UNIVERZITA PALACKÉHO V OLOMOUCI
PŘÍRODOVĚDECKÁ FAKULTA
KATEDRA INFORMATIKY

BAKALÁŘSKÁ PRÁCE

ASP. NET KOMPONENTA PRO OLAP ANALÝZU MULTIDIMENZIONÁLNÍCH DAT

2010 Robert Hanzel

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně

Datum : ……

Anotace

Cílem práce je vytvoření webového ovládacího prvku pro ASP.NET 2.0 a vyšší. Tento prvek bude umožňovat jednoduchou analýzu multidimenzionálních struktur analytických databází analytického serveru MS SQL SERVER.

Annotation

A purpose of this work is creating a web control for ASP.NET 2.0 and higher. This feature will allow a simple analysis of multidimensional structures of analytical databases of analytical server MS SQL SERVER.

Děkuji Ing. Jiřímu Hronkovi za odborné vedení mého bakalářského projektu.

OBSAH

Seznam obrázků:

1. Specifikace softwarového produktu

1.1 Zadání a cíl projektu

Cílem práce je vytvoření webového ovládacího prvku pro ASP.NET 2.0 a vyšší. Tento prvek bude umožňovat jednoduchou analýzu multidimenzionálních struktur analytických databází analytického serveru MS SQL SERVER. Po přidání prvku do ASP.NET stránky umožní vystavené vlastnosti webového prvku nastavení údajů pro připojení ke konkrétní databázi a vybrání jedné z dostupných datových struktur. Dále bude možno nakonfigurovat barevný vzhled prvku u klienta. Prvek bude zobrazovat hierarchickou strukturu vybrané datové kostky. Uživatel bude mít možnost definovat požadavek do databáze zadáním požadovaných údajů pro 2 osy dotazu a výběrem měrné jednotky. Navrácená data bude možno libovolně hierarchicky procházet podle obou os.

1.2 Požadované artefakty

- zadání a specifikace projektu
- analytická dokumentace
- programátorská dokumentace
- uživatelská příručka
- finální verze programu

2. Úvod do technologie OLAP

2.1 OLTP databáze

Ve většině organizací a firem se dnes využívá nějaký informační databázový systém. Účelem těchto systémů je shromažďovat a udržovat důležitá data pro existenci firmy. Uchovávaná data jsou aktuální, popisují skutečný stav věcí v reálném čase. Nad takovou množinou dat se obecně provádí různé operace - hlavně výběr, vkládání a úprava existujících dat. Proto se tato data často označují jako operativní a jsou uložena v operativních databázích. Systémy, které pracují tímto způsobem s daty, umožňují svým uživatelům vykonávat velké množství obchodních nebo jiných transakcí online. Označují se proto za systémy OLTP (*Online Transaction Processing*) a databáze, se kterými tyto systémy spolupracují, se také nazývají transakční nebo také OLTP databáze. Účelem využití OLTP databází je zautomatizování běžných firemních činností.

Výhody využití OLTP databází:

- umožňují realizovat velký počet současných přístupů k datům
- velká rychlost při změně uložených údajů
- operace nad daty jsou prováděny v transakcích
- data jsou uloženy v dostatečně strukturované formě

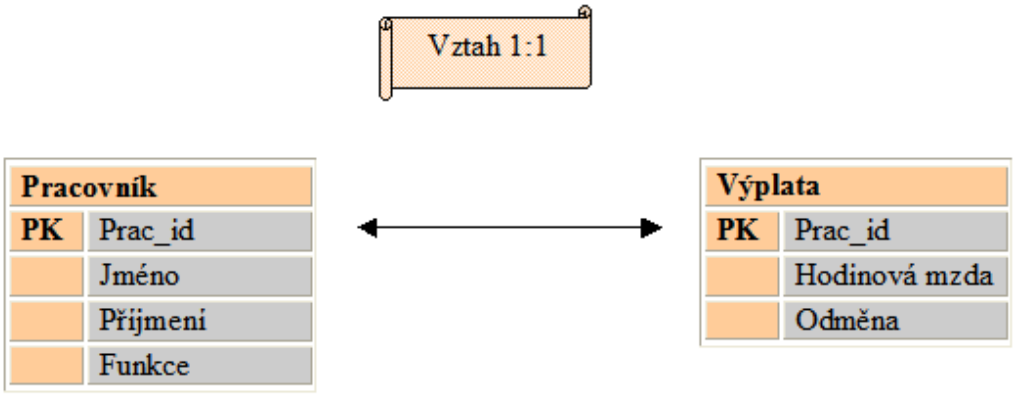
Zpracování dat v jednotlivých transakcích znamená, že je celá skupina operací nad množinou dat považována za jeden celek. Pokud všechny operace proběhnou úspěšně, potom je transakce potvrzena a vše je v pořádku. Ale může nastat případ, kdy se některá operace v transakci nezdaří a v tom případě je celá transakce vrácena zpět a data se vrátí do podoby před započetím transakce. Nemůže nastat situace, kdy se úspěšně provede jen některá z operací a transakce bude přesto dokončena. Tímto je zaručena konzistence dat při provádění jednotlivých transakcí. Výše uvedené výhody OLTP databází jsou umožněny modelem dat, na kterém jsou postaveny. Jedná se o relační datový model.

2.2 Relační datový model

Relační datový model je postaven na tabulkách (relacích) a vztazích mezi nimi. Tabulka představuje nějaký objekt nebo událost reálného světa modelovanou do podoby 2 – rozměrného pole složeného z jednotlivých sloupců a řádků. Může obsahovat několik atributů (domén, záhlaví sloupců), které vyznačují konkrétní charakteristiky objektu. Řádky tabulky představují jednotlivé záznamy (entity, instance objektu). Jednotlivé záznamy jsou v tabulce jednoznačně určeny primárním klíčem. Primární klíč je jeden nebo více zvolených atributů, které mají pro každý řádek jinou hodnotu. Tím je zaručena jednoznačnost záznamu a hovoříme o zajištění **entitní integrity**. Uložitelná hodnota záznamu pro jednotlivé atributy se stanovuje zřízením datového typu atributu, případně ještě dalším omezením přípustných uložitelných hodnot. Tímto způsobem se zajišťuje **doménová integrita** dat**.** Pokud vypozorujeme určitou vazbu mezi modelovanými objekty, zřídíme příslušný vztah mezi tabulkami v databázi. Vztahy mohou být třech druhů:
Vztah 1:1, 1:M, M:N. Díky zřízení vztahů mezi tabulkami můžeme najednou zobrazovat data z více tabulek. Zřízením správných vztahů mezi jednotlivými tabulkami zajistíme naší databázi integritu na úrovni vztahů, nebo též **referenční integritu.**

VZTAH 1:1 (one-to-one)

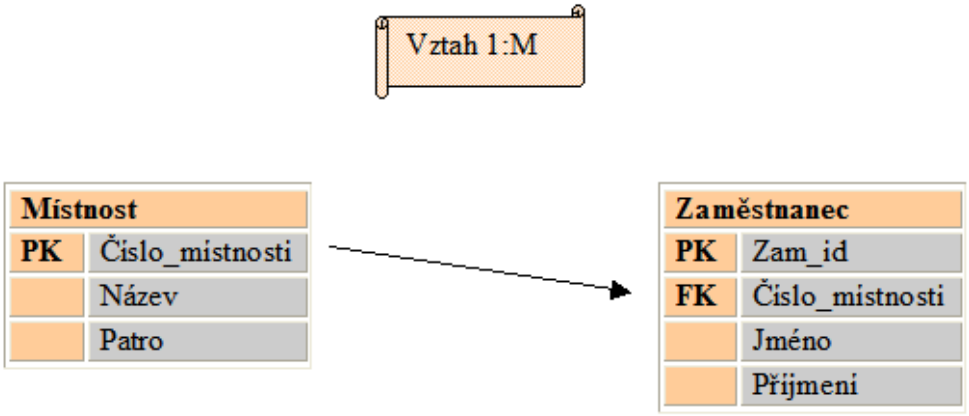
Jeden záznam (entita) z tabulky A je ve vztahu maximálně s jedním záznamem z tabulky B. Tento druh vztahu zajistíme tak, že kopii primárního klíče z tabulky A vložíme do tabulky B, kde bude opět představovat primární klíč. Jedna z tabulek je označena za primární, druhá je označena za sekundární.



Obrázek č.1 - Vztah 1:1

VZTAH 1:N (one-to-many)

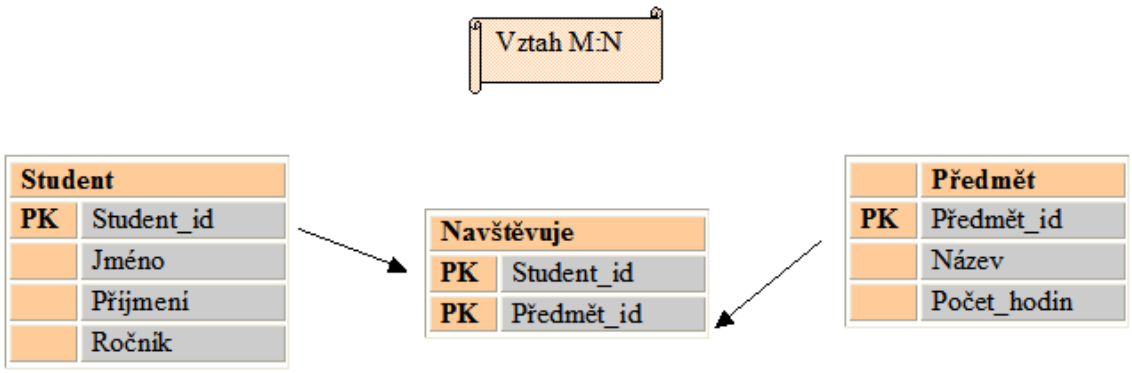
Tento vztah mezi dvěma tabulkami zřizujeme v případě, že jednomu záznamu z tabulky A přísluší jeden nebo více záznamů z tabulky B. Tento vztah se zajišťuje kopií primárního klíče tabulky A v tabulce B. Ta již má svůj primární klíč a kopie primárního klíče z tabulky A zde vystupuje jako **cizí klíč**.



Obrázek č.2 - Vztah 1:M

VZTAH M:N (many-to-many)

V tomto případě se k jednomu záznamu z tabulky A vztahuje jeden nebo více záznamů z tabulky B. A naopak, k jednomu záznamu z tabulky B se vztahuje jeden nebo více záznamů z tabulky A. Pro realizaci tohoto druhu vztahu použijeme třetí, tzv. vazební tabulku.
Do vazební tabulky umístíme kopie primárních klíčů obou tabulek. Oba klíče dohromady budou ve vazební tabulce plnit roli primárního klíče a každý zvlášť bude představovat cizí klíč do příslušné tabulky. Kromě kopií primárních klíčů obou tabulek může vazební tabulka obsahovat i další atributy, které vyjadřují určitou vlastnost vztahu.



Obrázek č.3 - Vztah M:N

K odstranění redundantních dat a optimalizaci výkonu databáze je vhodné, aby byly tabulky v normalizované podobě. Je celkem 5 normálních forem. V běžné praxi se setkáme se strukturami ve 2. nebo 3. normální formě.

1. normální forma (1NF)

Tabulka se nachází v první normální formě, pokud jsou všechny atributy atomické, nadále nedělitelné. To znamená, že můžou obsahovat jen skalární hodnoty. Pokud tabulka nesplňuje pravidla 1NF, nachází se v nulté normální formě (NFNF – non-first normal form).

Jméno	Příjmení	Adresa
Jiří	Novák	Havířov, 77412
Libor	Sedlář	Olomouc, 77200

Obrázek č.4 - Příklad tabulky v nulté normální formě

Jméno	Příjmení	Město	PSČ
Jiří	Novák	Havířov	77412
Libor	Sedlář	Olomouc	77200

Obrázek č.5 - Příklad tabulky v první normální formě

2. normální forma (2NF)

Tabulka splňuje podmínky druhé normální formy, pokud se nachází v první normální formě a všechny atributy kromě primárního klíče musí být plně závislé na celém primárním klíči.
Zkoumat splnění podmínek 2. normální formy má smysl jen u tabulek s vícesložkovým primárním klíčem. Pokud má tabulka primární klíč založený na jediném atributu, potom je podmínka 2. normální formy splněna automaticky.

Zboží	Dodavatel	E-mail	Cena
Mražená zelenina	Mrazírny Opava	mrz.op@seznam.cz	45
Mražené kuře	Mrazírny Opava	mrz.op@seznam.cz	110
Bílé víno	Vinařství Markov	vin.markov@atlas.cz	80

Obrázek č.6 - Příklad tabulky nesplňující pravidla 2NF

V zobrazené tabulce je primární klíč určen atributy Zboží, Dodavatel. Atribut Cena je závislý na celém primárním klíči. Jeho hodnota je dána jednak dodavatelem, tak i zbožím, které dodavatel nabízí. Atribut E-mail je závislý jen na části primárního klíče. Jeho hodnota závisí na dodavateli, nikoliv už na druhu zboží. Na následujícím obrázku je situace vyřešena v souladu se zásady 2NF.

Dodavatel_id	Dodavatel	E-mail
1	Mrazírny Opava	mrz.op@seznam.cz
2	Vinařství Markov	vin.markov@atlas.cz

Obrázek č.7 - Situace řešená dle pravidel 2NF

Zboží	Dodavatel_id	Cena
Mražená zelenina	1	45
Mražené kuře	1	110
Bílé víno	2	80

Obrázek č.8 - Situace řešená dle pravidel 2NF

3. normální forma (3NF)

Tabulka se nachází ve třetí normální formě v případě, že je ve tvaru 2NF a mezi neklíčovými atributy tabulky neexistují žádné závislosti.

Zákazník_id	Jméno	Příjmení	Město	PSČ
1	Jiří	Novák	Havířov	77412
2	Libor	Sedlář	Olomouc	77200

Obrázek č.9 - Příklad tabulky nesplňující 3NF

V uvedeném případě existuje vazba mezi neklíčovými atributy Město a PSČ. Daná situace má řešení vyobrazené na následujícím příkladě:

Zákazník_id	Město_id	Jméno	Příjmení
1	1	Jiří	Novák
2	2	Libor	Sedlář

Obrázek č.10 - Vyřešení problému s 3NF

Město_id	Město	PSČ
1	Havířov	77412
2	Olomouc	77200

Obrázek č.11 - Vyřešení problému s 3NF

Relační databáze jsou vynikajícím řešením pro běžné firemní systémy, v prostředí, kde je kladen důraz na zabezpečený paralelní přístup k datům, na rychlost vykonání každodenních operací s daty. Relační schéma splňuje tyto podmínky na velmi vysoké úrovni. Jak již bylo zmíněno dříve, data v OLTP databázích jsou udržována takovým způsobem, aby co nejpřesněji popisovala předmět podnikání firmy v současné situaci. V databázi se neudržují historická data. Pokud dochází ke změnám v datech, tyto se přepisují, popřípadě doplňují tak, aby byla vždy aktuální. Pokud bychom chtěli nad daty v relační databázi provádět nějaké analytické operace, budeme se potýkat s určitými problémy. Nedá se říct, že by analýza relačních dat byla nemožná, ale rozhodně to není důvod, pro který by byly relační databáze používány. A schéma relační databáze není navrženo pro optimální analýzu uložených dat. Nevhodnost použití relačních databázových systémů pro analýzu dat by se dala shrnout do několika bodů:

1) **Relační schéma** - data v relačních databázích bývají uložena v normalizovaných tabulkách, obsahují atomické údaje. Data jsou rozesety ve více vzájemně propojených tabulkách. Analýza takto uložených údajů je zdlouhavá a neefektivní. Velmi těžko se v takto strukturovaných datech hledají jednotlivé závislosti vybraných veličin.

2) **Zatížení systému** – pokud se používají primární informační systémy pro běžnou práci a zároveň i pro analýzu dat, zákonitě dochází k přetěžování systému a doba odezvy u běžných operací se zdelšuje.

3) **Absence historických dat** – v běžných relačních systémech se neudržují vhodná data pro analýzu. Při analýze určité datové veličiny sledujeme její vývojový trend podle určitých kritérií, ale hlavně v čase. Jak jsem se již zmínil, v transakčních databázích se historická data běžně nevyskytují.

4) **Roztroušenost dat** – je běžnou praxí, že firemní data se nacházejí v rámci podniku ve více datových zdrojích, pokaždé v jiném tvaru. Pokud chceme analyzovat množinu dat, která je roztroušena na více místech, znamená to požadovaná data přetransformovat a integrovat do centrálního datového úložiště a teprve pak můžeme začít provádět příslušnou analýzu.

Řešením problému s analýzou dat je použití databází vhodných pro práci s analytickými daty. Jedná se o multidimenzionální, nebo také o OLAP databáze.

2.3 Databáze OLAP

OLAP (Online Analytical Processing) značí online analytické zpracování dat a určuje způsob práce s daty. Na rozdíl od OLTP databází tedy OLAP databáze neslouží k práci s operačními, nýbrž výhradně s analytickými daty. Data jsou v této struktuře vhodně formovány do tvaru vhodného pro analýzu.

2.3.1 Datové struktury multidimenzionálních databází

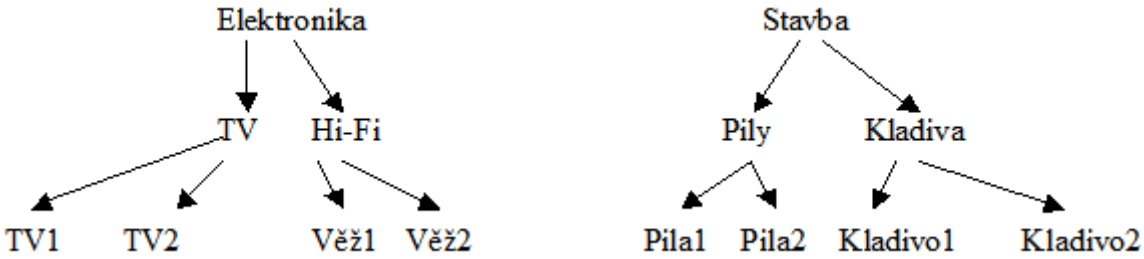
Multidimenzionální databáze sestávají z těchto základních strukturálních prvků:

1) **Multidimenzionální krychle, nebo také krychle OLAP**
2) **Měrné jednotky - fakta**
3) **Dimenze, které mohou, ale nemusí být sdílené**
4) **Hierarchie dimenzí**

Multidimenzionální krychle je základním stavebním kamenem analytických databází. Jestliže je v relačních databázích základem datová tabulka, můžeme si multidimenzionální kostku představit jako její protipól v analytické databázi. Je to základní struktura pro uchovávání dat. Stejně jako tabulek v relační databázi může být i krychlí v analytické databázi více. OLAP krychle sestává z měrných jednotek a dimenzí.
Měrné jednotky jsou veličiny, které potřebujeme analyzovat. Může jít o objem prodeje, o průměrný věk obyvatel, nebo třeba o počet klientů mobilního operátora. Najednou můžeme analyzovat celou skupinu měrných jednotek, neboli faktů.
Dimenze krychle OLAP můžeme brát za specifické pohledy, kterými se díváme na analyzovaná data. Pokud nás zajímá objem prodeje, lze na něj pohlížet různorodým způsobem. V případě dimenze *Produk*t si můžeme zvolit produkt, subkategorii nebo kategorii produktů, jejíž prodejnost chceme prozkoumat. Můžeme analyzovat prodejnost v určitých oblastech, potom je oblast druhou dimenzí krychle. Prodej stejného výrobku je zcela jistě proměnný v čase. Čas tedy představuje další, třetí dimenzi krychle. Počet dimenzí není omezen na 3, jak by se mohlo zdát. Jde sice o krychli, ale dimenzí může být daleko více. U Microsoft SQL Serveru je tento počet omezen na 64, což je více než dost. Fakta můžeme analyzovat i přes několik dimenzí najednou. Pokud chceme znát objem prodeje konkrétního výrobku ve vybraném regionu za minulý rok, nastavíme každou ze tří uvedených dimenzí na správnou hodnotu a výsledkem budou data odvozená a sumarizovaná podle našich požadavků v průsečíku hodnot všech dimenzí.
Sdílené dimenze jsou takové, které může používat více multidimenzionálních krychlí v rámci jedné databáze.
Další možností je vytvoření **víceúrovňové hierarchie** uvnitř dimenze. Regulováním úrovně hierarchie upravujeme **granularitu** měřených dat.
Pokud jsou u všech dimenzí nastaveny hierarchie na nejnižší úroveň, zkoumáme fakta na úrovni jednotlivých záznamů. V opačném případě jsou hierarchie nastavené na nejvyšším stupni a výsledkem jsou data pro dotyčné dimenze maximálně sumarizovaná.
Hierarchie dimenzí produkt, region a čas by mohly vypadat následovně:

Hierarchie dimenze produkt :

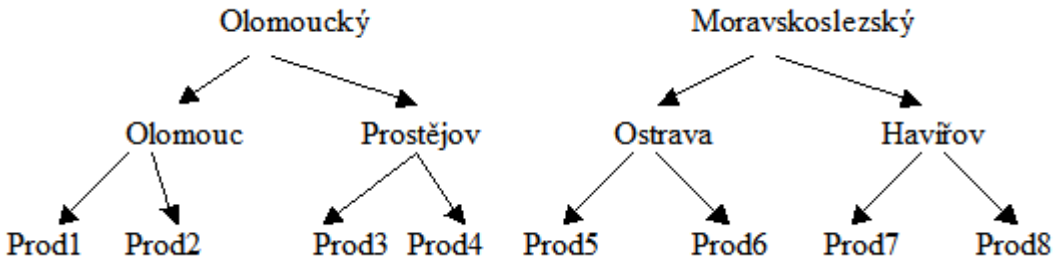
a) *Kategorie produktů*
b) *Subkategorie produktů*
c) *Produkt*



Obrázek č.12 - Příklad hierarchie dimenze produkt

Hierarchie dimenze oblast:

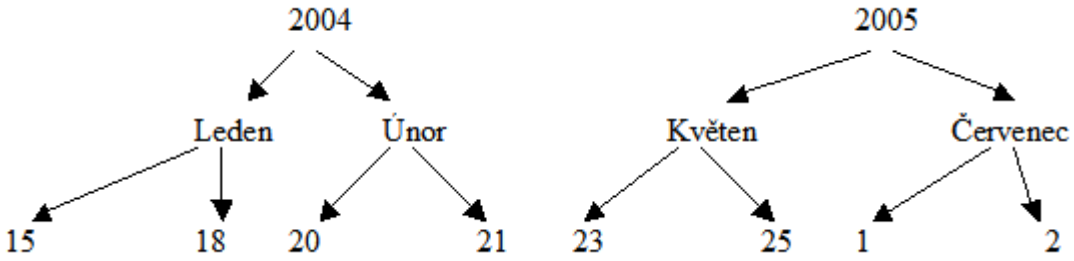
a) *Kraj*
b) *Město*
c) *Prodejna*



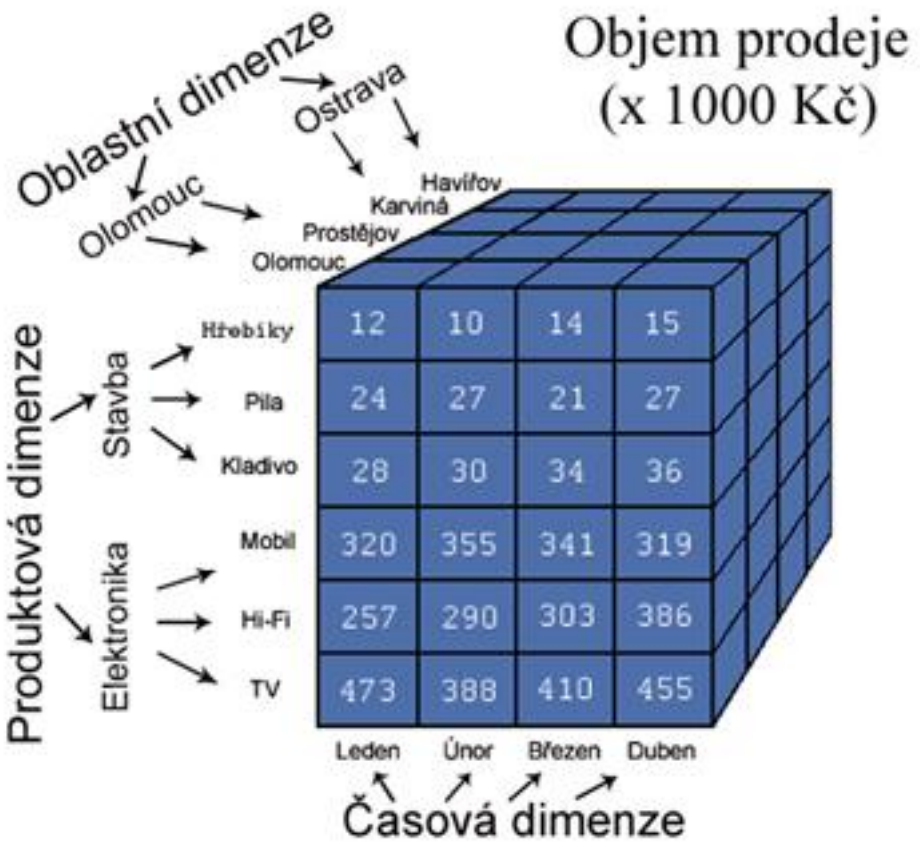
Obrázek č.13 - Příklad hierarchie dimenze oblast

Hierarchie dimenze čas:

a) *Rok*
b) *Měsíc*
c) *Den*



Obrázek č.14 - Příklad hierarchie dimenze čas



Obrázek č.15 - Příklad části krychle OLAP

Hierarchické procházení dimenzí je užitečné v případech, že nás zaujmou nějaké anomálie v datech. Pokud je např. roční obrat pro určitý rok abnormálně vysoký či nízký, budeme chtít zjistit, čím byla tato anomálie způsobena. Postoupíme tedy v dané dimenzi pro tento rok o úroveň níž a máme možnost sledovat údaje pro všechny měsíce daného roku. Pokud nás zaujme určitý měsíc, projdeme si pro něj vnořenou úroveň a budeme kontrolovat údaje za jednotlivé dny.
Multidimenzionální databáze eliminuje popsané nevýhody u relačních databází:

1) **Multidimenzionální schéma uložených dat** – v krychli jsou uložena nenormalizovaná data. Ty jsou často duplikována z důvodu realizace jednotlivých sumarizací faktů pro konkrétní úrovně hierarchií existujících dimenzí. Ve své podstatě představuje analyzování čehokoliv v reálném světě pohlížení na dotyčný subjekt z různých pohledů. Přesně na tomto principu je postavena architektura multidimenzionální krychle s jejími fakty a dimenzemi. Multidimenzionální schéma je tedy naprosto vhodnou strukturou pro analytická data.

2) **Zatížení systému** – použitím analytické databáze pro analýzy docílíme toho, že operační databáze zůstane primárně cílem pro vykonávání transakčních operací podnikového systému a samotná analýza bude probíhat odděleně, tudíž nebude ubírat na výkonu transakčnímu zpracování.

3) **Absence historických dat** – v relačních databázích dbáme na aktuálnost dat, ty jsou tedy průběžně upravovaná tak, aby popisovala současný stav. V datové krychli jsou ale data z převážné většiny statická a neměnná, změny se provádí jen výjimečně. Ukládají se zde údaje, které mají tzv. časové razítko, jsou tedy proměnná v čase. V OLAP krychli každopádně nepostrádáme historičnost dat jako jejich vlastnost důležitou pro analytické zpracování.

4) **Roztroušenost dat** – nasazení multidimenzionální databáze předpokládá existenci centrálního datového úložiště, do kterého jsou firemní data transformována a integrována z různých, zpravidla heterogenních systémů. Z tohoto úložiště jsou potom data nahrávána do multidimenzionální struktury. V multidimenzionální databázi tedy pracujeme s kompletními daty a máme jistotu, že prováděné analýzy zahrnují všechny důležité dostupné datové zdroje.

2.4 Datové sklady (Data warehouse)

Při probírání problematiky datových skladů často narazíme na pojem **Business Intelligence.** Ve stručnosti by se dalo říct, že se jedná o proces převodu údajů na informace a následný převod informací na poznatky. Na základě poznatků lze docházet k důležitým obchodním rozhodnutím. Hrají tedy důležitou roli v procesu manažerského rozhodování. Rozdíl mezi údaji a informacemi by se dal popsat následovně:
Údaje (data) chápeme jako jednotlivé záznamy v databázových tabulkách. **Informacemi** se data stávají ve chvíli, kdy jsou zpracovány takovým způsobem, že pro nás mají jasný význam. Pokud s informacemi pracujeme dále vhodným způsobem, můžeme na jejich základě docházet k určitým poznatkům.
Datový sklad můžeme chápat jako strukturované integrované datové úložiště pro velké objemy dat. Potřebné údaje se většinou nacházejí v různých, zpravidla nehomogenních systémech. Podobná data jsou také většinou uložena v každém systému jiným způsobem, mají tedy různý formát. Před uložením údajů do datového skladu se musí všechny uvažované údaje uvézt do jednotného a konzistentního stavu. Etapa, která zahrnuje extrakci dat z operačních zdrojů, jejich transformaci a následné nahrání do datového skladu se nazývá **ETL (Extraction, Transformation, Loading).** Data se nahrávají do relační databáze, která ale není organizovaná běžným způsobem. Tabulky zde nevystupují v normalizované formě, atd. Schéma databáze datového skladu je v takovém tvaru, aby se uložené údaje mohly pohodlně přehrávat do **multidimenzionálních databází** pro následnou analýzu nebo **data mining** (dolování dat). Nejznámější definice datového skladu pochází od Billa Inmona:

Datový sklad je podnikově strukturovaný depozitář subjektově orientovaných, integrovaných, časově proměnlivých, historických dat použitých na získávání informací a podporu rozhodování. V datovém skladu jsou uložena atomická a sumární data.

Základní vlastnosti dat uložených v datovém skladu by se dali podle definice shrnout do 4 bodů:

- **subjektová orientace** – data jsou do skladu zapisována podle předmětu, na který jsou orientována. Jsou tedy kategorizována podle jednotlivých subjektů – zákazník, prodejce, oblast. .

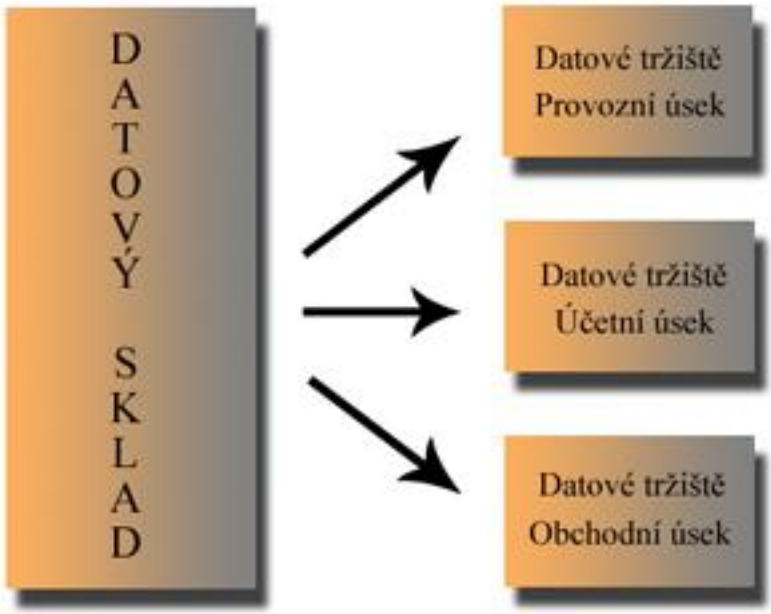
- **integrovanost** – data, která chceme zavádět do datového skladu, pochází většinou z nekonzistentního a neintegrovaného prostředí. Proto se musí před zavedením ‘vyčistit’ a uvézt do konzistentního a integrovaného stavu.

- **časová proměnnost –** data se do datového skladu ukládají jako série časových snímků. V operační databázi je platnost dat vyjádřena aktuálním stavem, v datovém skladu jsou data platná pro konkrétní časový snímek.

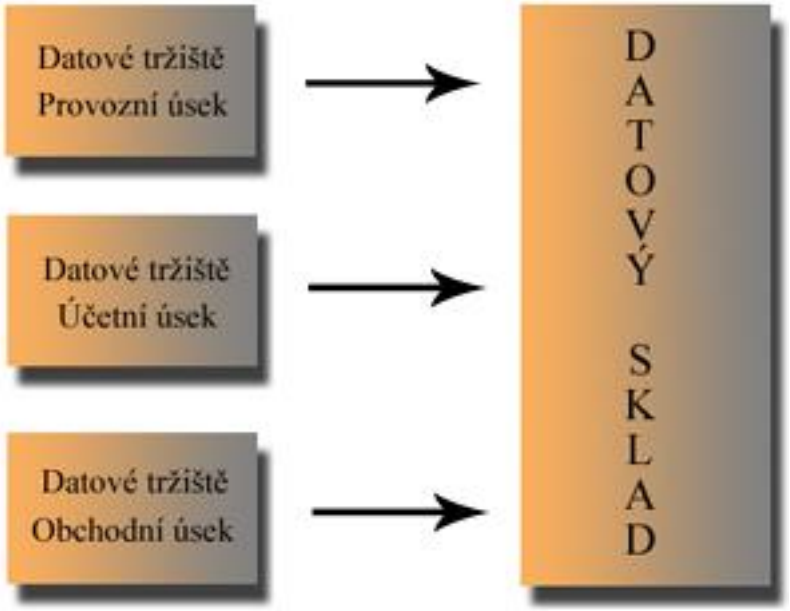
- **neměnnost –** zatímco v operačních databázích se data často upravují, v datovém skladu se již jednou uložená data nemění.

Náklady na budování a provoz datového skladu jsou poměrně vysoké, využívají se dnes hlavně u větších organizací, které dlouhodobě zpracovávají velké množství údajů a pečlivá analýza těchto údajů je pro ně velice důležitá (mobilní operátoři, banky, obchodní řetězce. .). Tvorba datového skladu je velice náročná a zdlouhavá činnost a vyžaduje opravdu velkou pečlivost. Je výhodnější analýzu neprovádět vůbec, než ji provádět nad nepřesnými údaji.
Ucelená část datového skladu se nazývá **datové trhy** (datamart). Může se jednat o realizaci části datového skladu určitého firemního oddělení.
Koncepce výstavby datového skladu postupným slučováním jednotlivých trhů se nazývá **přírůstková metoda 'zdola nahoru'.**
Opačný postup, kdy se vytvoří celý datový sklad a později se z něj vyčleňují jednotlivé datové trhy se nazývá **přírůstková metoda 'shora dolů'.**

Obrázek č.16 - Schéma datového skladu



Obrázek č.17 - Přírůstková metoda 'shora dolů'



Obrázek č.18 - Přírůstková metoda ‘zdola nahoru’

2.5 Analýza OLAP

Analýza OLAP představuje zpracování dat uložených v datovém skladu do podoby, která je vhodná pro koncové uživatele (manažery, ekonomy. .).
F. E. Codd popisuje analýzu OLAP souborem dvanácti pravidel:

- **Multidimenzionální konceptuální model** – uživatel by měl mít k dispozici multidimenzionální datový model umožňující efektivní analýzu.
- **Transparentnost** – celá technologie OLAP by měla být pro uživatele transparentní.
- **Dostupnost** – Systém OLAP by měl mít přístup jen k údajům potřebným k analýze. Zároveň by měl mít možnost přistupovat ke všem datům, bez ohledu na to, z jakého podnikového zdroje pocházejí, jak často jsou aktualizována atd.
- **Konzistentní vykazování** – I přes postupný nárůst počtu záznamů a velikosti databáze by uživatel neměl zaznamenávat podstatné snižování výkonu.
- **Architektura klient – server** – Systém OLAP by měl odpovídat principu architektury klient – server.
- **Generická dimenzionalita** – Každá dimenze údajů musí být ekvivalentní ve struktuře i operačních schopnostech.
- **Dynamické ošetření řídkých matic** – Systém OLAP by měl mít schopnost přizpůsobit své fyzické schéma konkrétnímu analytickému modelu, využívajícímu ošetření řídkých matic, bez snižování výkonu.
- **Podpora pro více uživatelů** – Systém OLAP musí být schopen podporovat víceuživatelský přístup ke konkrétnímu modelu.

- **Neomezené křížové dimenzionální operace** – Systém OLAP musí umět rozeznat hierarchické úrovně dimenzí a automaticky vykonat příslušné kumulované kalkulace v rámci dimenzí i mezi nimi.

- **Intuitivní manipulace s údaji** – Systém OLAP by měl umožnit konsolidované přeorientování cest na detailní úroveň a zpět (operade drill-down, drill-up). Uživatelské rozhraní by mělo všechny operace umožňovat způsobem 'ukázat a klepnout, případně zachytit a přemístit v buňkách krychle'.

- **Flexibilní vykazování** – Řádky, sloupce a buňky by měli být uspořádány takovým způsobem, který umožní analýzu a intuitivní vizuální prezentaci analytických sestav.

- **Neomezené dimenze a úrovně agregace** – V závislosti na požadavcích může mít analytický model více dimenzí, přičemž každá z nich může obsahovat vícenásobné hierarchie. Systém OLAP by neměl zavádět žádné umělé omezení počtu dimenzí nebo úrovní hierarchie.

Základem OLAP analýzy jsou data uložená v analytické databázi. Analytická databáze obsahuje jednu nebo více multidimenzionálních krychlí, které představují základní datové struktury v analytické databázi. Krychle dále sestává z faktů jako měrných jednotek a dimenzí jako různorodých pohledů na ně. Data se do analytické databáze neukládají přímo. Jak již bylo zmíněno, nejdříve se čerpají z operačního prostředí. Z různých operačních zdrojů se data nahrávají do datového skladu. Ale nejdříve se musí různými transformacemi uvézt do konzistentního a jednotného tvaru. Datový sklad je v podstatě relační databáze, která využívá nenormalizované tabulky, které se daleko více uplatní pro uchovávání dat sloužících jako podklad pro analytické operace. Správně uložená data se z datového skladu nahrávají do multidimenzionálních krychlí. Jedná se o tzv. **processing** krychle. Tyto operace vyžadují vcelku velký časový prostor. Data se v multidimenzionálních databázích nikdy neupravují. Jednotlivý processing znamená nahrání zcela nových informací do krychle. Proto se plnění analytických databází provádí jako plánovaná akce většinou přes noc, kdy nejsou aktuální analýzy nad databází prováděny. Každá krychle OLAP vyžaduje existenci svého protějšku v relační databázi datového skladu ve formě datového zdroje. Jedná se o podkladové schéma složené z tabulek 2 typů:

- **tabulky faktů** – zde jsou zaznamenávány veličiny, které budou předmětem analýzy.

- **tabulky dimenzí** – zde se ukládají informace, dle kterých jsou fakta odvozována.

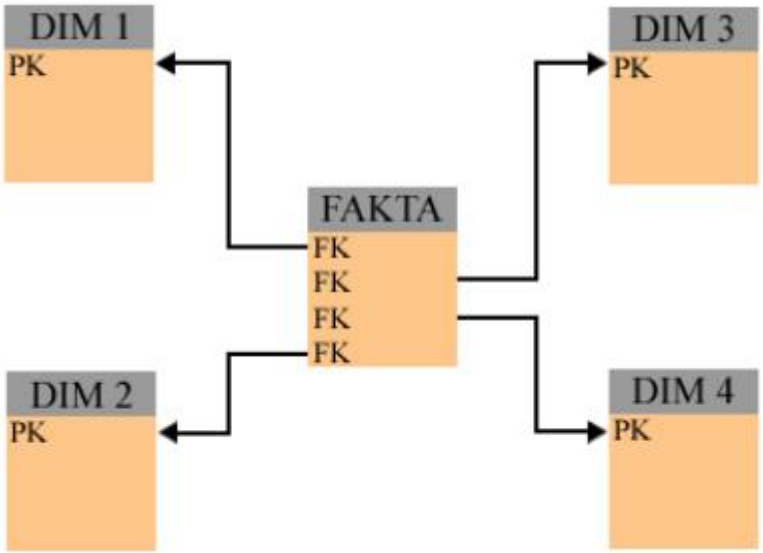
Spojení tabulek faktů a dimenzí je realizováno přes primární klíče v tabulkách dimenzí a cizí klíče v tabulkách faktů. Existují dva typy multidimenzionálních datových schémat:

2.5.1 Schéma hvězda

Jedná se o jednodušší variantu. Na zpravidla jednu tabulku faktů jsou navázány tabulky dimenzí. Pro každou dimenzi jedna tabulka. Všechny tabulky dimenzí se nachází v nenormalizovaném tvaru. Nevýhoda tohoto modelu spočívá v časově náročnějším procesu zavádění údajů. Na druhou stranu tento model poskytuje vysoký dotazovací výkon.

Cas_id	Datum	Rok	Kvartal	Mesic	Den
1	20.2.2006	2006	1	2	20
2	4.7.2007	2007	3	7	4
3	23.11.2007	2007	4	11	23

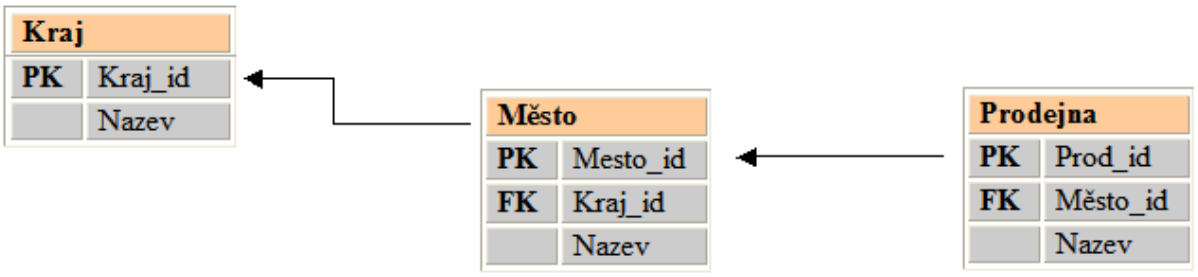
Obrázek č.19 - Příklad tabulky nenormalizované dimenze



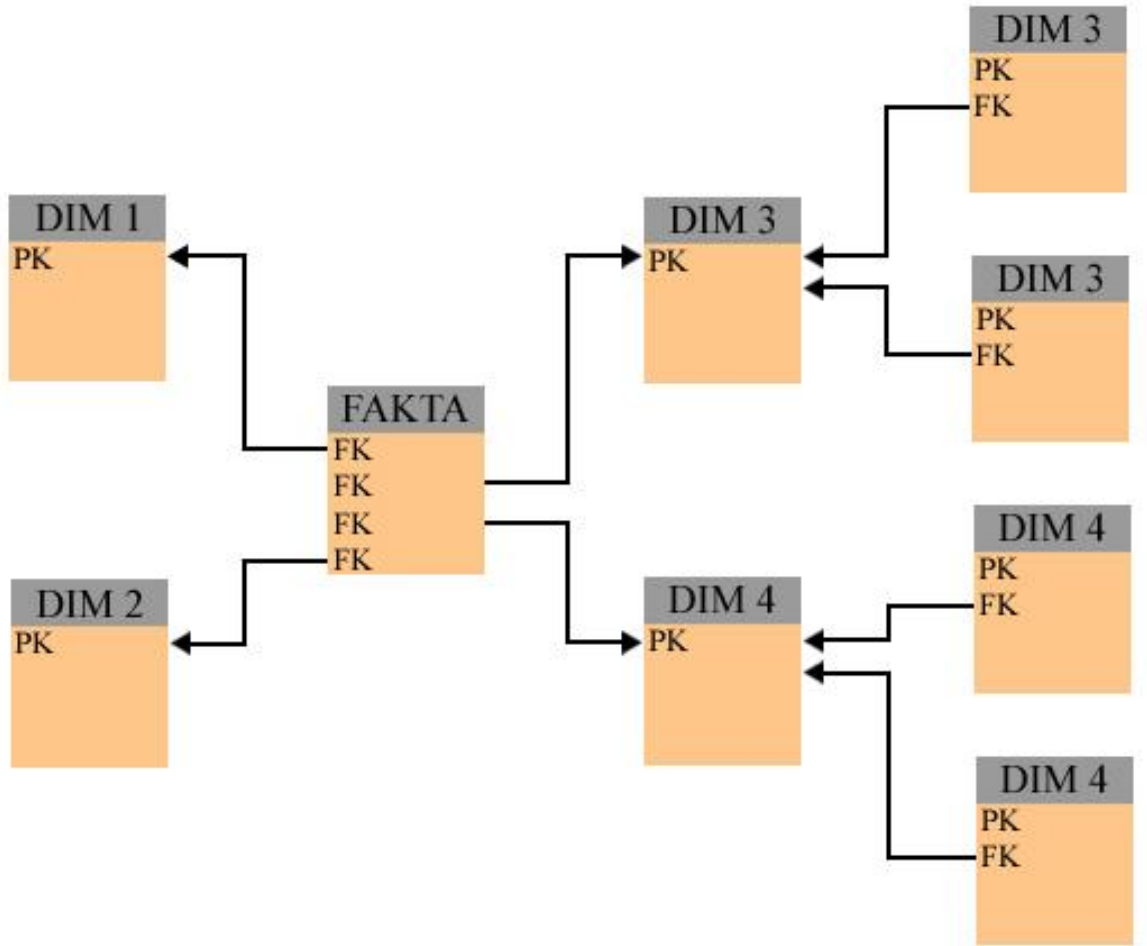
Obrázek č.20 - Hvězdicové schéma

2.5.2 Schéma sněhová vločka

Model sněhové vločky je návrhově složitější variantou. Jednotlivé dimenze mohou být složeny z jedné nebo více tabulek. Tabulky tvořící složenou dimenzi bývají v normalizovaném tvaru. Proces nahrání dat do tohoto schématu je díky normalizovaným tabulkám rychlejší než v případě hvězdy. Nevýhodou je v tomto případě nižší dotazovací výkon, jelikož se často musí přistupovat k více relačně svázaným tabulkám.



Obrázek č.21 - Příklad složené dimenze



Obrázek č.22 - Schéma sněhová vločka

2.5.3 Úložiště multidimenzionálních údajů

- **MOLAP** – multidimenzionální OLAP ukládá analytická data do svých vlastních multidimenzionálních struktur. To znamená, že jsou uložena přímo v analytické databázi. Zde je viditelná hlavní nevýhoda tohoto typu úložiště. Je to duplicita dat. Ty jsou uložena jak v datovém skladu, tak i v multidimenzionální databázi. Veškeré možné souhrny jsou předem spočítány a uloženy do analytických struktur. Z tohoto důvodu poskytuje MOLAP maximální dotazovací výkon.

- **ROLAP** – relační OLAP je koncipován odlišným způsobem. Data zůstávají uložena v datovém skladu a ROLAP poskytuje jen tzv. multidimenzionální pohledy na ně. Pohledy jsou realizovány jazykem SQL nad relační databází datového skladu. Uživatelské požadavky na analytické údaje jsou transformovány na příkazy SQL a vrácená data jsou předkládána uživatelům. ROLAP eliminuje problém s redundancí údajů. Ovšem dotazovací výkon je nižší než u mechanismu MOLAP.

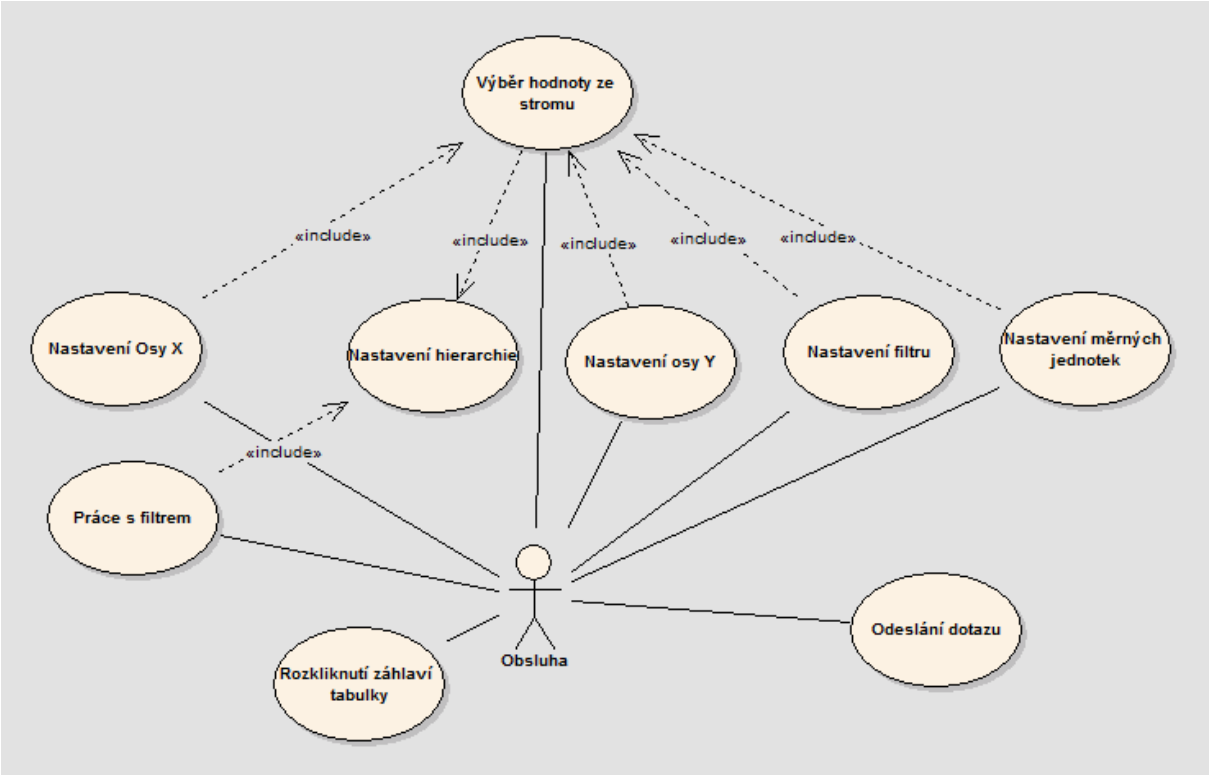
- **HOLAP** – hybridní OLAP je funkčním průnikem obou předchozích technologií. Snaží se vhodným způsobem eliminovat jejich nevýhody a využít jejich výhod. Jednotlivé údaje zůstávají uloženy v relační databázi a sumace se ukládají do multidimenzionální databáze.

2.5.4 Přístup k analytickým údajům

Aplikace zpracovávající analytická data jsou většinou koncipovány jako **klient – server.** Pokud je většina aplikační logiky vykonávána na straně serveru, jedná se o ‘tenkého’ klienta. V opačném případě se klient nazývá ‘tlustým’. V prvním případě zajišťuje klient hlavně prezentační funkce, popřípadě vykonává validační operace s formulářovými daty. Analytická databáze je součástí analytických služeb databázového serveru a je tedy umístěna na serverové straně. Mezi používané klienty analytického serveru patří například aplikace kancelářského balíku **Microsoft Office.** Tyto aplikace používají k zobrazování analytických údajů kontingenční tabulku (pivot table).
Další možností je použít komponenty **OWC (office web components).** Jedná se o kontingenční tabulku a kontingenční graf (Pivot table, Pivot chart). Lze je umístit do webové stránky jako objekty a využívat jejich programové rozhraní pomocí klientského skriptovacího jazyka (např. JavaScript). Vyžaduje se však existence a registrace příslušné knihovny v operačním systému. Z výše uvedeného rozdělení klient – server aplikací plyne, že se tímto způsobem budují ‘tlustí’ klienti analytického serveru. V tomto případě se totiž podstatná část aplikace skrývá v podobě klientské komponenty. Tato komponenta poskytuje grafické rozhraní pro výběr a prezentaci údajů a zasílá patřičně formulované požadavky na data analytickému serveru.
‘Tenký’ klient vyžaduje komunikaci s analytickým serverem přímo na serverové straně. K realizaci můžeme využít některý ze serverových skriptovacích jazyků jako je např. PHP. Nebo lze sáhnout po ucelené webové technologii jakou představuje třeba ASP.NET. V těchto případech slouží klient výhradně k prezentačním úkolům a ke komunikaci s webovým serverem. Předmětem této bakalářské práce je vytvoření serverové komponenty, která by umožňovala vývoj analytických klient – server aplikací za využití ‘tenkého’ klienta.

3. Analytická dokumentace

3.1 Případy užití



Obrázek č.23 - Use Case

3.1.1 Use Case *Výběr hodnoty ze stromu*

1) Obsluha provede navigaci stromem
2) Obsluha klikne myší na požadovanou hodnotu stromu
3) Systém porovná typ vybrané hodnoty ze stromu s typem aktivního checkboxu formuláře
4) Pokud se typy neshodují, neprovede se žádná další akce
5) Pokud se typy shodují, systém na základě typu aktivního checkboxu rozhodne o další akci
6) V případě, že je aktivní checkbox pro nastavení osy X, osy Y nebo měrných jednotek, zkopíruje se hodnota vybraná ze stromu do textového pole přidruženého k aktivnímu checkboxu.
7) V případě, že je aktivní checkbox pro nastavení filtru, použije se Use Case *Nastavení hierarchie*

3.1.2 Use Case *Nastavení osy X*

1) Obsluha vybere textové pole pro nastavení osy X aktivací *checkboxu* přidruženého k textovému poli pro nastavení osy X
2) Použije se *Use Case Výběr hodnoty ze stromu*

3.1.3 Use Case *Nastavení osy Y*

1) Obsluha vybere textové pole pro nastavení osy Y aktivací *checkboxu* přidruženého k textovému poli pro nastavení osy Y
2) Použije se *Use Case Výběr hodnoty ze stromu*

3.1.4 Use Case *Nastavení měrných jednotek*

1) Obsluha vybere textové pole pro nastavení měrných jednotek aktivací *checkboxu* přidruženého k textovému poli pro nastavení měrných jednotek
2) Použije se *Use Case Výběr hodnoty ze stromu*

3.1.5 Use Case *Nastavení filtru*

1) Obsluha aktivuje *checkbox* pro nastavení filtru
2) Použije se Use Case *Výběr hodnoty ze stromu*

3.1.6 Use Case *Nastavení hierarchie*

1) Obsluze se v novém okně zobrazí stromová struktura hierarchie
2) Obsluha provede navigaci hierarchií
3) Obsluha provede nastavení hierarchie
4) Obsluha stiskne tlačítko *OK* a okno se zavře

3.1.7 Use Case *Práce s filtrem*

1) Obsluha stiskne tlačítko *Zobrazit filtr*
2) Obsluze se zobrazí nové okno se seznamem hierarchií
3) Pokud obsluha klikne myší na některou z hierarchií, použije se Use Case *Nastavení hierarchie*
5) Po stisknutí tlačítka *Vyřadit vybrané* dojde k odstranění vybraných hierarchií
6) Po stisknutí tlačítka *Vyřadit všechny* dojde k odstranění všech hierarchií
7) Po stisknutí tlačítka *OK* dojde k zavření celého okna

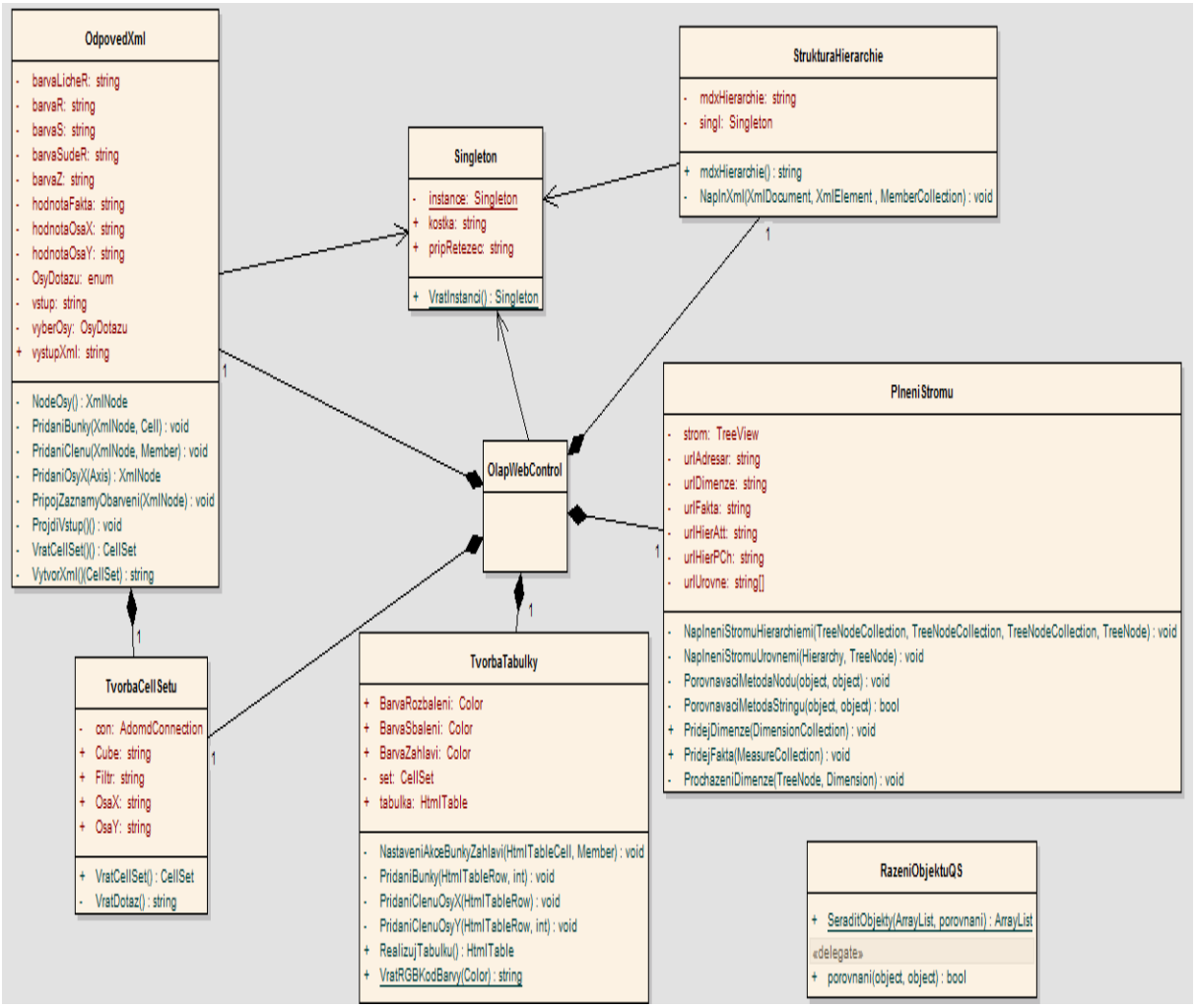
3.1.8 Use Case *Odeslání dotazu*

1) Obsluha stiskne tlačítko *Odeslat dotaz*
2) Systém odešle celou stránku na server
3) Systém zobrazí aktualizovanou stránku

3.1.9 Use Case *Rozkliknutí záhlaví tabulky*

1) Obsluha klikne myší na buňku záhlaví tabulky
2) Systém aktualizuje tabulku

3.2 Diagram tříd



Obrázek č.24 - Diagram tříd

4. Programátorská dokumentace

4.1 Použité technologie při realizaci komponenty

a) Vývojové prostředí ***Visual Studio 2008***
b) Platforma pro vývoj webových aplikací ***ASP.NET 3.5***
c) Programovací jazyk ***C#***
d) Objektový model ***ADOMD.NET*** pro práci s analytickým serverem
e) Analytické služby ***SQL Serveru 2008***
f) Jazyk ***MDX*** pro manipulaci s multidimenzionálními daty
g) Scriptovací jazyk ***JavaScript*** s využitím objektového modelu ***HTML DOM***
h) Značkovací jazyk ***HTML*** (Hypertext Markup Language)
i) Kaskádové styly ***CSS*** (Cascading Style Sheets)
j) Značkovací jazyk ***XML*** (Extended Markup Language)
k) Technologie ***AJAX*** (Asynchronous JavaScript and XML) pro asynchronní volání serveru

4.2 Seznam naprogramovaných klientských funkcí

4.2.1 Skript *RizeniAnalyzy.js*

OsetreniKliku ()
ZpracovaniCallbacku (par, dalsi)
VytvoreniXml ()
NoveXml (vstup)
VratNodeElementu (nazev)
NaplneniElementuBunkyZahlavi (elementBunka, jmenoBunky, textBunky)
NaplneniOsyX ()
NaplneniOsyY ()
NaplneniHodnot (elementHodnoty)
VratJmenaPrenesenychZahlavi (nazevOsy)
VratOdesilanyRetezec (osaBunky, idBunky, seznamClenu)
AktualizujData (retXml)

4.2.2 Skript *FiltrovaniDotazu.js*

TestujFiltr (nazevHierarchie)
OtevriOknoHierarchie (idHierarchie)
OtevriOknoFiltru ()
PridejHierarchii (retHierarchie)
NaplneniHierarchie (plnenyNode, poleNoduKNaplneni)
ZobrazHierarchii (doc, idHierarchie)
ZobrazClenyHierarchie (doc, poleClenu, elementSeznam, JeOtevrenyNadrazenyClen)
ExistujeHierarchie (idHierarchie)
VratElementXml (doc, typ, id)
VratElementHtml (doc, typ, id)
VratNoveTlacitko (doc, stav, id)
VratNovyCheckBox (doc, id, value)
VratNovouPolozkuSeznamu (doc, id)
VratNovySeznam (doc, id)
VratNovyPopis (doc, text)
OsetreniKlikuTlacitka ()
OsetreniZatrzeniCheckBoxu ()
ProvedZmenuNaVsechnyPotomky (chckBox, zatrzeno)
ProvedZmenuNaVsechnyPredchudce (chckBox)
VratSeznamPodrizenyCheckBoxu (chckBox)
ProvedZmenuCheckBoxuSeznamu (seznam, zatrzeno)
ProvedZmenuPredchudce (chckBox, zatrzeni, elementXml)
ProjdiZatrhnutiPolozekSeznamu (polePolozek)
VratZatrhnutiCheckBoxu (polozkaSeznamu)
ZmenZatrhnutiAktualizujXml (checkBox, zatrhnuti)
VratElementXmlPrislusnyCheckBoxu (checkBox)
ProvedZatrzeniPotomkuElementuXml (elementClen, zatrzeni)
VratIdHierarchie (idClenu)
ProvedZmenuVzhleduPopisku (popisek, alternativniVzhled)
VratNazvyVybranychHierarchii ()
PridejFiltrDoFormulare ()
VratRetezecFiltru ()
VratVybraneClenyHierarchie (elementHierarchie)
VratVybranePotomkyClenu (elementClen)
UlozFiltrJakoCookie ()
ZavriOknoHierarchie ()
ZavriOknoFiltru ()
OdebraniHierarchiiZFiltru ()
ZatrhniVsechnyClenyHierarchie (hierarchie)

4.2.3 Skript *ProchazeniServerovehoXml.js*

ProjdiDosleXml (nazevOsy, xml)
ProjdiPodleX (xml)
ProjdiPodleY (xml)
DoplnRadekHodnot (radekHodnotMistnihoXml, poziceVlozeni, radekHodnotDoslehoXm)
VytvorPotomkaZahlavi (zahlaviDoslehoXml)

4.2.4 Skript *TvorbaDotazu.js*

OsetreniPrepinacu (prep)
KlikStromu ()
HodnotaNodu (element)
KliknutoNaNod (element)
ZobrazitFakta (hodnotaElementu)
ZobrazitHodnotuNodu (hodnotaElementu)
NastavitFiltr (hodnotaElementu)
VratZapnutyRadButton ()
VratTypPovolenehoVstupu (prepinac)

4.2.5 Skript *RozeviraniZahlavi.js*

RozevriBunku (elementBunky)
RozevriBunkuPodleX (elementBunky)
RozevriBunkuPodleY (elementBunky)
DoplnRadek (cisloRadku, elementBunky, poziceNaOse, poziceAbsolutni)
VytvorRadek (elementBunky, poziceOtevrene, poziceVsechny)
VlozBunku (radek, pozice, elementBunky)
Struktura (nalezeno, pozice)
VratTextNodeElementu (element)
JeViditelneZahlavi (elementZahlavi)

4.2.6 Skript *ZaviraniZahlavi.js*

ZavriBunku (elementBunky)
ZavriBunkuPodleX (elementBunky, poziceNaOse)
ZavriBunkuPodleY (elementBunky, poziceNaOse)
VratPocetOtevrenychPotomku (elementZahlavi)

4.2.7 Skript *SpolecneFunkce.js*

PoziceElementuZahlaviNaOse (elementZahlavi, kolekceElementu, jenViditelne)
OznacAObarviBunku (bunka, stav)
VratOsuElementu (element)
MaElementPotomky (element)
MaPotomky (textBunky)
JeElementOtevreny (element)
MaElementPotomkyPreneseny (element)

4.3 Použité XML formáty

4.3.1 XML pro popis struktury hierarchie

4.3.1.1 Seznam elementů

<hierarchie>
atributy: id (string), nazev (string)
<clen>
atributy: id (string), nazev (string), zatrzeni (*uplne – castecne - zadne*)

4.3.1.2 Struktura

```
<hierarchie>
    <clen>
        <clen></clen>
        .
        .
        <clen></clen>
    </clen>
    .
    .
    <clen>
        <clen></clen>
        .
        .
        <clen></clen>

    </clen>
</hierarchie>
```

4.3.2 XML pro popis výsledků dotazů MDX (osa X)

4.3.2.1 Seznam elementů

<osa>
<radek>
<zahlavi>
atributy: jmeno (string), potomci (bool)
<hodnota>

4.3.2.2 Struktura

```
<osa>
    <radek>
        <zahlavi></zahlavi>
        .
        .
        <zahlavi></zahlavi>
    </radek>
    <radek>
        <hodnota></hodnota>
        .
        .
        <hodnota></hodnota>
    </radek>
    .
    .
    <radek>
        <hodnota></hodnota>
        .
        .
        <hodnota></hodnota>
    </radek>
<osa>
```

4.3.3 XML pro popis výsledků dotazů MDX (osa Y)

4.3.3.1 Seznam elementů

<osa>
<radek>
<zahlavi>
atributy: jmeno (string), potomci (bool)
<hodnota>

4.3.3.2 Struktura

```
<osa>
    <radek>
        <zahlavi></zahlavi>
        <hodnota></hodnota>
        .
        .
        <hodnota></hodnota>
    </radek>
    .
    .
    <radek>
        <zahlavi></zahlavi>
        <hodnota></hodnota>
        .
        .
        <hodnota></hodnota>
    </radek>
</osa>
```

4.3.4 XML pro zaznamenání aktuálního stavu výsledkové tabulky

4.3.4.1 Seznam elementů

<osaX>
<osaY>
<zahlavi>
atributy: jmeno (string), potomci (bool), klient (bool), otevreno (bool)
<hodnoty>
<radek>
<hodnota>

4.3.4.2 Struktura

```
<osaX>
        <zahlavi>
                <zahlavi></zahlavi>
                .
                .
                <zahlavi></zahlavi>
        </zahlavi>
        .
        .
        <zahlavi>
                <zahlavi></zahlavi>
                .
                .
                <zahlavi></zahlavi>
        </zahlavi>
</osaX>
<osaY>
        //struktura elementu stejná jako u <osaX>
</osaY>
<hodnoty>
        <radek>
                <hodnota></honota>
                .
                .
                <hodnota></honota>
        </radek>
        .
        .
        <radek>
                <hodnota></honota>
                .
                .
                <hodnota></honota>
        </radek>

</hodnoty>
```

4.3.5 XML pro zaznamenání informací o filtru

4.3.5.1 Seznam elementů

<filtr>
<hierarchie>
atributy: id (string), nazev (string), vybrano (bool)
<clen>
atributy: id (string), nazev (string), vybrano (bool),
zatrzeni (*uplne – castecne – zadne*), otevreno (bool)

4.3.5.2 Struktura

<filtr>
<hierarchie>
<clen>
<clen></clen>
.
.
<clen></clen>
</clen>
.
.
<clen>
<clen></clen>
.
.
<clen></clen>
</clen>
</Hierarchie>
.
.
<hierarchie>
- *stejná vnitřní struktura všech elementů <hierarchie>*
</hierarchie>
</filtr>

4.4 Popis programu:

Komponenta je programově rozdělena na část serverovou a část klientskou. Na straně serveru se komponenta připojuje k analytickému serveru, kde získává potřebná data a realizuje prvek do HTML, který se vkládá do stránky odesílané klientovi. U klienta má prvek na starosti interakci s uživatelem a vysílání požadavků na server. Obě části spolu komunikují a informace si vyměňují za využívání technologie AJAX.

4.4.1 Třída webového prvku

Základem prvku je samotná jeho třída *OlapWebControl* z identického jmenného prostoru. Je odvozená z bázové třídy *WebControl.* Poskytuje základní sadu vlastností umožňující prvku získat nezbytné informace pro jeho bezchybné fungování. *PripojovaciRetezec* je vlastnost, ve které se ukládá název analytického zdroje dat, s nímž bude prvek spolupracovat, název analytické databáze a přihlašovací údaje. *Kostka* obsahuje informaci o názvu vybrané multidimenzionální struktury. Zbylé vlastnosti již nejsou pro chod prvku nezbytné a jejich zadání není striktně vyžadováno. Jedná se o barvu pozadí prvku a zabarvení jednotlivých segmentů výsledkové tabulky. Při nevyplnění těchto vlastností se při inicializaci prvku nastaví jejich implicitní hodnoty. Při inicializaci prvku se jako první spouští metoda *OnInit.* V té se získávají URL adresy webových zdrojů zahrnutých do projektu. Jedná se o obrázky, skripty a soubor kaskádových stylů. Tyto adresy používá klientský prohlížeč pro jejich identifikaci na serveru. Dále se zde zřizuje reference zpětného volání klienta, která představuje kostru pro realizaci asynchronní komunikace mezi klientem a serverem. Metoda *CreateChildControls* obsahuje kód pro tvorbu dceřinných webových prvků. Jde především o vizuální prvky jako textová pole, tlačítka nebo složitější *TreeView*. Metoda RenderContents se spouští v závěru životního cyklu prvku a zajišťuje jeho převedení do HTML kódu. V této fázi se vytváří samotné HTML značky doplněné o HTML obsah jednotlivých dceřinných prvků. Aby měl prvek možnost přistupovat k datům *PostBacku*, musí implementovat rozhraní *IPostBackDataHandler*. Toto rozhraní definuje signatury dvou metod: *LoadPostData* přistupuje k samotným datům *PostBacku* a *RaisePostDataChangedEvent* umožňuje odpalovat dodatečné události po aplikaci změn provedených na základě dat *PostBacku.* Pro realizaci *CallBacku* je nutné implementovat rozhraní *ICallbackEventHandler*, které rovněž definuje signatury pro 2 metody:
Při zpětném volání vysílá klient na server společně s požadavkem jeden parametr řetězce, který obdrží metoda *RaiseCallbackEvent.*
Metoda *GetCallbackResult* se spouští jako reakce serveru na zpětné volání a posílá na klienta s odpovědí také jeden řetězcový parametr.

4.4.2 Průběh výpočtu

4.4.2.1 Realizační proces prvku

Po provedení metody *OnInit* webového prvku se dostává ke slovu metoda *CreateChildControls*, která má za úkol vytvořit podřízené prvky. Při volání této metody se rozlišuje, zda se jedná o prvotní inicializaci, nebo zda k inicializaci dochází v reakci na zpětné odeslání stránky na server. V prvním případě se místo vytvoření výsledkové tabulky pošle klientovi zpráva o tom, že žádné výsledky ještě nejsou k dispozici. V druhém případě již stránka předala prvku informace potřebné pro zhotovení výsledkové tabulky a dojde tedy k jejímu vytvoření. Nejdříve je vytvořena instance třídy *TvorbaCellSetu.* V konstruktoru třídy se předává název krychle, MDX výraz pro horizontální a vertikální osu dotazu a hodnoty pro filtrování dotazu. Třída je zodpovědná za poskládání syntakticky správného dotazu, který se následně použije pro získání výsledné sady dat z analytické databáze. Tuto výslednou množinu typu *CellSet* vrací metoda *VratCellSet*. Objekt ze třídy *TvorbaTabulky* vytváří Html tabulku naplněnou výsledky. V konstruktoru této třídy se jako jeden z parametrů předává objekt typu CellSet. Jako další v pořadí má za úkol metoda *CreateChildControls* vytvořit instanci třídy

TreeView, která bude plněna informacemi o kompletní struktuře vybrané krychle. K tomuto účelu se inicializuje instance třídy *PlneniStromu*. Metoda *PridejFakta* přebírá jako parametr kolekci *MeasureCollection*, obsahující všechny dostupné měrné jednotky krychle. Těmito údaji se postupně plní interní objekt *TreeView*. Obdobně metoda *PridejDimenze* s předanou kolekcí *DimensionCollection* naplní *TreeView* dostupnými dimenzemi kostky. Každá dimenze se do stromu přidá i se všemi svými hierarchiemi a každá hierarchie se svými úrovněmi. Metoda *VratNaplnenyStrom* vrací kompletně zrealizovaný *TreeView*. Zbytek kódu metody *CreateChildControls* vytváří ostatní jednoduché prvky jako jsou tlačítka a textová pole. Metoda *RenderContents* zapisuje do výstupního proudu posloupnost HTML kódu. K tomuto účelu slouží systémová třída *HtmlTextWriter,* pomocí které lze postupně zadávat jednotlivé HTML tagy i s jejich atributy. V místě, kde je potřeba umístit kód některého z dceřinných ovládacích prvků, se zavolá jeho metoda *RenderControl*. Ta do výstupního proudu přidá kompletní HTML dotyčného prvku. Hotový HTML výstup se předá webové stránce, která se následně odesílá klientskému prohlížeči.

4.4.2.2 Rozhraní prvku

Při prvním načtení stránky do prohlížeče má uživatel za úkol nastavit obě osy dotazu a typ měrné jednotky, která bude předmětem analýzy. V případě potřeby lze libovolnou hierarchii připojit k filtru dotazu. Webový prvek v prohlížeči vizuálně sestává z následujících funkčních ovládacích prvků:
TreeView s metadaty krychle, trojice párů *textbox – checkbox* pro práci s osami x ,y a s fakty, dále *checkbox* pro nastavení filtru, tlačítko pro zobrazení filtru, tlačítko pro odeslání stránky na server a z interaktivní tabulky výsledků dotazu. Jakmile dojde ke kliknutí do sekce stromu, spustí se funkce *KlikStromu* umístěná ve skriptu *RizeniAnalyzy.js*. Funkce zjistí, zda došlo ke kliku na některý node prvku *TreeView*. Potom dojde k přenesení hodnoty ze stromu do některého z *textboxů*, nebo se otevře nové okno pro práci s hierarchií. Vždy musí být ve shodě typ označeného uzlu stromu s aktivním *checkboxem*. V případě, že ke shodě nedojde, neprovede se žádná změna.

4.4.2.3 Výběr hierarchie ze stromu

Jestliže se má provést scénář s otevřením nového okna, je k tomuto účelu zavolána funkce *TestujFiltr* s parametrem názvu hierarchie. Funkce je definována ve skriptu *FiltrovaniDotazu.js*. Tato funkce nejdříve testuje, zda již byla dotyčná hierarchie použita. V tom případě jsou již všechna data přenesená na klienta a vykoná se volání funkce *OtevriOknoHierarchie* s parametrem obsahujícím název hierarchie. Jestliže tomu tak není, musí se nejprve potřebná data získat ze serveru. Vytvoří se proměnná obsahující název hierarchie a na začátek se doplní o znak ‘f’, který značí, že data za ním obsahují právě název hierarchie. Tato proměnná se předá jako parametr funkci *VolejServer*. Tato funkce byla vytvořena při inicializaci webového prvku a představuje referenci zpětného volání. Tedy při jejím spuštění dojde k asynchronnímu volání serveru.
Na serveru je spuštěna metoda *RaiseCallbackEvent*. V těle metody se provede testování prvního znaku parametru. Zjistí se, že parametr obsahuje název hierarchie. Následně se vytvoří instance třídy *StrukturaHierarchie*, které se pomocí konstruktoru předá název hierarchie. Návratová hodnota z *CallBacku* se složí z příznaku ‘f’ a výsledku volání metody *VratStrukturuHierarchie*. V těle metody se z analytického serveru získají data hierarchie a vytvoří se z nich XML popisující strukturu hierarchie. XML řetězec se stává výsledkem volání metody.
Řízení výpočtu se po provedení metody *GetCallbackResult* třídy *OlapWebControl* zajišťující odpověď na *CallBack* klienta předává zpět klientské části prvku.
Ve skriptu *RizeniAnalyzy.js* je definovaná funkce *ZpracovaniCallbacku,* která obsluhuje výsledek volání *Callbacku*. V těle této funkce se provede volání funkce *AktualizujData* s parametrem řetězce. Ten obsahuje XML obdržené ze serveru.
Funkce *AktualizujData* testováním prvního znaku řetězce zjistí, že hodnota proměnné obsahuje XML hierarchie. Provede se volání funkce *PridejHierarchii* ze skriptu *FiltrovaniDotazu.js*.
Funkci se parametrem předá XML hierarchie. V těle funkce *PridejHierarchii* se provede přidání došlé hierarchie do klientské XML globální proměnné, v které se uchovávají data o všech použitých hierarchiích. Po přidání hierarchie se volá funkce *OtevriOknoHierarchie* s parametrem názvu

hierarchie. Tato funkce dynamicky vytvoří HTML dokument vizuálně zobrazující celou hierarchii, který potom otevře v novém okně prohlížeče.
Tvorba dokumentu probíhá na základě parsování XML hierarchie. Při kliknutí na tlačítko u jakéhokoliv členu hierarchie dojde k zavolání funkce *OsetreniKlikuTlacitka*.
Ve funkci se vyhledá patřičný člen hierarchie v XML a atribut viditelnosti podřízených členů se v elementu členu nastaví na odpovídající hodnotu. V HTML dokumentu se potom vyhledá prvek seznamu obsahující podřízenou větev členů a viditelnost tohoto prvku se přizpůsobí prováděné akci. Tedy podřízená větev doposud viditelná se zneviditelní a naopak.
Při změně zatržení *checkboxu* u jakéhokoliv členu dojde k zavolání funkce *OsetreniZatrzeniCheckBoxu,* ve které se volají funkce *ProvedZmenuNaVsechnyPotomky* a *ProvedZmenuNaVsechnyPredchudce*. V těchto funkcích se nejdříve pracuje s XML hierarchie. Vyhledá se element členu, u kterého nastala změna a hodnota jeho atributu zatržení se nastaví na odpovídající hodnotu. V XML se postupně hledají všechny přímo nadřazené elementy, u kterých je též potřeba změnit hodnoty atributů zatržení. Dále se vyhledá HTML prvek změněného *checkboxu* a nejdříve se všechny podřízené checkboxy nastaví na stejnou hodnotu jako nadřízený *checkbox*. Potom se adekvátně jako u XML hledají přímo nadřazené *checkboxy* a provádí se u nich změna zatržení (pokud je vyžadována).

4.4.2.4 Zobrazení filtru

Další možností uživatele při práci s prvkem je stisknutí tlačítka *ZobrazitFiltr*. V tom případě dojde k otevření okna se seznamem hierarchií zahrnutých do filtru dotazu. Tato akce se provede vykonáním volání funkce *OtevriOknoFiltru*.
V těle funkce se provede konstrukce HTML dokumentu, který z XML filtru vytvoří seznam použitých hierarchií. Dokument se poté otevře v novém okně. V okně budou kromě seznamu hierarchií také tlačítko pro vymazání vybraných hierarchií z filtru a tlačítko pro vymazání celého filtru. Výmaz hierarchie se uskuteční zavoláním funkce *OdebraniHierarchiiZFiltru*.
Funkce vyhledá záznam dotyčné hierarchie v XML filtru a provede přenastavení atributu, jehož hodnota určuje zařazení celé hierarchie do filtru. V dalším kroku se zavolá funkce *ZatrhniVsechnyClenyHierarchie,* která projde XML hierarchie a u všech jejich členů nastaví atribut zatrhnutí na true.

4.4.2.5 Odeslání stránky

Pokud se při práci s prvkem provede stisknutí tlačítka *OdeslatDotaz*, dojde k odeslání celé stránky na server. Údaje o filtru poskytne funkce *VratRetezecFiltru.* Práce této funkce je popsána níže při popisu rozevírání záhlaví tabulky. Pokud jsou osy i fakta správně nastaveny, aktualizovaná stránka bude obsahovat na rozdíl od prvního načtení do prohlížeče i tabulku s výsledky dotazu.

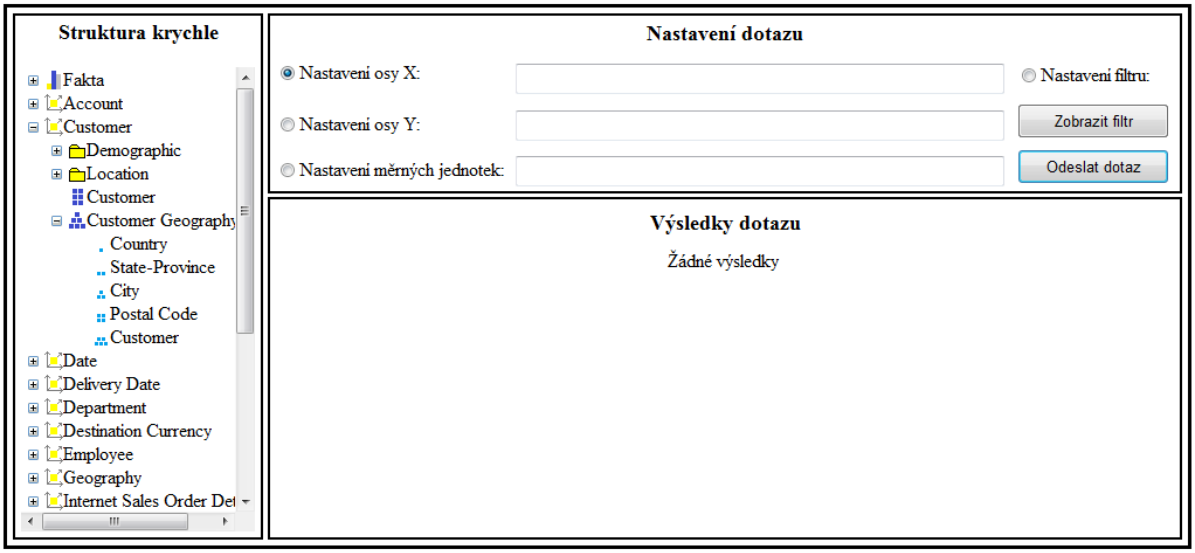
4.4.2.6 Práce s tabulkou výsledků

S touto tabulkou lze pracovat klikáním na buňky obou záhlaví. Po kliknutí na některou z těchto buněk se provede funkce *OsetreniKliku* ze skriptu *RizeniAnalyzy.js*. Ve funkci se nejdříve testuje existence globální XML proměnné, ve které se budou udržovat informace o probíhající analýze tabulky. V případě, že tato proměnná ještě nebyla inicializovaná, zavolá se funkce *VytvoreniXml*, která ji vytvoří.
Přechází se k testování, zda lze s buňkou vůbec pracovat. Pokud buňka nemá ve svém XML obrazu žádné potomky, neprovede se nic. Pokud nějací potomci existují, testuje se dále, jestli již buňka není otevřena. V tom případě se volá funkce *ZavriBunku* ze skriptu *ZaviraniZahlavi*.js. Funkci se parametrem předává XML element buňky. Ve funkci se XML elementu buňky nastaví atribut otevření na *false*, zjistí se, jestli je buňka umístěna na ose X nebo na ose Y a v závislosti na příslušnosti buňky k dané ose dojde k odstranění podřízených sloupců, respektive řádků tabulky. Nakonec se změní zbarvení buňky. V případě, že je buňka při kliknutí zavřená, testuje se dále, zda jsou údaje o potomcích součástí XML. Jestliže ano, volá se funkce *RozevriBunku* ze skriptu *RozeviraniZahlavi.js*. Provedou se analogické operace jako v případě funkce *ZaviraniZahlavi*. Přidávané řádky či sloupce

tabulky obsahují informace čerpané z XML obrazu tabulky. Jestliže se má provést otevření buňky a informace o potomcích dosud nejsou součástí klientského XML, musí se o ně požádat server. Nejdříve se zkonstruuje řetězec obsahující *mdx* výrazy pro obě osy a filtr. Tento řetězec vrací funkce *VratOdesilanyRetezec*. Při volání se této funkci parametricky předávají název buňky, název osy, z které buňka pochází a pole názvů všech buněk z druhé osy, které již mají údaje o svých potomcích přeneseny do XML. Pro získání tohoto pole je potřeba vyvolat funkci *VratJmenaPrenesenychZahlavi***.**
Ve funkci *VratOdesilanyRetezec* se pro konstrukci výrazu filtru volá funkce *VratRetezecFiltru* ze skriptu *FiltrovaniDotazu.js*. Ta nedělá nic jiného, než že prochází XML filtru a název každé hierarchie použité ve filtru přidává k výsledku volání funkce. Ke zkonstruovanému filtru se potom dodá výraz pro fakta a vytvoří se řetězec obsahující tzv. *tuple*, které později na serveru umožní zpracovat fakta i filtr společně ve filtrovací ose dotazu. Vytvořený řetězec vrácený funkcí *VratOdesilanyRetezec* se doplní na začátku o znak ‘g’ , představující příznak použitý později při zpracování *CallBacku***.** Provede se zavolání funkce *VolaniServeru* s parametrem právě zkonstruovaného řetězce. Tato funkce provede asynchronní volání serveru (*CallBack*). Další výpočet pokračuje již na serveru. V metodě *RaiseCallbackEvent* třídy webového prvku dojde nejprve k testování počátečního znaku došlého řetězce. Zjistí se, že došlá data představují informace pro konstrukci *mdx* dotazu do analytické databáze. Potřebná data se získají z instance třídy *OdpovedXml* přečtením vlastnosti *VystupXml***.**
Konstruktor třídy *OdpovedXml* nejdříve volá metodu *ProjdiVstup*, která projde vstupní řetězcový parametr obsahující data, z kterých se poskládá dotaz. Z řetězce vytvoří 3 proměnné obsahující výrazy pro obě osy a pro filtr. Metoda *VratCellSet* vrací datovou množinu za použití instance třídy *TvorbaCellSetu.* Tato množina se poskytuje jako vstup volání metody *VytvorXml*. Výsledek volání této metody obsahuje zkonstruované XML, které se uloží do interní proměnné *vystupXml*. Hodnota této proměnné je klientskému kódu třídy *OdpovedXml* zpřístupněna přes property *VystupXml***.**
V metodě *RaiseCallbackEvent* se řetězci obsahujícím právě vytvořené XML přiřadí na začátek příznak ‘g’ pro identifikaci u klienta. Tento řetězec je nakonec metodou *GetCallbackResult* odeslán zpět na klienta. Na klientovi se dostane ke slovu metoda *ZpracovaniCallBacku,* která volá metodu *AktualizujData* s parametrem řetězce obdrženého ze serveru. Z počátečního znaku zjistí, že zaslaná data představují XML obsahující výsledky databázového dotazu. Provede se volání funkce *ProjdiDosleXml* ze skriptu *ProchazeniServerovehoXml.js***.** Funkce spouští proces, který na základě vstupního XML ze serveru patřičně aktualizuje klientské XML uchovávající stav analýzy. Po provedení funkce *ProjdiDosleXml* následuje volání funkce *RozevriBunku*, která zabezpečí aktualizaci HTML tabulky prvku na stránce.

5. Uživatelská dokumentace

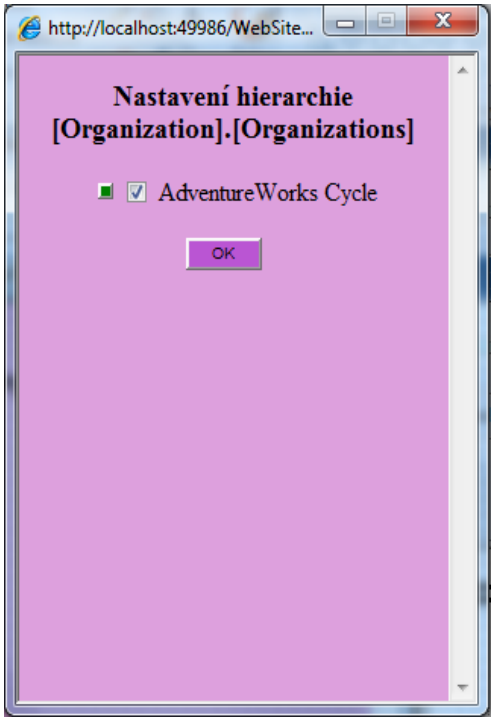
Po prvním načtení stránky se uživateli zobrazí základní rozvržení prvku. V levé části je sekce *Struktura krychle*, v které se nacházejí data popisující kompletní strukturu krychle. Nejvrchnější uzel stromu obsahuje množinu dostupných měrných jednotek krychle. Pod ním jsou potom seřazeny uzly jednotlivých dimenzí. Každá dimenze je složena ze svých hierarchií. Ty můžou být ve stromu seskupeny pod stejným uzlem adresáře. Hierarchie je dále složena z jedné nebo více úrovní. Atributová hierarchie obsahuje jedinou úroveň.
Vpravo nahoře je sekce *Nastavení dotazu,* sloužící pro nastavení parametrů dotazu odesílaných do databáze na serveru.
Vpravo dole je pak sekce *Výsledky dotazu* pro zobrazení výsledných dat vrácených ze serveru.



Obrázek č.25 - Pohled na základní rozvržení prvku

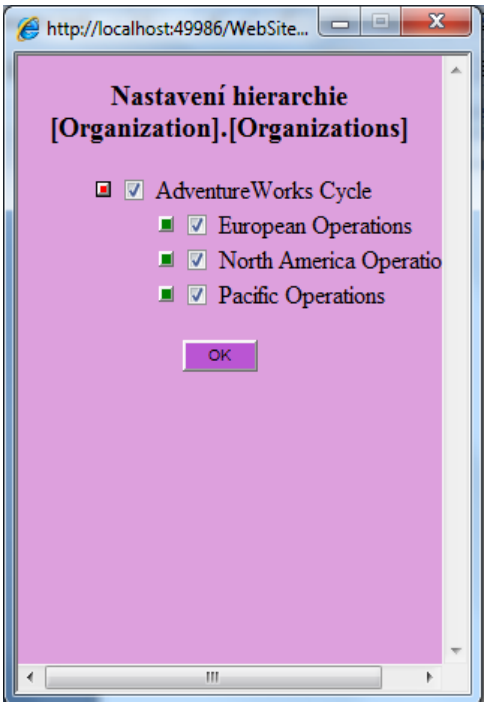
Nastavení parametrů dotazu se provádí aktivací *checkboxu* u některého z názvů os nebo u názvu *Nastavení měrných jednotek* a následným kliknutím na patřičný uzel stromu, představující požadovanou hodnotu. Hodnota uzlu se potom překopíruje do textového pole u aktivního *checkboxu*. Kopírování se neprovede v případě, kdy vybraná hodnota ze stromu nekoresponduje s typem požadované hodnoty ukládané do textového pole.
Do textových polí obou os lze uložit název hierarchie nebo název některé její úrovně.
Do textového pole měrné jednotky nelze samozřejmě uložit nic kromě názvu měrné jednotky.

Aktivace *checkboxu Nastavení filtru* umožní ve stromu vybrat jakoukoliv hierarchii kromě těch z nich, jež náleží do stejné dimenze jako hierarchie či úrovně hierarchií použité v obou osách dotazu. Výběrem správné hierarchie ze stromu dojde k otevření nového okna hierarchie. Zde je možno nastavit hierarchii pro začlenění do filtru dotazu.



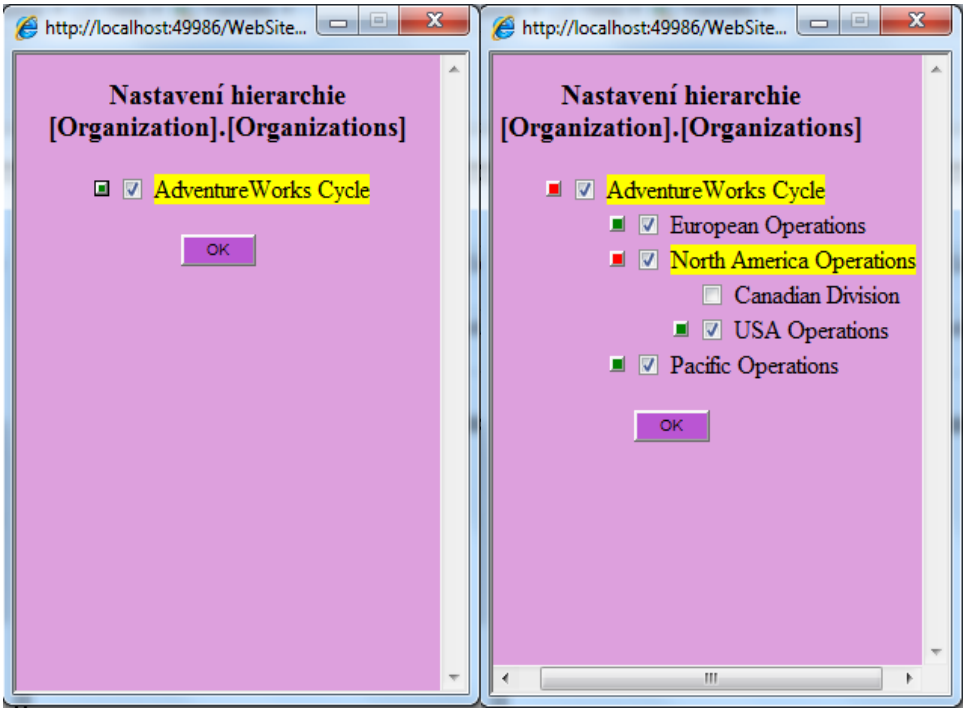
Obrázek č.26 - Okno pro nastavení filtrovací hierarchie

Zobrazování a skrývání jednotlivých úrovní členů hierarchie umožňuje stisk tlačítka u příslušného názvu členu. Pokud je úroveň skrytá, má tlačítko zelenou barvu. Po stisknutí se zobrazí všechny podřízené členy a tlačítko změní barvu na červenou.



Obrázek č.27 – Zobrazení úrovně členů

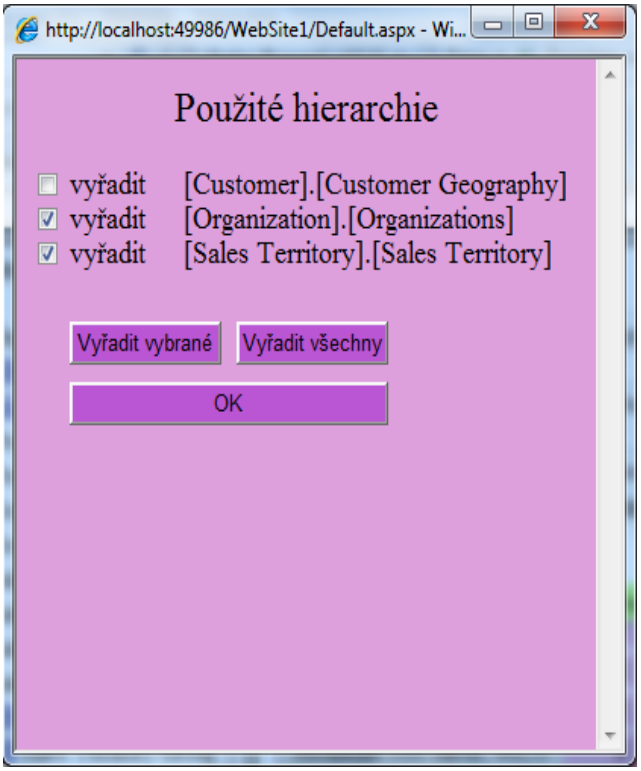
Zatržítko u každého členu hierarchie signalizuje zahrnutí tohoto členu do filtru. Pokud jsou zatrženy všechny členy hierarchie, z filtru se tato hierarchie vypouští. Pokud není zatržen ani jediný člen hierarchie, znamená to opět vypuštění hierarchie z filtru. Jestliže má určitý člen zatržené jen některé své potomky, je taková situace znázorněna zatržením členu a podbarvením jeho názvu. Tato změna se promítne i do všech přímo nadřazených členů. Jestliže je například nejvyšší člen zatržený a podbarvený, znamená to, že alespoň jeden z jeho potomků není zatržen. Nemusí to být ale nutně člen hned na následující úrovni pod ním.



Obrázek č.28 - Nezatržený potomek se nachází ve 2. úrovni pod výchozím členem

Pokud je hierarchie nastavena, tlačítkem OK se uloží do filtru a okno se zavře. Při příštím nastavení se hierarchie zobrazí ve stejném stavu, v jakém se nacházela při posledním nastavení.

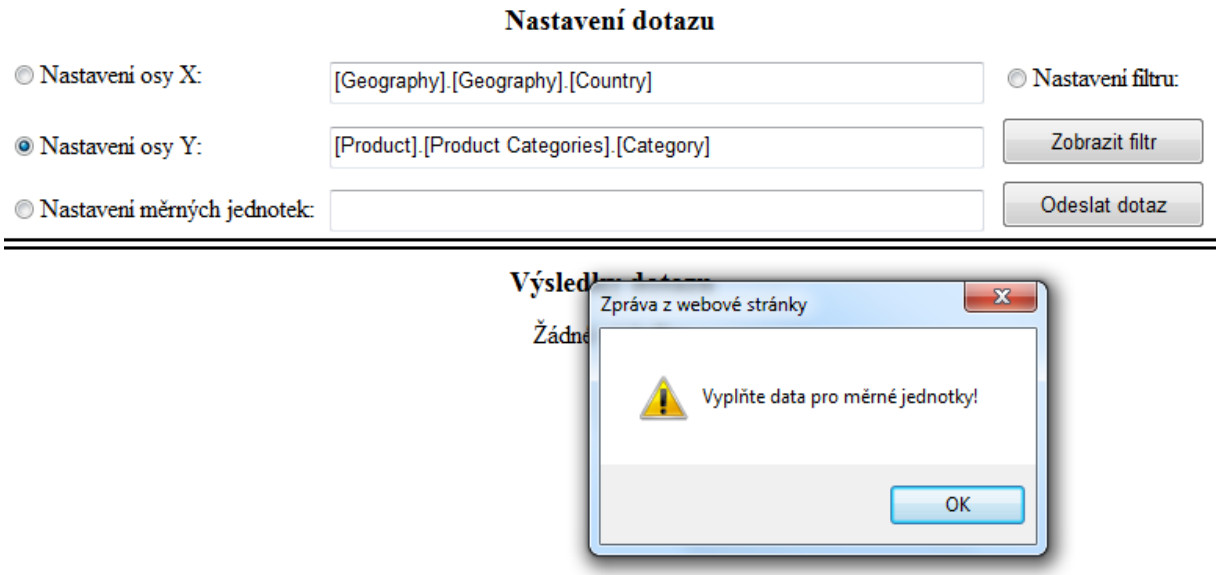
Po stisknutí tlačítka *Zobrazit Filtr* ze sekce *Nastavení* dotazu dojde k otevření nového okna se seznamem použitých hierarchií.



Obrázek č.29 - Okno se seznamem použitých hierarchií

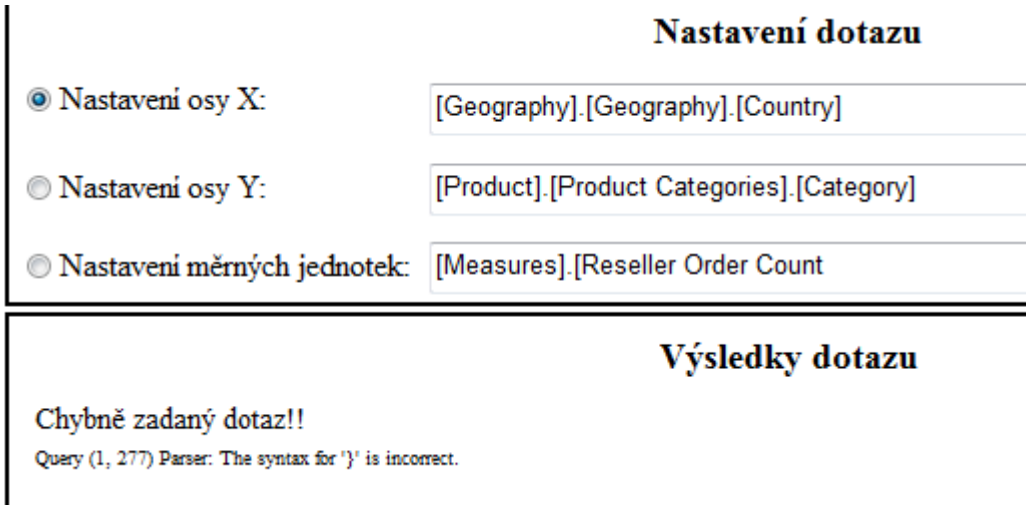
Kteroukoliv ze zobrazených hierarchií můžeme z filtru odstranit jejím zatržením a stisknutím tlačítka *Vyřadit vybrané*. Všechny hierarchie se odstraní stisknutím tlačítka *Vyřadit všechny*. Zobrazit podrobnosti o nastavení některé hierarchie ze seznamu lze provést kliknutím myší na název hierarchie. Potom dojde k otevření nového okna hierarchie. Okno seznamu hierarchií se ukončí po stisknutí tlačítka *OK*.

Tlačítkem *Odeslat dotaz* ze sekce *Nastavení dotazu* dochází k odeslání nastavených parametrů dotazu na server. Ještě předtím dochází k ověření, zda byly dodrženy základní pravidla při nastavování parametrů dotazu. Stránka se neodešle, pokud nejsou nastaveny obě osy i měrné jednotky dotazu. Za chybu je považováno také nastavení obou os dotazu na stejnou hodnotu.



Obrázek č.30 - Při chybném nastavení parametrů se zobrazí chybové hlášení

Pokud při zpracování dotazu na serveru dojde k nějaké chybě, místo vráceného výsledku se v sekci *Výsledky dotazu* zobrazí chybové hlášení.



Obrázek č.31 - Při zpracování dotazu na serveru došlo k chybě

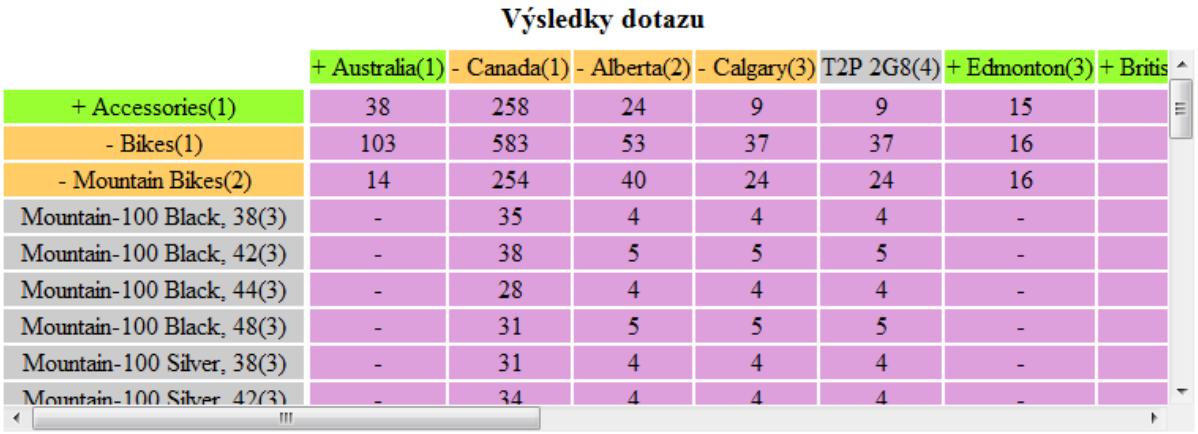
Při úspěšném zpracování dotazu se ze serveru vrací výsledná data ve formě tabulky. Ta jsou zobrazena v sekci *Výsledky dotazu.* Tabulka se skládá z horizontálního a vertikálního záhlaví, obě záhlaví jsou složena z množiny buněk s názvy jednotlivých členů hierarchie a s číselným označením úrovně, ve které se člen nachází. Tělo tabulky obsahuje hodnoty analyzované měrné jednotky.

Výsledky dotazu

	+ Australia(1)	+ Canada(1)	+ France(1)	+ Germany(1)	+ United Kingdom(1)	+ United States(1)
+ Accessories(1)	38	258	71	46	78	824
+ Bikes(1)	103	583	156	91	148	2072
+ Clothing(1)	62	478	129	72	126	1543
+ Components(1)	85	474	134	93	130	1730

Obrázek č.32 - Tabulka s výsledky dotazu

S tabulkou lze pracovat klikáním na buňky jejího záhlaví. Buňka, kterou lze otevřít, je označena znaménkem ‘+‘. Po kliknutí na ni se tabulka doplní o všechny viditelné podřízené členy. Naopak buňka, která je již otevřena, nese označení ‘-’. Po jejím zavření jsou z tabulky odstraněny všechny podřízené členy. Buňky, obsahující názvy členů poslední úrovně, kterou již nelze dále rozvíjet, jsou bez znaménka. Uvedené 3 typy buněk záhlaví jsou kromě existence znaménka rovněž odlišeny vlastním zabarvením. Buňka umožňující otevření či zavření členu mění kurzor myši, pokud se nachází nad ní.

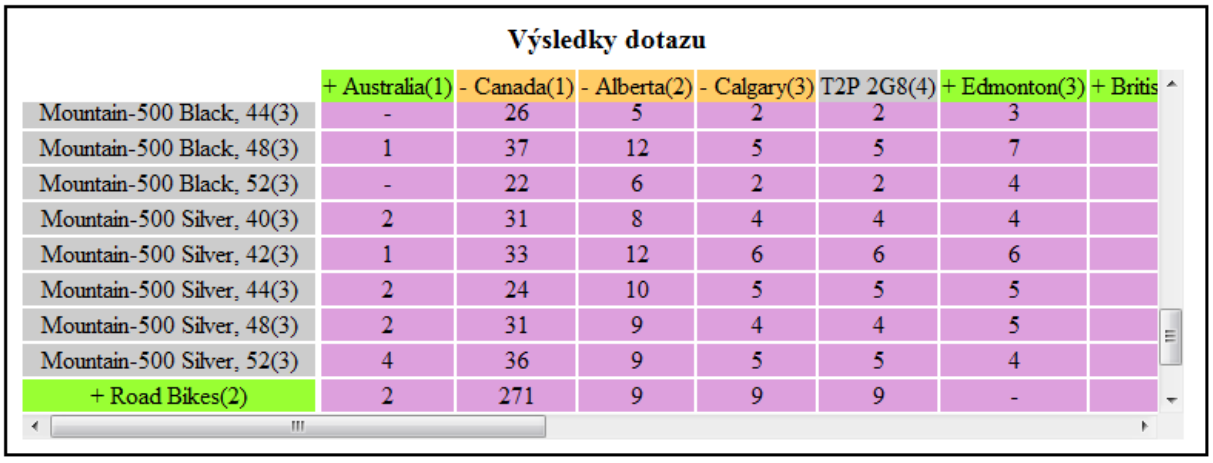


Výsledky dotazu

	+ Australia(1)	- Canada(1)	- Alberta(2)	- Calgary(3)	T2P 2G8(4)	+ Edmonton(3)	+ Britis
+ Accessories(1)	38	258	24	9	9	15	
- Bikes(1)	103	583	53	37	37	16	
- Mountain Bikes(2)	14	254	40	24	24	16	
Mountain-100 Black, 38(3)	-	35	4	4	4	-	
Mountain-100 Black, 42(3)	-	38	5	5	5	-	
Mountain-100 Black, 44(3)	-	28	4	4	4	-	
Mountain-100 Black, 48(3)	-	31	5	5	5	-	
Mountain-100 Silver, 38(3)	-	31	4	4	4	-	
Mountain-100 Silver, 42(3)	-	34	4	4	4	-	

Obrázek č.33 - Tabulka s otevřenými členy

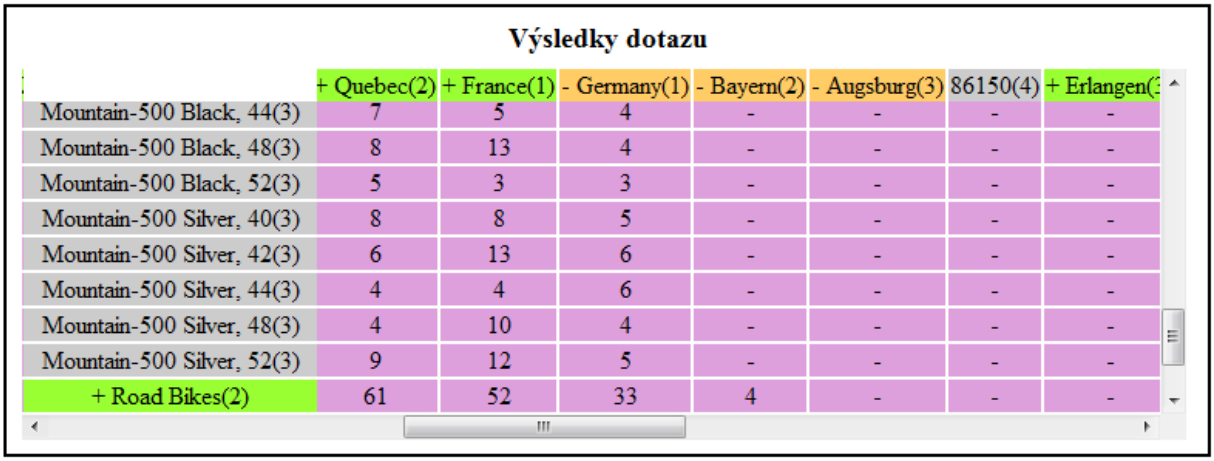
Pokud jsou data rozsáhlejší, jsou k tabulce připojeny posuvníky. Při scrollování tabulkou zůstává záhlaví fixované na místě a tím pádem bude vždy viditelné.



Výsledky dotazu

	+ Australia(1)	- Canada(1)	- Alberta(2)	- Calgary(3)	T2P 2G8(4)	+ Edmonton(3)	+ Britis
Mountain-500 Black, 44(3)	-	26	5	2	2	3	
Mountain-500 Black, 48(3)	1	37	12	5	5	7	
Mountain-500 Black, 52(3)	-	22	6	2	2	4	
Mountain-500 Silver, 40(3)	2	31	8	4	4	4	
Mountain-500 Silver, 42(3)	1	33	12	6	6	6	
Mountain-500 Silver, 44(3)	2	24	10	5	5	5	
Mountain-500 Silver, 48(3)	2	31	9	4	4	5	
Mountain-500 Silver, 52(3)	4	36	9	5	5	4	
+ Road Bikes(2)	2	271	9	9	9	-	

Obrázek č.34 - Při vertikálním posuvu zůstává záhlaví vždy viditelné



Výsledky dotazu

	+ Quebec(2)	+ France(1)	- Germany(1)	- Bayern(2)	- Augsburg(3)	86150(4)	+ Erlangen(3
Mountain-500 Black, 44(3)	7	5	4	-	-	-	-
Mountain-500 Black, 48(3)	8	13	4	-	-	-	-
Mountain-500 Black, 52(3)	5	3	3	-	-	-	-
Mountain-500 Silver, 40(3)	8	8	5	-	-	-	-
Mountain-500 Silver, 42(3)	6	13	6	-	-	-	-
Mountain-500 Silver, 44(3)	4	4	6	-	-	-	-
Mountain-500 Silver, 48(3)	4	10	4	-	-	-	-
Mountain-500 Silver, 52(3)	9	12	5	-	-	-	-
+ Road Bikes(2)	61	52	33	4	-	-	-

Obrázek č.35 - Při horizontálním posuvu zůstává záhlaví vždy viditelné

6. Závěr (Enclosing)

Vypracování bakalářské práce mi umožnilo prohloubit znalosti v oblasti technologií OLAP, ASP.NET 3.5, Javascript a XML.

Build up of Bachelor's work made me possible to deepen my knowledge with technologies OLAP, ASP.Net 3.5, Javascript and XML.

7. Seznam literatury

Lacko L. – Datové sklady, analýza OLAP a dolování dat, Computer Press, 2003.
MacDonald M., Szpuszta M. – ASP.Net 2.0 a C#. Zoner Press, 2006.
Andrew J.Brust, Sdtephen Forte - Mistrovství v programování SQL Serveru 2005.
SQL Server 2008 Product Documentation [MSDN Library]