VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

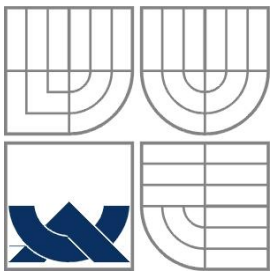
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

VYUŽITÍ OLAPU V ELEKTRONICKÉM OBCHODĚ

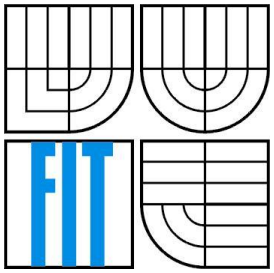
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE JAN LYSÝ
AUTHOR

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

VYUŽITÍ OLAPU V ELEKTRONICKÉM OBCHODĚ

APPLICATION OF OLAP IN AN E-SHOP

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JAN LYSÝ
AUTHOR

VEDOUCÍ PRÁCE — Ing. OTA JIRÁK
SUPERVISOR

BRNO 2010

Abstrakt

Cílem práce je navrhnout elektronický obchod, ve kterém bude využita OLAP databáze. Pro tuto databázi bylo navrženo multidimenzionální schéma. S využitím nástroje Business Intelligence, který je součástí SQL Server 2008 Enterprise, je podle schématu realizována databáze. Data z OLAP systému jsou dostupná formou grafů nebo tabulek přes administrátorské rozhraní internetového obchodu, implementovaném v ASP.NET.

Abstract

The aim of this thesis is to design an e-shop that uses OLAP database. Multidimensional scheme was assembled for the database. Using Business Intelligence tool that is part of SQL Server 2008 Enterprise the database was realized according to the scheme. Data from the OLAP system are accessible through administrator interface implemented with ASP.NET in the form of graph or table.

Klíčová slova

OLAP databáze, OLAP kostka, multidimenzionální databáze, multidimenzionální kostka, elektronický obchod, Business Intelligence, SQL Server 2008.

Keywords

OLAP database, OLAP cube, multidimensional database, multidimensional cube, e-shop, Business Intelligence, SQL Server 2008.

Citace

Lysý Jan: Využití OLAPu v elektronickém obchodě, bakalářská práce, Brno, FIT VUT v Brně, 2010

Využití OLAPu v elektronickém obchodě

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Oty Jiráka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Jan Lysý
17.5.2010

Poděkování

Chtěl bych poděkovat vedoucímu práce Ing. Otu Jirákovi, za ochotu při řešení problému a věnovaný čas na konzultacích.

Obsah

1 Úvod

OLAP systémy jsou důmyslnou technologií, které se při cíleném použití stávají pro společnost velmi užitečným prostředkem. Data, která jsou v nich uložena, mohou být po zpracování k dispozici formou reportů nebo různých grafů. Tento výstup z OLAP systému je možné použít k podpoře rozhodování nebo analýze dat. Uložená data mohou být až historická, mohou tedy požadavky na diskový prostor být značně vysoké.

Tato problematika spadá do odvětví dolování dat, které se používá nejen v informačních systémech, ale postupně se dostává i do lékařství nebo oblastí výzkumu a vývoje. Může mít vliv na řízení celých firem, protože umožňuje přístup ke kvalitně zpracovaným datům, která jsou pak předmětem dalšího plánovaní nebo návrhů.

Vlivem vzrůstu počtu elektronických obchodů se zvyšuje konkurence v jednotlivých oblastech prodeje. Proto je předložená práce zaměřena na rozvoj nové koncepce vedoucí ke zvýšení efektivity prodeje a tím i získání předstihu před konkurencí v oblasti prodeje. Jako možné řešení je zde předloženo plnohodnotné využívání OLAP technologie.

Celá práce je rozdělena do několika kapitol. Ve druhé kapitole jsou shrnuty nejdůležitější pojmy z teorie. V úvodu je stručně popsána OLTP a OLAP databáze. V závěru kapitoly je potom shrnuto jejich vzájemné porovnání. pro názornost je přiložen jednoduchý příklad.

Třetí kapitola je věnována návrhu celého systému. Návrh je rozdělen do tří částí, kde je popsána struktura databáze, grafický vzhled a služby, jimiž by měl obchod disponovat po kompletní implementaci, která je popsána v následující kapitole.

Popis implementace je zaměřen na funkci použitých prvků a snahu zmínit většinu variant, které by mohly být použity pro realizaci obchodu. Dále je zde popsán způsob tvorby OLAP kostky s nástroji od firmy Microsoft.

Závěrečná kapitola popisuje dojem z použitých nástrojů a vlastní zhodnocení celé práce.

2 Teoretické základy

2.1 OLTP databáze

Běžná databáze by měla umožňovat velké množství operací v reálném čase. Takové databáze jsou označovány jako OLTP (On-line Transaction Processing). Data jsou zde ukládána v dvojrozměrných databázových tabulkách. Každý řádek v tabulce má jednoznačný identifikátor. Mezi tabulkami existují relace, které popisují vzájemné vztahy.

Realizaci každé databáze, by měl předcházet návrh relačního schématu. U OLTP databází se při návrhu využívá normalizace, která by měla usnadnit následující práci s databází. Nejčastěji se využívají databáze ve třetí normální formě.

2.2 Datové sklady a OLAP

Pod pojmem datový sklad je možné si představit rozsáhlou databázi oddělenou od operační databáze, ovšem pro organizaci dat zde platí jiná pravidla. Tabulky nemusejí být normalizované. Hlavním cílem datových sladů je podpora rozhodování.

Asi nejznámější definice datového skladu od Billa Inmona[1]:

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

Subjektová orientace:

Data se do datového skladu zapisují spíš podle předmětu zájmu, než podle aplikace, ve které byla vytvořena. Při orientaci na subjekt jsou data v datovém skladu kategorizovaná podle subjektu, kterým může být například zákazník, dodavatel, zaměstnanec, výrobek a podobně. Orientace na aplikaci naproti tomu znamená, že data jsou v systému uložena podle jednotlivých aplikací, například data aplikace pro odbyt, data aplikace pro fakturaci, data aplikace pro personalistiku.

Integrovanost:

Datový sklad musí být jednotný a integrovaný. To znamená, že data týkající se konkrétního předmětu se do datového skladu ukládají jen jednou. Proto musíme zavést jednotnou terminologii, jednotné a konzistentní jednotky veličin. Není to snadný úkol, protože data přicházejí do datového skladu z nekonzistentního a neintegrovaného operačního prostředí. Proto musí být data v etapě přípravy a zavedení upravená, vyčištěná a sjednocená. Pokud data nejsou konzistentní a důvěryhodná, datový sklad ztrácí význam.

Časová variabilita:

Data se ukládají do datového skladu jako série snímků, z nichž každý reprezentuje určitý časový úsek. Na rozdíl od operačního prostředí, kde jsou data platná v okamžiku přístupu, v datových skladech jsou data platná pro určitý časový moment, časový snímek. Zatímco v operačním databázovém prostředí jsou uložena data za kratší časové období, většinou za několik dnů, maximálně měsíců, v datovém skladu jsou data za delší časové období, typicky za několik roků. Klíčové atributy v datovém skladu obsahují čas, který v operačních databázích nemusí být uváděný. Jakmile je v datovém skladu zaznamenaný konkrétní snímek dat z operativní databáze, nemohou už být data v datovém skladu modifikovaná.

Neměnnost:

V operačních transakčních databázích jsou data do databáze jednak vkládána, jednak modifikována a mazána. Data v datovém skladu se obvykle nemění ani neodstraňují, jen jsou v pravidelných intervalech přidávána nová data. Proto je manipulace s daty v datových skladech daleko jednodušší. V zásadě můžeme připustit jen dva typy operací. Zavedení dat do datového skladu a přístup k těmto datům. Žádné změny dat nejsou přípustné. Z toho vyplývá, že většina metod pro optimalizaci a normalizaci dat a transakční přístup k datům je v datovém skladu nepotřebná.

OLAP systémy

OLAP je zkratka pro Online Analytical Processing, která na první pohled nic konkrétnějšího neřekne. Systémy OLAP pomáhají analyzovat velké množství dat, ze kterých je možno vytvářet tabulky, reporty, grafy a další možné souhrnné zprávy. Tyto výstupy pomohou poměrně rychle analyzovat velké množství dat uložených v databázi a napomohou při rozhodování.

E.F.Codd, tvůrce relačního databázového modelu popsal výraz OLAP pomocí 12 pravidel, která jsou převzata z [1].

1. **Multidimenzionální konceptuální model:** OLAP by měl poskytovat uživateli multidimenzionální model odpovídající jeho podnikatelským potřebám tak, aby tento model mohl využívat pro analýzu shromážděných dat.
2. **Transparentnost:** Technologie systému OLAP, podřízená databáze a architektura výpočtu by měli být pro uživatele transparentní, aby uživatel mohl naplno využívat svou produktivitu a odbornost při použití front-end nástrojů a prostředí. Pro platformu Microsoft je možné jako klientský nástroj použít například program Excel z balíku Office, který bude napojený na OLAP Server. Důležitá je samozřejmě i heterogennost vstupních dat, kterou zajišťuje proces ETL. Pro tento proces jsou na platformě SQL Server 2008 k dispozici integrační služby.
3. **Dostupnost:** Systém OLAP by měl přistupovat k těm datům, které jsou potřebné pro analýzu. Systém by měl být navíc schopný přistupovat ke všem datům potřebným pro analýzu,

nezávisle na tom, ze kterého heterogenního zdroje tato data pocházejí, jak často jsou obnovovaná a podobně.

4. **Konzistentní vykazování:** I když počet záznamů, a tedy i velikost databáze, postupem času roste, uživatel by neměl pocítit žádné podstatné snížení výkonu.
5. **Architektura klient-server:** Systém OLAP musí odpovídat principům architektury klient-server s přihlédnutím na maximální cenu a výkon, flexibilitu a interoperabilitu.
6. **Generická dimenzionalita:** Každá dimenze dat musí být ekvivalentní ve struktuře i operačních schopnostech.
7. **Dynamické ošetření řídkých matic:** Systém OLAP by měl být schopný adaptovat svoje fyzické schéma na konkrétní analytický model, který optimalizuje ošetření řídkých matic, přičemž dosáhne a udrží požadovanou úroveň výkonu.
8. **Podpora pro více uživatelů:** Systém OLAP musí být schopný podporovat pracovní skupinu uživatelů pracujících současně na konkrétním modelu.
9. Neomezené křížové dimenziální operace: Systém OLAP musí dokázat rozeznat dimenzionální hierarchie a automaticky provést asociované kumulované kalkulace v rámci dimenzí, i mezi nimi.
10. **Intuitivní manipulace s daty:** Pravidlo definuje konsolidované přeorientování cest na detailní úroveň a zpět. Uživatelské rozhraní by mělo umožňovat všechny manipulace způsobem „ukázat a klepnout, případně zachytit a přemístit“ v buňkách kostky.
11. **Flexibilní vykazování:** Musí existovat schopnost uspořádat řádky, sloupce a buňky způsobem, který umožní analýzu intuitivní vizuální prezentací analytických sestav.
12. **Neomezené dimenze a úrovně agregace:** V závislosti na požadavcích podnikání může mít analytický model více dimenzí, přičemž každý z nich může mít vícenásobné hierarchie. Systém OLAP by neměl zavádět (například z technických důvodů) žádné umělé omezení počtu dimenzí nebo úrovní agregace.

2.3 Multidimenzionální datový model

Tak jako se pro zobrazování dat v relačních databázích používají tabulky. U multidimenzionálních databází jsou to multidimenzionální kostky. Tyto kostky nejsou úplně běžné, jak je známe z reálného života, mohou být totiž n-rozměrné. Každá kostka je definována dimenzemi a fakty. Výpočet kostky je velmi časově náročný kvůli zpracování a průchodu velkého množství dat.

Fyzicky jsou data uložena v tabulkách relační databáze, multidimenzionálního schématu se dosáhne pomocí relačních vazeb mezi tabulkami faktů a tabulkami dimenzí.

2.3.1 Fakta

Fakta jsou data, která se nacházejí uvnitř kostky. Většinou jsou to numerické měrné jednotky. Představují množství, na jehož základě se analyzují vztahy mezi dimenzemi. Například to mohou být počty položek v nákupu, nebo cena nákupu.
Všechna fakta bývají uložena v jedné tabulce. Tato tabulka mívá největší počet záznamů. Taky se do ní nejčastěji připisují nové položky. Zároveň obsahuje cizí klíče do jednotlivých dimenzí.

2.3.2 Dimenze

Pod dimenzí si můžeme představit něco jako hranu datové kostky. Údaje jsou zde hierarchicky nebo logicky uspořádané, což umožňuje vytváření agregačních výpočtů. Nejčastěji používané dimenze jsou časové, produktové, geografické.
Tabulky dimenzí jsou mnohem menší než tabulky faktů a taky se v nich data nemění tak často. Většina tabulek dimenzí obsahuje relativně stabilní data, snad pouze dimenze zákazníků se bude měnit častěji.

2.3.3 Příklad kostky

Nejčastěji zmiňovaným příkladem je datová kostka prodeje s dimenzemi *produkt, čas* a *region.* Příklad této trojdimenzionální kostky je vhodný z důvodu jeho jednoduchosti a názornosti.
Dimenze produkt představuje prodávané zboží, díky ní je možno rychle vyhodnotit prodávanou jednotlivých produktů. Další dimenzi je čas, díky můžeme sledovat prodej zboží v jednotlivých časových úsecích. Poslední dimenzí je region, tato dimenze umožňuje sledovat a porovnávat mezi sebou různé geografické lokality, například kraje nebo státy, kde by firma měla jednotlivé pobočky.
Dimenze mají hierarchickou strukturu, která by u jednotlivých dimenzí mohla být například následující:

Region

● Stát
●● Kraj
●●● Město

Produkt

● Druh produktu
●● Kategorie
●●● Podkategorie
●●●● Název produktu

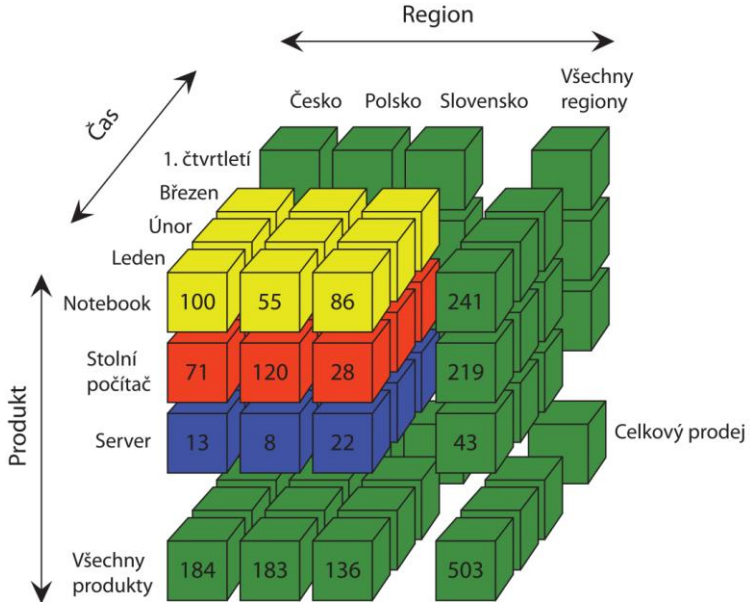
Čas

● Rok

●● Čtvrtletí

●●● Měsíc

●●●● Týden



Obrázek 2.1 Příklad multidimenzionální kostky

Na obrázku 2.1 je názorně vidět princip multidimenzionální databáze. Hodnoty na obrázku jsou smyšlené, mají význam pouze ilustrativní. Mohly by představovat například počet prodaných kusů v tisících.

2.3.4 Schémata multidimenzionálních databází

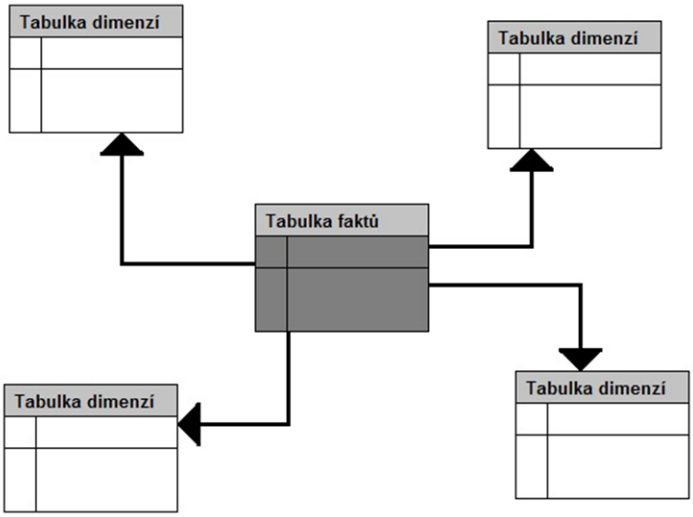
Když chceme realizovat multidimenzionální kostku, je potřeba mít určitý podklad, ze kterého vycházíme a podle nějž se řídíme. Tímto řídícím prvkem je schéma. Rozlišujeme tři druhy schémat:

- hvězda,
- sněhová vločka,
- souhvězdí.

Stručně zde rozeberu všechny tři.

Schéma hvězdy

Schéma hvězdy se skládá z jedné tabulky faktů a několika tabulek dimenzí. Tabulka faktů obsahuje měrnou jednotku a cizí klíče do tabulek dimenzí. Každé dimenzi přísluší pouze jedna tabulka v nenormalizovaném stavu. Je zde tedy vysoká redundance dat. Příklad schématu je na obrázku 2.2. Důsledkem redundance dat, je pomalá tvorba modelu. Naopak jeho výhodou je vysoký dotazovací výkon.

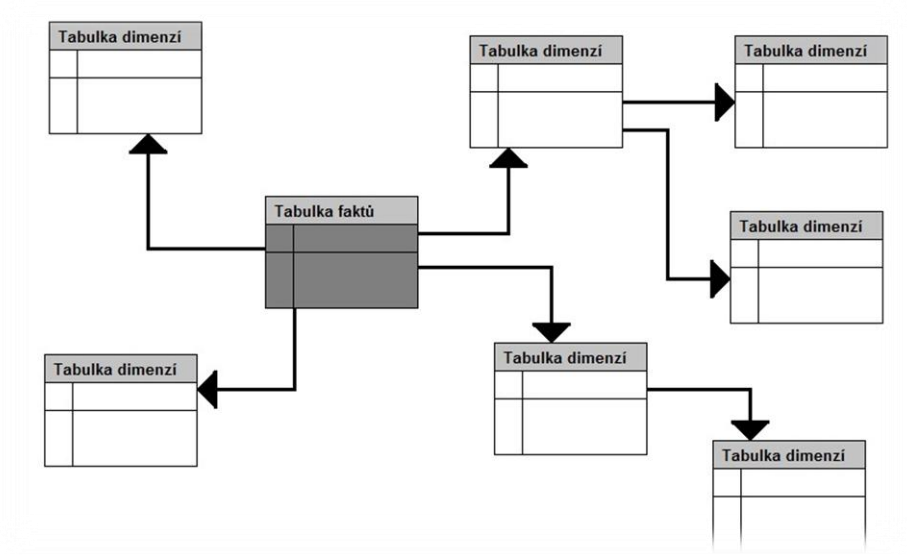


Obrázek 2.2 Schéma hvězdy

Schéma sněhové vločky

Sněhová vločka má také pouze jednu tabulku faktů, od hvězdy se liší v tom, že dimenze může být tvořena několika relačně svázanými tabulkami. Dimenze zde tedy mohou být normalizovány. Dochází zde tedy ke snížení redundance. Vztahy mezi dimenzemi jsou zřetelně viditelné na obrázku 2.3.

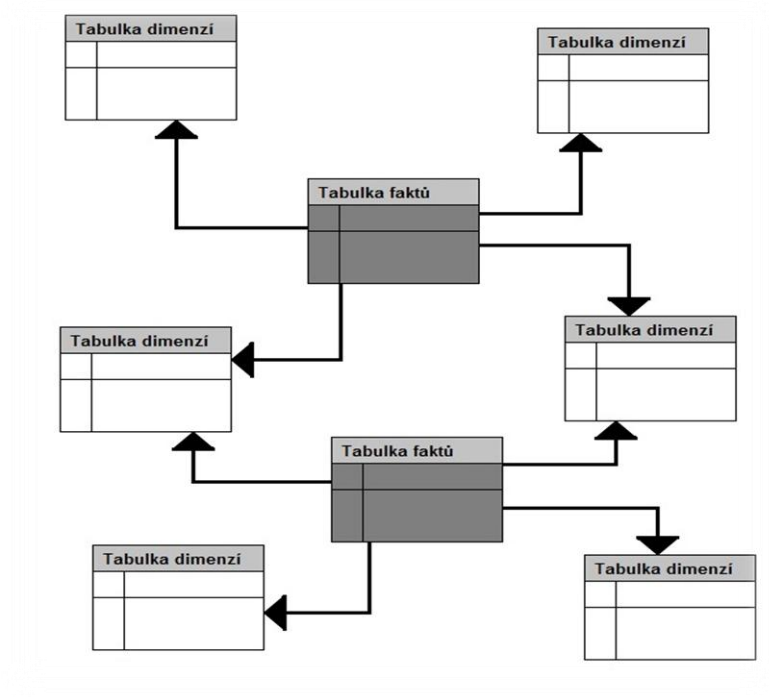
U tohoto modelu je výhodou rychlejší zavedení dat do tabulek. Oproti hvězdicovému schématu je zde nižší dotazovací výkon, kvůli spojeným tabulkám v dimenzích.



Obrázek 2.3 Schéma sněhové vločky

Schéma souhvězdí

Rozsáhlé databázové systémy si vyžádali více tabulek faktů, které mají společné některé tabulky dimenzí. To je vnímáno jako spojení několika hvězd, proto název souhvězdí. Příklad schématu si pravděpodobně každý dokáže představit, ale pro názornost je ukázka na obrázku 2.4.



Obrázek 2.4 Schéma souhvězdí

2.3.5 OLAP servery

Pokud chceme využít OLAP technologie máme k dispozici tři základní serverové architektury. Všechny tyto architektury nám umožňují pracovat s multidimenzionálními daty, aniž bychom věděli, jak jsou data uložena.

Multidimenzionální OLAP (MOLAP)

Multidimenzionální OLAP získává data z operačních zdrojů, nebo z datového skladu. Získaná data jsou poté uložena ve vlastní databázové struktuře. Data se ukládají v multidimenzionální databázi, jako předem vypočítané pole. Struktura databáze umožňuje rychlé získání dat i z více dimenzí. Část dat se může uložit na straně klienta, což umožní rychlou analýzu bez velkého zatížení sítě. Hlavní výhodou této architektury, je vysoký výkon vzhledem k dotazům uživatele. Velkou nevýhodou může být naopak redundance dat, protože data jsou uložena jak v relační databázi, tak v databázi multidimenzionální. Při vyšším počtu dimenzí tedy značně narůstají požadavky na diskový prostor.

Relační OLAP (ROLAP)

Relační zpracování dat získává data pro analýzu z relačního datového skladu. Zpracovaná data se předkládají uživateli jako multidimenzionální pohled. Data a metadata jsou uloženy jako záznamy v relační databázi. OLAP server používá tato metadata na generování SQL příkazů, které jsou potřeba

k získání dat, které požaduje uživatel. Je zde odstraněn problém s redundancí, protože data jsou uložena pouze v relační databázi. Oproti architektuře MOLAP je zde výrazně nižší výkon.

Hybridní OLAP (HOLAP)

Hybridní OLAP jak název napovídá je kombinaci úložišť MOLAP a ROLAP. Jsou zde využity výhody obou úložišť a snaha minimalizovat jejich nevýhody. Množství detailních dat se ukládá do relační databáze a do multidimenzionální databáze se ukládají sumární data.

2.3.6 Měrné jednotky

Hodnoty uvnitř datové kostky jsou definovány dimenzemi. Měrná jednotka může být vyhodnocena v libovolném bodě kostky. Výpočet měrné jednotky závisí na hodnotách dimenzí dané kostky. Druhy měrných jednotek a princip pochází z [6].

Distributivní

Výsledné hodnoty se získávají distribuovaným výpočtem. V případě rozdělení kostky na několik částí lze celkovou hodnotu funkce získat výpočtem z jednotlivých „mezivýsledků“, aniž by vznikla chyba. Můžeme sem zařadit například součet.

Algebraické

Algebraické jednotky mohou být vypočítány pomocí několika distributivních funkcí. Typickou algebraickou funkcí je algebraický průměr, který může být spočítán jako součet/počet.

Holistické

Jednotky, pro které neexistují žádné algebraické funkce. Příkladem může být funkce median ().

2.3.7 Operace v OLAP systémech

Datové sklady jsou budovány především kvůli podpoře při rozhodovaní a analýze dat. Samozřejmě to lze provádět i bez OLAP systému, ale bylo by to mnohem náročnější. Manipulaci s OLAP kostkami umožňují následující operaci. Jejich princip pochází z [6].

Roll-lup

Posun v hierarchii dimenze o jednu úroveň výše. Po této operaci je pohled na data „hrubší“ v dimenzi, ve které byla provedena změna.

Drill-down

Operace je opakem *roll-up*. Posuneme se tedy v hierarchii vybrané dimenze o úroveň níže. Pohled na data bude tedy „jemnější“.

Slice

Tato operace funguje na principu selekce nad jednou dimenzí. Pomocí *slice* můžeme tedy u OLAP kostky vybrat pouze data splňující nějakou podmínku. Například jsou vybrána data pouze za jeden týden.

Dice

Dice je rozšířením operace *slice*. Představuje selekci nad několika dimenzemi. Například jsou vybrána data za jeden týden pouze z města Brno.

Pivoting

Změna pohledu na multidimenzionální data. Počet pohledu je roven permutaci počtu dimenzí.

2.4 OLTP vs. OLAP

OLTP

Tyto systémy jsou určeny pro každodenní užívání mnoha uživateli. Mohou data upravovat, vkládat, mazat a prohlížet s rychlou odezvou. Přístup k datům je založen na krátkých atomických transakcích. K dispozici jsou zde pouze aktuální data z jednoho zdroje, který je uchovává ve formátu nevhodné k analýze.

OLAP

OLAP systémy jsou schopny zpracovávat data z několika databází. Data, která jsou v nich uložena, neslouží ke zpracování jednoduchých transakcí, ale zpracovávají se jako celek. Výsledkem toho zpracování jsou analýzy sloužící jako podpora při rozhodování analytiků a manažerů.
V těchto systémech se mohou nacházet záznamy i několik let zpět.

Shrnutí

Oba systémy jsou určeny ke zcela odlišným účelům. Výhodou OLTP je jeho „trvanlivost“ pokud je správně navržen. V případě, že spolupracuje s kvalitním uživatelským rozhraním, může ho využívat téměř kdokoliv. Jednou z jeho silných zbrání je spolupráce s datovými sklady.
OLAP systémy se jeví jako silná zbraň pro komplexní analýzy, modelování a prognózy. Jsou schopny přistoupit k velkému množství dat v relativně krátkém časovém úseku. OLAP má malou nevýhodu v jeho požadavcích na velikost diskového prostoru, protože může uchovávat opravdu mnoho záznamů.
V následující tabulce 2.1 jsou porovnány oba systémy ohledně různých vlastností.

Vlastnost	OLTP	OLAP
Charakteristika	Provozní zpracování	Informační zpracování
Orientace	Transakční	Analytická
Uživatel	Úředník, databázový administrátor	Znalostní pracovník (manažer, analytik)
Funkce	Každodenní operace	Dlouhodobé informační požadavky, podpora rozhodování
Návrh databáze	E-R základ, aplikačně orientovaný	Multidimenzionální schéma, věcná orientace
Data	Současná, zaručeně aktuální	Historická
Sumarizace dat	Základní, vysoce detailní	Shrnutá, kompaktní
Náhled	Detailní	Shrnutý, multidimenzionální
Jednotky práce	Krátké, jednoduché transakce	Komplexní dotazy
Přístup	Číst a zapisovat	Většinou pouze číst
Zaměření	Vkládání dat	Získávání informací
Počet dostupných záznamů	Desítky	Miliony
Počet uživatelů	Tisíce	Stovky
Velikost databáze	100 MB až GB	100 GB až TB
Přednosti	Vysoký výkon, vysoká přístupnost	Vysoká flexibilita, nezávislost koncového uživatele
Míry hodnocení	Propustnost transakcí	Propustnost dotazů a doba odezvy

Tabulka 2.1 Porovnání OLAP a OLTP [4]

3 Návrh systému

Vzhledem k tomu, že by měla OLAP databáze být využita v elektronickém obchodě, je třeba nejprve realizovat elektronický obchod, který je nezbytnou součástí této práce.

Zjištění požadavků

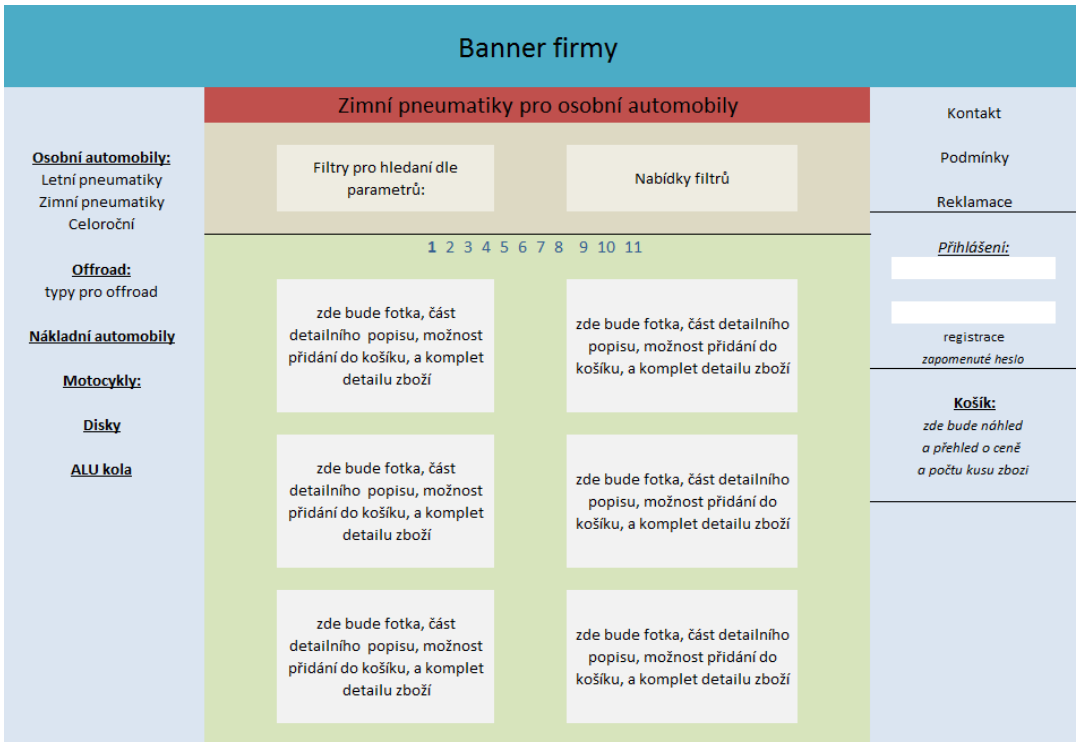
Po konzultaci se zástupcem firmy, pro kterou je elektronický obchod určen, byly sepsány základní požadavky, kterými by elektronický obchod měl disponovat.

Vytvoření návrhu

Výsledný návrh systému se skládá ze tří částí. Jednou částí je grafický vzhled, druhou částí je databázový systém a jako poslední prvek celého návrhu jsou poskytované služby.

3.1 Grafický vzhled

Pomocí programu Microsoft Excel byla vytvořena kostra vzhledu, jak by si autor představoval a rozvrhl rozhraní stránek. Tento vzhled byl dodán zástupci firmy, se kterým byl autor práce v kontaktu. Po určité době zástupce firmy poskytnul grafický vzhled stránek, který byl pak použit. Oba náhledy jsou přiloženy jako obrázky, předběžný návrh z obrázku 3.2 je možné porovnat s návrhem skutečným na obrázku 3.3.



Obrázek 3.1 Předběžný návrh vzhledu

Skutečný návrh se od předběžného návrhu hodně liší. To ovšem vůbec nevadí. Cílem mého předběžného návrhu bylo, aby vyjádřil představu o rozložení stránek. I přesto, že skutečný grafický návrh v nepatrných maličkostech rozchází s mou myšlenkou, základní kostra byla zachována.

Obrázek 3.2 Skutečný návrh

3.2 Návrh databáze

V žádném elektronickém obchodě nesmí chybět její základní stavební prvek, kterým je databáze. V našem případě nebude využita běžná relační databáze, ale pokusíme se využít databázi OLAP.

V teoretické části jsem zmínil tři druhy OLAP serveru. Já jsem si k realizaci zvolil Relační OLAP server. Pokud jsem pochopil jeho návrh správně. Je třeba vytvořit multidimenzionální schéma, které se převede do relační databáze.

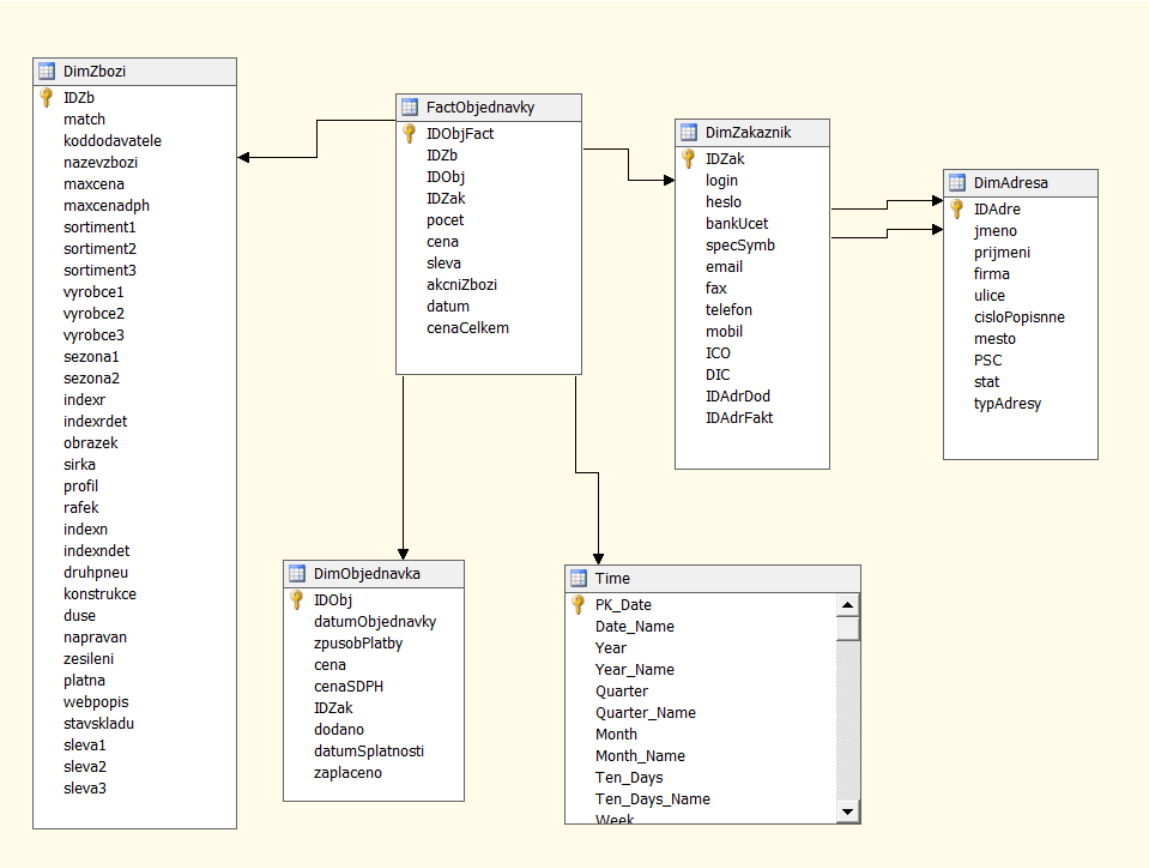
Schéma se skládá z pěti tabulek dimenzí a jedné tabulky faktů. Základními čtyřmi dimenzemi jsou čas, zboží, objednávka a zákazník. Struktura celého schématu je vidě na obrázku 3.3.

Vzhledem k tomu, že jsem s OLAP systémem nikdy dřív do styku nepřišel, zvolil jsem za vhodné východisko tento relativně jednoduchý návrh schématu. Za dimenzi *zákazník* je přidána dimenze *adresa*, která je vázána dvěma vztahy, jeden představuje vztah k adrese fakturační a druhý k adrese dodací. Ne vždy se musí tyto adresy shodovat, proto jsem zvolil tuto možnost. I přesto, že by se adresy shodovaly, musí být vytvořeny dva záznamy. Aby si zákazník mohl dodatečně obě adresy změnit.

Dimenze *zboží* byla vytvořena na základě dat, která mi byla poskytnuta. Tato data se nacházela ve formě XML souboru, který obsahoval položky zboží, které by elektronický obchod měl nabízet.

Další dimenzí je *objednávka*, ve které jsou uloženy informace potřebné k fakturaci.

Dimenze času, byla vygenerována pomocí nástroje *Analysis Services*. Je tedy rozsáhlejší než, kdyby ji definoval autor sám. Obsahuje hierarchii dnů, týdnů, měsíců, čtvrtletí a roků. Většina časových údajů je v tabulce zastoupena slovním i číselným vyjádřením, je tabulka tak rozsáhlá.



Obrázek 3.3 Multidimenzionální schéma

Tabulka faktů bude obsahovat každou prodanou položku zboží, která je obsažena v objednávce. Musí být také provázána se všemi ostatními dimenzemi. Obsahuje tedy čtyři cizí klíče. Jedním z cizích klíčů je datum, kdy byl výrobek objednán, protože dimenze *Time* má jako primární klíč celé datum. Mimo vlastní primární klíč jsou v tabulce uloženy ještě hodnoty počet objednaných kusů, cena za kus a celková cena konkrétního zboží.

3.3 Poskytované služby

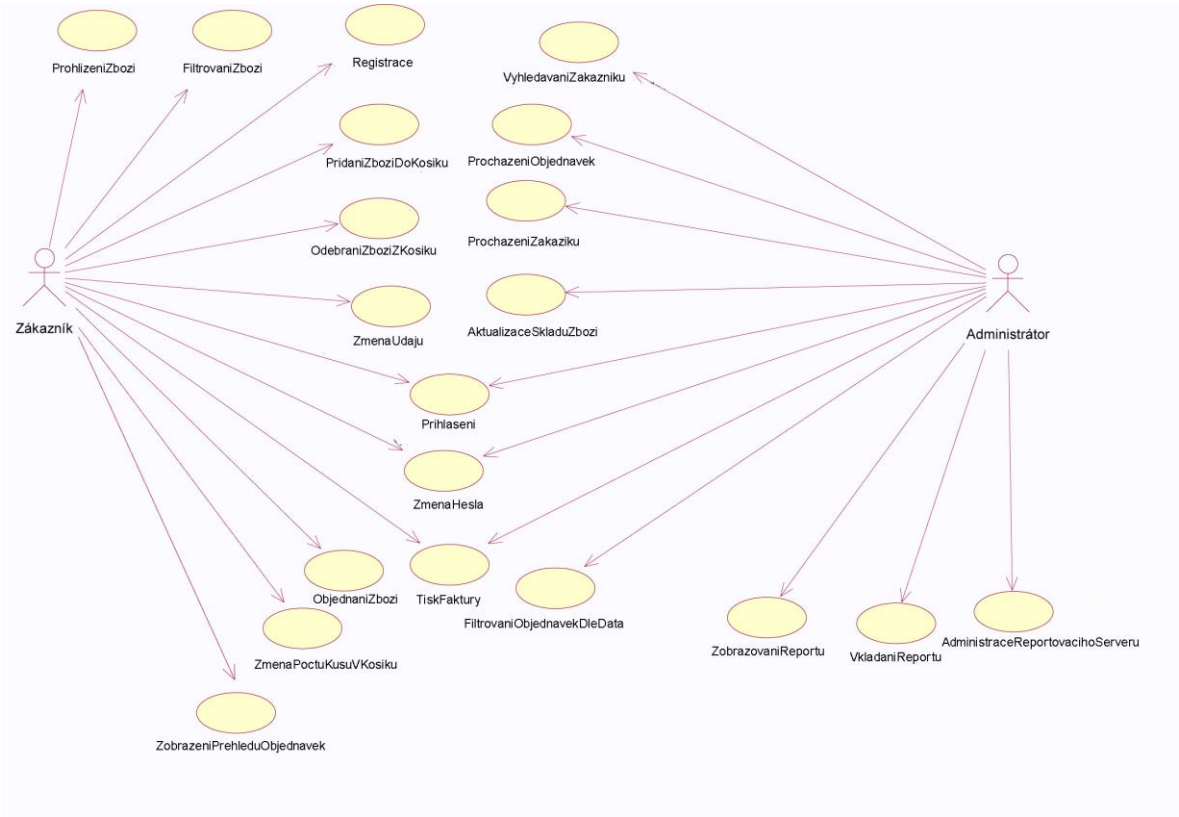
Mezi poskytované služby jsou zařazeny akce, které elektronický obchod nabízí pro zákazníka a pro administrátora. K poskytovaným službám byl vytvořen Use Case diagram, který je možné si prohlédnout na obrázku 3.4.

Služby pro zákazníka

Zákazník může procházet nabídku zboží, kterou je možné filtrovat dle určitých parametrů. Dále je zobrazené zboží je možno přidávat do košíku. Počet přidávaných kusů lze definovat v textovém poli

vedle tlačítka, přidávající zboží. V případě, že by objednávka nebyla provedena před zavřením internetového prohlížeče, měl by zůstat obsah košíku uložen.

Aby, si zákazník mohl objednat zboží, je nutná jednoduchá registrace. Pro registraci je nutné vyplnit základní kontaktní údaje a adresu, na kterou bude zboží dodáno. Ve chvíli kdy se registrovaný zákazník přihlásí, je mu umožněno dokončení objednávky. V případě, že by chtěl zákazník změnit dodací nebo fakturační adresu, měla by mu být tato služba poskytnuta po přihlášení.



Obrázek 3.4 Use Case diagram

Služby pro administrátora

Administrátorovi by mělo být umožněno nahlédnout na všechny objednávky. Všechny objednávky jsou filtrovány dle data nebo zákazníka, který objednával. U zobrazených objednávek jsou zobrazeny informace o zákazníkovi, který zboží objednal. Podobný princip funguje i u zákazníků.

Dále bude moct administrátor, nebo manažer vytvářet analýzy prodeje na základy výstupu z OLAP databáze. Pod výstupem OLAP databáze je možné si představit různé typy grafů nebo sestav.

Zboží do databáze nebude vkládáno pomocí formulářů, ale bude se načítat z XML souborů.

4 Implementace

Základ elektronického obchodu bylo možné implementovat pomocí PHP (Hypertext Preprocessor) nebo ASP.NET. K oběma jazykům je potřeba mimo jiné znalost HTML a CSS. Po zvážení možností bylo rozhodnuto, že by bylo vhodné implementovat elektronický obchod v ASP.NET.

4.1 Databáze

Všechna data, která elektronický obchod zobrazuje a ukládá, jsou zpracována a uložena pomocí Microsoft SQL Serveru 2008. Tento software je nabízen v několika variantách. V našem případě musí verze SQL Serveru podporovat nástroje pro práci s OLAP systémy. Bylo potřeba tedy zvolit verzi SQL Server 2008 Enterprise.

Verze Enterprise oproti ostatním verzím obsahuje nástroje Analysis Services, které umožňují tvorbu OLAP kostek a další možnosti dolování dat.

4.1.1 Tvorba databáze

Pomocí nástroje Microsoft SQL Server Management Studio byla vytvořena základní struktura databáze. Tabulky bylo možno vytvářet pomocí SQL skriptu nebo manuálně pomocí studia.

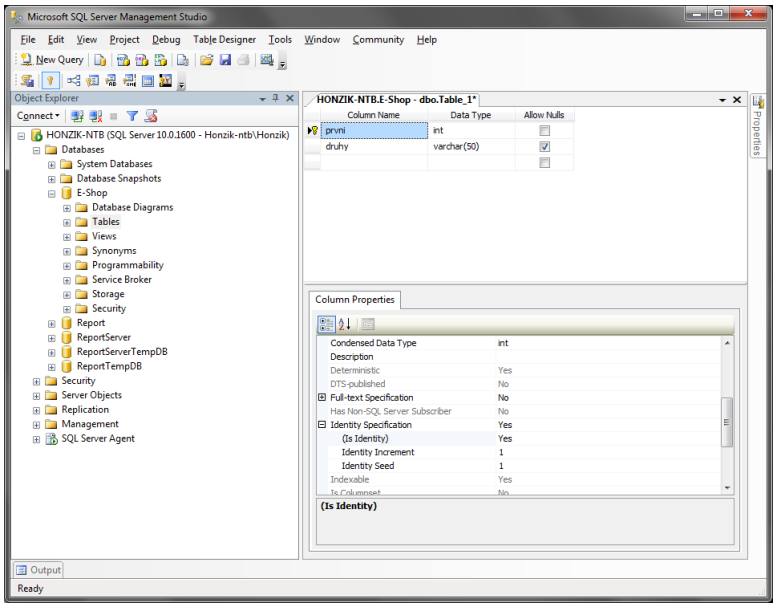
Po spuštění nástroje Server Management Studio je potřeba se přihlásit k nainstalovanému SQL Serveru. Je možné si vybrat z několika druhů serveru, jak je vidět na obrázku 4.1. V tuto chvíli postačil typ *Database Engine*, který umožňuje práci se základní SQL Databází.

Obrázek 4.1 Přihlašovací dialog

Po připojení k serveru se ukáže stromová struktura dat uložených na serveru. Ve složce *Databases* byla vytvořena databáze *E-shop*,se kterou autor celou dobu pracoval. Novou databázi lze vytvořit dvěma způsoby. První možností je spuštění příkazové řádky. Pravým tlačítkem myši klikneme na *Databases* a vybereme možnost *Start powerShell*. Druhou možností je pravým tlačítkem myši

kliknout na složku *Databases*, kde ze zobrazeného dialogu vybereme *New database*. Otevře se nám nové okno, v něm se nastavuje jméno databáze, zpětná kompatibilita databáze až po SQL Server 2000. A spoustu dalších pro nás v tuto chvíli nepodstatných věcí.

Jednotlivé tabulky se vytvářejí podobně. Opět máme dvě možnosti jako u tvorby databáze. Popíši zde druhou z možností. Zde se nám neotevře nové okno, ale pouze se otevře nová záložka v pravé části okna. Jak vypadá rozvržení okna je názorně vidět na obrázku 4.2.



Obrázek 4.2 Tvorba tabulky

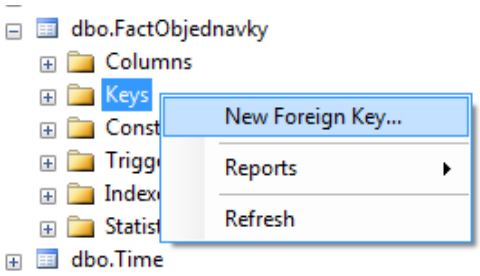
Tato záložka je rozdělena na dvě části. V horní části jsou sloupce tabulky. Nový sloupec přidáme tím, že začneme psát do prázdného posledního řádku, nebo pravým kliknutím na některý z řádku, pokud bychom chtěli vložit nový sloupec tabulky mezi již vytvořené sloupce.

V prvním sloupci tabulky se nastavuje název sloupce, ve druhém datový typ a ve třetím je možné vybrat, zda sloupec může obsahovat hodnoty *null*. Každá tabulka musí obsahovat primární klíč. Ten zvolíme pravým kliknutím na řádek, kde z dialogu vybereme *Set Primary Key*.

Ve spodní části záložky se nastavují vlastnosti jednotlivých řádku. Pro sloupec primárního klíče je zde důležitá vlastnost *Identity Increment*, která zajišťuje, že se hodnota primárního klíče nastavuje automaticky při vkládání nového záznamu do tabulky. Hodnota této vlastnosti se automaticky změní na hodnotu *1* po změně vlastnosti *Is identity*.

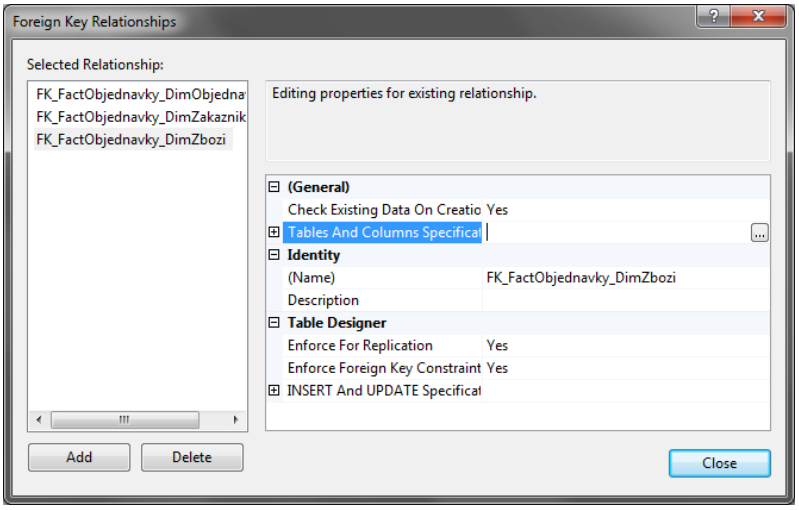
Ve chvíli kdy jsou vytvořeny všechny sloupce tabulky, je možné tabulku uložit. Při ukládání nastavíme název tabulky.

Ve chvíli kdy jsou vytvořeny všechny tabulky, je následujícím krokem vytvoření relací mezi jednotlivými tabulkami. Princip vytvoření relace spočívá v označení cizích klíčů mezi tabulkami. Abychom definovali relace mezi tabulkami, najdeme ve stromové hierarchii tabulku, která obsahuje cizí klíč, který chceme označit. Klikneme pravým tlačítkem na složku *Keys* a z nabídky vybereme *New Foreign Key*.



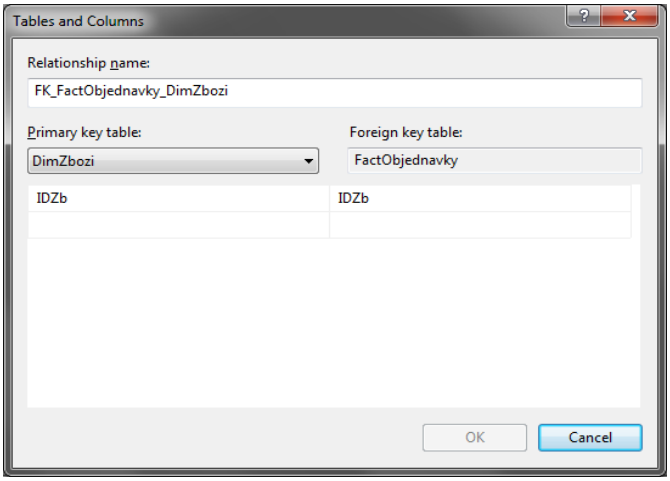
Obrázek 4.3 Nový cizí klíč

Nyní se otevře nové okno, které je vidět na obrázku 4.4. V něm se nastaví název vztahu. Pro lepší orientaci při další práci byla jako název zvolena jména obou tabulek. Po vyplnění názvu je nutné definovat vztah. To umožní okno, které se otevře kliknutím na ikonku tří teček, v řádku *Table And Columns Specification*.



Obrázek 4.4 Vlastnosti relace

V otevřeném dialogu, jehož ukázka je na obrázku 4.5, vybereme tabulku a sloupec, ve které se náš cizí klíč nachází jako primární klíč.



Obrázek 4.5 Nastavení sloupců cizího klíče

Ve chvíli kdy jsou vytvořeny všechny relace podle multidimenzionálního schématu je možné s vytvořenou databází plnohodnotně pracovat.

4.1.2 Propojení databáze s webovou aplikací

ASP.NET nabízí několik možností jak pracovat s daty v SQL databázi. Abychom k datům měli přístup, je nutné nejprve nastavit *Connection String*. *Connection String* je seznam nastavení/hodnot, které umožňují přístup ke vzdáleným datům. Na pořadí hodnot a velikosti písmen v řetězci nezáleží. Musí tam být obsaženy ovšem všechny potřebné informace k připojení, které bývají následující:

- Server -kde je databáze uložena
- Název -databáze, se kterou chceme pracovat
- Způsob -ověření vaší totožnosti

Connection String je možno zapsat přímo do souboru web.config, nebo jej vytvořit pomocí panelu Server Explorer.

Ve chvíli kdy je aplikace schopna připojení k databázi, je možné pracovat s daty dvěma způsoby.

SQL Data Source

SQL Data Source je z velké části postaven na znalosti jazyka SQL. Vrací tedy data z tabulek. Jeho silnou zbraní je jednoduchost a praktičnost. Umožňuje parametrické dotazování, které může získat parametr pro SQL dotaz z několika dalších zdrojů. Výstup dotazu je pak založen na hodnotě tohoto parametru. Parametry pro dotazy lze čerpat z následujících zdrojů:

Control – ovládací prvek na stránce, například *TextBox*

Cookie – hodnota uložena v cookie

Form – pole formláře předané pomocí POST

Profile – hodnota v uživatelském profilu

Query String – hodnota proměnné v URL adrese

Session – hodnota proměnné v session

None – hodnota přiřazená „na pevno“

LINQ

Druhým způsobem je využití technologie LINQ, což znamená Language INtegrated Query. Využití LINQ je možné od verze C# 3.0 nebo Visual Basic 9. Principem LINQ je převod dat na objekty, se kterými lze v kódu později jednoduše pracovat. Nad těmito objekty lze provádět dotazy podobně jako v SQL, to je umožněno díky novým klíčovým slovům.

LINQ je ovšem velice dynamický nástroj a rozlišuje se v těchto základních implementacích:

- LINQ to SQL
- LINQ to XML
- LINQ to Objects
- LINQ to Dataset

Není vyloučeno si ovšem připsat nějakou vlastní implementaci, která by byla potřeba.

Přestože je hlavním úkolem LINQ se na data dotazovat, dokáže je i velice snadno modifikovat nebo vkládat. Modifikace je založena na získání konkrétního objektu, po kterém následuje změna údajů a následné odeslání změn jediným příkazem.

4.2 Rozhraní nabízených služeb

4.2.1 Tvorba kostry stránek

ASP.NET nabízí k tvorbě internetových stránek velmi vyspělé prvky. Na vytvoření základní kostry internetových stránek byla použita technologie Master Pages. Master Pages využívá k sestavení výsledné stánky dva soubory.

Hlavní soubor má příponu *.master. Obsahuje prvky, které jsou pro všechny stránky našeho webu společné, tedy kde se zobrazí banner, jak bude velký, seznam odkazů v menu apod. Dále do něj je nutno umístit prvek *ContentPlaceHolder*, který zajišťuje propojení se stránkou Content Page.

Contetn Page je stránka, která se zobrazí v místě kde je umístěn *ContentPlaceHolder*. Propojení s *ContentPlaceHolder* zajišťuje prvek *Content*.

Pro zobrazení samotných informací v prohlížeči se dále již na stránku Master vůbec neodkazuje, vždy je v odkazu použit pouze Content Page, která se o výsledné sestavení postará sama.

4.2.2 Přihlašování a registrace

Pro práci s registrací a přihlašováním je v ASP.NET technologie Membership. Dokáže programátorovi velice snadno a rychle vytvořit formulář pro registraci, přihlášení, změnu hesla apod. Bohužel má jednu malou nevýhodu, a tou je výchozí provider.

Provider je třída, která zajišťuje obsluhu pro práci s uživateli. Jsou v ní implementovány metody pro různé akce s uživatelským účtem. Ovšem výchozí implementace této třídy si vytváří vlastní tabulky v databázi, které jsou nepřehledné a až zbytečně rozsáhlé.

Přepsání celé třídy by bylo velmi složité a není cílem této práce, tak jsem implementoval pouze nejnutnější metody. Ostatní metody, které musely být přepsány, vrací výjimku.

Registrace

Při registraci zákazníka, jsou data ukládána do dvou tabulek. Nebylo zde tedy použito Membership, ale celý formulář byl autorem vytvořen.

Kostra formuláře je vložena do prvku Wizard, který umožňuje tvorbu průvodce. Tento průvodce se může skládat z několika kroků. Mezi těmito kroky lze procházet pomocí navigačních tlačítek. Informace pro registrování uživatel vyplní do prvků typu *TextBox*. Většina vyplněných údajů je ověřována nástrojem *RegularExpressionValidator*. Tento prvek porovnává vložené informace s regulárním výrazem a v případě neshody nedojde k odeslání formuláře a zobrazí se nastavené

varování. Podobný nástroj byl použit pro ověření hesla. Ten ovšem neporovnává zadanou hodnotu s regulárním výrazem, ale porovnává dvě hodnoty mezi sebou na shodu.
Po správném vyplnění formuláře se data vkládají pomocí LINQ do databáze a uživatel se může přihlásit.

Přihlášení

Pro přihlášení uživatele je již použit prvek spolupracující s Membership. Jedná se o prvek *Login*. Dokáže vytvořit jednoduchou přihlašovací stránku, kde se pro ověření přihlašovacích údajů použije metoda z providera.

4.2.3 Nákupní košík

Pro realizaci nákupního košíku je využito cookies a jednu vlastní třídu. Cookies je zvoleno z důvodu možnosti uchování informací i po zavření prohlížeče. Jako další možnost bylo uvažováno ukládání dat do databáze, což bylo shledáno jako zbytečně složité a neumožňovalo by to uložení košíku nepřihlášeným uživatelům. Podobný problém by byl u session.
Cookies nejsou určeny pro uchovávání citlivých údajů, proto se do nich v tomto případě ukládá pouze primární klíč z tabulky zboží a počet objednávaných kusů. Tyto údaje jsou odděleny dvojtečkou. V případě objednávání více druhů zboží jsou jednotlivé druhy odděleny středníkem.
S uloženým cookies dále pracuje třída *kosik*, ve které se ukládají informace o celkové ceně apod., které jsou na stránce nákupního košíku zobrazovány. Hlavním úkolem této třídy je však manipulace s položkami, tedy jejich přidávání, odebírání a změny. Jednotlivé položky v košíku jsou také realizovány jako třída.
Po jakékoliv manipulaci s nákupním košíkem, se pomocí metody *ToString()* ukládají změněné informace zpět do cookies.

4.2.4 Zobrazování dat

Informace uložené v databázi by nebyly příliš užitečné, kdyby k nim neměl přístup zákazník v jednoduše čitelné formě. K zobrazování dat z databáze má ASP.NET velmi širokou nabídku možností. Stručně jsou popsány pouze varianty, které byly využity.

GridView

Je jedním ze základních ovládacích prvků pro zobrazování dat. Nabízí velmi rozsáhlé možnosti pro zobrazování dat z databáze nebo jiných datových zdrojů. Zobrazená data se načtou do tabulky, kterou je možné lehce naformátovat, přidat stránkování, editaci řádků a spoustu dalších věcí.
Výstup GridView lze generovat automaticky na základě datového zdroje. Touto cestou se zobrazují všechny sloupce, což není příliš praktické. Většinou je třeba některé sloupce skrýt nebo změnit pořadí, popřípadě vytvořit odkaz apod. Bližší informace o tomto prvku je možné najít například v [5].

DetailsView

Jak již název napovídá, nezobrazuje se zde několik záznamů najednou, ale pouze jeden. Prvek má opět velmi širokou nabídku možností. Lze využít stránkování, takže uživatel pak může přecházet mezi jednotlivými záznamy. Také jsou zde možnosti editace záznamu, odstranění nebo přidání nového. Bližší informace se dozvíte například v [5].

ListView

ListView je novějším prvkem než GridView a DetailsView. Pro správnou funkčnost potřebujeme definované dvě základní šablony. Jsou jimi *LayoutTemplate* a *ItemTemplate*. Lze je definovat ručně nebo je nechat vygenerovat automaticky. Oproti ostatním možnostem je ListView nejflexibilnější. Lze snadno využít pro výpis jednoho prvku nebo více prvků. Podporuje také vkládání a úpravu záznamů v databázi. Pro každou akci se nastavuje samostatná šablona.

Objekt tabulka

Protože nebyl nalezen jednoduchý a elegantní způsob výpisu nákupního košíku. Byl použit objekt *Table*, který byl vložen na stránku a označen identifikátorem. Při načítání stránky byla tabulka přizpůsobena aktuálním potřebám. Vytvořeny byly objekty *TableRow* a *TableCell,*které byly postupně přidávány do tabulky. Jako velmi zajímavé se jevilo jednoduché přidávání dalších prvků do tabulky, například tlačítek a textových polí.

4.2.5 Import zboží

V případě tabulky obsahující informace o nabízeném zboží, je tabulka plněna daty z XML souboru. Po načtení XML do objektu *XmlDocument* se vyberou hodnoty „uzlů“, které mají požadovaný název a uloží se do objektu *XmlNodeList*, který se projde pomocí cyklu *foreach* a uloží do databáze záznamy, které obsahují informace o jednotlivých položkách zboží. Hodnoty pro záznam, který se uloží do databáze, jsou načteny z atributu „uzlu“.

Důležitou roli u procesu importu hrálo čištění dat, které je poměrně dost pracné. Bylo důležité dohledat jaké hodnoty, která vlastnost může mít. Informace v XML souboru nemají dost často jednotný formát. Před vložením dat do databáze je velké množství hodnot upravováno pomocí podmínek a příkazu *switch.*

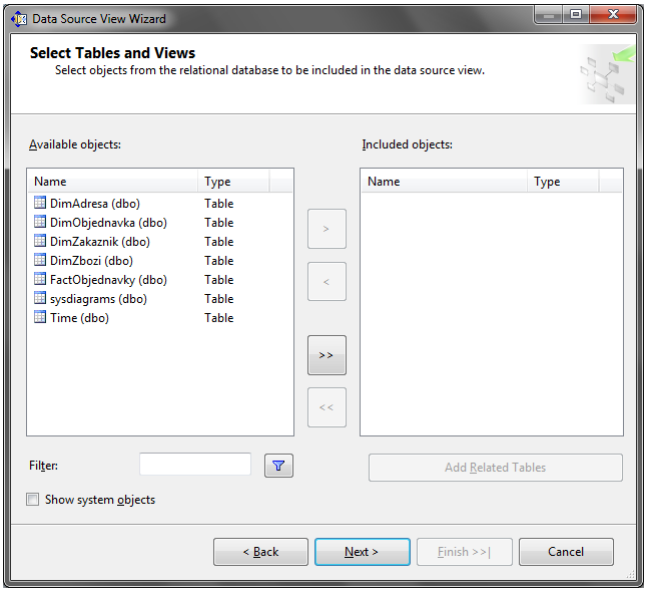
4.3 Práce s OLAP daty

Pokud chceme vytvářet analýzy nad OLAP databází, je třeba tyto data zpracovat. To vyžaduje nejprve vytvoření OLAP kostky a jednotlivých dimenzí. Pro vytvoření kostky byl použit nástroj Analysis Service, který spadá do Business Intelligence nástrojů.

Business Intelligence nástroje jsou určeny pro generování sestav a analýzu dat. Od verze 2008 je dostupná funkce generování grafů. Bližší informace jsou uvedeny v [3].

4.3.1 Definování zdrojů OLAP kostky

OLAP kostku je třeba vytvořit ve zcela novém projektu, ve kterém vybereme jako šablonu Analysis Services Project. Jako zdroj dat pro další práci poslouží databáze, která je využívána v elektronickém obchodě. Vytvoření datového zdroje je velmi podobné tvorbě *Connection Stringu*. Po nastavení základních informací o serveru, na kterém se databáze nachází, se vybírá způsob autentizace. Zde je třeba upozornit, že je nutné vybrat možnost, kde se zadává uživatelské jméno a heslo uživatele windows. V ostatních případech nelze kompilovat multidimenzionální kostku ani jednotlivé dimenze. Definováním připojení k databázi příprava pro přístup k datům nekončí. Je nutné vytvořit pohled na datový zdroj (Data Source View). V případě práce s rozsáhlou databází je potřeba pracovat pouze s vybranou částí databáze, což nám umožňují *Data Source Views*. Datový pohled lze vytvořit snadno pomocí průvodce, kde si uživatel vybere tabulky, které má pohled obsahovat. Jak krok s výběrem vypadá je vidět na obrázku 4.6.



Obrázek 4.6 Průvodce *Data Source Views*

V datovém pohledu je možné dodefinovat vazby mezi tabulkami, které můžeme k práci potřebovat. Další možnosti, které pohled nabízí, jsou například organizace diagramů nebo tvorba dotazů. Jednou z jeho hlavních funkcí je obnovení pohledu, protože struktura pohledu vychází ze struktury datového zdroje při jeho vytváření, může se po určité době struktura datového zdroje změnit. V případě její změny se musí pohled přizpůsobit a reagovat na změnu.

4.3.2 Vytvoření dimenzí

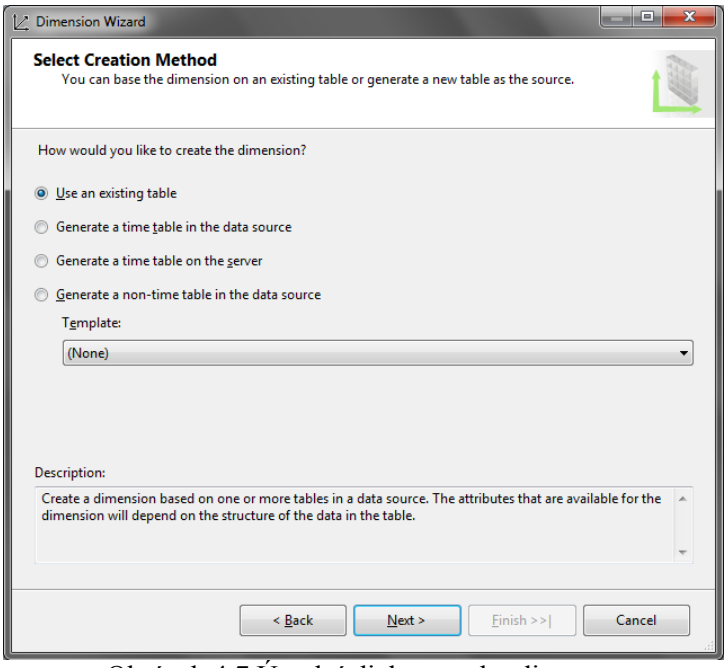
Když už máme přístup k datům v databázi, je možné se věnovat tvorbě jednotlivých dimenzí.

Časová dimenze

Tvorba časové dimenze se od ostatních dimenzí dost liší, a to z důvodu, že je vygenerována automaticky. Navrhnuta byla pouze její hierarchie. Pokud by dimenze byla vytvářena ručně, musela by tabulka obsahovat záznamy jednotlivých dnů v kalendáři, které jsou při automatickém generování doplněny dle zvoleného rozsahu.

Dimenze se vytvářejí také pomocí průvodce. V úvodním dialogu lze zvolit ze čtyř možností. Vzhled celého dialogu vypadá je zachycen na obrázku 4.7. V případě časové dimenze lze vybrat následujících dvou:

- Generování časové tabulky v datovém zdroji
- Generování časové tabulky na serveru



Obrázek 4.7 Úvodní dialog tvorby dimenze

Zvolena byla možnost vygenerování časové tabulky v datovém zdroji. Lze s ní tak pracovat pomocí nástroje SQL Server Management Studio.

Dalším krokem v případě časové dimenze bylo zvolení hierarchie. K tomu slouží jednouchý dialog, který je na obrázku 4.8. V prvních dvou řádcích se volí rozsah dat, která bude obsahovat vygenerovaná tabulka. Hierarchie časové dimenze je následující:

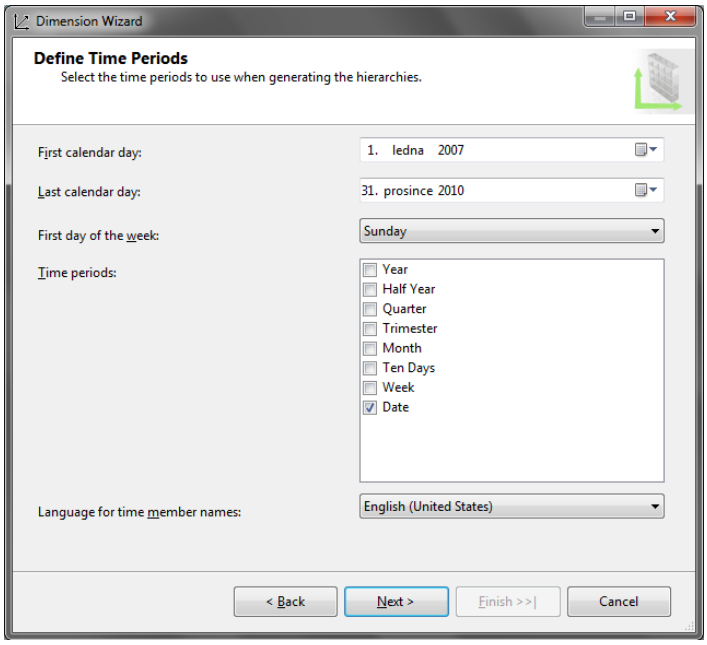
● Rok

●● Čtvrtletí

●●● Měsíc

●●●● Den

●●●●● Datum



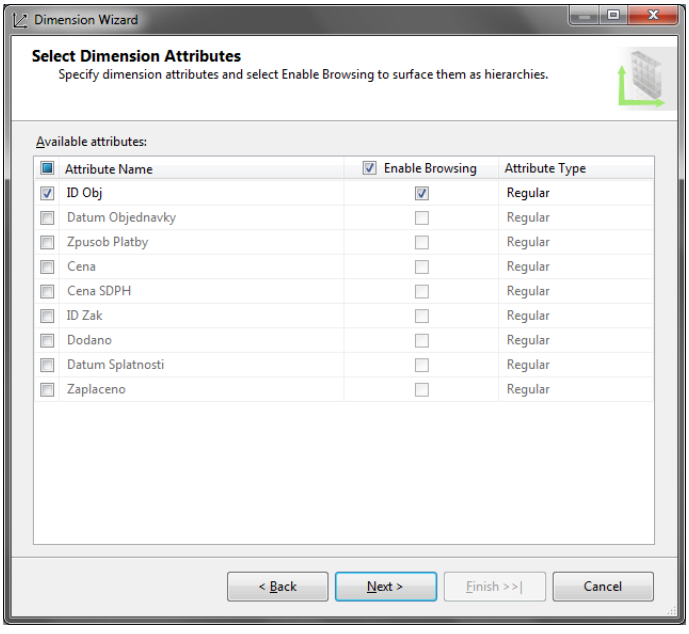
Obrázek 4.8 Hierarchie časové dimenze

Ve zbývajících krocích si lze prohlédnout zvolenou hierarchii.

Dimenze objednávek

Při tvorbě dalších dimenzí nebylo využito automatického generování, ale byly vytvořeny na základě existujících tabulek. Použita tedy byla první možnost z nabídky, která je na obrázku 4.7. Následujícím krokem je volba tabulky, která je pro dimenzi zdrojem dat. Ve stejném kroku je volen i klíčový sloupec.

U zvolené tabulky je třeba zvolit atributy dimenze. U zvolených atributů lze měnit typ atributu. Typem atributu lze určit, že se u daného sloupce jedná například o město nebo kategorii zboží. Vzhled dialogu pro volbu atributu je na obrázku 4.9. Atributy se volí podle hierarchie dimenze.



Obrázek 4.9 Dialog pro nastavení atributů dimenze

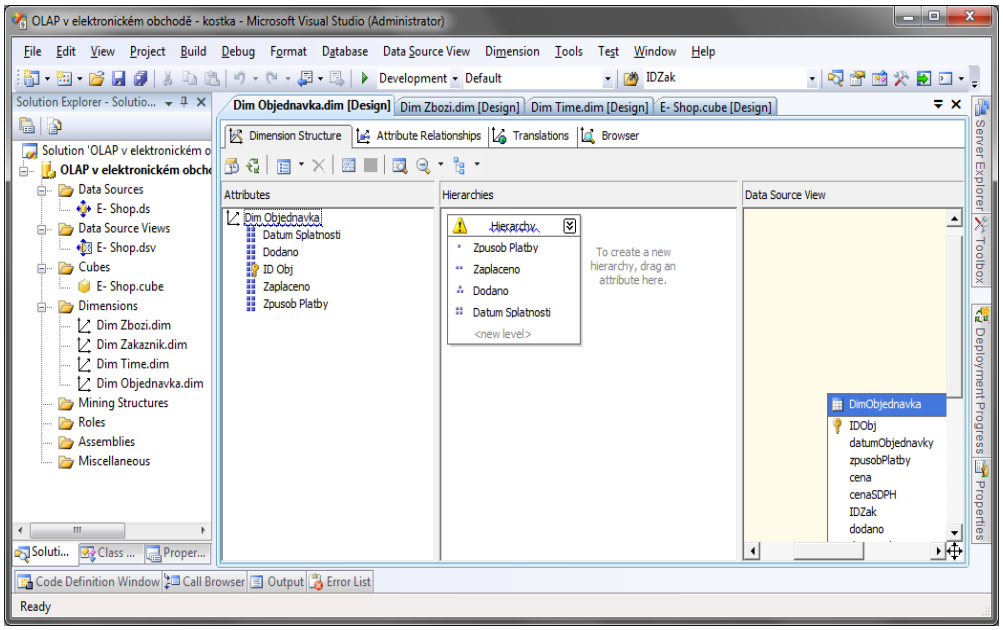
Dimenze objednávek by jí mohla mít například následující:

● Způsob platby

●● Zaplaceno

●●● Dodáno

●●●● Datum splatnosti



Obrázek 4.10 Tvorba hierarchie

Po dokončení průvodce lze atributy přidávat z *Data Source View*. Zvolené atributy se pak skládají do hierarchie. Hierarchie se vytváří způsobem „drag and drop“.

Ve vývojovém prostředí je možné s dimenzemi dále pracovat. Lze vytvořit pro jednu dimenzi více hierarchií, anebo mezi atributy vytvářet vztahy. Vytvoření vztahů mezi atributy zrychluje práci s velkými objemy dat. Ovšem správné navrhnutí vztahů není úplně jednoduché.

Zbylé dvě dimenze se tvořily podobně jako dimenze objednávek. Hlavní hierarchie dimenzí jsou následující:

Zákazník

● Stát

●● PSČ

●●● Město

●●●● Ulice

Zboží

● Výrobce

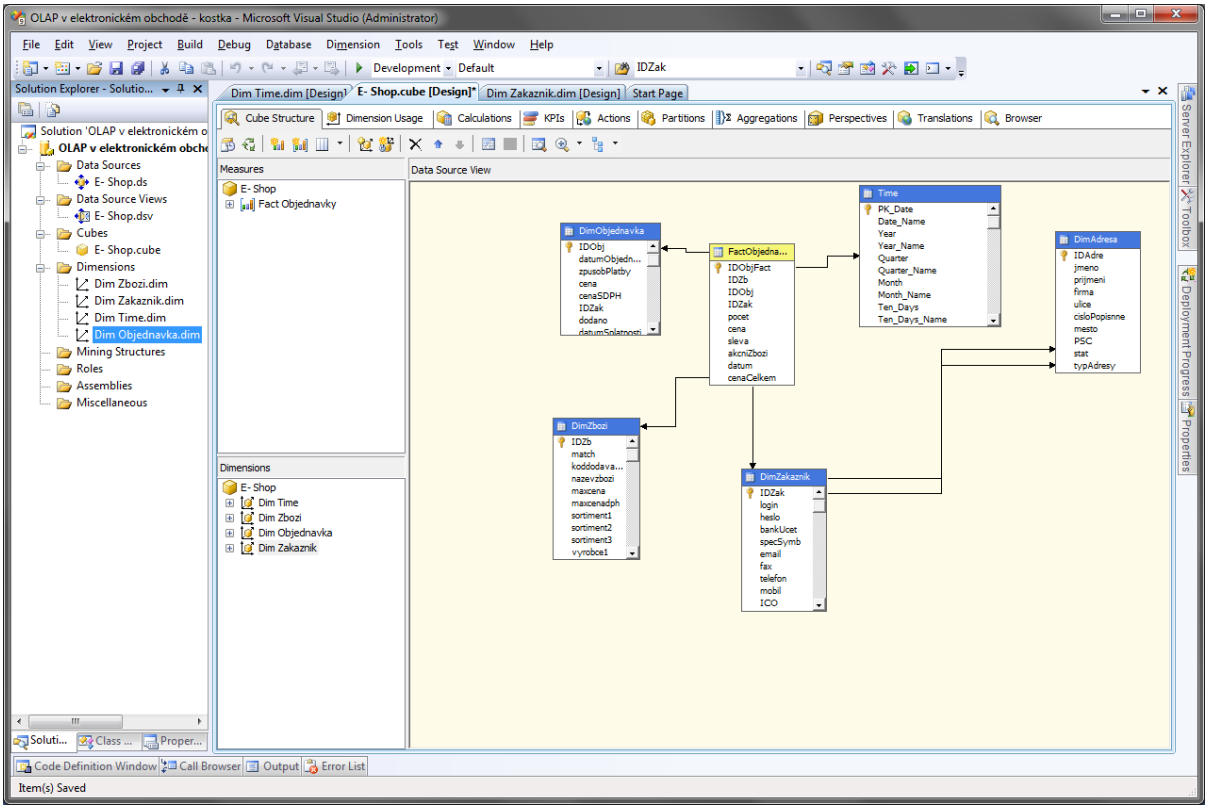
●● Sortiment 1

●●● Sortiment 2

●●●● Název

4.3.3 Vytvoření kostky

Když jsou hotové všechny potřebné dimenze, zbývá vytvořit kostku, která bude z těchto dimenzí složena. Kostku lze nechat vygenerovat automaticky nebo ji vytvořit prázdnou. Byla použita prázdná kostka, do které byly postupně přidávány dimenze a měrné jednotky. Vývojové prostředí pro kostku je vidět na obrázku 4.11. Jistě stojí za povšimnutí kolik možností je zde k dispozici.



Obrázek 4.11 Prostředí pro návrh kostky

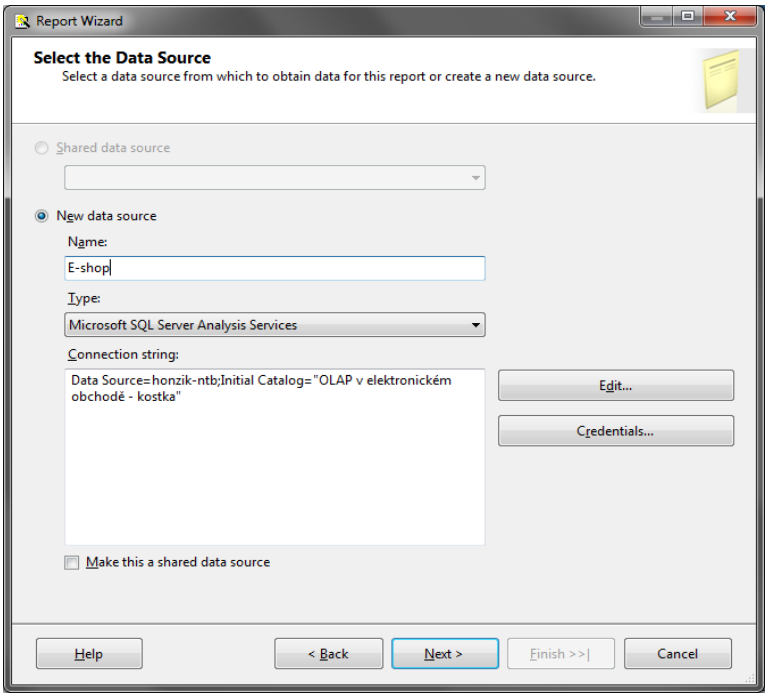
Ve chvíli kdy je kostka sestavena, můžeme si prohlédnout data, která jsou v ní uložena. Lze to udělat pomocí záložky *Browser*. Aby bylo možné zde vidět zadaná data, musíme provést sestavení projektu a odeslat jej pod správu analytického serveru. To lze udělat pomocí tlačítka se zelenou šipkou nebo přes příkazy *Build* a *Deploy*.

4.3.4 Reprezentace dat v internetovém obchodě

Informace shromážděné v OLAP databázi je nutné nějak reprezentovat uživateli, který by je měl využít k pomoci při nějakém rozhodnutí. K tomuto účelu slouží reporty. Pro vytvoření reportů je třeba založit opět nový projekt. Jako šablonu vybereme *Report Server Project Wizard*.

Po založení projektu se hned nabídne průvodce pro vytvoření datového zdroje. Po zadání názvu vybereme jako typ zdroje *Microsoft SQL Server Analysis Services*, který je určen pro zpracování OLAP dat. Dále musíme zadat *Connection string*, pomocí kterého se budeme připojovat k datovému zdroji. Jako zdroj se zde nevyužívá stejná databáze jako pro elektronický obchod, ale projekt s OLAP kostkou. Vzhled úvodního kroku průvodce je ukázán na obrázku 4.12. Výhodnou volbou je vytvořit

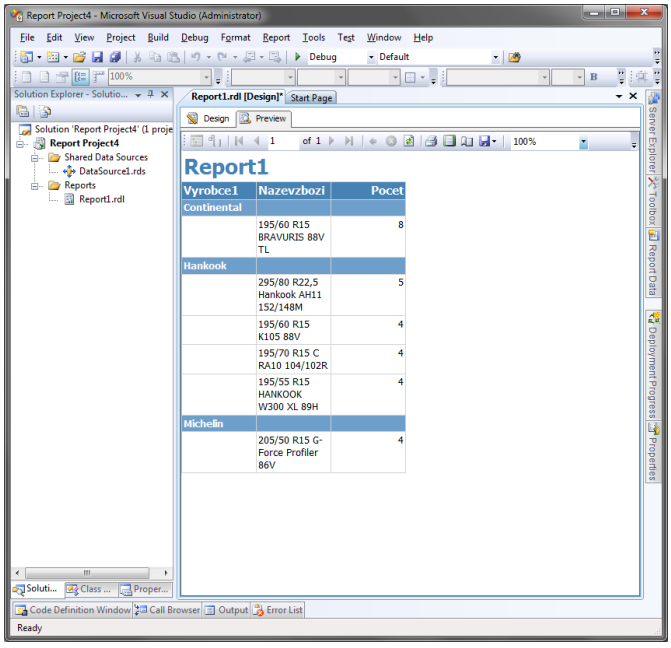
zdroj dat jako sdílený, abychom ho bylo možné využít i pro další reporty. Pokud není vytvořen sdílený zdroj, musí se pro každý report nastavit vlastní zdroj dat.



Obrázek 4.12 Nastavení datového zdroje reportu

Po nastavení datového zdroje následuje vytvoření dotazu, podle kterého budou zobrazena data. K tomu slouží nástroj *Query Designer*. Lze v něm poměrně snadno definovat, podle jakých dimenzí se data budou zobrazovat a v jakých měrných jednotkách. Nástroj je založen na principu „drag and drop“, vytvoření dotazu je tedy poměrně snadné a rychlé.

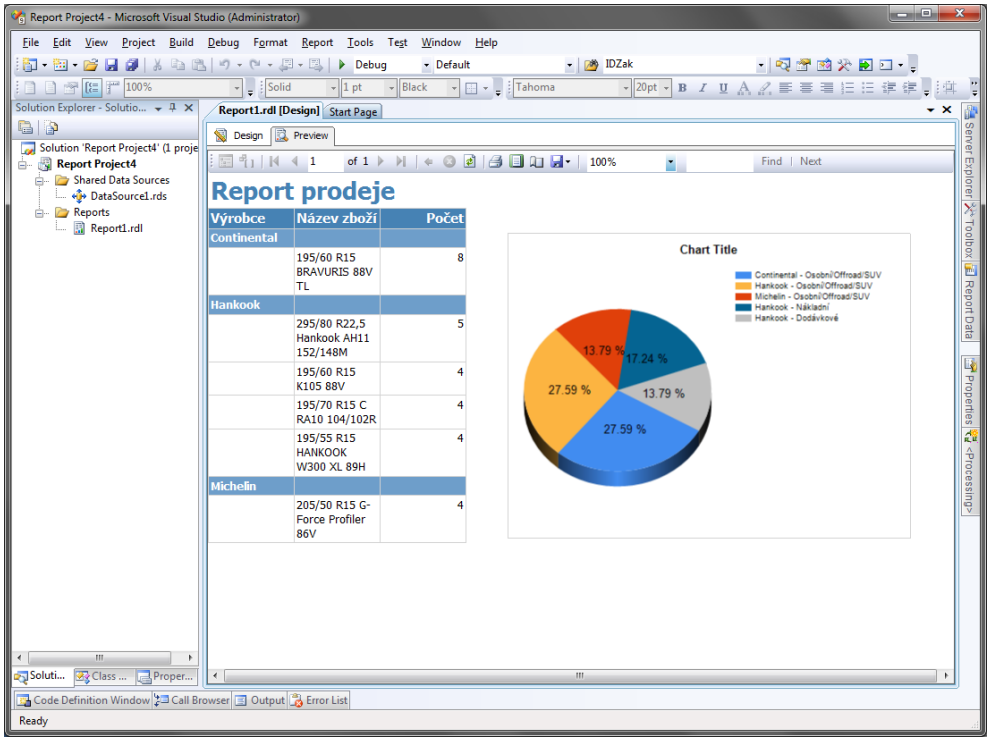
Následující kroky průvodce nastavují vzhled a seskupování údajů, popřípadě stránkování. Výsledný report může vypadat například jako na obrázku 4.13.



Obrázek 4.13 Ukázka výsledného reportu

V záložce *Design* lze vzhled reportu dále upravovat a přizpůsobit našim potřebám. Je možné do něj přidat i další prvky jako je například graf. Vložení grafu lze provést pomocí postranního panelu *ToolBox*. Po vložení grafu dostaneme na výběr z několika desítek různých vzhledů. Po vybrání vzhledu musíme nastavit datová pole pro graf. Rozlišují se tři kategorie:

- Category field – kategorie, podle kterých se graf člení
- Series field – podkategorie, na které se dělí kategorie
- Data field – data, která se vyhodnocují



Obrázek 4.14 Ukázka reportu s grafem

Na obrázku 4.14 je ukázka grafu. Za povšimnutí stojí, že do grafu lze přidat například podíl jednotlivých výrobců v procentech. Do *series field* jsem nastavil kategorii výrobku, tím se rozdělili výrobci na další kategorie.

Zobrazení reportu na internetových stánkách

Po vytvoření reportu je nutné provést další kroky, abychom mohli report zobrazit na internetových stránkách. Je potřeba mít k dispozici reportovací server, který je součástí Reporting Services. Ten lze nainstalovat společně s SQL Serverem. Dříve než ho začneme využívat, musí se správně nastavit. To umožňuje nástroj Reporting Services Configuration Manager.

Nastavují se v něm přihlašovací údaje k serveru, databáze a URL, přes kterou lze k serveru přistoupit. Po úspěšném nastavení report serveru zbývá nastavit u vytvořeného reportu cílovou URL serveru, aby bylo možné provést *Build* a *Deploy*. Po úspěšném dokončení tohoto kroku, se report zobrazí na internetových stránkách.

K zobrazení reportu v projektu s internetovými stránkami slouží komponenta *MicrosoftReportViewer*. Nastaví se u ní URL report serveru a cesta k reportu na straně serveru. Je možné jí pomocí různých vlastností upravovat vzhled. Velikou výhodou je, že prvek *MicrosoftReportViewer* umožňuje export do souborů běžných formátu jako *.pdf nebo *.csv.

5 Možná rozšíření

Pro elektronický obchod by mohlo být realizováno velké množství rozšíření. Zmíněny jsou zde pouze ty zajímavější návrhy.

Grafy cen u jednotlivých produktů

Pokud by se vhodně upravila databáze. Mohl by u každého produktu být vygenerován graf, zobrazující historický vývoj ceny. Úprava databáze je vyžadována kvůli přepisování aktuální ceny. Grafické znázornění vývoje cen je v dnešní době poměrně populární. Proto, bych toto rozšíření doporučoval.

Spolupráce s více dodavateli

Tento elektronický obchod, byl navržen tak, aby nebylo žádné zboží fyzicky skladem. Spolupracuje však teoreticky pouze s jedním dodavatelem. V případě zlepšení toho systému by mohl využít služeb, několika dodavatelů. Každý dodavatel, od kterého by zboží bylo odebíráno, by musel umožnit dotazování do jeho skladů a nabídky pomocí vzdáleného volání procedur. Pokud by toto rozšíření bylo vhodně realizováno, mohlo by být velmi prospěšné.

6 Závěr

Během vypracování předložené práce se autor seznámil s teorií OLAP systému a výhodami a nevýhodami oproti běžným OLTP databázím. Pokus o využití v praxi umožnil lépe pochopit problematiku datových skladů. I přesto nebylo možné se zcela seznámit s kompletní problematikou OLAP systému, neboť tato technologie je velmi rozsáhlá a k jejímu komplexnímu zvládnutí je třeba dalšího studia problematiky a vypracování rozsáhlejších projektů.

Pro využití v praxi bylo potřeba realizovat funkční elektronický obchod. Obchod byl implementován v ASP.NET, kde byl ze dvou možných jazyků zvolen C#. K rychlejšímu zvládnutí práce s programovacím jazykem C# bylo využito znalostí objektově orientovaného programování. Tato možnost tvorby internetových stránek se ve srovnání s použitím PHP jazyku jevila jako velmi efektivní a elegantní.

Pro realizaci OLAP systému byla využita technologie Business Intelligence od firmy Microsoft, která byla nainstalována společně s Microsoft SQL Serverem 2008 Enterprise. Při práci s těmito nástroji se objevily různé potíže, jejichž řešení bylo mnohdy obtížné. Po zprovoznění všech nástrojů, potřebných k využívání OLAP, bylo užití jejich vývojového prostředí velmi intuitivní. Zobrazení dat z kostky na internetové stránky bylo provedeno prostřednictvím služeb reportovacího serveru, který umožní zobrazení reportu v prohlížeči. Report lze skládat z navržených dimenzí využité v multidimenzionálním schématu. Výstupem byly vhodně naformátované grafy a tabulky.

Jako možné rozšíření bylo uvažováno využití grafického výstupu z databáze, který by mohl být prospěšný i pro zákazníka.

Závěrem je nutné podotknout, že nebylo možné během testování dostat smysluplné výsledky, které by pomohly k podpoře rozhodování. Podmínkou pro zisk plnohodnotných výsledku je provedení analýzy nad reálnými daty.

Literatura

[1] LACKO, Ľuboslav. *Business Intelligence v SQL Serveru 2008 : reportovací, analytické a další datové služby*. Vyd. 1. Brno : Computer Press, 2009. 456 s. ISBN 978-80-251-2887-9.

[2] PUŠ, Petr. *Vývojář.cz* [online]. 24. 1. 2008 [cit. 2010-05-16]. Úvod do LINQ. Dostupné z WWW: <http://www.vyvojar.cz/Articles/563-uvod-do-linq.aspx>.

[3] *Business Intelligence* [online]. 2008 [cit. 2010-05-16]. Microsoft. Dostupné z WWW: <http://www.microsoft.com/cze/sqlserver2008/business-intelligence.mspx>.

[4] *Datové sklady a OLAP* [online]. 28. 10. 2002 [cit. 2010-05-16]. Data Mining Solutions. Dostupné z WWW: <http://datamining.xf.cz/view.php?cisloclanku=2002102808>.

[5] MACDONALD, Matthew; SZPUSZTA, Mario; ALLEN, K. *ASP.NET 2.0 a C# : tvorba dynamických stránek profesionálně*. Vyd. 1. Brno : Zoner Press, 2006. 1376 s. ISBN 80-86815-38-2.

[6] ZENDULKA, Jaroslav. Získávání znalostí z databází. Brno, 2009. 160 s. Studijní opora. FIT VUT Brno.