klienta je tomu presne naopak. Tlustý klient tak kladie väčšie hardwarové i softwarové nároky na klienta, tenký naopak na stranu serveru a na komunikaciu [12]. Tlustý klient má väčšinou nižší objem prenesených dát než tenký klient. Rozdelenie klientských aplikáci na tenkého a tlustého klienta nie je striktné, rada aplikáci je na pomedzí týchto skupín.

My sa pri návrhu programu sústredíme na bohatého klienta. Tomu zodpovedá aj výsledná architektúra a jednotlivé vrstvy a ich role v architektúre. Dvojvrstvá architektúra vychádza z vrstvy klientské a vrstvy dátovej. Globálny pohľad a OLAP aplikáciu ukazuje následujúci obrázok.



Obrázek 9: Dvojvrstvá architektúra

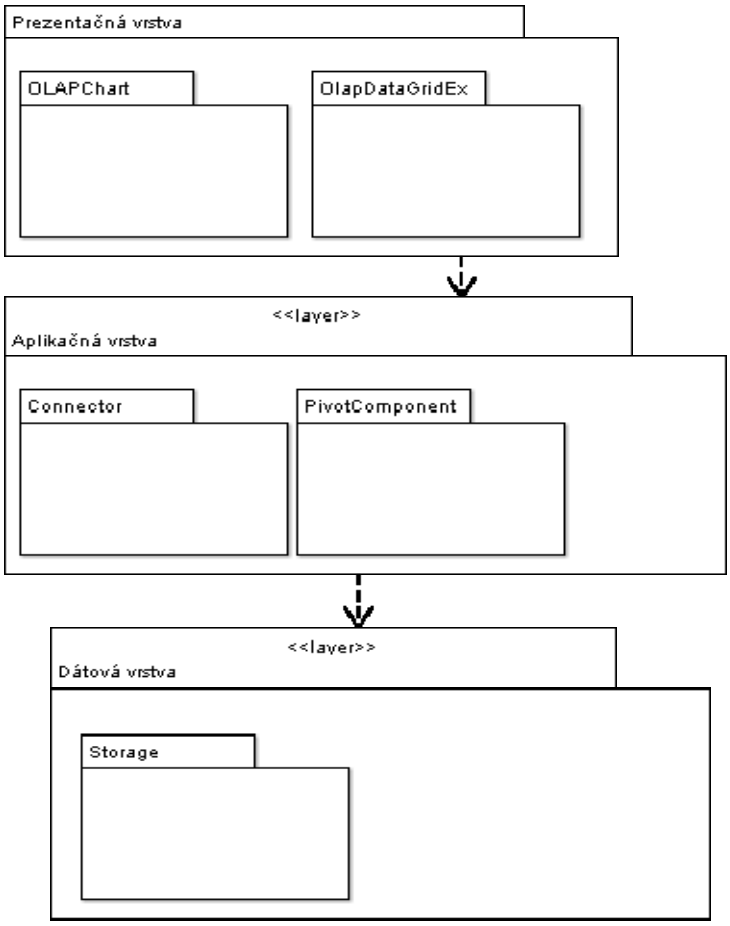
Klient obsahuje celú aplikačnú logiku, s ktorou pracuje priamo nad dátovým zdrojom. Na strane klienta je tiež OLAP engine, ktorý je súčasťou FLEX frameworku. V našom scénári je zdroj dát reprezentovaný relačnou databázou a klient využívá jazyk SQL pre prácu s dátami. Mezi dalšie typy dátového zdroja patrí XML súbory, ktoré slúžia na perzistenciu výsledkov analýz.

Dátová vrstva

Táto vrstva slúži ako dátová základna pre náš datový sklad postavený na relačnej databáze. Do tejto vrtvy patrí tiež uložisko pre výsledky analýz vo forme XML súboru.

Aplikačná/Klient vrstva

Aplikačná alebo tiež klient vrstva v podstate tvorí obal pre aplikačnú logiku a prezentačné prostriedky. Táto vrstva zaisťuje prístup k dátam datového skladu, ich spracovanie prostredníctvom OLAP enginu a následnej vizualizácií analýz. Komplexnosť aplikačnej vrstvy ma viedla k jej ďalšiemu rozdeleniu na vrstvy. Toto rozčlenenie pomohlo definovať odpovednosť jednotlivých vrstiev v rámci aplikačnej vrstvy. Vrstvy sú definované ako prezentačná (presentation), aplikačná (business) a dátová (persistence). Na pomedzí týchto vrstiev leží napríklad MVC (Model View Controller), obr.č. 10.



Obrázek 10: Aplikačná vrstva

Reprezentantmi jednotlivých vrstiev sú komponenty resp. triedy, ktoré som rozdelil na dva logické celky. A to doménovo a aplikačne špecifické. V nasledujúcom texte uvediem iba implementačne zaujmavé. Pomocné triedy a komponenty neuvádzam.

Domenovo špecifické komponenty

Sú sem zaradené komponenty, ktoré rozširujú OLAP engine, poskytnutý vo FLEX frameworku a kedže bolo myslené aj na ich znovupoužitelnosť, je možné ich začlenenie v ďalších aplikáciach a to i samostatne. Ide o nasledujúce komponenty:

- PivotComponent,
- OLAPChart,
- OLAPDataGridEx.

Aplikačne špecifické komponenty

Patria sem komponenty či triedy , ktoré sa starajú o prácu s dátovými zdrojmi, zabezpečujú integráciu s operačným systémom a poskytujú grafické rozhranie. Ich učelnosť je čisto jednostranná. Výčet komponent je nasledovný(nie je kompletný):

- Storage,
- Connector,
- OLAP.

View

Zabezpečuje vizualizačné nástroje. Popis čitateľ nájde v časti venovanej implementácií.

Controller

Zabezpečuje prácu s databázou, odchytáva a posiela události. Je prostredníkom medzi zvyškom aplikácie a OLAP enginom. Popis čitateľ nájde v časti venovanej implementácií.

Model

Zabezpečuje prostriedky pre prácu s databázov a súborovým systémom operačného systému. Tvorí ju trieda Storage. Popis čitateľ nájde v časti venovanej implementácií.

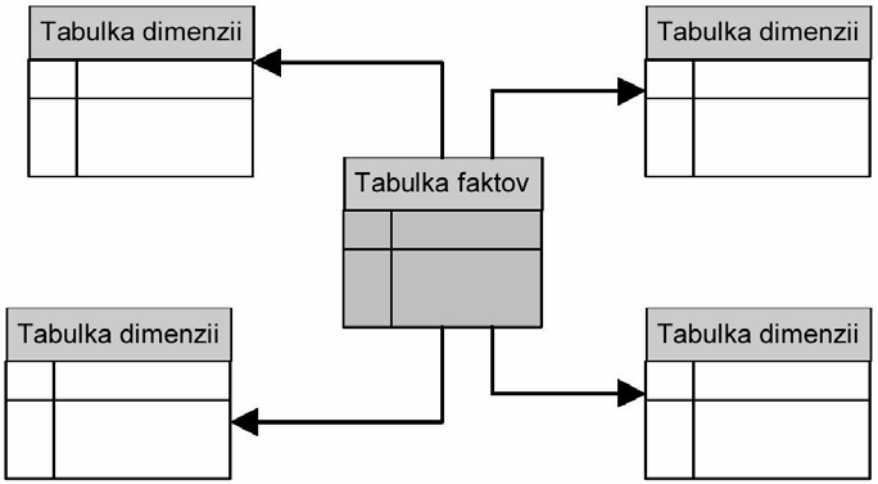
7.2 Návrh dátového skladu

Schémata multidimenzionálnych databázi

U datových skladov sa používa multidimenzionálny model, a to vo forme schématu hviezdy, snehovej vločky alebo súhvezdia. Ako klúčové pri modelovaní databázi je identifikácia zaujmavých faktov a vlastností [13].

Schéma hviezdy

Ako schéma pre modelovanie datového skladu bolo určené schéma hviezdy (obr. č. 11). Toto schéma bolo vybrané pre jeho jednoduchosť a rýchlosť dotazovania. Obsahuje jednu centrálnu tabuľku (tabuľku faktov), ktorá obsahuje velké množstvo dát bez redundancie. Ďalej obsahuje množinu menších tabuliek dimenzií, kde každá z nich obsahuje informacie o jednej z dimenzií.



Obrázek 11: Schéma hviezdy

Každá dimenzia je reprezentovaná jednou tabuľkou s niekoľkými atribútmi. V tejto tabuľke vznikajú redundancie. Mimo to, atribúty v tabuľke dimenzií môžu tvoriť hierarchie.

Charakteristika tabuľky v databáze

Tabulka 3: Charakteristika tabuľky

Množina	Entita
Popis	Atribúty
Dimenzionalita	Dvojdimenzionálna tabulka
Čas	Elementárna manipulácia
Modelovanie	*Schéma hviezdy*
Prístup k dátum	SQL

7.3 Agregačné funkcie

Agregovanie hodnôt a tutiž celú analýza má na starosti klientská FLEX aplikácia s OLAP enginom. Ten je zodpovedný za vytvorenie dátovej kocky. Bod v datovej kocke je definovaný množinou hodnôt dimenzií. Merná jednotka dátovej kocky je numerická funkcia, ktorá môže být vyhodnotená v každém bode kocky. Výpočet sa uskutoční pomocou agregacie dát korešpondujúcich s danými hodnotami dimenzií. Existujú tri základné typy týchto jednotiek:

- **Distributivní**: distribuovaný výpočet. Patria sem funkcie count(), sum(), min(), max().
- **Algebraické:** výsledok funkcie, ktorá má M parametrov, z nich každý môže byť získaný výpočtom distributivnej funkcie. Príkladom je avg().

7.4 Návrh formy perzistencie analýz

Jednou s požiadaviek kladných na projekt, bola možnosť perzistencie výsledkov OLAP analýz. Pre tento účel bol ako najvhodnejší určený súbor vo formáte XML. A to nie len pre jeho relatívnu zrozumitelnosť, ale predovšetkým pre jednoduchú implementáciu.

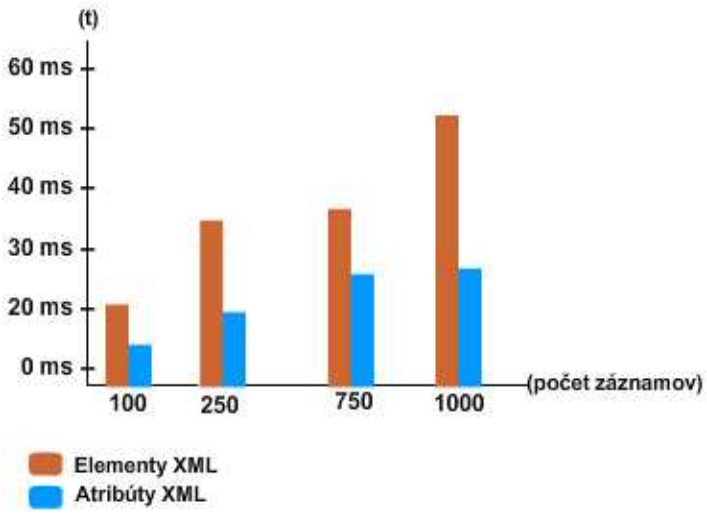
Štruktúra výsledku je značne variabilná, čo do počtu atribútov v jednotlivých záznamoch. Ako príklad uvádzam výsledok analýzy „*predajnosť a cena produktu za jednotku času(rok)*“

```
<root measure="Cost,Sales" dimension="Product,Year" slicer=" "
columns=" Year " rows="Product">
 <data Product="(Total)" Year ="2008" Cost="102.5" Sales="185" />
 <data Product="Flexo" Year ="2008"  Cost="15" Sales="10" />
…
</root>
```

Význam jednotlivých uzlov a atribútov:

- *root* – koreňový uzol (nemenný), definuje nasledujúce atribúty .
  - *atribút measure* – výčet použitých mier.
  - *atribút dimension* – výčet použitých dimenzií.
  - *atribút slicer* – výčet dimenziií na rezovej osi.
  - *atribút rows* – výčet dimenziií na riadkovej osi.
  - *atribút columns* – výčet dimenziií na stĺlpcovej osi.
- *data* – prázdny uzol s variabilným počtom atribútov. Jeho atribútmi sú prvky z uzlu *root* a hodnotami sú výsledky analýz.

Ešte by bolo dobré spomenúť, prečo bola štruktúra postavená na atribútoch uzlov a nie na samotných elementoch. Je to z dôvodu uskutočnenej rýchlostnej analýzy XML parsera vo Flash Playery, na základe ktorej bolo ako efektívnejší zvolený prístup cez atribúty. Výsledky rýchlostnej analýzy:



Obrázek 12: Rýchlostný test

7.5 Popis a diagram tried

Popis tried nie je úplný, má skôr ilustratívny charakter. Triedy som vyberal podľa ich OLAP funkcionality a práce s OLAP enginom.

PivotComponent

PivotComponent je triedou, ktorá posiela a odchytáva udalosti pre *OLAPChart* a *OLAPDataGridEx*. Má na starosti operácie s kockou a predávanie výsledkov do spomínaných komponent *OLAPChart* a *OLAPDataGridEx*.

Popis dôležitých metód a udalosti:

- *set DataProvider(o:Object)* - priradenie datového zdroja.
- *set cube(value:OLAPCube)* – nastavenie OLAP kocky.
- *set dimensions(value:Array)* – určenie množiny dimenzií OLAP kocky.
- *set displayedFacts(value:Arra y)* – určenie mier, ktoré sa zobrazia vo výsledku.
- *set rowFields(value:Array)* – určenie dimenzií, ktoré sa objavia na riadkovej osi.
- *set columnFields(value:Array)* - určenie dimenzií, ktoré sa objavia na stĺpcovej osi.
- *set slicerFields(value:Array)* - určenie dimenzií, ktoré sa objavia na rezovej osi.

- *dataChanged event* – posiela pri zmene vstupných dát.
- *queryChanged event* – posiela pri zmene dotazu.
- *dimensionDelete event* – ochytáva zmenu počtu dimenzií.
- *filterChanged event* – ochytáva zmenu filtru.
- *resetPivotTable event* – odchytáva pri resete Pivot tabuľky.
- *resetChart event* – odchytáva pri resete grafu.
- *CUBE_COMPLETE event* – odchytáva pri inicializácii OLAP kocky.

OLAPChart

Je rozšírením grafovej komponenty. Nadobúda štyri typy: stĺpcový, zónový, líniový a bodový.

Popis dôležitých metód a události:

- *set DataProvider(o:Object)* - priradí výsledkok agregácií.
- *generateSeries(olapData:Object)* - vygeneruje jednotlivé série, ktoré budú zobrazené v grafe.
- *generateCategoryValue(index:int)* – generuje kategórie.
- *set type(type:String)* - definuje typ grafu.
- *resetChart event* – posiela pri resete grafu.

OLAPDataGridEx

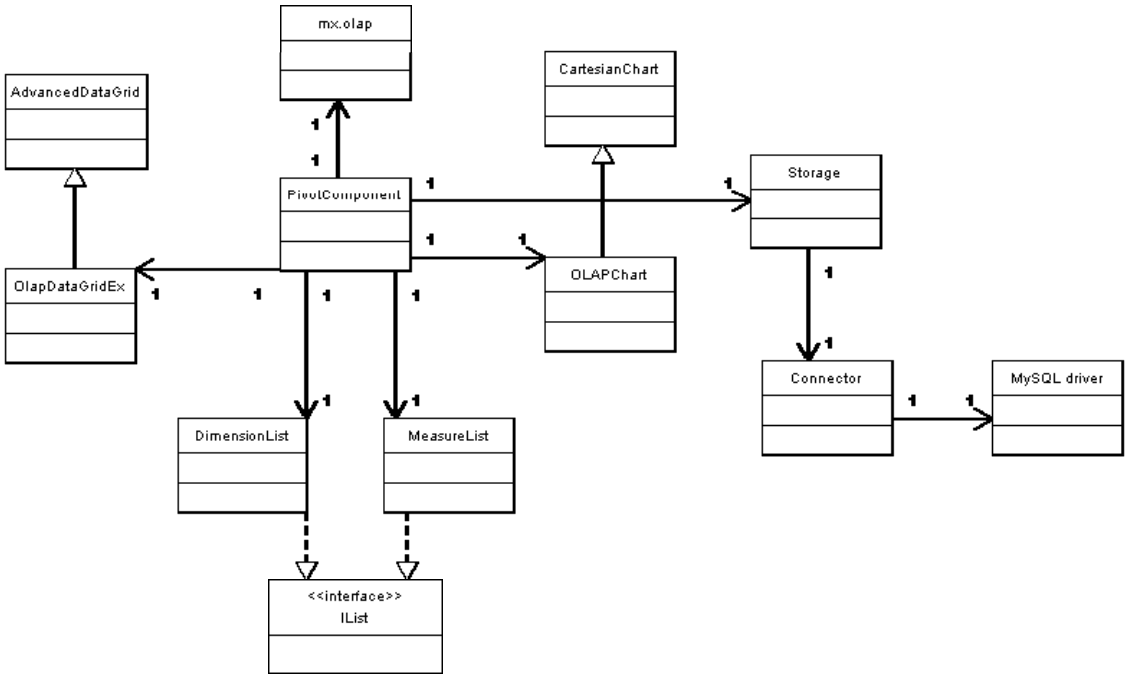
Rozširuje štandartnú komponentu tabuľky, ktorá je dostupná vo FLEX frameworku.

Popis dôležitých metód

- *set DataProvider(o:Object)* - s priradeného výsledku agregácií, vytvorí dimenzie.
- *prepareNewQuery()* – v prípade, že uživateľ vloží novú dimenziu či mieru do tabuľky, spôsobom Drag&Drop, pripraví a zavolá sa nový dotaz nad kockou prostredníctvom komponenty *PivotComponent*.
- *getNewSet(Field:Array, isMeasure:Boolean, isSlicer:Boolean)*– na základe filtru vygeneruje výslednú konfiguráciu pre dotaz nad kockou.
- *dataChanged event* - odchytáva pri zmene dát.
- *queryChanged event* - odchytáva pri zmene dotazu.
- *delete event* – odchytáva pri zmene dimenzií.

Diagram tried

Diagram tried popisuje hlavné triedy jadra aplikácie. Nezobrazuje pomocné triedy.



Obrázek 13: Class Diagram

8 Implementácia

V kapitole sú zhrnuté implementačné detaily aplikácie. Čitateľ tu nájde popis, ako bola aplikácia vyvíjaná a s akými problémami sa behom vývoja bolo nutné vysporiadať. Pri implementácií bol využitý OOP prístup s využitím event-modelu podobného, ako napríklad v jazyku JAVA. Tento model je založený na tzv. subscribe, publish objektoch. Aplikácia vo forme inštalátora a zdrojových kódov spolu s uživateľským manuálom, je súčasť prílohy [1].

Slovník pojmov

V nasledujúcom texte uvediem niekoľko pojmov, ktoré súvisia s *OLAP API* a pojmov z terminológie vizualizácie grafov. V ďalšom texte sa budem na tieto pojmy odvolávať.

OLAPCube – trieda reprezentujúca OLAP kocku

OLAPResult - trieda reprezentujúca výsledok dotazu nad OLAP kockou. Výsledok je definovaný ako množinu os.

- *COLUMN_AXIS* – stĺpcová osa,
- *ROW_AXIS* – riadková osa,
- *SLICER_AXIS* – rezová osa.

OLAPQuery – trieda reprezentujúca dotaz vykonaný na Olap kocke. Je definovaná ako dvojdimenzionálna tabuľka stĺpcov a riadkov.

OLAPMember – trieda reprezentujúca člen dimenzie Olap Kocky

Séria

Pod pojmom séria budeme označovať prvky, ktoré špecifikujú dáta zobrazované v grafoch. V našom prípade sú to fakta, ktoré budú reprezentovať série.

Kategória

Týmto názvom označíme súbor hodnôt pre ktoré zobrazujeme dáta. V našom prípade sú to dimenzie a ich členy.

8.1 Prístup a práca s databázovým serverom

Pri implemetácií prístupu k databázovému serveru som narazil na prvú prekážku, ktorá spočívala v absencií natívnej schopností priamej komunikácie s databázou. Ako bolo spomenuté v kapitolách o technológiach FLEX a AIR, tieto nástoje poskytujú spojenie s databázou len prostredníctvom

aplikačného serveru napr. PHP, PYTHON apod.. Tento problém som odstránil použitím open-source ovladaču [14].

V skratke o použitom ovladači:

- Ovládač pre MySQL,
- Implementácia v jazyku ActionScript 3.0,
- Obálka nad binárnými soketmi vo Flash Playery,
- Momentálne vo verzií Beta,
- Synchrónny prístup k dátam – ma za následok časové prodlevy.

Medzi ďalšie dôvody, ktoré ma viedli k použitiu MySQL boli:

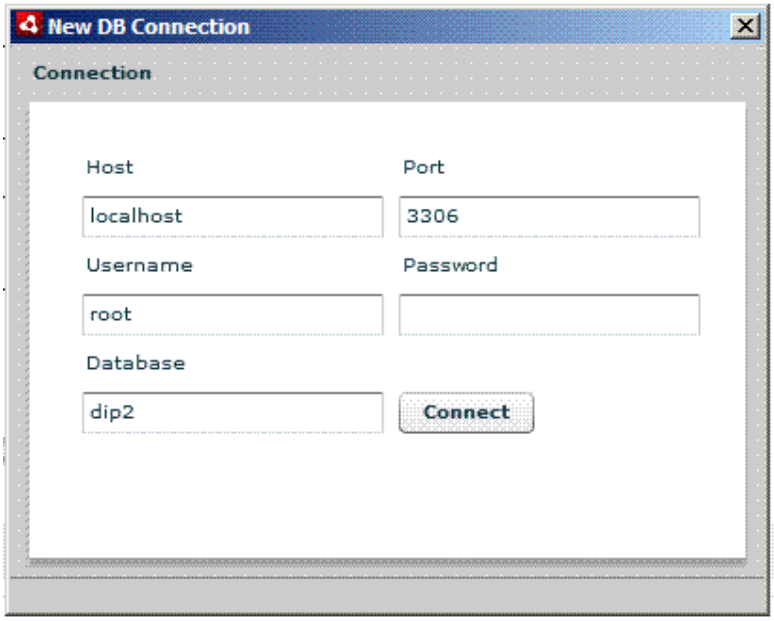
- Predošlé pracovné skúsenosti,
- Nenáročný na hardware.

8.1.1 Proces tvorby spojenia

Spojenie s databázou sa vyžaduje na začiatku práce s OLAP aplikáciou, kedy je na klienta potreba dostať dáta z dátového skladu. Pre prácu s databázou bola implementovaná trieda *Connector*, ktorá volá API vyššie spomenutého ovládača. Pre spojenie s databázou je potrebné zadať parametre pripojovacieho reťazca prostredníctvom dialogového okna viď obr. č. 14.

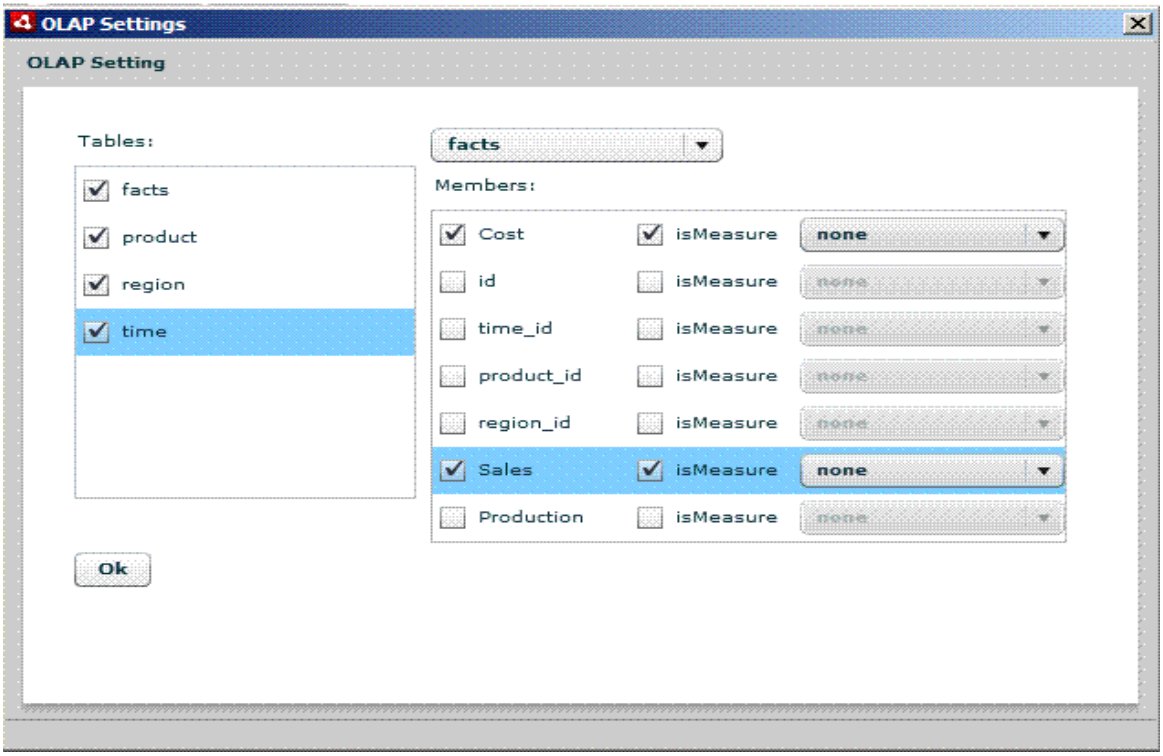
Medzi tieto parametre patrí:

- *Host* – adresa databázového servera,
- *Port* – číslo portu(3306),
- *Db* – meno databázy,
- *Username* – uživateľské meno,
- *Password* - uživateľské heslo.



Obrázek 14: Vytvorenie spojenia

Po úspešnom pripojení trieda *Connector* získa schému databázy a predá jú komponente *OLAPWindowSetings*, ktorá je zodpovedná za zobrazenie konfiguračného nástroja s danou schémou.



Obrázek 15: Konfigurácia dimenzií, mier

Schéma obsahuje tabuľky – dimenzie a stĺpce – členy, ktoré sa použijú na analýzu. Výsledok konfigurácie sa predá objektu *OLAPConfig*, ktorý sa predá metóde *buildSQL* objecktu *Connector*. Ako už názov napovedá metóda vytvorí SQL dotaz. Tento dotaz pozostáva s klauzule *SELECT* a klauzulí *INNER JOIN*.

```
SELECT time.month, time.quarter, day.id, product.brand, product.id,
product.name, facts.Sales, facts.Cost FROM facts inner join day on day.id=
facts.day_key inner join product on product.id= facts.prd_key
```

V tomto momente, kedy je dotaz skompletizovaný nastáva synchrónne volanie na databázu. Toto miesto je kritické, pretože si dokážeme predstaviť ako dlho asi trvá kým sa prenese väčší objem dát. Počas tohoto trvania je aplikácia neaktívna a nereaguje. Po získaní výsledku sú dáta v aplikácií reprezentované ako flat-data a predané do kolekcie *ArrayCollection.* Tá slúži ako *dataProvider* pre *PivotComponent.*

8.2 Proces tvorby dotazu

Obrázek 16: Schéma tvorby dotazu

Po inicializácií dataProvideru komponenty *PivotComponent,* sa ten následne predá inštancií kocky *OLAPCube*. Tvorba dotazu spočíva v interakcií uživatela s komponentami uživatelského rozhrania. Tieto komponenty popíšem v nasledujúcom texte.

DimensionList

Je inicializovaná pri získaní dát z databázy. Obsahuje zoznam vybraných dimenzií, tak ako si ich zvolil uživateľ v dialogovom okne *OLAPWindowSetting*.

MeasureList

Je inicializovaná pri získaní dát z databázy. Obsahuje zoznam vybraných mier. Každá položka zo zoznamu je inštanciou *ComboBox Factory*, čo znamená, že po vizualizačnej stránke je to vysúvací zoznam s položkami prednastavených agregátov SUM, AVG, COUNT, MIN, MAX (Obr. č. 16).

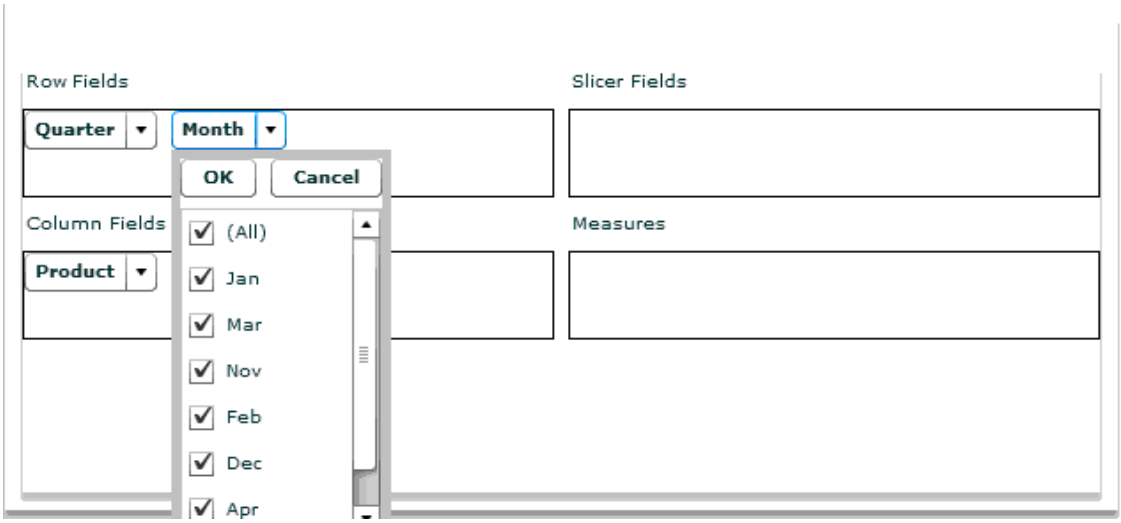
Obrázek 17: Zoznam mier a agregačných funkcií

Po označení danej agregačnej položky, pošle táto komponenta udalosť *„change“***,** ktorú odchytí *PivotComponent* a jej metóda *„modifyFacts“,* ktorá následne volá metódy OLAP kocky na zmenu dimenzií. Pri každej zmene OLAP kocky je vyvolaná udalosť *„progress“*, ktorej úlohou je zaznamenať časovy priebeh vyvolanej zmeny v komponente *ProgressBar*.

Ovládacie prvky Row, Column, Slicer a Measure Box

Je to skupina komponent, majú charakter zoznamu a platia pre ne nasledujúce zásady

- Implementujú rozhranie Ilist,
- Vkladanie prvkov do boxov prebieha spôsobom Drag&Drop,
- Odstranenie prvku prebieha po stlačení klávesy *„Del“,*
- Prvkom zoznamu je trieda ComboBox Factory, ktorá slúži ako renderer položky,
- Prvky ComboBox Factory slúžia ako filter členov danej dimenzie,
- Pri zmene filtra posiela udalosť *„selected“,*
- Pri zmene prvku v zozname posiela udalosť *„change“+ meno boxu,*
- Nedovolujú duplicity dimenzii, prebieha kontrola naprieč všetkými boxami,
- Prebieha kontrola charakteru vkladaného prvku a to v zmysle, že nie je možné vložiť napríklad dimenziu do MeasureBoxu.



Obrázek 18: Vybrané dimenzie a miery

Význam jednotlivých zoznamov napovedajú ich mená. To znamená, že po pridaní dimenzie do zoznamu RowBox sa do OLAP kocky pridá na riadkovú os dimenzia. Obdobne pre ostatné zoznamy. Pre *MeasureBox* navyše platí, že každý vložený prvok je filtrom. Predstavme si, že máme množinu mier cena, predaj a tržba. Prednastavené chovanie *PivotComponent*y je také, že zobrazí v grafe aj v tabuľke všetky tieto miery. Avšak po vložení miery do *MeasureBoxu* sa zobrazí práve táto miera spolu s už vloženými.

8.3 Vizualizácia výsledkov analýz

Výsledky analýz sú v aplikácií reprezentované prostredníctvom grafov *OLAPChart* a kontingečnej tabulky *OLAPDataGridEx*.

OLAPChart

Táto komponenta je rozšírením triedy *CartesianGraph*, ktorá je bázovou triedou pre grafy, ktoré poskytujú štandartné pravouhlé dvojdimenzionálne zobrazenie. Sú podporované tieto typy grafov **Area, Column, Line, Plot.** Prednastaveným typom je Line.

Generovanie jednotlivých kategórií grafu nie je nijak deterministické v zmysle poradia a záleží na štruktúre vrátenej z *OLAPCube* v podobe *OLAPResult*. V konečnom dôsledku to znamená, že poradie môže byť rôzne a záleží už na poradí vstupných dát z databázy. Tento nedostatok som sa snažil aspoň čiastočne odstániť tým, že som implementoval funkciu radenia chronologicky podľa jednotky času. Radí sa podľa mesiaca, štvrťroka a roka. Spôsob implemetácie je však závislý na hodnotách, ktoré reprezentujú dátové polia pre jednolivé záznamy. Napríklad pre radenie podľa nazvov mesiaca, aplikácia predpokladá, že názvy polí pre jednotlivé mesiace majú tvar *„Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec“*.

Graf disponuje nápovedou, ktorá sa zobrazí po prejdení kurzora myši cez položku grafu a zobrazuje numerickú hodnotu v danom bode grafu. Implementuje ju metóda *defaultDataTipFunction*, ktorá dostane ako parameter objekt triedy *HitData*, ktorá špecifikuje informáciu o bode vo vybranom mieste. Súčasťou grafu je legenda, ktorá zobrazuje vybrané fakty.

Proces generovania grafu

Postup:

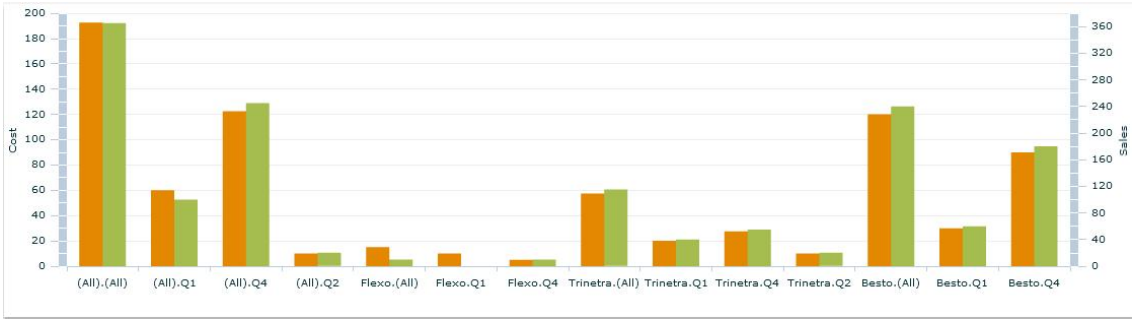
1. Generovanie začiná inizializáciou dataProvidera grafu výsledkom z *OLAPCube*, ktorý sprostredkuje *PivotComponent.*
2. Zavolá sa *prepareMembers*, ktorá z ROW_AXIS získa kolekciu reprezentantov *OLAPMember.*
3. Na základe typu grafu sa vygeneruje pole objektov, ktoré reprezentujú série grafu
4. *generateCategoryValue()* vygeneruje dáta pre série.
5. Nasleduje vytvorenie kategórií a generovanie dát pre ne pomocou metódy *getDataForCategory()* a *generateCategoryValue().*
6. *createRenderer()* vytvorí renderer pre kategórie.

Pre generovanie kategórií platí zásada, že každá kategória je výsledkom konkatenácie členov jednotlivých dimenzií, oddelených bodkami, pre ktoré je daný výsledok. Majme dimenziu Rok s členmi 2007,2008 a dimenziu Auto s členmi Audi, BMW. Potom by kategórie mohli vypadať nasledovne *„2007.Audi“, „2007.BMW“, „2008.Audi“, „2008.BMW“*

Pri výpočte analýzy je aplikácia nastavená tak, že bere do úvahy aj celkovú agregáciu(napr. všetkých produktov). Tieto agregácie sú vo výsledku označené ako členy (All).

Zoom

Pre lepšie zobrazenie výsledku v grafovej podobe bolo implementované jednoduché približovanie grafu. Pri zmene polohy jazca v komponente *Hslider* sa pošle udalosť *change*, na ktorú sa reaguje zväčšenim šírky grafovej komponenty *OLAPChart.*



Obrázek 19: OLAPChart

OLAPDataGridEx

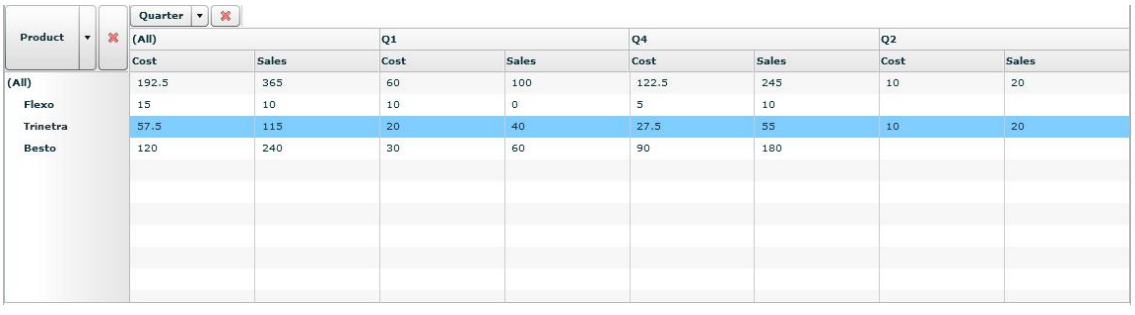
Komponenta rozširuje triedu *OLAPDataGrid*, ktorá poskytuje maticové zobrazenie dát. Poskytuje prostriedky pre filtrovanie dát, interaktívne vkladanie dimenzií.

Po označení riadka tabuľky sa zobrazí 1 až n koláčových grafov, kde n je počet vybraných faktov. Obsahom týchto grafov sú položky z označeného riadku tabuľky.

Implementácia filtru

Ako hlavičkový renderer tzv. *headerRenderer* bol implementovaný triedou *PivotPopUpButton*, ktorý vlastne reprezentuje vysúvacie menu s checkboxam. Pri každej zmene výberu je volaná metóda *makelistData* triedy *PivotListData*. Táto trieda dedí z triedy *AdvancedDataGridListData* a pridáva dve vlastnosti, menovite:

- *hasMembers*: Boolean,
- *members* : ArrayCollection.



Obrázek 20: OLAPDataGridEx

Pri zmene kolekcie members sa upravý výsledok a pošle udalosť na zmenu dataProvideru tabuľky.

Ukladanie a nahrávanie výsledkov analýz

Na perzistenciu a nahrávanie výsledkov analýz vo forme súbora XML bola implementovaná trieda Storage. Pre ukladanie slúži metóda SaveXML() a na nahrávanie LoadXML(). Obe tieto metódy

volajú File API runtimu AIR. Je použité synchrónne volanie. Štruktúra výsledného XML je popísaná v kapitole venovanej analýze.

Integrácia s OS

Aplikácia beží nad runtimom AIR, čo sa využilo pri tvorbe GUI a samotnej aplikácie. Je využitý natívny oknenný systém operačného systému. Každé dialogové okno, spolu s oknom samotnej aplikácie dedí od tried TitleWindow resp. Window, čo má za následok, že má k dispozícií vlastný chrome s ovládacími prvkami – min, max, zavrieť.

Distribúcia aplikácie

Aplikácia je distribuovaná vo forme klasického inštalatóra, ktorý poznáme u bežných desktopových aplikácií.

Export do pdf

Pre export bolo zvolený formát PDF pretože Flash Player má podporu exportu do tohoto formátu. Je však potrebné, mať naištalovanú PDF tlačiareň. Ja som použil PDF tlačiareň BullZip.

8.4 Implementácia uživateľského rozhrania

Pri návrhu uživateľského rozhrania boli zohľadnené predovšetkým požiadavky na logické členenie jednotlivých častí aplikacie, intuitivné ovládanie. Preto je systém navrhnutý s jednoduchým designom. Z tohoto dôvodu bolo rozhodnuté, že systém bude rozčlenený do dvoch sekcií.

Hlavné menu

V tejto sekcií sú všetky ovládacie prvky, ktoré poskytujú správu databázového zdroja, ukladanie a nahrávanie výsledkov dotazov a jednoduchý nástroj na radenie.

Pracovná plocha

Obsahuje komponentu Tabnavigator. Panel zo záložkami, ktorý reprezentuje pohľad na dáta.
Pohľady sú dva - grafový a tabuľkový. Sú tu koncentrované všetky mechanismy pre proces analýzy ako ovladacie prvky DimensionList, MeasureList, pre ovládacie boxy popísané vyššie.

9 Zhodnotenie

V tejto kapitole uvediem súhr poznatkov a výsledky testov aplikácie.

9.1 Experimentálne testy

V testoch som sa zameral na meranie časovej výkonnosti aplikácie, kedže tento ukazateľ je pri analýzach kritický. Kedže v našej aplikácií sme použily OLAP kocku, ktorá je celá v pamäti počítača, počet záznamov hral úlohu vo výkonnosti aplikácie. Tento ukazateľ, teda počet záznamov, však nebol pre výkonnosť tak kritický, ako počet agregovaných dimenzií a hierarchií. Výsledky ukazuje tabuľka č. 6.

Testovacie dáta mali nasledujúci charakter:

Tabulka 4: Charakter testovacích dát

Názov dimenzie	Počet záznamov
Zákaznik	100
Produkt	20
Pobočka	5
Typ objednávky	5
Čas	100

Tabuľka faktov nadobudla postupne hodnoty 2000, 5000, 10000 záznamov.

Tabulka 5: Výsledky experimentálnych testov

Počet záznamov	Počet dimenzií	Doba analýzy (s)
2000	3	2,6
2000	4	3,4
2000	5	4,1
5000	3	3,5
5000	4	4,0
5000	5	4,9
10000	3	7,9
10000	4	8,1
10000	5	8,7

Z testov sa dá odvodiť fakt, že koncepcia OLAP kocky na strane klienta je určená predovšetkým pre menšie a stredné business riešenia. V prípade nasadenia na skutočné databázy s niekoľko GB dát je tento koncept prakticky nepoužitelný. Na druhej strane to ale určite nie je *„šliapnutím vedľa“* a aplikácie s myšlienkou OLAP kocky tzv. in-memory si svoje miesto isto nájdu. Napríklad ako aplikácie na analýzu návštevnosti stránok resp. rôzne e-shopy a podobne.

9.2 Zhrnutie použitých technológií

Úzkym hrdlom AIRu je jeho rozširiteľnosť. Ak má runtime AIR konkurovať ostatným runtimom ako je Java či NET musí v tejto oblasti ešte veľa spraviť. Na druhej strane je AIR iba vo verzií 1.0 a veľa z vlastnosti sa v nasledujúcich verziách zmení a pridá.

Počas vývoja aplikácie som narazil na ďalšie nástroje, ktoré ponúkajú podobnú funkcionalitu ako AIR. Ide predovšetkým o runtime CURL [15]. Ten je taktiež určený pre integráciu webových riešení na desktop, kde výrobca deklaruje vynikajúce výkonnostné parametre. Webový framework FLEX je na scéne už pomerne dosť dlhú dobu, za ktorú sa stihol dostať na popredné priečky obľúbenosti vývojárov. Je to hlavne vďaka aplikačnému modelu, ktorý umožňuje doslova *„sypať“* aplikácie z rukáva.

Veľkou devízou použitia AIRu je možnosť použiť jeden vývojový proces pri tvorbe aplikácie. Totiž aplikácie vyvýjané pre webový prehliadač sa dajú veľmi jednoducho a rýchlo integrovať na desktop a vice-versa. V tomto zmysle bola vyvýjana aj naša OLAP aplikácia. Je to vlastne sada komponent, ktoré sa dajú znovupoužiť a to nielen v desktop aplikáci, ale je ju možnoť integrovať do našej hotovej webovej aplikácie s obmedzením na perzistenciu dát na lokálnom disku uživateľa.

9.3 Vylepšenia

Kedže práca a samotná aplikácia bola myslená ako *„proof of concept“* sú oblasti rozšírenia a vylepšenia dosť značné. Ako vylepšenia by som uviedol použitie trojvrstvé architektúry, ktorá je priamou evolúciou, z dnešného pohľadu už koncepčne zastaralej, dvojvrstvovej architektúry. Zrejme neboli dosiahnuté ani všetky výkonnostné možnosti použitých nástrojov, i keď som sa snažil dodržiavať všetky tzv. *„best-practice“* daných technológií. Aby bolo možné aplikáciu rozšíriť, je i napriek snahe o dodržanie princípov OOP, nutné refaktorovať. Aplikácia je závislá na schéme dátového skladu a preto by bolo možné uvažovať o jej rozšírení na ostatné schémata a použitie metadat. Kedže aplikácia je vo fáze tzv. prvej iterácie, kde výsledkom je funkčný prototyp, je možné pri testovaní naraziť na implementačné nedostatky a neočakávané chovanie, preto si táto oblasť zasluhuje povšimnutie pri ďaľšom vývoji.

10 Záver

Oblasť OLAP analýzy sa za posledných niekoľko rokov dostala do popredia záujmu, vďaka požiadavkám na podporu analytického spracovania údajov. S ňou úzko súvisí aj multidimenzionálny databázový model, z ktorého vychádza.

Cieľom práce bolo priblíženie logického modelu, ilustrácia oblasti využitia a analýza možností a využitia modelu. OLAP možnosti ponúkajú veľa implementačných variantov a vo svojej práci, v jej implementačnej časti som sa snažiť vybudovať prototyp s využitím vybraných technológií. Ako implementačné technológie boli vybrané FLEX a AIR, a to z dôvodu ich platformovej nezávislosti, jednoduchosti nasadenia v prostredí webu a atraktívnosti. FLEX architektúra poskytuje všetky prostriedky pre vybudovanie a prácu OLAP aplikácie. Hlavnou ideou je možnosť reprezentácie OLAP kocky na strane klienta tzv. „in-memory“ a ukladať často používané výsledky agregačných funkcií na uzivateľský disk. OLAP kocka je potom dotazovaná pomocou OLAP API. Výsledky OLAP dotazov sú zobrazené v OLAP kontingenčnej tabuľke a detaily môžu byť zobrazené ďalej v grafoch. Prínosom práce je analýza problematiky OLAP, použitých nástrojov, implementácia funkčného prototypu.

Literatúra

[1] LACKO, Ľ. . *Databáze: datové sklady, OLAP a dolování dat s příklady* v. [s.l.] : Computer Press, 2003.

[2] REMBEL, R., KONCILIA, Ch.: *Data Warehouses and OLAP: Concepts, Architectures, and Solutions*. [s.l.] : Idea Group Inc (IGI), 2007.

[3] RAFANELLI, M.: *Multidimensional Databases: Problems and Solutions*. [s.l.] : Idea Group Inc (IGI), 2003.

[4] JARKE, M.: *Fundamentals of Data Warehouses*. [s.l.] : Springer, 2003.

[5] LECHTENBORGER, J.: *Data Warehouse Schema Design*. [s.l.] : IOS Press, 2001.

[6] PENDSE, N.: *The OLAP Report: What is OLAP?* [online]. 2005 [cit. 2008-05-15]. Dostupný z URL: http://www.olapreport.com/fasmi.htm/.

[7] CHAMERS, Mike . *AIR for FLEX Developers* . [s.l.] : O'Reilly, 2007.

[8] LARSON, B.: *Microsoft SQL Server 2005 Reporting Services 2005*. McGraw-Hill, 2006.

[9] STACKOWIAK, R., RAYMAN J., GREENWALD, R.: *Oracle Data Warehousing and Business Intelligence Solutions*. Wiley Publishing, 2007.

[10] VOLITICH D.: *IBM Cognos 8 Business Intelligence: The Official Guide*. McGraw-Hill, 2008.

[11] JELEN B., ALEXANDER M.: *Pivot Table Data Crunching (Business Solutions)*. Que Publishing, 2006.

[12] KANTER J.: *Understanding Thin-Client/Server Computing*. Microsoft Press, 1998.

[13] ADAMSON Ch.: *Mastering Data Warehouse Aggregates: Solutions for Star Schema Performance*. Wiley Publishing, 2006.

[14] *Actionscript 3 MySql Driver*. [online], [cit. 2008-05-15]. Dostupný z URL: http://code.google.com/p/assql/.

[15] *CURL runtime*. [online], [cit.2008-05-15]. Dostupný z URL: http://www.curl.com/.

Zoznam príloh

Príloha 1. CD médium s aplikáciou OLAP, zdrojové kódy, manuál