Technická univerzita v Liberci
Hospodářská fakulta

Studijní program: 6209 – Systémové inženýrství a informatika
Studijní obor: Manažerská informatika

Název práce

Řešení problematiky změn dimenzí při tvorbě datového skladu a následné OLAP analýzy

Číslo diplomové práce: DP – MI – KIN – 2006 01

VÁCLAV BAHNÍK

Vedoucí práce: Prof. Ing. Jan Ehleman, CSc., KIN
Konzultant: Ing. Zádová Vladimíra, KIN

Počet stran: 70 Počet příloh: 1
Datum odevzdání: 6. 1. 2006

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, zejména § 60 - školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce a konzultantem.

Datum: 6.1.2006

Podpis: …….....

Resumé:

Tato diplomová práce se zabývá řešením problematiky měnících se dimenzí v prostředí datového skladu a následné OLAP analýzy a jejím cílem je přinést ucelený pohled na danou problematiku.

V první části jsou představeny principy a nástroje Business Intelligence, které s měnícimi se dimenzemi souvisí.

Poté následuje shrnutí teoretických poznatků a definice vhodných postupů, které zajistí ošetření problémů měnících se dimenzí

V další části jsou teoreticky popsané postupy aplikovány na dva konkrétní příklady – v diplomové práci je podrobně popsána implementace postupu řešení měnících se dimenzí od načtení zdrojových dat do dočasného úložiště, přes nutné transformace prováděné při ukládaní načtených dat do datového skladu, až po úpravu metadat OLAP datových kostek pro splnění požadavků kladených na řešení měnících se dimenzí.

Resumé:

This thesis deals with solving of Changing dimensions dilemma in the field of Data Warehouse and consecutive OLAP analysis. Its goal is provide integrated view on the dilemma.

In the first part there are introduced principles and instruments of Business Intelligence which relate to Changing dimensions.

After this there is a summary of theoretical pieces of knowledge and a definition of suitable procedures that ensure treatment of Changing dimensions.

In the next part there are the theoretically described procedures applied to two concrete examples - in the thesis there is in detail defined implementation of solving procedure of Cchanging dimensions, starting with loading source data into Data Staging Areas, over necessary transformation taken during saving of unloaded data into a Data Warehouse, up to the adjustment of metadata in OLAP cubes for fulfilment of requirements desired from solving of Changing dimensions.

Klíčová slova:

Měnící se dimenze, Pomalu se měnící dimenze, ETL procesy, Datový sklad, OLAP, Multidimenzionální úložiště dat, datová kostka, Microsoft SQL 2000 Server, Microsoft SQL 2000 Server Analysis Services, Microsoft SQL 2000 Server Data Transformation Services, Transact-SQL, datová integrace

Key words:

Changing dimensions, Slowly changing dimensions, ETL processes, OLAP, data cubes, Multidimensional data structure, Microsoft SQL 2000 Server, Microsoft SQL 2000 Server Analysis Services, Microsoft SQL 2000 Server Data Transformation Services, Transact-SQL, data integration

OBSAH:

Seznam použitých zkratek:

Zkratka	Anglický termín	Český termín
BI	Business Intelligence	Business Intelligence
CRC	Cyclic Redundant Checksum	Kontrolní součet cyklické redundance
CRM	Customer Relationship Management	Řízení vztahů se zákazníky
ČR		Česká republika
ČSÚ		Český statistický úřad
DM	Data Mining	Dolování dat
DTS	Data Transformation Services	Služby transformací dat
DW	Data Warehouse	Datový sklad
DWH	Data Warehouse	Datový sklad
EAI	Enterprise Aplication Integration	Integrace podnikových aplikací
EIS	Executive Information System	Systém pro podporu exekutivy
ERP	Enterprise Resource Planning	Plánování podnikových zdrojů
ETL	Extraction Transformation Loading	Extrakce Transformace Uložení
GB	Gigabyte	Gigabyte
HOLAP	Hybrid OLAP	Hybridní OLAP
HTML	Hyper Text Markup Language	neexistuje
HW	Hardware	Hardware
ICT	Information and Communication Technology	Informační a komunikační technologie
IS	Information System	Informační systém
IT	Information Technology	Informační technologie
LAN	Local Area Network	Místní síť
MB	Megabyte	Megabyte
MDX	MultiDimensional eXpression	neexistuje
MOLAP	Multidimensional OLAP	Multidimenzionální OLAP
ODBC	Open Database Connectivity	Otevřený přístup k databázi
OLAP	On Line Analytic Processing	neexistuje
ORP		Obec s rozšířenou působností

OWC	Office Web Components	Webové komponenty Office
POU		Pověřený obecní úřad
ROLAP	Relational OLAP	Relační OLAP
SCD	Slowly Changing Dimensions	Pomalu se měnící dimenze
SCM	Supply Chain Management	Řízení dodavatelských řetězců
SQL	Structured Query Language	Strukturovaný dotazovací jazyk
SW	Software	Software
TB	Terabyte	Terabyte
WAN	Wide Area Network	Vzdálená síť

1. Úvod

Žijeme v době, kdy téměř všechny organizace v naší „rozvinuté“ civilizaci využívají pro podporu svých činností Informační a komunikační technologie. Výrobním a obchodním společnostem slouží výpočetní technika k zajištění všech svých procesů od komunikace s dodavateli, přes fázi vývoje, výroby, skladování, až po péči o zákazníka. Státní a samosprávné orgány nasazují prostředky IS/ICT prakticky ve všech oblastech své působnosti, jako je například výběr daní, vyřizování dokladů, evidence grantových programů, výměna dat s občany, či mezi sebou navzájem.

Světová globalizace, která boří geografické hranice, vede k neustálé nejistotě budoucnosti organizace a k růstu konkurence. V důsledku toho se organizace, která chce na trhu uspět, musí neustále zabývat tím, zda to, co a jak dělá, vede k dosažení vytyčených cílů, zda je naplňována její strategie. K porovnávání dosažených výstupů s předem definovanými cíli, je třeba obě hodnoty, cíl i skutečnost, nějakým způsobem sledovat, zaznamenávat, uchovávat a poté analyzovat. Výsledky těchto analýz by pak měly tvořit podklad pro kvalifikované rozhodování vedoucích pracovníků organizací. IS/ICT, tj. informační systémy založené na informační a komunikační technologii je možné z tohoto hlediska rozdělit na produkční (transakční), které slouží k sledování činností podniku a systémy Business Intelligence. Produkční systémy, které jsou primárně určeny pro podporu procesů uvnitř i vně organizace, zaujímají samozřejmě v procesu tvorby podkladů pro rozhodování významné místo - jsou vhodné pro sledování, zaznamenávání a částečně pro uchovávání údajů o dosažených výstupech. Nejsou však příliš vhodné pro jejich analýzu. A to proto, že za prvé ve většině případů uchovávají pouze část dat potřebných pro komplexní analýzu. Za druhé nedisponují výkonnými a efektivními analytickými nástroji. Na tuto oblast, tedy na integraci a transformaci dat z výše popsaných zdrojových systémů se zaměřuje Business Intelligence, která tak přináší nástroje pro výkonné analýzy, jejíž výstupy by měly tvořit podklady pro rozhodování.

Při analýzách je velice důležitý rozměr času. Skutečnost je porovnávána s historií a na tomto základě se organizace pokouší odhadnout trendy, kterými by se okolní prostředí mohlo ubírat. Tyto trendy pak zásadně ovlivňují chování organizací, kterým se snaží na

odhadované trendy připravit a změnit tak odhad na konkurenční výhodu. Při porovnávání dat získaných v přítomnosti s daty historickými se velmi často objevují zásadní problémy, které spolu úzce souvisí, a které výnamně ovliňují kvalitu trendu. Pro odhad trendů se používají matematické a statistické metody, které jsou tím přesnější, čím delší je sledovaná časová řada. Prvním problémem je tedy fakt, že organizace musí mít uchovávána historická data. Druhou věcí, která je pro jakoukoli analýzu a následný odhad trendu naprosto kritická, je konzistence sledovaných a porovnávaných atributů – musíme se dívat stále na totéž. A to je v dnešním turbulentním prostředí velice obtížné.

Představme si, že jako manažeři sledujeme výkonnost nějakého prodejního oddělení. Ekonomický systém zaznamenává jednotlivé prodeje konkrétních obchodních zástupců, nicméně příslušnost obchodního zástupce k prodejnímu oddělení uchovává pouze pro okamžik přítomnosti – „jak je to nyní“. Obchodní zástupci však mezi odděleními v průbehu času přechází, oddělení se slučují, štěpí. Jak ale má manažer analyzovat výkonnost za delší časový úsek na základě dat z takového systému, když např. ví, že mu dva obchodní zástupci v průběhu sledovaného období přibyli a naopak tři odešli do jiného oddělení – dva příchozí si sebou „přinesli“ své prodeje ze svého minulého oddělení, naopak tři odchozí si své prodeje „odnesli“ na své nové působiště? A co teprve když manažer neví přesně, kdy ke změnám došlo? Asi bude nucen si od personalistů vyžádat údaje o datech přestupu obchodních zástupců, bude muset získat údaje o prodejích nejen svého oddělení, ale i oddělení, do kterého jeho bývalí obchodní zástupci přešli, a všechny údaje bude muset pospojovat. Tak získá přehled, jak bylo jeho oddělení úspěšné. V dalším období bude muset podobnou proceduru absolvovat znovu. Je to efektivní? Určitě ne. A to jsme jen na úrovni oddělení, teď k tomu přidejme vyšší patro, kdy se sejdou jednotliví vedoucí oddělení a diskutují s nadřízeným o svých ekonomických úspěších. Každý si přinese vlastní podklady, někdo ve svých výpočtech výše popsané persnální změny uvažoval, jiný nikoli – celková suma prodejů jednotlivých oddělení pak nesouhlasí s celkovým objemem prodeje organizace získaném z ekonomického software… A tak bychom mohli pokračovat. Mohou být takovéto údaje považovány za podklady určené pro kvalifikovaná rozhodnutí, např. zrušení nebo naopak podpora neperspektivních prodejních odděleních? Jsem přesvědčen, že nikoli.

Moje diplomová práce se zabývá postihnutím právě takových změn v čase, aby tvorba časových řad a následný odhad trendů vycházely z konzistentních a konsolidovaných dat.

Cílem diplomové práce je přinést ucelený pohled na danou problematiku. V první části jsou představeny principy a nástroje Business Intelligence, které s měnícími se daty v čase souvisí. Poté následuje shrnutí teoretických poznatků a definice vhodných postupů, které zajistí ošetření vytyčených problémů. V další části jsou teoreticky popsané postupy aplikovány na dva konkrétní příklady – v diplomové práci je podrobně popsána implementace teoreticky objasněného postupu řešení od načtení zdrojových dat do dočasného úložiště, přes nutné transformace prováděné při ukládaní načtených dat do datového skladu, až po úpravu metadat OLAP datových kostek.

Toto téma jsem si vybral proto, že pracuji ve společnosti PVT, a.s. v týmu, který se zabývá vývojem a implementací řešení Business Intelligence. Účastnil jsem se projektů, ve kterých bylo řešení změn dat v čase požadováno a podílel jsem se na vývoji řešení.

2. Představení problematiky měnících se dimenzi

Jelikož se má diplomová práce zabývá zachycením změn dat v čase pro věrohodnou analýzu, bude řešení vztahováno k systémům primárně určených k uchovávání a analýze historických dat heterogenních zdrojů – k nástrojům Business Intelligence. Tento pojem je možné definovat jako ucelený a efektivní přístup k práci s daty organizace, který má vliv na správnost strategických rozhodnutí a přináší schopnost efektivně využívat tato nashromážděná data k tvorbě informací a znalostí. Pokud se v systémech Business Intelligence nějakým způsobem sledují změny dimenzionálních dat v čase, hovoří se o „Měnících se dimenzích“. V publikacích věnovaných této problematice se Měnící se dimenze dále dle fekvence změn dělí na „Pomalu se měnící dimeze“ a „Rychle se měnící dimenze“. Protože má práce je zaměřená na souhrn poznatků z praxe a s reálným požadavkem na řešení Rychle se měnících dimenzí jsem se dosud nesetkal, v dalším textu se zaměřuji poze na Pomalu se měnící dimenze.

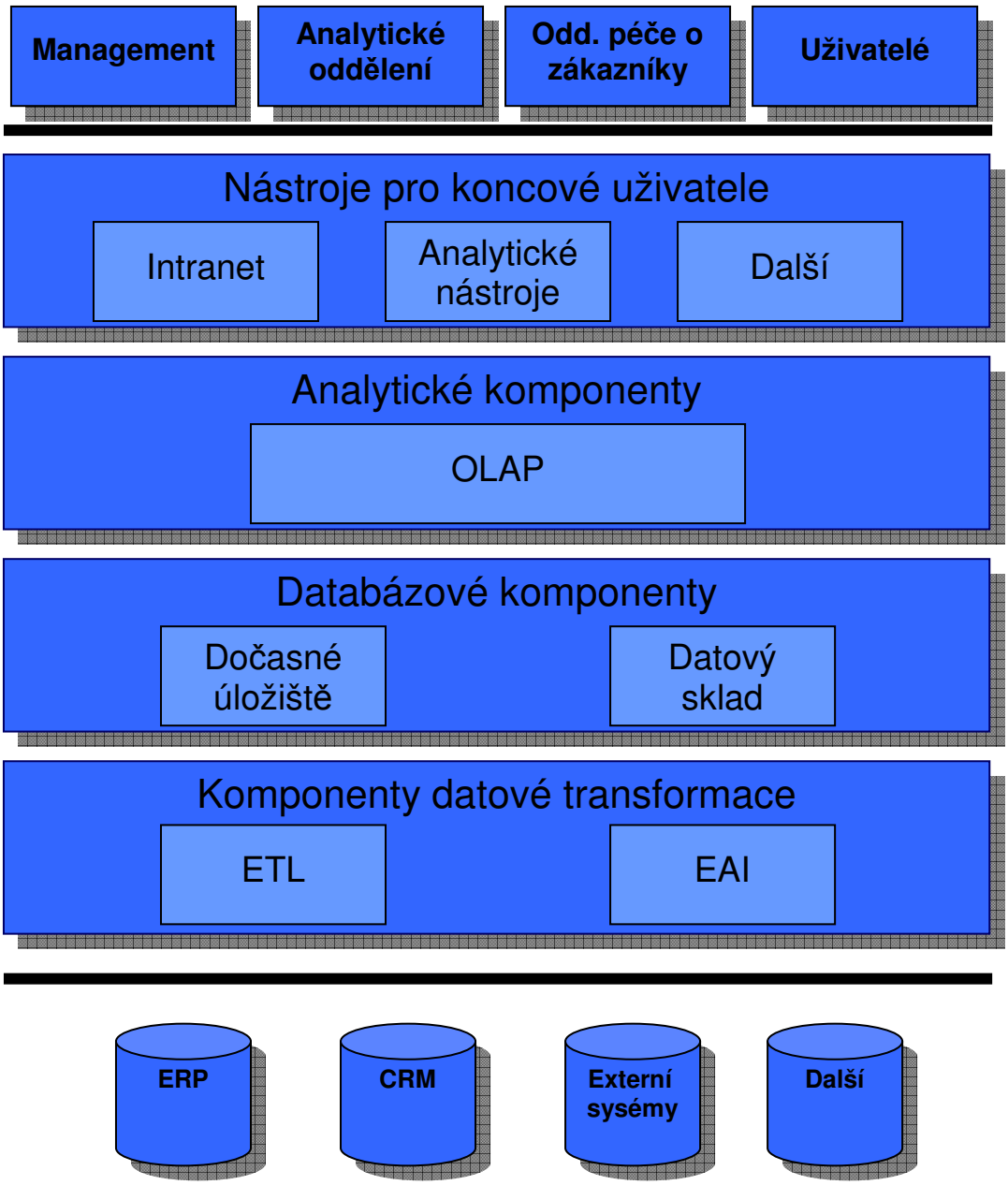
2.1. Business Intelligence

V první části teoretické pasáže považuji za vhodné představit stěžejní nástroje Business Intelligence, které mají vazbu k ošetření problémů Pomalu se měních dimenzí.

Architektura BI řešení

Pokud se na „klasické“ řešení BI (viz Obrázek č.1) podíváme z technologického pohledu, spatříme několik vrstev, které se „nabalují“ na zdrojové produkční systémy.

Obrázek č.1 – Architektura BI řešení



Zdroj: [přizpůsobený NOV]

Ve zkratce lze Obrázek č.1 popsat těmito slovy: Data z produkčních systémů jsou zdrojem dat pro Dočasné úložiště a následně i pro Datový sklad. V datovém skladu jsou data připravena pro další nástroje BI, jako je třeba OLAP analýza, díky nimž jsou výstupy zpřístupněny uživateli.

2.1.1. Zdrojové systémy

Provozní systémy, tedy především systémy SCM, ERP a CRM, nedokáží řešit úlohy typu BI z několika zásadních důvodů:

- **Nedostatečná historie dat** – provozní systémy udržují z kapacitních důvodů data stará pouze několik měsíců, starší data se přehrávají do archívu, kde mají velice omezenou využitelnost především z důvodu vysoké přístupové doby, která je v mnoha případech kritickým hlediskem.

- **Nedostatečné techniky a nástroje pro zpracování dat –** provozní systémy většinou disponují sadou připravených reportů, které jsou navíc orientovány spíše na sledování jednotlivých transakcí, případně jejich agregací, než na globální pohled.

- **Nemožnost zpracovat data z jiných aplikací** – v žádné společnosti není používán pouze jeden systém, byť existují velmi mohutná řešení, jako je např. SAP R/3. Řada dat navíc vzniká mimo provozní systémy třeba i jako původně osobní aplikace. Důležitá data rovněž pocházejí z externích zdrojů - ať se jedná např. o údaje z finančních trhů nebo data o průzkumu trhu.

- **Nepříznivý dopad na výkonnost provozních systémů** – zpracování dat v provozním systémech k souhrnným přehledům a analýzám představuje další zátěž a vede k prodlužování doby odezvy transakčních systémů pro běžné uživatele.

2.1.2. Datový sklad

Výše uvedené důvody vedou k zásadnímu konceptu datových skladů – požadavky na provozní systémy a na zpracování dat pro podporu rozhodování jsou natolik rozdílné, že vyžadují dva druhy systémů – provozní systémy a datový sklad.

Datový sklad je fyzicky a logicky oddělen od provozních systémů a představuje další krok v budování informačních systémů, přičemž data z provozních systémů se převádějí do datového skladu, kde se po transformaci ukládají způsobem, který vyhovuje analytickému a prezentačnímu zpracování výstupů.

Podstata datového skladu

Datový sklad představuje uložení dat, které má následující charakteristiky:

- Integruje data z různých zdrojů do jednoho systému
- Obsahuje historii - jsou k dispozici data i za několik minulých let
- Data jsou uložená na různých úrovních sumarizace
- Data jsou z provozních systémů načítána periodicky
- Uživatelé data většinou pouze čtou, tj. neprovádí jejich zadávání, ani je nemění (výjimkou jsou některé pokročilé metody OLAP analýzy, jako je například využití možnosti zápisu do datové kostky užívaný například při tvorbě variantních modelů nebo plánování)
- Data z datového skladu se využívají pomocí širokého spektra metod pro prezentace a analýzy dat

Skutečnost, že datové sklady jsou systémy zásadně odlišné od provozních transakčních systémů vyplývá z následující srovnávací tabulky.

Tabulka č.1 – Srovnání provozních systémů a datového skladu

Hlavní účel	
Provozní systémy	**Datový sklad**
Automatizace operací nebo procesů	**Poskytování optimálních informací pro rozhodování**
Koncepční rozdíly	
Provozní systémy	**Datový sklad**
Úkol: dostat data do systému	Úkol: Dostat informace ze systému
Uživatelé mají možnost zadávat, měnit, rušit a číst data	Uživatelé mají možnost data pouze číst
Zajišťují automatizaci rutinních činností	Umožňují kreativitu uživatelů při práci s daty (analýzy, prezentace)
Aplikace jsou v podstatě statické (požadavky na funkčnost aplikace jsou poměrně stálé)	Aplikace jsou dynamické (požadavky na funkčnost aplikací se mění)
Podporují každodenní firemní aktivity	Podporují dlouhodobé strategie firmy
Orientované na výkonnost	Poskytují konkurenční výhodu
Proces implementace a využívání je poháněn technologií (tj. impulsem k inovaci systému je nové systémové prostředí, nová verze databáze atp.)	Proces implementace a využívání je poháněn potřebami organizace (tj. impulsem k inovaci systému jsou nové potřeby uživatelů)
Technologické rozdíly	
Provozní systémy	**Datový sklad**
Zpracovávají velké objemy malých transakcí	Zpracovávají malý počet komplexních dotazů
Transakce neustále přidávají a aktualizují data	Data se načítají dávkově
Důležitým hlediskem je omezení redundance dat	Důležitým hlediskem je rychlý přístup k datům pro účely analýz a prezentací
Integrita dat se zajišťuje datovým modelem a aplikacemi	Integrita dat se zajišťuje při dávkových načítacích procesech (transformace dat)
Datové modely jsou optimalizované pro online aktualizace a rychlé zpracování transakcí	Datové modely jsou optimalizované pro rychlé zpracování výstupů
Používají se převážně normalizované relační datové modely	Používá se kombinace datových modelů (normalizované a denormalizované relační modely, sumarizované tabulky, star schema datové modely, snow flake datové modely, multidimenzionální datové modely)

Zdroj: [PVT]

2.1.2. ETL nástroje – nástroje pro plnění datového skladu

Do datového skladu se data nezadávají, ale načítají se z provozních systémů. Načítání se většinou provádí v čase, kdy nejsou provozní systémy příliš zatíženy, aby se neprodlužovala doba odezvy pro uživatele těchto systémů.

Při plnění datového skladu je nutné realizovat tyto hlavní kroky:

- **E**xtrakce vstupních dat z provozních (transakčních) systémů
- **T**ransformace vstupních dat
- **L**oad (uložení) dat do datového skladu

Extrakce vstupních dat

Nástroje datového skladu musí umožňovat extrakci dat z provozních systémů, což ve většině případů znamená komunikovat určitým způsobem (ODBC, nativní drivery, textové soubory) s relační nebo síťovou databází či případně systémem souborů.

Transformace vstupních dat

Data v datovém skladu jsou uložena jiným způsobem než ve zdrojových systémech. Provozní systémy používají v naprosté většině normalizovaný entitně relační datový model. U datového skladu se naproti tomu používá kombinace několika datových modelů (schéma hvězda, schéma sněhová vločka, normalizovaný entitně relační model, denormalizovaný entitně relační model, multidimenzionální datový model).

Transformace dat se skládá z těchto dílčích operací:

- Validace – ověření správnosti dat

- Čištění – odstranění či změna nesprávných dat
- Integrace – dosažení konzistence dat pocházejících z různých systémů (datové typy, formáty..)
- Derivace – vytvoření derivovaných dat na základě vstupních dat
- Denormalizace – snížení potřeby spojování tabulek při využívání datového skladu
- Sumarizace – vytvoření požadovaných souhrnů z detailních dat

Prostředí datového skladu musí poskytovat dostatek nástrojů a metod pro zvládnutí všech kroků. Vzhledem k povaze těchto procesů nepostačuje pouze SQL jazyk, ale je nutné používat specializované prostředky, jako jsou například vyšší programovací jazyky.

Velmi důležitou fází je validace a čištění, protože základní podmínkou využitelnosti datového skladu je to, že obsahuje důvěryhodná a správná data. Jakkoliv sofistikované metody na využití dat jsou v podstatě bezcenné, pokud nepracují se správnými daty.

2.1.3. OLAP analýza

Prověřenou a nejčastěji používanou metodou je OLAP analýza (On-Line Analytic Processing), která je podrobněji popsána v této části a jejíž způsoby řešení jsou nastíněny v části 4.2.3. OLAP analýza – MS SQL Server 2000 Analysis Services.

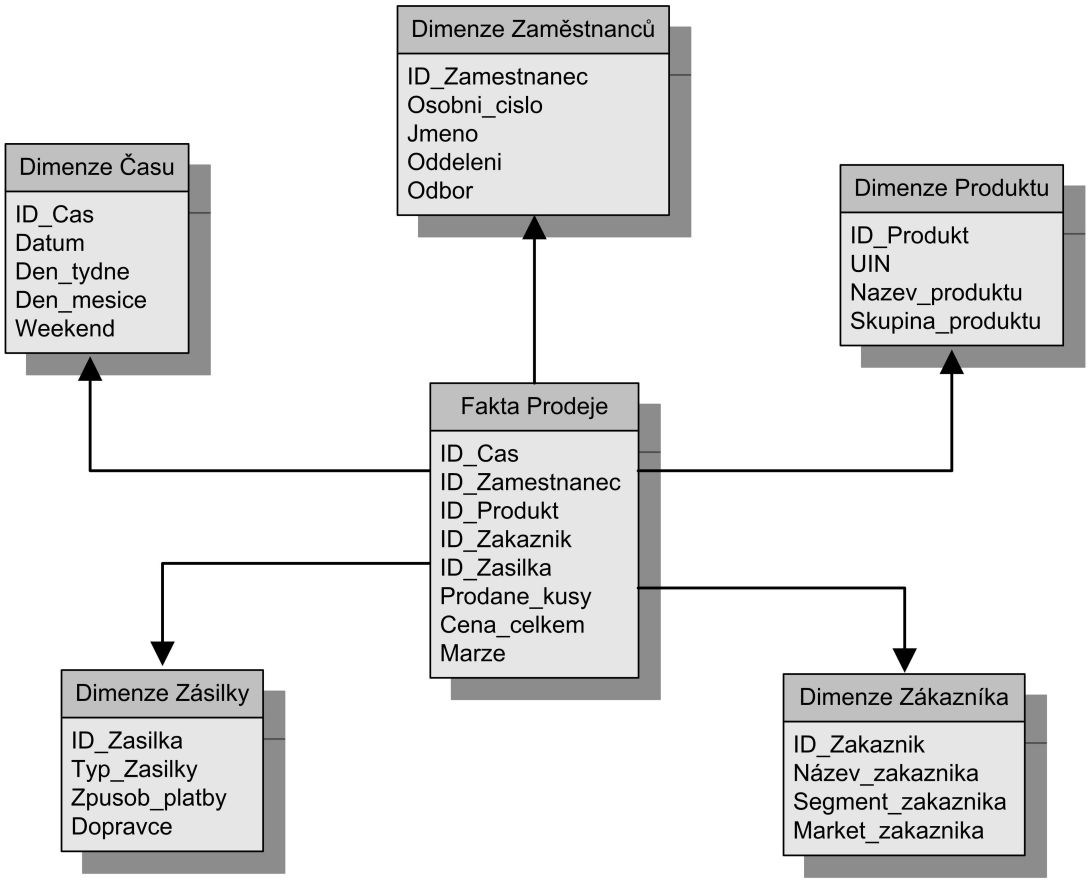
2.1.4. Datová kostka

OLAP analýza nejčastěji pracuje nad datovými kostkami, což jsou jakési multidimenzionální struktury vytvořené právě nad datovým skladem. Základem datových kostek jsou dimenze a fakty.

2.1.5. Star schema

Oproti transakčním systémům využívá datový sklad většinou jiný datový model úložiště pro OLAP analýzu. Schéma jedné centrální tabulky faktů s několika dimenzionálními tabulkami spojenými cizími klíči, které tvoří primární klíč centrální tabulky faktů, se díky svému tvaru říká *Star schéma (Hvězdicové schéma)*.

Obrázek č.2 – Star schema



Zdroj: [vlastní]

Na obrázku je znázorněno Star schema zachycující tržby a počty prodaných kusů, které má jednu tabulku faktů a pět tabulek dimenzí. V tomto příkladu jsou dimenzionální tabulky velmi zjednodušeny, v praxi většinou mají bohatou strukturu, vlastní hierarchii s několika úrovněmi (level). Hierarchií může být v rámci jedné dimenze i více (například u časové dimenze nepřekvapí hierarchie kalendářního a fiskálního roku. Hierarchie umožňují provádět výše uvedené techniky drillování, řezání, atd.

Obecně platným pravidlem je, že dimenzionální tabulky jsou záměrně denormalizované. Primárním hlediskem při analýzách je totiž rychlost vybavení dotazu, která se stupněm normalizace klesá. V případě komplexních dotazů do datových kostek, s kterými se počítá, ustupuje požadavek na normalizaci do pozadí. Ani argument úspory místa normalizované struktury neobstojí – dimenzionální tabulky tvoří zlomek celkového objemu dat. Případná úspora, řádově v MB, je naprosto zanedbatelná při srovnání s objemem dat tabulek faktů, které se zpravidla pohybují v řádech GB až TB.

Přesto je samozřejmě možné dimenzionální tabulky normalizovat a vyjádřit tak explicitně hierarchie v dimenzích.

Dimenze

Dimenze představují hrany pomyslné krychle a zobrazují úhly pohledu, pod kterými se můžeme na analyzovaná data dívat, jako je například organizační struktura společnosti, čas, účetní osnova nebo nákladová střediska.

Fakty

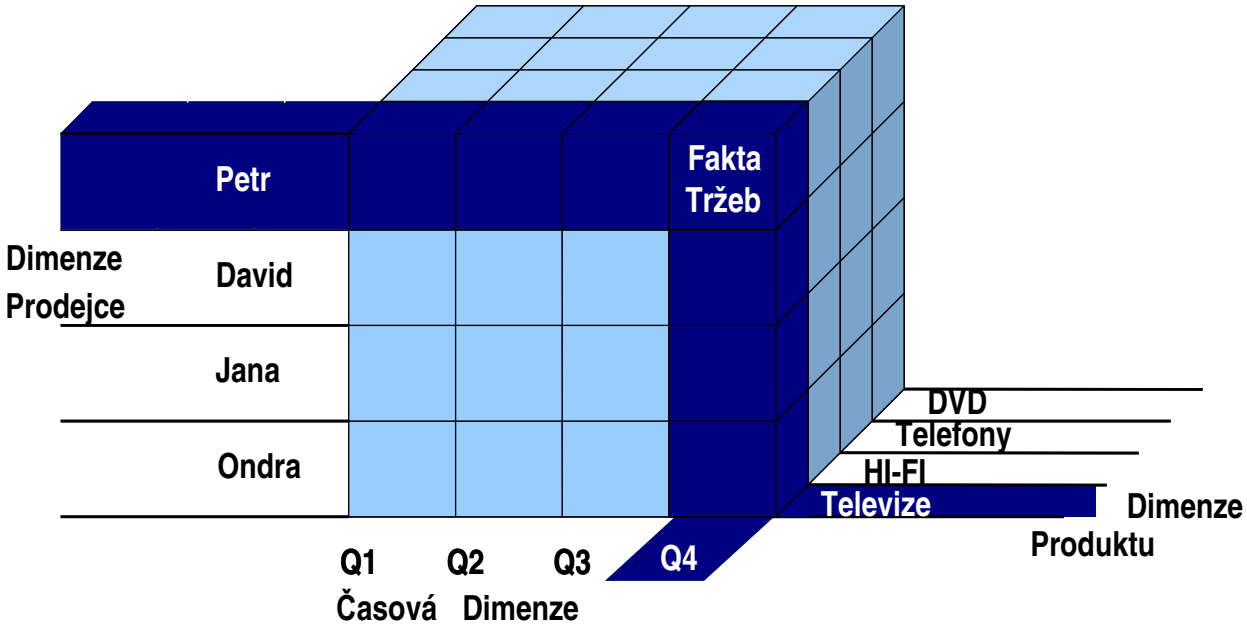
Fakty lze definovat jako ukazatele, pomocí nichž lze měřit výkony vzhledem k dimenzím. Mohou to být například koruny, různé počty, váha, atd., které chceme přes dimenze sledovat. Konkrétní hodnota faktu leží vždy na pomyslném průsečíku příslušných dimenzí.

Práce s datovou kostkou

Díky své struktuře lze s datovou kostkou provádět mnoho zajímavých věcí. Data je možné třídit, filtrovat, dělit do skupin, tyto skupiny mezi sebou porovnávat. Dimenze lze mezi sebou měnit, řadit za sebe a vzájemně kombinovat. Současně se lze do dimenze vnořovat – drill-down a vynořovat se z ní – drill-up. Dokonce je možné přecházet mezi

různými krychlemi, přičemž zůstává zachován filtr, který jsme si nastavili – tzv. drill-accross. Na následujícím obrázku je znázorněno, jakým způsobem se lze z datové kostky dozvědět výši tržeb (sumu všech jednotlivých transakcí) prodaných televizí prodejcem Petrem ve 4.kvartálu.

Obrázek č.3 – Datová kostka

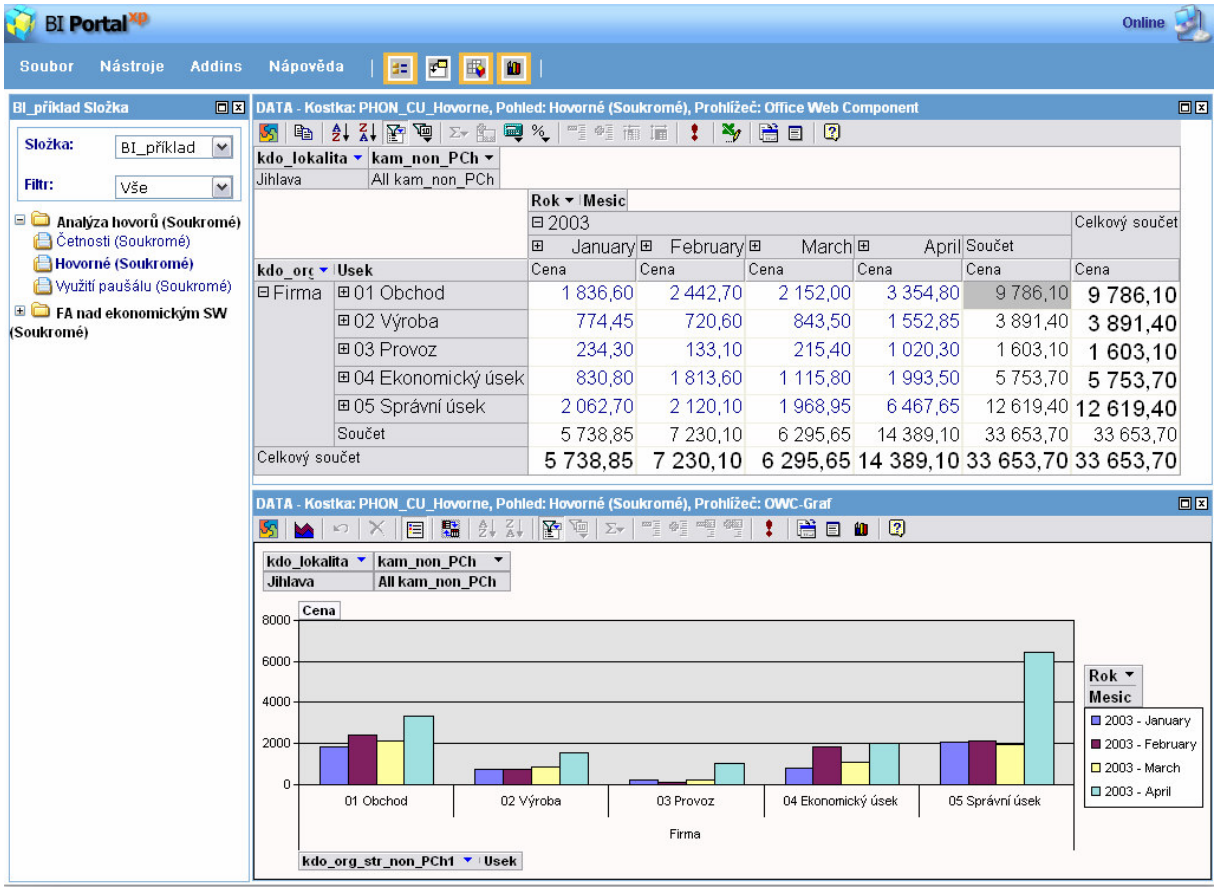


Zdroj: [MS1]

Znázorněná kostka je pouze třídemenzionální, nicméně v definici bylo uvedeno, že se jedná o multidimenzionální strukturu s teoreticky neomezeným počtem dimenzí. Jelikož průměrný člověk z reálného světa běžně uvažuje maximálně ve třech dimenzích, je multidimenzionalita datové kostky transformována do dvou, maximálně tří rozměrů. Třídimenzionální znázornění využívá zobrazovací komponenty založené na virtuální realitě a poskytuje velice zajímavé výstupy analýz, přesto je nejrozšířenějším zobrazením výstupů dvojrozměrná tabulka. Další dimenze se buď včleňují do jednoho ze dvou rozměrů – jedna dimeze ja zařazena za druhou a člení ji na menší části, nebo se další dimenze používají jako filtr, který z datové kostky poskytuje pouze řez. Představme si totiž, že by

v obrázku č.3 byla pro každý průsečík všech tří dimenzí zobrazená konkrétní hodnota. Zobrazení by bylo velice nepřehledné. Proto by se v praxi zřejmě jedna z dimenzí použila jako filtr a zbylé dvě dimenze by se použily jako sloupce a řádky dvojrozměrné tabulky. Příklad transformace více dimenzí do dvourozměrného prostoru je znázorněn na obrázku č.4, který ukazuje zobrazovací komponentu Microsoftu. Jedná se OWC (Office Web Components), což je vlastně kontingenční tabula aplikace MS Excel umístěná na webovou stránku.

Obrázek č.4 – Datová krychle zobrazená v OWC komponentě



Zdroj: [PVT]

Obrovskou výhodou využití počítače pro výstupy prováděných analýz je také možnost zobrazit data ve formě přehledného grafu, který je pro rychlý přehled mnohem výhodnější.

3. Návrh obecného řešení problému

3.1. Definice pojmu Pomalu se měnící dimenze – Slowly changing dimensions

Jak již bylo řečeno, výstupy datového skladu se nejvíce zaměřují na reportování a analyzování faktů neboli metrik. Fakty jsou však relevantními daty pouze pokud jsou definovány v kontextu ke správným dimenzím. Dimenze a jejich atributy jsou relativně stálé, přesto však v průběhu času dochází k jejich změnám. Může se měnit adresa jednotlivce, přicházejí a odcházejí zaměstnanci, společnosti představují nové produkty a vyřazují starý sortiment, atp. Správa těchto změn v čase je jedním z nejošidnějších aspektů navrhování datového skladu.

Pojmem pomalu se měnící dimenze (*slowly changing dimensions – SCD*) tedy rozumíme obměnu dimenzionálních atributů v čase. Termín pomalu se může zdát v některých případech zavádějící (například dimenze obchodníků nebo sortimentu se v případě reorganizace podniku mění relativně rychle), při srovnání s proměnlivostí metrik v tabulkách faktů se však dimenzionální data mění velmi pomalu.

3.1. Obecné principy řešení

V současné teorii se hovoří o čtyřech základních typech řešení tohoto problému – Typ 1 – přepsání historie, Typ 2 – zachování historie, Typ 3 – zachování verze historie. Nejvhodnější způsob řešení závisí především na požadavcích kladených na nutnost zachování historie a očekávanou frekvenci změn.

3.1.1. Typ 1 – přepsání historie

V případě změny atributu dimenze je původní hodnota nahrazena novou aktuální hodnotou. Všechny zaznamenané hodnoty tak odpovídají poslednímu stavu. Výhodou

tohoto řešení je jeho rychlost a jednoduchost. Změna se provádí pouze v příslušné tabulce dimenze, tabulka faktů zůstává nedotčena. Kritickou slabinou tohoto přístupu je ztráta historických vazeb, ztráta historie změn. Současně je v případě změn nutné provést kompletní „přepočítání“ (rebuild) datové kostky, neboť agregovaná data musí vycházet z atomických záznamů, které se po změně váží na jiné místo v dimenzi.

3.1.2. Typ 2 – zachování historie

Jak již bylo řečeno, jestliže nastávají změny v dimenzích (změny v pozici v hierarchii či změny ve vlastnostech), měly by být existující transakce (fakty) spojovány se starými hodnotami, to je s těmi, které existovaly před změnou. Jen nové transakce by měly být spojovány s novou formou člena. U Typu 2 pomalu se měnících dimenzí [KIM, str.185], je tato historická informace o členovi udržována. Když se člen mění, záznam v dimenzi není přímo aktualizován. Místo toho je vytvořen pro téhož člena dimenze nový záznam. Novému členu je přidělen i nový primární klíč v dimenzionální tabulce - od toho okamžiku se na všechny nové transakce tohoto člena používá nový primární klíč, který se označuje jako umělý, odvozený klíč (surrogate key). Přirozený klíč (natural key) však musí zůstat zachován v původní podobě. Současně je nový záznam označen aktuálním časovým razítkem. Použitím dvou nezávislých klíčů je zajištěna možnost tvorby rozdílných dotazů v závislosti na požadavcích analýzy. Umělý klíč fragmentuje data dle časových razítek, přirozený klíč naopak spojí v případě potřeby fragmentovaná data opět dohromady .

Největší výhodou je komplexnost tohoto přístupu a možnost inkrementálně znovuvytvářet již připravené dimenze a datové kostky, nevýhodou je pak nárůst objemu databáze a problém fragmentace dat – při změnách jsou data měněných členů ve faktových tabulkách od sebe oddělena, jejich spojení je možné pouze v případě zachování přirozeného klíče.

3.1.3. Typ 3 – zachování verze historie

Další možností zachycení historie je přidání dalšího atributu k měněnému členu dimenze, který bude uchovávat stav před změnou. Tento způsob je použitelný v případě, že požadavky na změnu jsou pouze omezené (např. zachycení rodinného stavu zákazníka). Z principu je však jasné, že tento způsob nezachytí kompletní historii při vícenásobné změně. I proto není tento způsob doporučován. Je však možností v případech, kdy není možné nebo vhodné řešit problém pomalu se měnících dimenzí pomocí Typu 2.

3.1.4. Hybridní způsob řešení

Hybridní způsob řešení kombinuje základní, výše popsané techniky ošetření pomalu se měnících dimenzí. V zásadě je možné hybridní způsob rozlišit na dva přístupu:

- předvídatelné změny s více verzemi překrytí – vychází z Typu 3 a dle počtu předvídatelných změn přidává sloupce k držení historických hodnot. Nejčastějším využití nachází při reorganizacích (prodeje, výroby, atd.)
- nepředvídatelné změny s jedinou verzí překrytí – používá se kombinace všech základních typů, tabulka dimenze obsahuje jak dodatečné sloupce pro řešení Typu 2, ale i další sloupce pro řešení Typu 3. Tento způsob bývá někdy označován jako Typ 6.

3.3. Praxe

Protože většina obchodních procesů vyžaduje flexibilně uchovávat údaje o historii, je nejvíce používaným řešením pomalu se měnících dimenzí Typ 2 – zachování historie, případně hybridní způsoby vycházející především z tohoto způsobu řešení.

4. Vlastní řešení

4.1. Úroveň nasazení řešení

Řešení problematiky Pomalu se měnících dimenzí je v podstatě možné implementovat na jakékoli úrovni – již v produkčním systému, v datovém skladu nebo až v nástrojích analýzy. Zpravidla se Pomalu se měnící dimenze řeší od vrstvy datového skladu výše, neboť datový sklad může sloužit jako zdroj dat nejen pro analytické nástroje, ale i pro další potřeby, jako jsou například dotazy z HTML stránek, poskytování dat dalším aplikacím, apod. Příkladem implemtace Pomalu se měnících dimenzí od úrovně datového skladu se budu zabývat v následujících kapitolách.

4.2. Použitá technologie společnosti Microsoft

Vlastní řešení měnících se dimenzí je demonstrováno na produktech společnosti Microsoft. Hlavním důvodem pro volbu Microsoftu je fakt, že drtivá většina řešení BI, na kterých jsem se podílel byla vyvíjena na produktech této společnosti. Dalším důvodem je pozice Microsoftu na trhu BI – dle konzultanské společnosti Gartner, Inc [GAR] jde o nejdynamičtěji se vyvýjející platformu v posledních čtyřech letech.

V listopadu roku 2005 Microsoft uvedl na trh novou verzi produktu MS SQL Server 2005 (komplexní nástroj pro řešení BI), která tyto prognózy potvrzuje. Oproti minulé verzi přináší jednak centralizovaný vývoj a správu všech služeb v jednom integrovaném nástroji (MS Visual Studio 2005), dále pak mnohá vylepšení funkcionality a několik naprosto nových funkcí a vlastností. Za všechny bych jmenoval tzv. „Proactive cache“ analytických služeb – systém monitoruje určené úložiště primárních transakčních dat a v případě změny

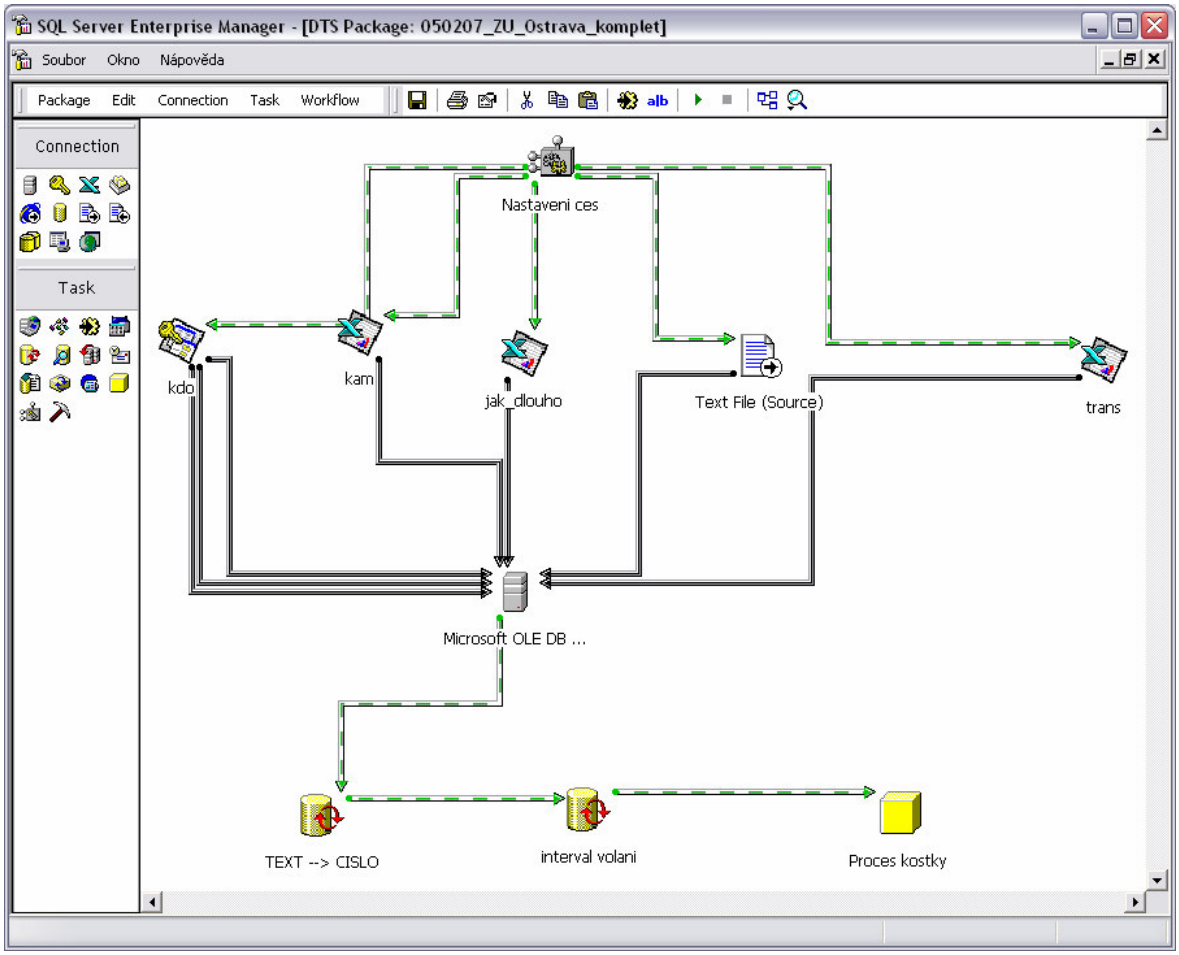
(přidání, smazání, modifikace) okamžitě tato změněná data načítá do vyrovnávací paměti. V případě dotazu do datové kostky, která využívá tuto funkci, je výsledek dotazu získaný z datové kostky v případě potřeby upraven na základě dat ve vyrovnávací paměti. Uživatel má tak k dispozici OLAP analýzu téměř aktuálních dat – tedy opravdový real-time OLAP. Druhou oblastí, která prošla spíše revolucí než evolucí, jsou nástroje pro vývoj a správu ETL procesů (v minulé verzi označeny Data Transformation Services, dnes Integrační služby (Integration Services)). Kromě již zmíněné integrace do jednotného vývojového prostředí, je nejviditelnější změnou logické oddělení workflow prováděných operací v rámci ETL procesu od definice vlastních úloh procesu, což velmi výrazně zjednodušuje a zpřehledňuje jeho vývoj a správu.

Protože však má diplomová práce vznikla jako výsledek dlouhodobé participace na různých projektech, je vlastní řešení měnících se dimenzí ukázáno v nástrojích starší verze tohoto produktu – v nástrojích Microsoft SQL Serveru 2000.

4.2.1. ETL vrstva – MS Data Transformation Services

Pro realizaci ETL procesů je využíván nástroj integrovaný do produktu MS SQL 2000 s označením Data Transformation Services. V něm datový "pumpař" vytváří jednotlivé "balíčky" (DTS packages). Každý balíček obsahuje definice připojení ke zdroji a k cíli, definice transformačních úloh a jejich pravidel (DTS tasks) a předpis toho, v jakém pořadí se mají jednotlivé úlohy provádět (workflow). Právě workflow je velmi užitečná vlastnost balíčků DTS, neboť umožňuje nastavit logiku datové transformace.

Obrázek č.5 – Konzole MS SQL Server 2000 (Data Transformation Services)



Zdroj: [Vlasntí]

Pro názornost je uveden příklad tvorby komplexního DTS balíčku, který řeší ETL procesy od počátečního importu dat ze zdrojových systémů a souborů, přes různé transformace a konsolidace těchto dat až po výsledné zprocesování OLAP kostky v datovém skladu.

Mezi hlavní výhody „Data Transformation Services“ patří: již zmiňované workflow; otevřená a rozšiřitelná architektura; možnost napojení na množství datových zdrojů a cílů (ODBC); programovatelnost a výkonnost.

4.2.2. Datový sklad - MS SQL Server 2000

Vlastní datový sklad je postaven na relačním databázovém produktu společnosti Microsoft – MS SQL Server 2000. Jedná se o jeden z nejrozšířenějších databázových serverů na trhu, který splňuje všechny požadavky, které jsou v současnosti na tento druh SW kladeny – jako jsou bezpečnost, robustnost, rychlost a spolehlivost.

4.2.3. OLAP analýza – MS SQL Server 2000 Analysis Services

Nástrojem, který zabezpečí realizaci OLAP analýzy, je MS SQL Server 2000 Analysis Services, což je analytická nadstavba SQL serveru, která umožňuje vytvářet a ukládat multidimenzionální datové kostky třemi způsoby:

- MOLAP (Multidimensional OLAP) – multidimenzionální úložiště s patentovanou technologií, v závislosti na požadovaném stupni agregace, požadované velikosti databáze a požadované rychlosti vybavení dotazu jsou agregované hodnoty předpočítány a uloženy přímo do multidimenzionální struktury, k dotazům se využívá jazyk MDX (MultiDimensional eXpression), tento způsob přináší nejvyšší rychlost odpovědí na dotazy při nejvyšších požadavcích na objem dat

- ROLAP (Relational OLAP) – data jsou ponechána v původních relačních tabulkách, případné předpočítané agregované hodnoty jsou uloženy do speciálních pomocných relačních tabulek, MDX dotazy jsou směrovány pomocí klasických SQL příkazů přímo do relačních tabulek, tento způsob je pomalý, není však tolik náročný na datový prostor. Určitého zvýšení rychlosti je možné dosáhnout použitím inteligentní cache, často kladené dotazy jsou vybavovány právě z ní

- HOLAP (Hybrid OLAP) – datová kostka se může skládat z několika oddílů (partition), v některých případech je vhodné, aby jeden oddíl byl typu MOLAP a druhý ROLAP. Oddíl MOLAP může například obsahovat data za poslední rok, do kterých jsou často kladeny dotazy (důležitá je rychlost), oddíl ROLAP pak obsahuje více historická data např. za posledních 10 let, která jsou používána řidčeji (důležitý je objem dat).

Zabezpečení přístupu k datům

Samostatnou kapitolu, o které je nutné se zmínit, tvoří zabezpečení přístupu k datům. Na úrovni analytického serveru, tzn. na úrovni konkrétních datových krychlí, lze vytvořit role, u kterých je možné nadefinovat, které fakty budou přístupné, které dimenze, od a do jaké hloubky budou viditelné, jakým způsobem se mají zobrazovat agregované hodnoty při omezeních. Do vytvořených rolí se poté přiřazují konkrétní uživatelé, přičemž je nutné vyzdvihnout možnost použití integrovaného ověřování v rámci domén systému MS Windows. Uživatel je ověřován na základě svého primárního přihlášení k podnikové doméně.

4.3. Pomalu se měnící dimenze

Jak bylo uvedeno v části 4.1. Úroveň nasazení řešení, úroveň nasazení řešení pomalu se měnících dimenzí byla stanovena od vrstvy datového skladu. Nasazení procesů, které zajistí ošetření problémů týkajících se pomalu se měnících dimenzí, by proto mělo pokrývat všechna patra řešení BI, která se dají využít jako zdroj dat – tedy datového skladu a OLAP datových kostek.

Pro uchování informace, kdy a k jakým změnám dimenzionálních dat došlo, je na úrovni datového skladu nutné upravit níže popsaným způsobem ETL procesy. A to nejen jejich strukturu, ale i pořadí, v jakém mají být jednotlivé kroky procesů provedeny.

Protože transformace, které se provádí v rámci ETL procesů pro zachycení SCD, musí mít vazbu na řešení SCD o patro výš – tedy na OLAP datovou kostku, začnu s praktickou ukázkou právě na úrovni analytického serveru, neboť v části věnované modifikaci ETL procesů budou znovu použity některé atributy vysvětlené v části týkající se datových kostek, ale již chápány v kontextu vyšší vrstvy řešení.

4.3.1. Vrstva analytického serveru - OLAP datových kostek

Příklad řešení problémů pomalu se měnících dimenzí vychází z [MS2] a je znázorněn v nástroji Microsoft SQL Server 2000 Analysis Services.

Předpokládejme obchodníka, který v čase přechází z jednoho útvaru do druhého a my chceme zhodnotit jeho osobní každoroční prodej. Jeho transakční historie je rozdělená do dvou či více útvarů. Jedná se o typický příklad, kde je vhodné nasadit řešení založené na Typu 2 – zachování historie.

Tabulka PRODEJ obsahuje jednotlivé případy prodejů. Tabulka obsahuje mimo jiné vazbu na prodejce přes sloupec ID_Obchodnik do tabulky OBCHODNIK.

Obrázek č.6 – Řešení SCD

2:Data in Table 'PRODEJ' in 'SCD' on 'BRNW06'

id_prodej	id_zbozi	id_obchodnik	Dat_prodeje	Mnozstvi	Cena
1	122	1	1/10/2003	20	1500
2	145	1	2/10/2003	50	2500
3	555	2	2/10/2003	10	2200
4	122	3	3/3/2003	40	1500
5	122	3	6/20/2003	10	1500
6	122	6	9/15/2003	100	1500
7	555	2	9/15/2003	25	2200

Zdroj: [PVT]

Tabulka OBCHODNIK obsahuje jednoznačný identifikátor ID_Obchodnik – umělý (surrogate) klíč, který je spojen s položkou ID_Obchodnik tabulky PRODEJ. Dále musí mít každý obchodník jednoznačný vnitřní identifikátor, který se během životního cyklu obchodníka u organizace nebude měnit – přirozený klíč (natural key). V příkladu bylo jako identifikátor zvoleno osobní číslo pracovníka. (Pokud by se předpokládala jeho změna, mohlo by to být číslo zjištěné např. jako Select MAX(Os_cis)+1). Dále tabulka obsahuje časové razítko platnosti OD_data a DO_data a položku Stav.

Obrázek č.7 – Řešení SCD

ID_Obchodnik	Os_cis	Od_data	Do_data	Prijmeni	Jmeno	ID_Utvar	Stav	Seznam
1	1669	1/1/2003	5/31/2003	Vachoušek	Vilém	1	0	$Moved
2	2001	1/1/2003	12/31/2020	Marounek	Matěj	1	1	$Original
3	1444	1/1/2003	12/31/2020	Veselá	Milena	1	1	$Original
4	1333	1/1/2003	12/31/2020	Boháč	Filip	2	1	$Original
5	1669	6/1/2003	8/31/2003	Vachoušek	Vilém	2	0	$Moved
6	1669	9/1/2003	12/31/2020	Vachoušek	Vilém	3	2	{[Obchodník].[Prijmeni].&[1],[Obchodník].[Prijmeni].&[5],[Obchodník].[Prijmeni].&[6]}

Zdroj: [PVT]

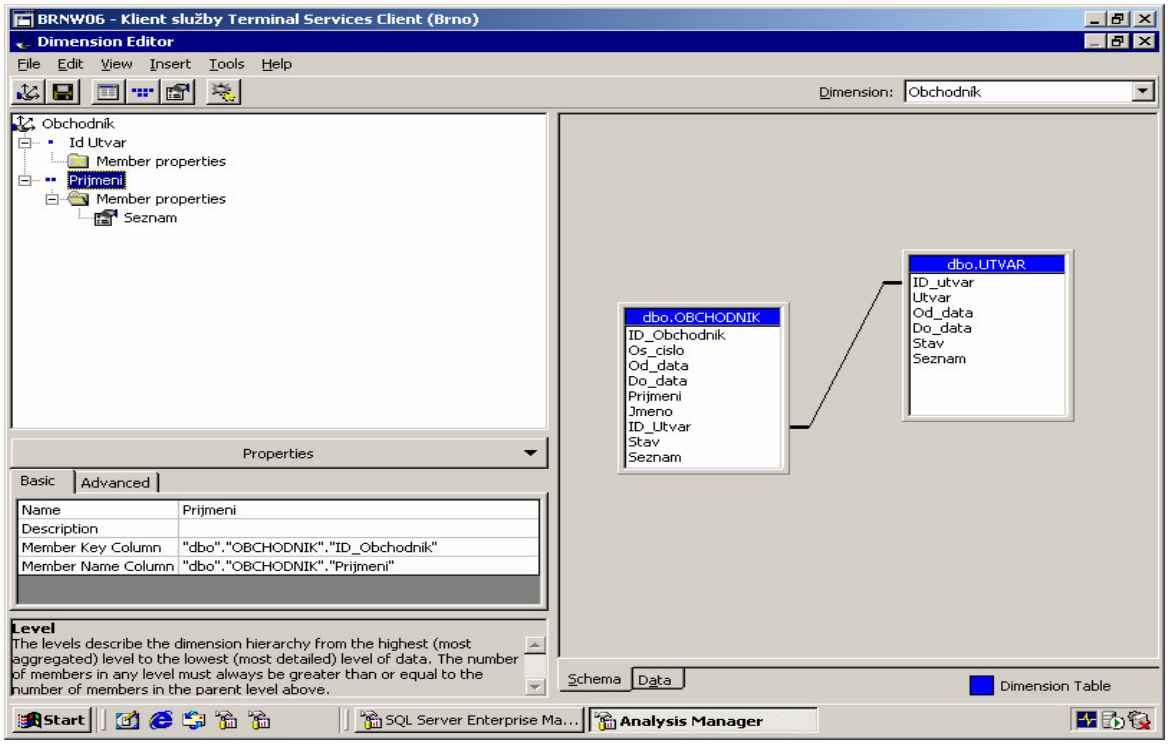
Při vložení nového obchodníka se naplní položka Stav hodnotou 1, Od_Data se naplní aktuálním časem a položka Do_Data vysokým datem např. rokem 2099. Tzn. u všech obchodníků, u kterých nedošlo k žádné změně, je položka Stav rovna hodnotě 1. Pokud dojde k libovolné změně v tomto číselníku, neprovede se aktualizace přímo u příslušného záznamu, ale vygeneruje se nový záznam se změněným atributem (Jméno, útvar, atd). U původního záznamu se zaktualizuje pouze položka Stav na hodnotu 0 a položka Do_data se změní na aktuální čas. U nového záznamu se položka Stav naplní na hodnotu 2 a položka OD_data aktuálním časem.

Z toho vyplývá, že aktuální obchodníky získáme např. jako SELECT * FROM OBCHODNIK WHERE Stav>0. Historii konkrétního obchodníka získáme např. jako SELECT * FROM OBCHODNIK WHERE Os_cislo=1669 ORDER BY OD_Data. Přitom zůstanou zachovány časové vazby do tabulky faktů (PRODEJ).

Pro řešení problému SCDs v datových skladech je vhodné podkladovou tabulku pro dimenzi Obchodník doplnit o další sloupec – např. SEZNAM – typu nvarchar. V tomto sloupci bude u obchodníka, u kterého dosud nenastaly žádné změny, hodnota $Original, u

obchodníka, který byl změněn, hodnota $Moved a u aktuálně platného obchodníka seznam všech předchozích členů. Tzn. že na úrovni datové pumpy je třeba vytvořit textový řetězec s touto informací. Dimenze Obchodník bude pak mít následující tvar:

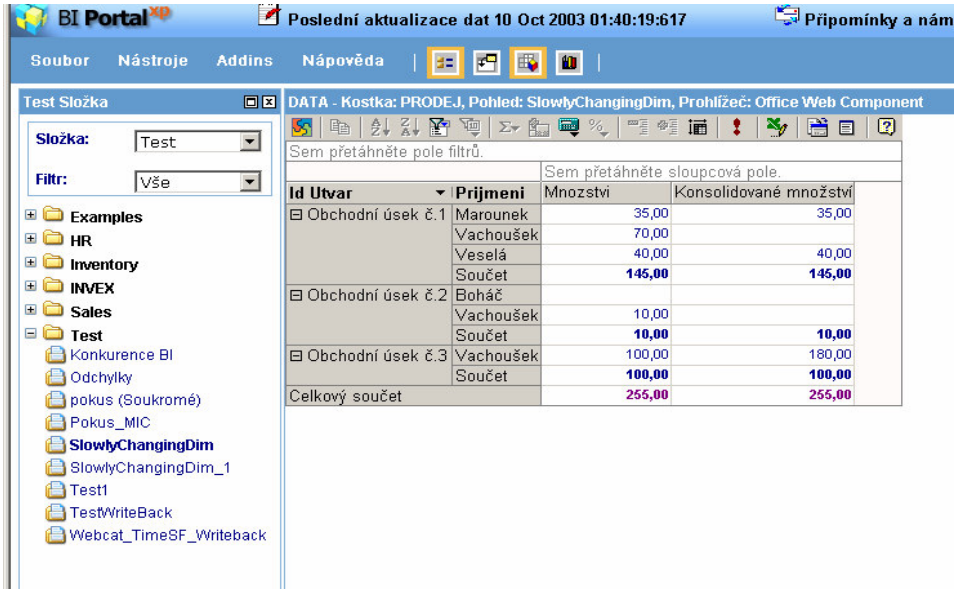
Obrázek č.8 – Řešení SCD



Zdroj: [PVT]

Number Key Column u úrovně Prijmeni musí být číselné pole. Member Properties u úrovně Prijmeni ukazuje na pole SEZNAM tabulky OBCHODNIK. Pohled do datové kostky ukazuje prodeje obchodníka Vachouška při jeho začlenění do jednotlivých útvarů.

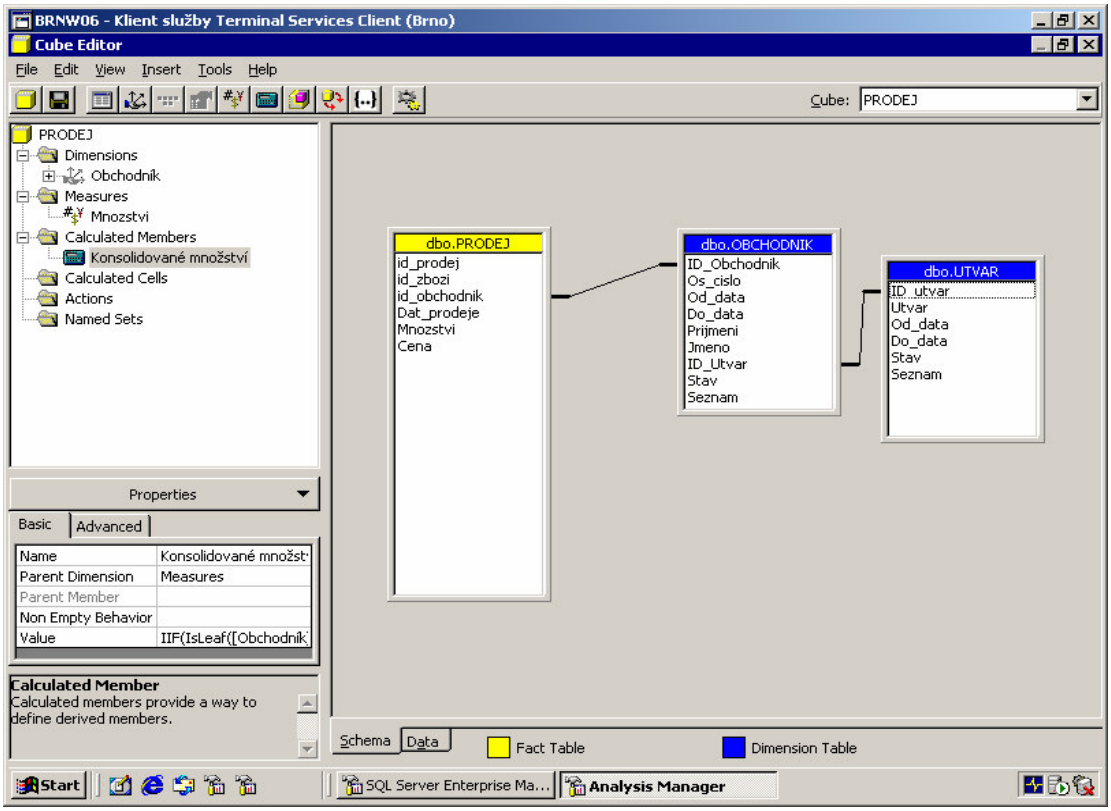
Obrázek č.9 – Řešení SCD



Zdroj: [PVT]

Pokud požadujeme vyčíslit realizované prodeje za obchodníka, bez ohledu na útvar v kterém v průběhu časového období působil, vytvoříme buď novou dimenzi Obchodníka bez útvarů, nebo pokud chceme použít již vytvořenou dimenzi Obchodnik, musíme problém vyřešit pomocí tzv. Calculated Members (on-line počítaná metrika podle předem definovaného vzorce).

Obrázek č.10 – Řešení SCD



Zdroj: [PVT]

Konsolidované množství je Calculated Member s hodnotou počítanou podle vzorce:

```
IIF(IsLeaf([Obchodník].currentmember),
IIF([Obchodník].currentmember.properties("Seznam")="$Original",[Measures].[Mnozstvi],
IIF([Obchodník].currentmember.properties("Seznam")="$Moved",NULL,
SUM(StrToSet([Obchodník].currentmember.properties("Seznam")),
[Measures].[Mnozstvi]))),[Measures].[Mnozstvi])
```

Pole SEZNAM jehož obsah je plněn do Member Properties dimenze Obchodník obsahuje seznam všech fragmentů obchodníka, což umožňuje rychlým a efektivním způsobem zobrazit kumulované hodnoty za konkrétního obchodníka, i když jsou jeho prodeje fragmentovány do více větví. Naopak výpočet není prováděn pro přesunuté členy, což výrazně zrychlí dotaz do datové kostky, protože v případě použití Calculated Member jsou i agregovaná data on-line počítána při každém dotazu.

Zachycení změn dimenzionálních změn v čase pro OLAP analýzu není triviální záležitostí, v literatuře ani na internetu jsem žádné konkrétní obecně použitelné řešení neobjevil. Výše popsané řešení je pevně spojeno s použitou technologií Microsoftu a je

vhodné pouze pro takové případy, kdy je měnících se dimenzí málo, neboť pro každou měnící se dimenzi je nutné duplikovat všechny fakya, resp. vytvářet nové kalkulované součty. Proto řešení této úrovně není tímto způsobem uspokojivě řešeno v obecněji chápané rovině.

V praxi se osvědčil jiný přístup, který však také není dle mého názoru ideální. V případě, že je nutné sledovat změny dimenzionálních dat, je vytvořena speciální datová kostka, která má ve faktové tabulce nastavenou vazbu na dimenze ze sloupce umělého klíče. Uživatel tak má pohromadě všechny měnící se dimenze bez nutnosti duplikovat fakty. Pokud naopak potřebuje zjistit údaje členů dimenze bez ohledu na změny v jejich postavení ve struktuře dimenze, je vytvořená druhá kostka, kde v tabulce faktů jsou vazby na dimenze provedeny přes sloupce klíče přirozeného. Tento způsob je z praktických zkušeností uživateli mnohem lépe přijímán, neboť je poměrně přímočarý. Mé výhrady však směřují k tomu, že nemám možnost měnit SCD úhel pohledu na stejná data v jednom okamžiku, že nemám obě hodnoty vedle sebe.

4.3.2. Vrstva datového skladu

K tomu, aby mohlo být ošetření Pomalu se měnících dimenzí implementováno na úrovni OLAP datových kostek, je předtím nutné zajistit zachycení změn dimenzionálních dat v čase v nižší vrstvě, která je plněná daty z produkčních systémů, a která slouží jako relační úložiště a zdroj dat ro datové kostky – v datovém skladu.

Pro hladký průběh načítání dat z produkčních systémů do datového skladu, se v praxi osvědčilo vytvořit tzv. „Data Staging Areas“ (dočasné úložiště), v praxi též označováno jako 0.vrstva datového skladu. Jedná se o oddělený datový prostor, ve kterém se budou provádět transformace. Do tohoto prostoru budou přímo načítána data ze zdroje. Po úspěšném dokončení všech transformací budou transformovaná data načtena do vyšší – 1.vrstvy datového skladu. Fyzicky se jedná o separátní databázi na stejném databázovém serveru.

ETL procesy musí zahrnovat nejprve zachycení změn dimenzionálních dat a poté naplnění faktových tabulek novými daty s vazbami na aktuální tabulky dimenzí. Na tomto místě je nutné poznamenat, že v organizaci by měl ke každé dimenzi existovat „správce“, zodpovědný za její aktuálnost. Pouze on má pravomoc k tvorbě změn zdrojových dat popisujících dimenze, jejich validaci a zveřejnění v rámci vyšších vrstev řešení.

ETL dimenzionálních dat

Princip ošetření problémů pomalu se měnících dimenzí a posloupnost jednotlivých kroků, které je nutné provést v rámci ETL dimenzionálních dat, je následující:

- v prvním kroku musí být do „Data Staging Area“ (0.vrstva datového skladu) načtena data z operativních zdrojových systémů
- pokud v dimenzi používáme umělý (surrogate) klíč, musíme uchovávat persistentní *hlavní tabulku s křížovým odkazem* (master cross-reference table), která zachycuje přiřazení umělého klíče ke klíči užitému ve zdrojovém systému v daném čase (příklad takové tabulky je uveden na Obrázku č.11).

Obrázek č.11 – Hlavní tabulka s křížovým odkazem

Hlavní tabulka s křížovým odkazem
Umělý klíč dimenze Klíč operativního zdroje Atribut dimenze 1...N Datum Efektivnosti řádku Datum Expirace řádku Poslední indikátor řádku dimenze Poslední Součet cyklické redundance (CRC)

Zdroj: [KIM]

- Jak je patrné z Obrázku č.11, jsou v dalším kroku porovnávány řádky získané ze zdrojového systému s hlavní tabulkou s křížovým odkazem. K rychlému určení, zda byl řádek ve zdrojovém systému změněn, je doporučováno [KIM, str.360] využít algoritmu *součtu cyklické redundance* (CRC). V komentáři ke

kódu příkladu bude vysvětlena implementace a rozdíl mezi řešením s a bez použití CRC.

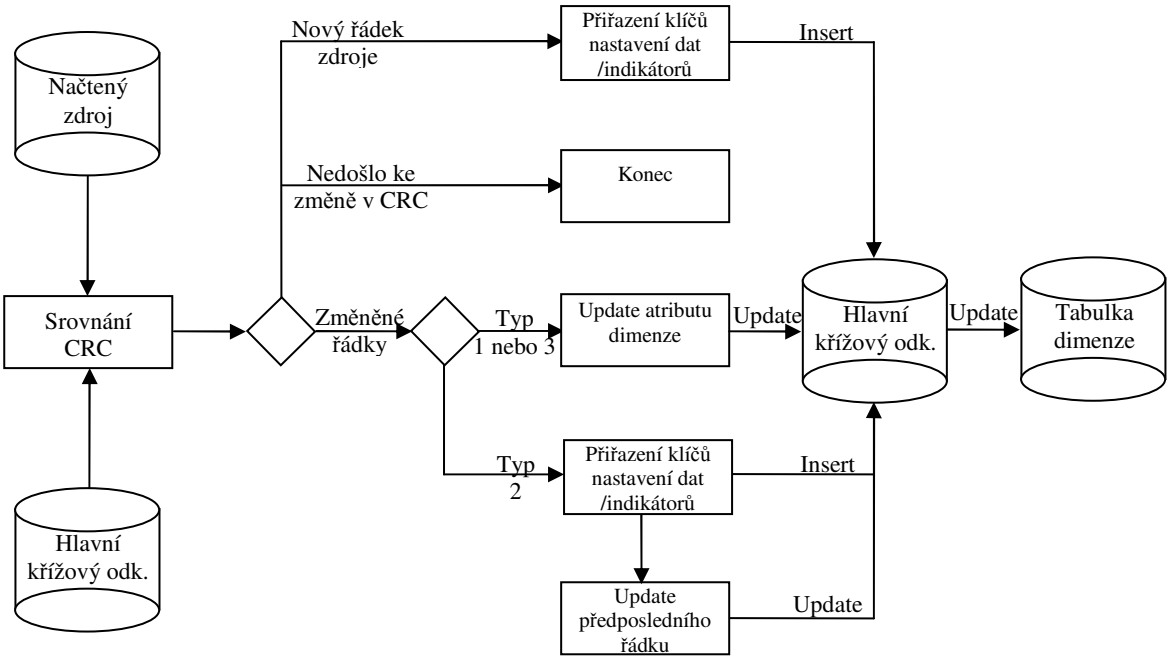
  - pokud je zdrojový řádek nový, není nalezen jeho odkaz v hlavní tabulce s křížovým odkazem, ETL proces mu přiřadí nový umělý klíč a přidá nový řádek do hlavní tabulky s křížovým odkazem

  - pokud je CRC identický pro načtený záznam a pro aktuálně platný řádek v hlavní tabulce s křížovým odkazem, nedošlo ke změně a my můžeme načtený řádek ignorovat

  - pokud se CRC pro načtený záznam liší od CRC aktuálně platného řádku v hlavní tabulce s křížovým odkazem, musíme zjistit, který sloupce načteného řádku se změnil, přiřadíme načtenému řádku nový umělý klíč, zneplatníme aktuálně platný řádek v tabulce s křížovým odkazem a zapíšeme nový řádek s novým umělým klíčem do tabulky s křížovým odkazem

- v posledním kroku je na základě hlavní tabulky s křížovým odkazem updatována příslušná tabulka dimenze.

Obrázek č.12 – Hlavní tabulka s křížovým odkazem



Zdroj: [KI1]

V některých případech není nutné používat pro porovnávání změn mezi daty ze zdrojových systémů a daty z datového skladu hlavní tabulku s křížovým odkazem, ale přímo použít tabulky dimezí datového skladu. Příklad takového řešení je uveden v druhém příkladu níže.

ETL faktů

ETL proces tabulek faktů by měl probíhat v následujících krocích:

- načtení dat faktů ze zdrojových systémů do „Data Staging Area“ a nastavení správné granularity
- transformace dat faktů, pokud je to třeba – aplikace součtových funkcí, konverzí dat, sjednocení měn nebo jednotek faktů, případné normalizace

- nahrazení klíčů ze zdrojových systémů umělými (surrogate) klíči a přidání nových umělých klíčů jako odkazů (cizích klíčů) na další dimenze – podle hlavní tabulky s křížovým odkazem
- ověření kvality dat faktů zpětným srovnáním s provozními systémy
- načtení transformovaných dat do datového skladu (přidání nových řádků do „ostrých“ tabulek faktů.

Příklad postupného řešení dimenzí

Prvním podrobně popsaným příkladem ošetření problémů pomalu se měnících dimenzí je znázorněn na části projektu z praxe, kde byla zpracovávána celá řada statistických dat, vzniklo tak Statistické datové tržiště. Dále bude popisována práce s číselníkem regionální dimeze, který obsahuje hierarchicky začleněné části obce do obcí, pověřených obecních úřadů (POU), obcí s rošířenou působností (ORP), okresů, krajů a oblastí. Primárním zdrojem dat je Český statistický úřad, který ho distribuuje v textovém souboru. Tato dimenze je velmi důležitá neboť se jedná o sdílenou dimezi nejen napříč datovým tržištěm, ale celým datovým skladem. Jelikož neexistuje vazba mezi POU a okresem a ORP a okresem, vzniká dimenze se dvěma hierarchiemi.

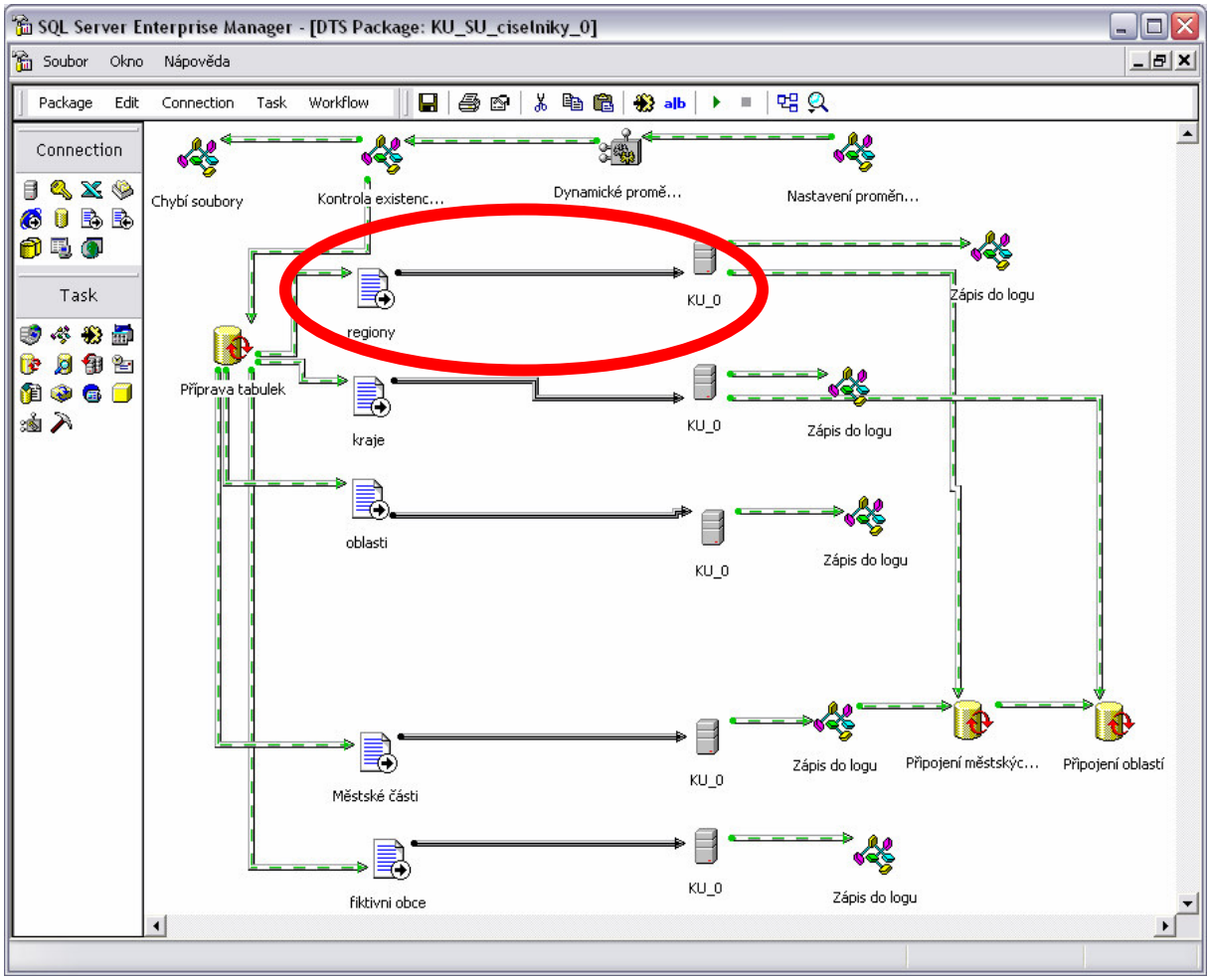
Postup řešení

Postup nahrávání těchto dat z ČSÚ až do datového skladu je následující:

- Akvizice textového souboru z ČSÚ mailem
- Načtení dat z textového souboru do 0.vrstvy DWH („Data Staging Area“) ETL procesy. Prioritou této fáze je převod z textového formátu do prostředí databázového serveru a rychlost, proto dochází k prostému kopírování („Bulk copy“) dat bez jakékoli transformace. Na Obrázku č.13 je znázorněna datová

pumpa, která toto nahrání zajistí. Načtení dat ze zdrojového textového souboru je zvýrazněna elipsou.

Obrázek č. 13 – Datová pumpa pro načtení dat z textového souboru



Zdroj: [PVT]

Jelikož je obsah dat regionálního číselníku dodávaných ČSÚ poněkud zvláštní (data jsou uváděna nejenom za konkrétní části obce, ale i za vyšší celky), bylo nutné upravit členy na nejnižší úrovni dimenze, musely být vytvořeny její nové fiktivní prvky. Jejich načítání probíhá v ostatních krocích zobrazeného transformačního balíčku.

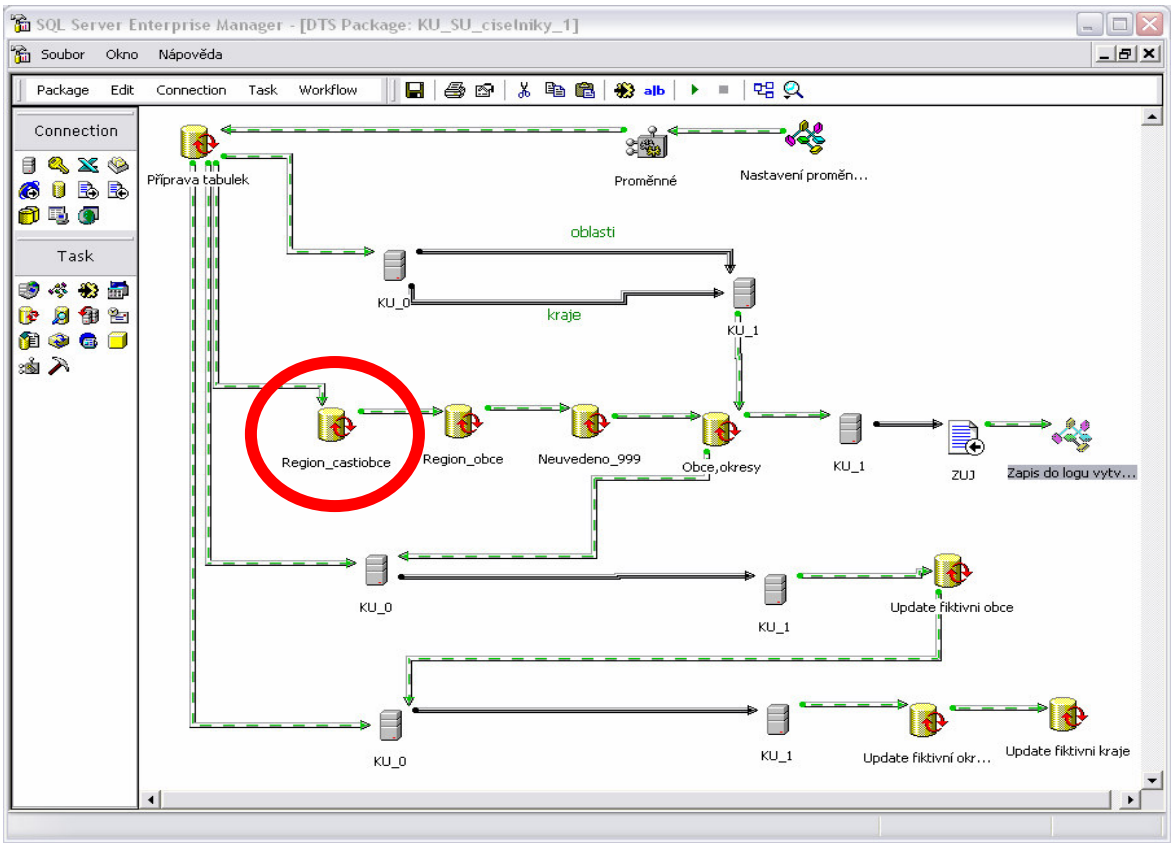
- V dalším kroku jsou data 0.vrstvy datového skladu načítána dalšími ETL procesy, transformována do podoby, která je výhodná pro efektivní získávání analytických výstupů. Následně jsou data uložena do 1.vrstvy datového skladu, která slouží jako relační úložiště pro datové kostky definované v analytickém

serveru. DTS balíček, který tento krok definuje je znázorněn na obrázku č.14, tento krok je podreběji popsán níže.

- Poté se načtou nová data faktů do tabulek faktů.

- Posledním krokem je update vazeb mezi tabulkami faktů a tabulkou regionální dimenze přes umělý klíč neboť tato vazba se používá k uchování historie. Ve zdrojových datech neexistovala, neboť vznikla až v průběhu transformace.

Obrázek č. 14 – Datová pumpa s krokem ošetřujícím SCD



Zdroj: [PVT]

Kruhem znázorněný krok transformace pojemnovaný „Region_castiobce“ definuje potřebné kontroly a případné transformace nutné k ošetření problémů pomalu se měnící regionální dimeze. V tomto případě se ke kontrolám aktuálnosti atributů prvku dimenze nepoužívá Hlavní tabulka s křížovým odkazem, ale nově načtená data jsou porovnávána přímo s daty v tabulce dimenze. Transformace také zajišťuje komplexní ošetření

případného update MDX formulí nutných pro řešení SCD na úrovni OLAP datových krychlí popsaném v části 4.3.1. Vrstva analytického serveru - OLAP datových kostek.

Na následujících stránkách je uveden kód transformace s komentářem k jednotlivým funkčním blokům. Komentář je uveden znaky „//“ Na úvod je nutné poznamenat, že kód je psán v jazyce Transact-SQL, což je implementace standardního jazyka SQL dle ANSI 92 společností Microsoft. To znamená, že standard SQL je rozšířen o další programové možnosti umožňující tvorbu pokročilých operací s daty. Pro porovnání nově načtených dat s daty dimenzí v datovém skladu byla použita tzv. „kurzorová tabulka“. Ta umožňuje načíst potřebná data a pak provádět požadované operace postupně s každým řádkem, což je přesně to, co je třeba pro požadované porovnání. Současně je v kódu znázorněno použití CRC součtu pro zjištění rozdílu v atributech v řádku nově načtených dat a dat v datovém skladu, jak bylo popsáno výše. To je zajištěno tak, že v tabulce dimenze je přidán speciální sloupec, jehož hodnota je pro každý řádek dána formulí BINARY_CHECKSUM (jmeno sloupce atributu 1, jmeno sloupce atributu 2,…). Tak je v tomto sloupci jedinečné číslo závislé na obsahu sledovaných atributů.

Krok „Region_castiobce“:

```
declare @Nazev_castiobce_new varchar(150)
declare @Kod_castiobce_new varchar(8)
declare @Nazev_obce_new varchar(150)
declare @Kod_obce_new varchar(8)
declare @Nazev_POU_new varchar(150)
declare @Kod_POU_new varchar(8)
declare @Nazev_ORP_new varchar(150)
declare @Kod_ORP_new varchar(8)
declare @Nazev_okresu_new varchar(150)
declare @Kod_okresu_new varchar(8)
declare @Nazev_kraje_new varchar(150)
declare @Kod_kraje_new varchar(8)
declare @Nazev_oblasti_new varchar(150)
declare @Kod_oblasti_new varchar(8)
declare @Datum_platnosti_new smalldatetime
```

// nejprve je nutné deklarovat používané proměnné, ty, které mají své jméno zakončené _new slouží k dočasnému uložení hodnot nově načtených dat 0.vrstvy datového skladu

```
declare @Nazev_castiobce_old varchar(150)
declare @Kod_castiobce_old varchar(8)
```

```
declare @Nazev_obce_old varchar(150)
declare @Kod_obce_old varchar(8)
declare @Nazev_POU_old varchar(150)
declare @Kod_POU_old varchar(8)
declare @Nazev_ORP_old varchar(150)
declare @Kod_ORP_old varchar(8)
declare @Nazev_okresu_old varchar(150)
declare @Kod_okresu_old varchar(8)
declare @Nazev_kraje_old varchar(150)
declare @Kod_kraje_old varchar(8)
```

// stejně tak slouží proměnné, které mají své jméno zakončené _old k dočasnému uložení hodnot dat již používaných v 1.vrstvě datového skladu

```
declare @Pocet_shodnych_radku int
```

// tato proměnná slouží k zjištění, zda prvek dimenze načtený z 0.vrstvě již existuje a má se porovnávat s existujícím nebo neexistuje a bude do tabulky dimenze vložen

```
declare @ID_Region_SCD bigint
```

// tato proměnná slouží k určení umělého klíče dimenze, který musí být naprosto jedinečný, praxe ukázala, že lepší než se spolehnout na číslo automaticky generované databázovým strojem (IDENTITY), je lepší administrovat vlastní tabulku, která bude sloužit k určení umělých klíčů

```
declare @CRC_Castiobce_old int
declare @CRC_Castiobce_new int
```

// tyto proměnné uchovávají hodnotu CRC pro „nová“ a „stará“ data

```
declare @ID_region_old bigint
declare @MDX_string varchar(1000)
```

// tyto proměnné slouží k zajištění korektního obsahu sloupce popisující MDX formuly, která slouží ke sloučení prvků dimenze se stejným přirozeným číslem pro správné zobrazení na úrovni OLAP datové kostky

```
declare Region_new cursor for

SELECT [Nazev_castiobce]
      ,[Kod_castiobce]
      ,[Nazev_obce]
      ,[Kod_obce]
      ,[Nazev_POU]
      ,[Kod_POU]
```

```
      ,[Nazev_ORP]
      ,[Kod_ORP]
      ,[Nazev_okresu]
      ,[Kod_okresu]
      ,[Nazev_kraje]
      ,[Kod_kraje]
      ,[Nazev_oblasti]
      ,[Kod_oblasti]
      ,[Datum_platnosti]
FROM  [KU_0].[dbo].[SU_CI_Dregiony]

open Region_new
```

// v této části se definuje kurzorová tabulka Region_new a načtou se do ní data regionální dimenze z 0.vrstvy datového skladu (databáze KU_0)

```
fetch next from Region_new into @Nazev_castiobce_new, @Kod_castiobce_new,
@Nazev_obce_new, @Kod_obce_new, @Nazev_POU_new, @Kod_POU_new,
@Nazev_ORP_new, @Kod_ORP_new, @Nazev_okresu_new, @Kod_okresu_new,
@Nazev_kraje_new, @Kod_kraje_new, @Nazev_oblasti_new, @Kod_oblasti_new,
@Datum_platnosti_new
```

// nyní se vezme první řádek z kurzorové tabulky a hodnoty sloupců se načtou do proměných _new

```
while (@@fetch_status<>-1)
```

// začátek smyčky, která se opakuje postupně pro každý řádek kurzorové tabulky

```
BEGIN

      SELECT @Pocet_shodnych_radku = COUNT(*)
      FROM [KU_1].dbo.SU_CI_DObce
      WHERE Kod_castiobce = @Kod_castiobce_new
```

// do proměnné @Pocet_shodnych_radku se zjistí, zda ve „starých“ datech v 1.vrstvě datového skladu již existuje řádek se stejným přirozeným klíčem jako má hodnota proměnné @Kod_castiobce_new

```
      IF @Pocet_shodnych_radku = 0
```

// pokud takový řádek ve „starých“ datech neexistuje, je třeba ho tam vložit

```
      BEGIN
```

```
SELECT @ID_Region_SCD = POSLEDNI_CISLO_CASTIOBCE
FROM KU_1.dbo.SU_CI_CISLA
```

// načtení posledního použité hodnoty umělého klíče z pomocné tabulky, která se přidá k nově vkládanému řádku do 1.vrstvy datového skladu

```
INSERT INTO [KU_1].dbo.SU_CI_DObce (ID_region_castiobce,
Nazev_castiobce, Kod_castiobce, MDX_castiobce, Nazev_obce,
Kod_obce, Nazev_POU, Kod_POU, Nazev_ORP, Kod_ORP,
Nazev_okresu, Kod_okresu, Nazev_kraje, Kod_kraje, Nazev_oblasti,
Kod_oblasti, Datum_platnosti, Od_data_castiobce,
Do_data_castiobce, Stav_castiobce)
VALUES(@ID_Region_SCD, @Nazev_castiobce_new,
@Kod_castiobce_new, '$Original', @Nazev_obce_new,
@Kod_obce_new, @Nazev_POU_new, @Kod_POU_new,
@Nazev_ORP_new, @Kod_ORP_new, @Nazev_okresu_new,
@Kod_okresu_new, @Nazev_kraje_new, @Kod_kraje_new,
@Nazev_oblasti_new, @Kod_oblasti_new,  @Datum_platnosti_new,
@Datum_platnosti_new, '2049', '1')
```

// vložení nového řádku do tabulky dimenze v 1.vrstvě datového skladu, do pole definující MDX formuly se zapíše hodnota '$Original', do pole držící hodnotu horní meze intervalu platnosti se zapíše nějaká hodně vysoká hodnota - '2049' a do sloupce platnosti se zapíše '1'

```
UPDATE KU_1.dbo.SU_CI_CISLA SET POSLEDNI_CISLO_CASTIOBCE
= POSLEDNI_CISLO_CASTIOBCE + 1
```

// následně se provede update pomocné tabulky, hodnota umělého klíče se zvýší o 1

END

// tím smyčka končí, pokračuje se dalším řádkem kurzorové tabulky

ELSE

// pokud existuje ve „starých“ datech řádek se stejným přirozeným klíčem jako je identifikátor nově načtených dat z 0.vrstvy, provádí se následující blok operací

BEGIN

```
SET @CRC_Castiobce_new =
BINARY_CHECKSUM(@Nazev_castiobce_new, @Kod_castiobce_new,
@Nazev_obce_new, @Kod_obce_new, @Nazev_POU_new,
@Kod_POU_new, @Nazev_ORP_new, @Kod_ORP_new,
```

```
@Nazev_okresu_new, @Kod_okresu_new, @Nazev_kraje_new,
@Kod_kraje_new, @Nazev_oblasti_new, @Kod_oblasti_new)
```

// nad hodnotami uvedených proměnných (atributy řádku nově načtených dat, které chceme porovnávat) se provede funkce BINARY_CHECKSUM a výsledek se uloží do proměnné @CRC_Castiobce_new

```
SELECT @CRC_Castiobce_old  = CRC_Castiobce
FROM dbo.SU_CI_DObce
WHERE Kod_castiobce = @Kod_castiobce_new
        AND Stav_castiobce = '1'
```

// do proměnné se @CRC_Castiobce_old uloží hodnota slupce držící BINARY_CHECKSUM hodnotu řádku ve „starých“ datech v 1.vrstvě datového skladu řádku se stejným přirozeným klíčem jako je identifikátor řádku „nových dat“, který je platný (může být jen jeden)

```
IF      @CRC_Castiobce_old <> @CRC_Castiobce_new
```

// pokud se hodnoty BINARY_CHECKSUM neliší, jsou porovnávané atributy „starých“ a „nových“ dat shodné a smyčka končí, pokračuje se dalším řádkem kurzorové tabulky

```
---     SELECT @Nazev_castiobce_old = Nazev_castiobce,
---     @Kod_castiobce_old = Kod_castiobce, @Nazev_obce_old =
---     Nazev_obce, @Kod_obce_old = Kod_obce, @Nazev_POU_old
=
---     Nazev_POU, @Kod_POU_old = Kod_POU, @Nazev_ORP_old =
---     Nazev_ORP, @Kod_ORP_old = Kod_ORP, @Nazev_okresu_old
=
---     Nazev_okresu, @Kod_okresu_old = Kod_okresu,
---     @Nazev_kraje_old = Nazev_kraje, @Kod_kraje_old =
Kod_kraje

---     IF      @Nazev_castiobce_old <> @Nazev_castiobce_new
---             OR @Nazev_obce_old <> @Nazev_obce_new
---             OR @Kod_obce_old <> @Kod_obce_new
---             OR @Nazev_POU_old <> @Nazev_POU_new
---             OR @Kod_POU_old <> @Kod_POU_new
---             OR @Nazev_ORP_old <> @Nazev_ORP_new
---             OR @Kod_ORP_old <> @Kod_ORP_new
---             OR @Nazev_okresu_old <> @Nazev_okresu_new
---             OR @Kod_okresu_old <> @Kod_okresu_new
---             OR @Nazev_kraje_old <> @Nazev_kraje_new
---             OR @Kod_kraje_old <> @Kod_kraje_new
```

// pro srovnání (při spuštění celého kroku se neprovede, protože je zneplatněn uvozením ---)je zde naznačen způsob, jak provést kontrolu bez využití CRC, z praxe lze konstatovat, že porovnávání velkého množství hodnot je mnohem náročnější než vytvoření CRC součtu, proto je použití CRC výhodnější

```
BEGIN

// pokud se hodnoty BINARY_CHECKSUM liší, bude nutné příslušný původní řádek v tabulce dimenze v 1.vrstvě datového skladu modifikovat a vložit řádek nový v souladu s postupem řešení SCD Typu 2

    SELECT @MDX_string = ''

    // do proměnné @MDX_string, v které se postupně bude tvořit MDX formule užitou v OLAP datové kostce se vloží prázdný řetězec

    declare MDX_string cursor for
    SELECT ID_region_castiobce FROM [KU_1].dbo.SU_CI_DObce
    WHERE Kod_castiobce = @Kod_castiobce_new

    open MDX_string

    // deklaruje se další kurzorová tabulka pro zajištění cyklu, kdy se postupně sestavuje požadovaná formule MDX z řádků „starých“ dat tabulky dimenze 1.vrstvě datového skladu se stejným přirozeným klíčem

    fetch next from MDX_string into @ID_region_old

    // hodnoty sloupců z prvního řádku kurzorové tabulky se načtou do proměnné

    while (@@fetch_status<>-1)

    // začátek smyčky, která se opakuje postupně pro každý řádek kurzorové tabulky

    BEGIN

        IF LEN(@MDX_string)>0 SELECT @MDX_string =
        @MDX_string + ','
```

```
SELECT @MDX_string = @MDX_string +
'[Dimenze].[Level].&[' +
RTRIM(CONVERT(char(1000),@ID_region_old)) + ']'
```

// pokud není proměnná @MDX_string prázdná, přidá se pro oddělení hodnot v MDX formuli a pak se do této proměnné přidá uvození '[Dimenze].[Level].&[', obsah hodnotu umělého klíče řádku kurzorové tabulky a znak ']'

```
fetch next from MDX_string into @ID_region_old
```

// načte se další řádek kurzorocé tabulky

```
END

close MDX_string

deallocate MDX_string
```

// smaže a z paměti se uvolní kurzorová tabulka pro sestavení MDX formule

```
SELECT @ID_Region_SCD = POSLEDNI_CISLO_CASTIOBCE
FROM KU_1.dbo.SU_CI_CISLA
```

// načtení poslední použité hodnoty umělého klíče z pomocné tabulky, která se přidá k nově vkládanému řádku do 1.vrstvy datového skladu a do MDX formule

```
SELECT @MDX_string = '{[Dimenze].[Level].&[' +
RTRIM(CONVERT(char(1000),@ID_Region_SCD)) + '],' +
@MDX_string + '}'
```

// k formuli MDX se přidá umělý klíč řádku nově načtených dat z 0.vrstvy datového skladu

```
UPDATE [KU_1].dbo.SU_CI_DObce SET Do_data_castiobce =
convert(smalldatetime,'31.12.'+convert(varchar(4),(YEAR(@Datum_
platnosti_new)-1)),104), Stav_castiobce = '2', MDX_castiobce =
'$Moved'
 WHERE Kod_castiobce = @Kod_castiobce_new
```

// provede se update dosud platného řádku „starých“ dat v tabulce dimenze v 1.vrstvě datového skladu – změní se datum horní meze platnosti, platnost a pole s hodnotou MDX formule se změní na '$Moved'

```
INSERT INTO [KU_1].dbo.SU_CI_DObce (ID_region_castiobce,
Nazev_castiobce, Kod_castiobce, MDX_castiobce, Nazev_obce,
Kod_obce, Nazev_POU, Kod_POU, Nazev_ORP, Kod_ORP,
Nazev_okresu, Kod_okresu, Nazev_kraje, Kod_kraje, Nazev_oblasti,
Kod_oblasti, Datum_platnosti, Od_data_castiobce,
Do_data_castiobce, Stav_castiobce) VALUES(@ID_Region_SCD,
@Nazev_castiobce_new, @Kod_castiobce_new,  @MDX_string,
@Nazev_obce_new, @Kod_obce_new, @Nazev_POU_new,
@Kod_POU_new, @Nazev_ORP_new, @Kod_ORP_new,
@Nazev_okresu_new, @Kod_okresu_new, @Nazev_kraje_new,
@Kod_kraje_new, @Nazev_oblasti_new, @Kod_oblasti_new,
@Datum_platnosti_new, @Datum_platnosti_new, '2049', '1')
```

// vložení nového řádku do tabulky dimenze v 1.vrstvě datového skladu, do pole definující MDX formuly se zapíše hodnota '$Original', do pole držící hodnotu horní meze intervalu platnosti se zapíše nějaká hodně vysoká hodnota - '2049' a do sloupce platnosti se zapíše '1'

```
UPDATE KU_1.dbo.SU_CI_CISLA SET POSLEDNI_CISLO_CASTIOBCE
= POSLEDNI_CISLO_CASTIOBCE + 1
```

// následně se provede update pomocné tabulky, hodnota umělého klíče se zvýší o 1

```
END

    END

fetch next from Region_new into @Nazev_castiobce_new, @Kod_castiobce_new,
@Nazev_obce_new, @Kod_obce_new, @Nazev_POU_new, @Kod_POU_new,
@Nazev_ORP_new, @Kod_ORP_new, @Nazev_okresu_new, @Kod_okresu_new,
@Nazev_kraje_new, @Kod_kraje_new, @Nazev_oblasti_new, @Kod_oblasti_new,
@Datum_platnosti_new
```

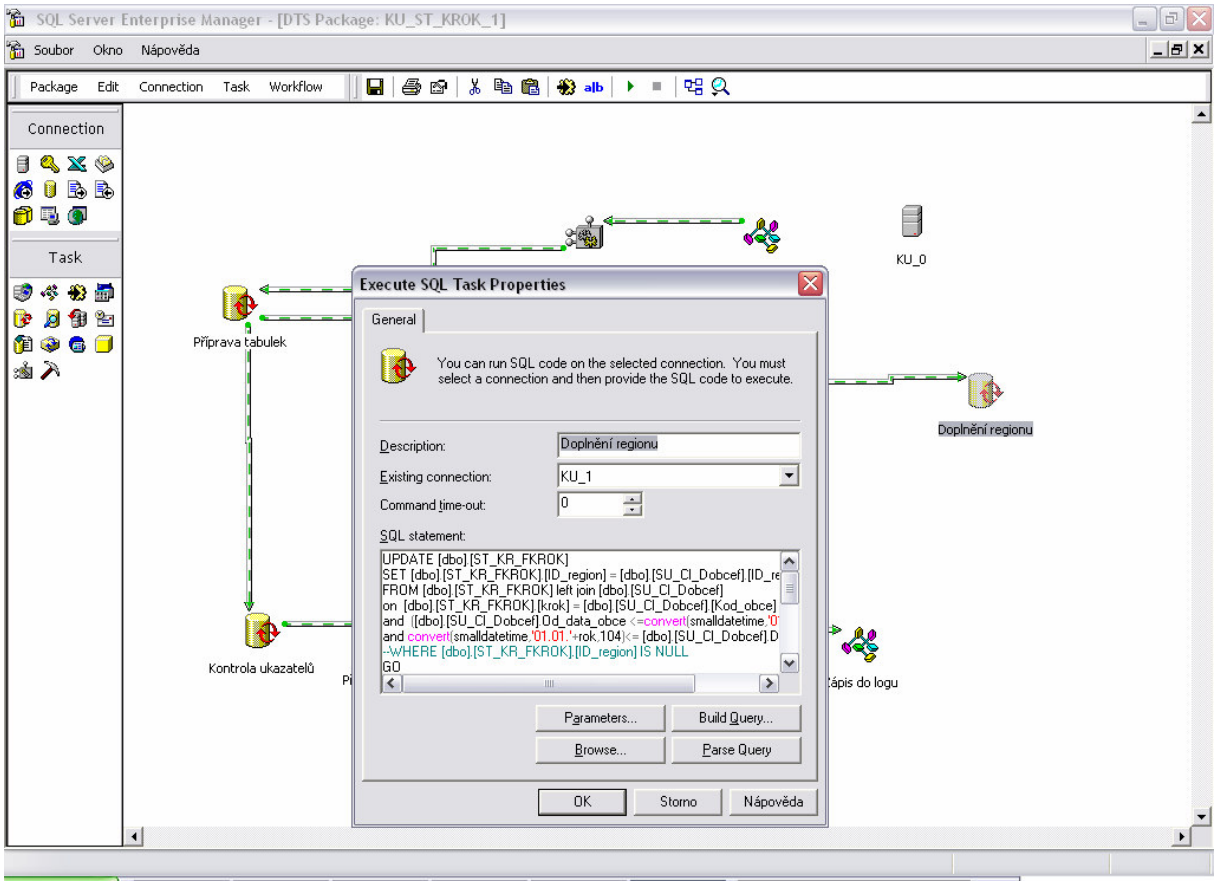
// načte se další řádek kurzorové tabulky

```
END

close Region_new

deallocate Region_new
```

// smaže a z paměti se uvolní kurzorová tabulka pro kontrolu

V tuto chvíli již tabulka SU_CI_Dobcef obsahuje aktualizovaná data. Dalším krokem je načtení nových dat faktů z 0.vrstvy do 1.vrstvy. Přestože se jedná o velmi důležitý a náročný krok plnění datového skladu, blíženi popisován nebude, neboť nemá s problematikou měnících se dimenzí nic společného. Po načtení faktů je však třeba provést update tabulek faktů pro zachycení vazby mezi fakty a dimenzí přes umělý klíč. Balíček, který zajistí tento update je znázorněn na Obrázku č.15.

Obrázek č. 15 – Update tabulek faktů



Zdroj: [PVT]

Samotný kód transformačního kroku pro dvě tabulky faktů [ST_KR_FKROK] a [ST_KR_FKROK_MOS] je následující:

```
UPDATE [dbo].[ST_KR_FKROK]
SET [dbo].[ST_KR_FKROK].[ID_region] = [dbo].[SU_CI_Dobcef].[ID_region]
FROM [dbo].[ST_KR_FKROK] LEFT JOIN [dbo].[SU_CI_Dobcef] ON
        [dbo].[ST_KR_FKROK].[krok] = [dbo].[SU_CI_Dobcef].[Kod_obce]
                and  ([dbo].[SU_CI_Dobcef].Od_data_obce
```

```
                    <=convert(smalldatetime,'01.01.'+rok,104)
            and convert(smalldatetime,'01.01.'+rok,104)
                    <=[dbo].[SU_CI_Dobcef].Do_data_obce)
GO

UPDATE [dbo].[ST_KR_FKROK_MOS]
SET [dbo].[ST_KR_FKROK_MOS].[ID_region] = [dbo].[SU_CI_Dobcef].[ID_region]
FROM [dbo].[ST_KR_FKROK_MOS] LEFT JOIN [dbo].[SU_CI_Dobcef] ON
      [dbo].[ST_KR_FKROK_MOS].[krok] = [dbo].[SU_CI_Dobcef].[Kod_obce]
            and  ([dbo].[SU_CI_Dobcef].Od_data_obce
                    <=convert(smalldatetime,'01.01.'+rok,104)
            and convert(smalldatetime,'01.01.'+rok,104)
                    <=[dbo].[SU_CI_Dobcef].Do_data_obce)
```

// hodnota umělého klíče z tabulky [SU_CI_Dobcef] je vložena do tabulek faktů do sloupce [ID_region] příslušného řádku dle rovnosti přirozených klíčů a rozmezí platnosti daného řádku v tabulce dimenze.

Příklad řešení všech dimenzí najednou

Další příklad, který bych chtěl podrobněji rozebrat a demonstrovat na něm řešení pomalu se měnících dimenzí, vychází z dalšího sektoru, na kterém jsem se podílel, tentokrát z oblasti kapitálového trhu. Zdrojem dimenzionálních dat je v tomto případě denně aktualizovaná dávka, která obsahuje všechny číselníky používané ve zdrojovém produkčním systému. Tato dávka se uloží do jediné tabulky v 0.vrstvě datového skladu a slouží jako zdroj pro načítání do vyšších vrstev řešení. Toto uložení není v tomto konkrétním případě výjimečně v kompetenci ETL procesů, ale z organizačních důvodů je zajišťeno jiným způsobem. Protože jsou všechny číselníky v jediné tabulce a přibývají v jednom časovém okamžiku se stejnou platností, využívá navržené řešení Hlavní tabulku s křížovým odkazem.

Postup řešení

Postup nahrávání dat zdrojových dimenzionálních dat do datového skladu je následující:

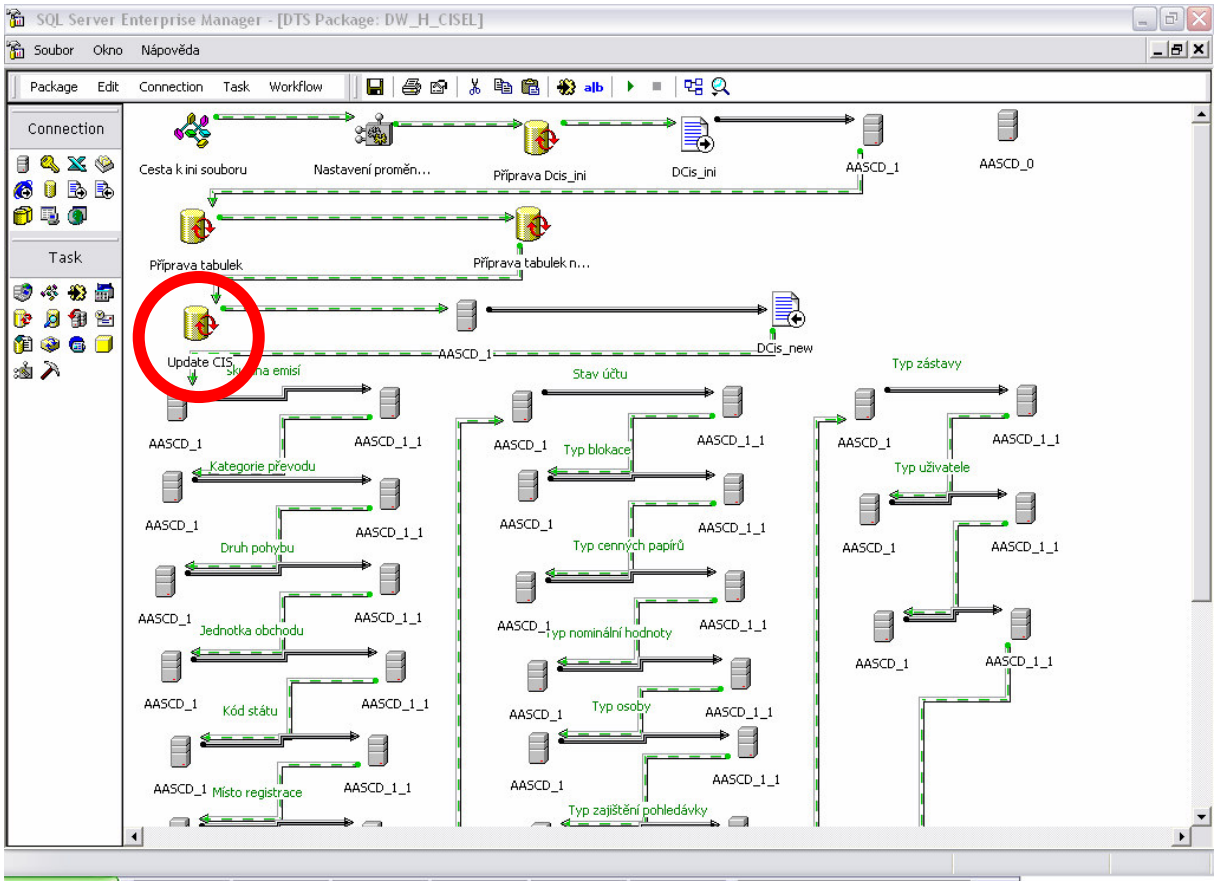
- Předpokladem je přítomnost dat v 0.vrstvě datového skladu

- Data 0.vrstvy datového skladu jsou načítána ETL procesy, transformována do podoby, která je výhodná pro efektivní získávání analytických výstupů. Následně jsou data uložena do 1.vrstvy datového skladu, která slouží jako relační úložiště pro datové kostky definované v analytickém serveru. Tento proces se provádí periodicky každý den. DTS balíček, který tento krok definuje je znázorněn na obrázku č.15, tento krok je podrobněji popsán níže.

- Poté se načtou nová data faktů do tabulek faktů.

- Posledním krokem je opět update vazeb mezi tabulkami faktů a tabulkou regionální dimenze přes umělý klíč neboť tato vazba se používá k uchování historie, v e zdrojových datech neexistovala, neboť vznikla až v průběhu transformace.

Obrázek č. 16 – Datová pumpa s krokem ošetřujícím SCD



Zdroj: [PVT]

Kruhem znázorněný krok transformace pojmenovaný „Update_CIS“ uchovává metadata o potřebných kontrolách a případných transformacích pro řešení problémů pomalu se měnících dimezí za použití Hlavní tabulky s křížovým odkazem.

Na následujích stránkách je opět uveden kód transformace s komentářem k jednotlivým funkčním blokům. Komentář je opět uveden znaky „//“. Protože jsou číselníky zdrojového systému ploché, je hierarchie dimenzí definována až na úrovni datového skladu – při primárním vkládání dat do datového skladu je hierarchie načtena z uživatelem definované tabulky. Z tohoto důvodu se v tomto příkladu nepoužívá porovnávání CRC součtů, ale přímé porovnání atributů.

Krok „Update_CIS“:

```
declare @Asl_id_new int
declare @TypAas_new varchar (9)
declare @Klc_new varchar(6)
declare @Hdn1_new varchar(25)
declare @DTZM_new datetime

// deklarace používaných proměnných, ty, které mají své jméno zakončené _new slouží
k dočasnému uložení hodnot nově načtených dat 0.vrstvy datového skladu.

declare @Typ_old varchar (25)
declare @Name_old varchar(25)

// stejně tak slouží proměnné, které mají své jméno zakončené _old k dočasnému uložení
hodnot dat již používaných v 1.vrstvě datového skladu

declare @Pocet_shodnych_radku int

// tato proměnná slouží k zjištění, zda prvek dimenze načtený z 0.vrstvě již existuje a má se
porovnávat s existujícím nebo neexistuje a bude do tabulky dimenze vložen

declare @Asl_id_max_old int
declare @Asl_id_max_new int

// tyto proměnné slouží k zjištění posledního čísla dávky vložení dat z provozního systému
do 0.vrstvy datového skladu (všecny řádky vložené jednou dávkou mají stejné číslo
dávky), toto číslo se uchovává ve speciální pomocné tabulce
```

```
declare @Cis_id_SCD int
```

// tato proměnná slouží k určení umělého klíče dimenze, který musí být naprosto jedinečný

```
SELECT @Asl_id_max_old = [LastAsl_Cis]
FROM [AASCD_1].[dbo].[DW_PIMPORT]
```

// do proměnné @Asl_id_max_old se uloží číslo poslední zpracované dávky číselníků

```
declare Ciselnik_new cursor for

SELECT CDCis.Asl_id, TypAas, Klc, Hdn1, DatUct
FROM dbo.CDCis JOIN dbo.CDCisTyp ON
        CDCis.CisTyp_id = CDCisTyp.CisTyp_id JOIN dbo.CDAsl ON
        CDCis.Asl_id = CDAsl.Asl_id
        WHERE TypAas IS NOT NULL
                AND CDCis.Asl_id > @Asl_id_max_old

open Ciselnik_new
```

// v této části se definuje kurzorová tabulka Ciselnik_new a načtou se do ní data všech číselníků z 0.vrstvy datového skladu

```
fetch next from Ciselnik_new into @Asl_id_new, @TypAas_new, @Klc_new,
@Hdn1_new, @DTZM_new
```

// nyní se vezme první řádek z kurzorové tabulky a hodnoty sloupců se načtou do proměnných _new

```
while (@@fetch_status<>-1)
```

// začátek smyčky, která se opakuje postupně pro každý řádek kurzorové tabulky

```
BEGIN

        SELECT @Pocet_shodnych_radku = COUNT(*)
        FROM AASCD_1.dbo.DW_DCis
        WHERE Cis_nid = @Klc_new AND Cis_name = @TypAas_new
```

// do proměnné @Pocet_shodnych_radku se zjistí, zda ve „starých“ datech v 1.vrstvě datového skladu již existuje řádek se stejným přirozeným klíčem jako má hodnota proměnné @Klc_new a protože přirozený klíč může být v různých číselnících stejný, musí se také shodovat jméno číselníku

```
IF @Pocet_shodnych_radku = 0
```

// pokud takový řádek ve „starých“ datech neexistuje, je třeba ho tam vložit

```
BEGIN

    SELECT @Cis_id_SCD = LastCis_SCD
    FROM AASCD_1.dbo.DW_PIMPORT
```

// do proměnné @Cis_id_SCD se uloží poslední použitá hodnota umělého klíče z pomocné tabulky, která se přidá k nově vkládanému řádku do 1.vrstvy datového skladu

```
    INSERT INTO AASCD_1.dbo.DW_DCis ([Asl_id], [Cis_name],
    [Cis_id], [Cis_nid], [Typ], [Nazev], [Datum_od], [Datum_do],
    [Datum_run])
    VALUES(@Asl_id_new, @TypAas_new, @Cis_id_SCD, @Klc_new,
    'Neudáno', @Hdn1_new, @DTZM_new, '2199-01-01', getdate())
```

// vložení nového řádku do tabulky dimenze v 1.vrstvě datového skladu, do pole držící hodnotu horní meze intervalu platnosti se zapíše nějaká hodně vysoká hodnota - '2199-01-01', do sloupce, který definuje hierarchii se vyplní hodnota 'Neudáno'

```
    UPDATE AASCD_1.dbo.DW_PIMPORT SET LastCis_SCD =
    LastCis_SCD + 1
```

// následně se provede update pomocné tabulky, hodnota umělého klíče se zvýší o 1

```
END
```

// tím smyčka končí, pokračuje se dalším řádkem kurzorové tabulky

```
ELSE
```

// pokud existuje ve „starých“ datech řádek se stejným přirozeným klíčem jako je identifikátor nově načtených dat z 0.vrstvy provádí se následující blok operací

```
BEGIN

        SELECT @Name_old = Nazev
        FROM AASCD_1.dbo.DW_DCis
        WHERE [Cis_nid] = @Klc_new
                AND [Cis_name] = @TypAas_new
                AND [Datum_do] = '2199-01-01'
```

// do proměnné @Name_old se uloží text prvku dimenze ze „starých“ z tabulky v 1.vrstvě datového skladu, který má stejný přirozený klíč jako řádek „nových“ dat načtený z 0.vrstvy datového skladu, který je ze stejného číselníku a zároveň je platný

```
        IF      @Hdn1_new <> @Name_old
```

// pokud se tyto dvě proměnné neliší, jsou porovnávané atributy „starých“ a „nových“ dat shodné a smyčka končí, pokračuje se dalším řádkem kurzorové tabulky

```
        BEGIN
```

// pokud se proměnné liší, bude nutné příslušný původní řádek v tabulce dimenze v 1.vrstvě datového skladu modifikovat a vložit řádek nový v souladu s postupem řešení SCD Typu 2

```
        SELECT @Cis_id_SCD = LastCis_SCD FROM
        AASCD_1.dbo.DW_PIMPORT
```

// do proměnné @Cis_id_SCD se uloží poslední použitá hodnota umělého klíče z pomocné tabulky, která se přidá k nově vkládanému řádku do 1.vrstvy datového skladu

```
        SELECT @Typ_old = Typ FROM AASCD_1.dbo.DW_DCis WHERE
        [Cis_nid] = @Klc_new AND [Cis_name] = @TypAas_new AND
        [Datum_do] = '2199-01-01'
```

// do proměnné @Typ_old se uloží hodnota, která popisuje hierarchii aktuálně platného prvku dimenze, protože zdrojová data tuto fodnotu neobsahují a je nutné ji zachovat i pro nově vložený řádek

```
        UPDATE AASCD_1.dbo.DW_DCis SET [Datum_do] = @DTZM_new -
        1 WHERE [Cis_nid] = @Klc_new AND [Cis_name] = @TypAas_new
        AND [Datum_do] = '2199-01-01'
```

// provede se update dosud platného řádku „starých“ dat v tabulce dimenze v 1.vrstvě datového skladu – změní se datum horní meze platnosti

```
            INSERT INTO AASCD_1.dbo.DW_DCis ([Asl_id], [Cis_name],
            [Cis_id], [Cis_nid], [Typ], [Nazev], [Datum_od], [Datum_do],
            [Datum_run]) VALUES(@Asl_id_new, @TypAas_new, @Cis_id_SCD,
            @Klc_new, @Typ_old, @Hdn1_new, @DTZM_new, '2199-01-01',
            getdate())
```

// vložení nového řádku do tabulky dimenze v 1.vrstvě datového skladu, do pole držící hodnotu horní meze intervalu platnosti se zapíše nějaká hodně vysoká hodnota - '2199-01-01'

```
            UPDATE AASCD_1.dbo.DW_PIMPORT SET LastCis_SCD =
            LastCis_SCD + 1
```

// následně se provede update pomocné tabulky, hodnota umělého klíče se zvýší o 1

```
            END

        END

fetch next from Ciselnik_new into @Asl_id_new, @TypAas_new, @Klc_new,
@Hdn1_new, @DTZM_new
```

// načte se další řádek kurzorové tabulky

```
END

close Ciselnik_new
deallocate Ciselnik_new
```

// smaže a z paměti se uvolní kurzorová tabulka pro kontrolu

```
SELECT @Asl_id_max_new = MAX(Asl_id)
FROM CDCis
```

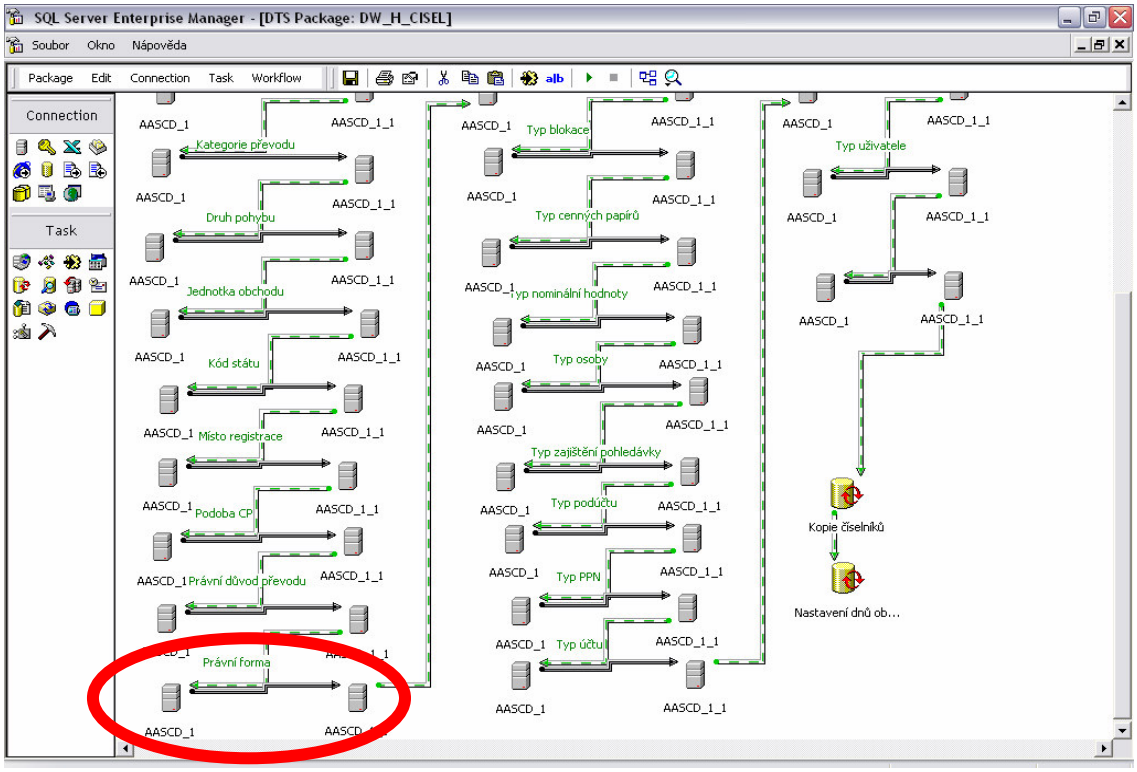
// do proměnné @Asl_id_max_new se zapíše nejvyšší hodnota čísla dávky případně dávek, které byly právě zpracovávány

```
UPDATE AASCD_1.dbo.DW_PIMPORT SET LastAsl_CIS = @Asl_id_max_new
```

// zjištěná hodnota posledního čísla dávky je vložena do pomocné tabulky

V dalších krocích DTS balíčku je nutné zajistit generaci konkrétních tabulek dimenzí z Hlavní tabulky s křížovým odkazem.

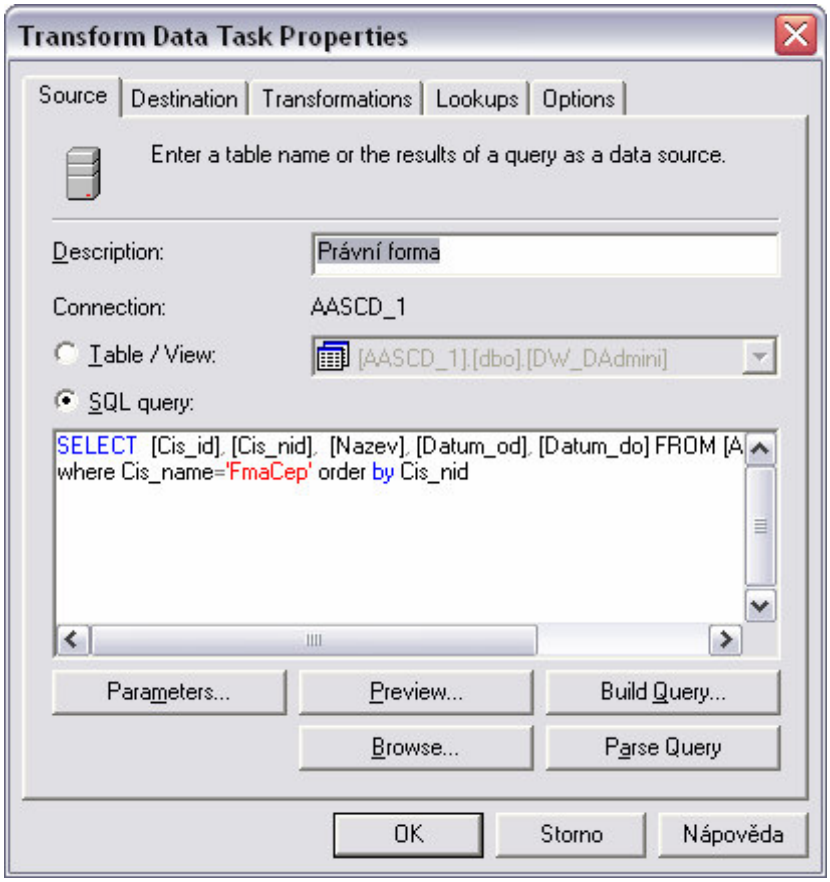
Obrázek č. 17 – Načtení dat do tabulek dimenzí



Zdroj: [PVT]

Jednotlivé kroky vytvářející tabulky dimenzí jsou zobrazeny na obrázku č. 17, podrobný popis jedné z nich (na celkovém obrázku je znázorněna elipsou), následuje.

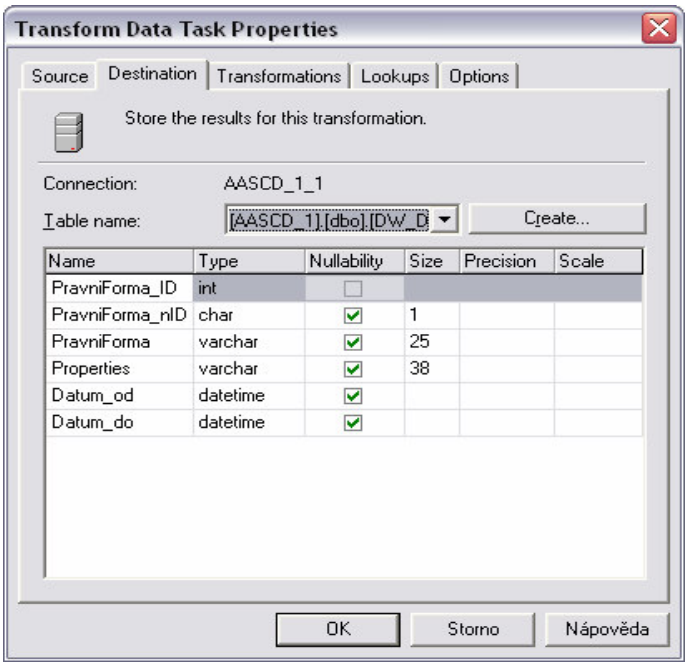
Obrázek č. 18 – Definice zdroje transformace



Zdroj: **[PVT]**

Zdrojem dat procesu vytvoření dimenze popisující právní formu je výběr z výše připravené Hlavní tabulky s křížovým odkazem, s omezením názvu číselníku.

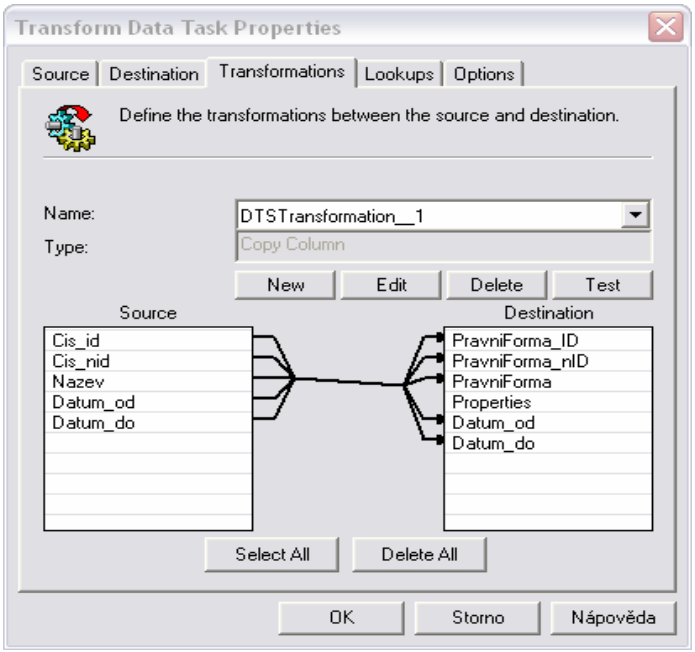
Obrázek č. 19 – Definice cíle tranformace



Zdroj: [PVT]

Cílem je pak tabulka dimeze Právní formy, která je definována také v 1.vrstvě datového skladu a slouží jako relační úložiště a zdroj dat pro OLAP datové kostky.

Obrázek č. 20 – Definice vlastní transformace



Zdroj: **[PVT]**

Mezi definovaným zdrojem a cílem je možné definovat různé pokročilé transformace, nicméně v popisovaném příkladu dochází k prostému kopírování, neboť všechny potřebné změny již byly ošetřeny v předchozím kroku.

Dalším krokem je načtení nových dat faktů z 0.vrstvy do 1.vrstvy. Stejně jako v prvním příkldu nebude podrobněji popsán. Po načtení faktů je třeba provést update tabulek faktů pro zachycení vazby mezi fakty a dimenzí přes umělý klíč.

Kód transformačního kroku pro tabulku faktů [DW_FEMISEOBD] je následující:

```
UPDATE AASCD_1..DW_FEMISEOBD
SET PravniForma_id = IsNull(AASCD_1..DW_DPravniForma.PravniForma_id,9999)
FROM AASCD_1..DW_FEMISEOBD LEFT JOIN AASCD_1..DW_DPravniForma ON
        AASCD_1..DW_FEMISEOBD.PravniForma_nid=AASCD_1..DW_DPravniFor
        ma.PravniForma_nid
        AND
        AASCD_1..DW_FEMISEOBD.DaTZM>=AASCD_1..DW_DPravniForma.Datu
        m_Od
        AND
        AASCD_1..DW_FEMISEOBD.DaTZM<=AASCD_1..DW_DPravniForma.Datu
        m_Do
GO

// provede se update cizího klíče PravniForma_id v tabulce faktů umělým klíčem
z tabulky dimenze DW_DpravniForma dle rovnosti přirozených klíčů a rozmezí platnosti
daného řádku v tabulce dimenze, pokud by měl mít údaj hodnotu NULL, naplní se kódem
'9999'

UPDATE AASCD_1..DW_FEMISEOBD
SET TypCP_id = IsNull(AASCD_1..DW_DTypCP.TypCP_id,9999)
FROM AASCD_1..DW_FEMISEOBD LEFT JOIN AASCD_1..DW_DTypCP ON
       AASCD_1..DW_FEMISEOBD.TypCP_nid=AASCD_1..DW_DTypCP.TypCP_nid
       AND
       AASCD_1..DW_FEMISEOBD.DaTZM>=AASCD_1..DW_DTypCP.Datum_Od
       AND
       AASCD_1..DW_FEMISEOBD.DaTZM<=AASCD_1..DW_DTypCP.Datum_Do
GO

UPDATE AASCD_1..DW_FEMISEOBD
SET PodobaCP_id = IsNull(AASCD_1..DW_DPodobaCP.PodobaCP_id,9999)
FROM AASCD_1..DW_FEMISEOBD LEFT JOIN AASCD_1..DW_DPodobaCP ON
       AASCD_1..DW_FEMISEOBD.PodobaCP_nid=AASCD_1..DW_DPodobaCP.Pod
       obaCP_nid
```

```
        AND
        AASCD_1..DW_FEMISEOBD.DaTZM>=AASCD_1..DW_DPodobaCP.Datum_O
        d
        AND
        AASCD_1..DW_FEMISEOBD.DaTZM<=AASCD_1..DW_DPodobaCP.Datum_D
        o
GO

UPDATE AASCD_1..DW_FEMISEOBD
SET MistoRegistrace_id =
IsNull(AASCD_1..DW_DMistoReg.MistoRegistrace_id,9999)
FROM AASCD_1..DW_FEMISEOBD LEFT JOIN AASCD_1..DW_DMistoReg ON
        AASCD_1..DW_FEMISEOBD.MistoRegistrace_nid=AASCD_1..DW_DMistoReg
        .MistoRegistrace_nid
        AND
        AASCD_1..DW_FEMISEOBD.DaTZM>=AASCD_1..DW_DMistoReg.Datum_O
        d AND
        AASCD_1..DW_FEMISEOBD.DaTZM<=AASCD_1..DW_DMistoReg.Datum_D
        o
GO

UPDATE AASCD_1..DW_FEMISEOBD
SET TypNomHodnota_id =
IsNull(AASCD_1..DW_DTypNomHod.TypNomHodnoty_id,9999)
FROM AASCD_1..DW_FEMISEOBD LEFT JOIN AASCD_1..DW_DTypNomHod ON
        AASCD_1..DW_FEMISEOBD.TypNomHodnota_nid=AASCD_1..DW_DTypNo
        mHod.TypNomHodnoty_nid
        AND
        AASCD_1..DW_FEMISEOBD.DaTZM>=AASCD_1..DW_DTypNomHod.Datum
        _Od
        AND
        AASCD_1..DW_FEMISEOBD.DaTZM<=AASCD_1..DW_DTypNomHod.Datum
        _Do
GO

UPDATE AASCD_1..DW_FEMISEOBD
SET SkupEms_id = IsNull(AASCD_1..DW_DKodSkuEm.KodSkuEm_id,9999)
FROM AASCD_1..DW_FEMISEOBD LEFT JOIN AASCD_1..DW_DKodSkuEm ON
        AASCD_1..DW_FEMISEOBD.SkupEms_nid=AASCD_1..DW_DKodSkuEm.Kod
        SkuEm_nid
        AND
        AASCD_1..DW_FEMISEOBD.DaTZM>=AASCD_1..DW_DKodSkuEm.Datum_
        Od AND
        AASCD_1..DW_FEMISEOBD.DaTZM<=AASCD_1..DW_DKodSkuEm.Datum_
        Do
GO
```

```
UPDATE AASCD_1..DW_FEMISEOBD
SET Prevoditelnost_id = IsNull(AASCD_1..DW_DPrevod.Prevoditelnost_id,9999)
FROM AASCD_1..DW_FEMISEOBD LEFT JOIN AASCD_1..DW_DPrevod ON
        AASCD_1..DW_FEMISEOBD.Prevoditelnost_nid=AASCD_1..DW_DPrevod.Pr
        evoditelnost_nid
        AND
        AASCD_1..DW_FEMISEOBD.DaTZM>=AASCD_1..DW_DPrevod.Datum_Od
        AND
        AASCD_1..DW_FEMISEOBD.DaTZM<=AASCD_1..DW_DPrevod.Datum_Do
GO

UPDATE AASCD_1..DW_FEMISEOBD
SET Stat_id = IsNull(AASCD_1..DW_DKodStatu.KodStatu_id,9999)
FROM AASCD_1..DW_FEMISEOBD LEFT JOIN AASCD_1..DW_DKodStatu ON
        AASCD_1..DW_FEMISEOBD.Stat_nid=AASCD_1..DW_DKodStatu.KodStatu_
        nid AND
        AASCD_1..DW_FEMISEOBD.DaTZM>=AASCD_1..DW_DKodStatu.Datum_O
        d AND
        AASCD_1..DW_FEMISEOBD.DaTZM<=AASCD_1..DW_DKodStatu.Datum_Do
GO

UPDATE DW_FEMISEOBD
SET JednotkaObchodu_Id =
IsNull(DW_DJednotkaObchodu.JednotkaObchodu_ID,9999)
FROM DW_FEMISEOBD LEFT JOIN DW_DJednotkaObchodu ON
        DW_FEMISEOBD.JednotkaObchodu_nid=DW_DJednotkaObchodu.Jednotka
        Obchodu_nid
        AND DW_FEMISEOBD.DaTZM>=DW_DJednotkaObchodu.Datum_Od
        AND DW_FEMISEOBD.DaTZM<=DW_DJednotkaObchodu.Datum_Do
```

// stejným způsobem se provede update všech dalších cizích klíčů v tabulce faktů příslušným umělým klíčem z tabulky dimenze dle rovnosti přirozených klíčů a rozmezí platnosti daného řádku v tabulce dimenze, pokud by měl mít údaj hodnotu NULL, naplní se kódem '9999'.

Ruční úprava hierarchie dimenzí

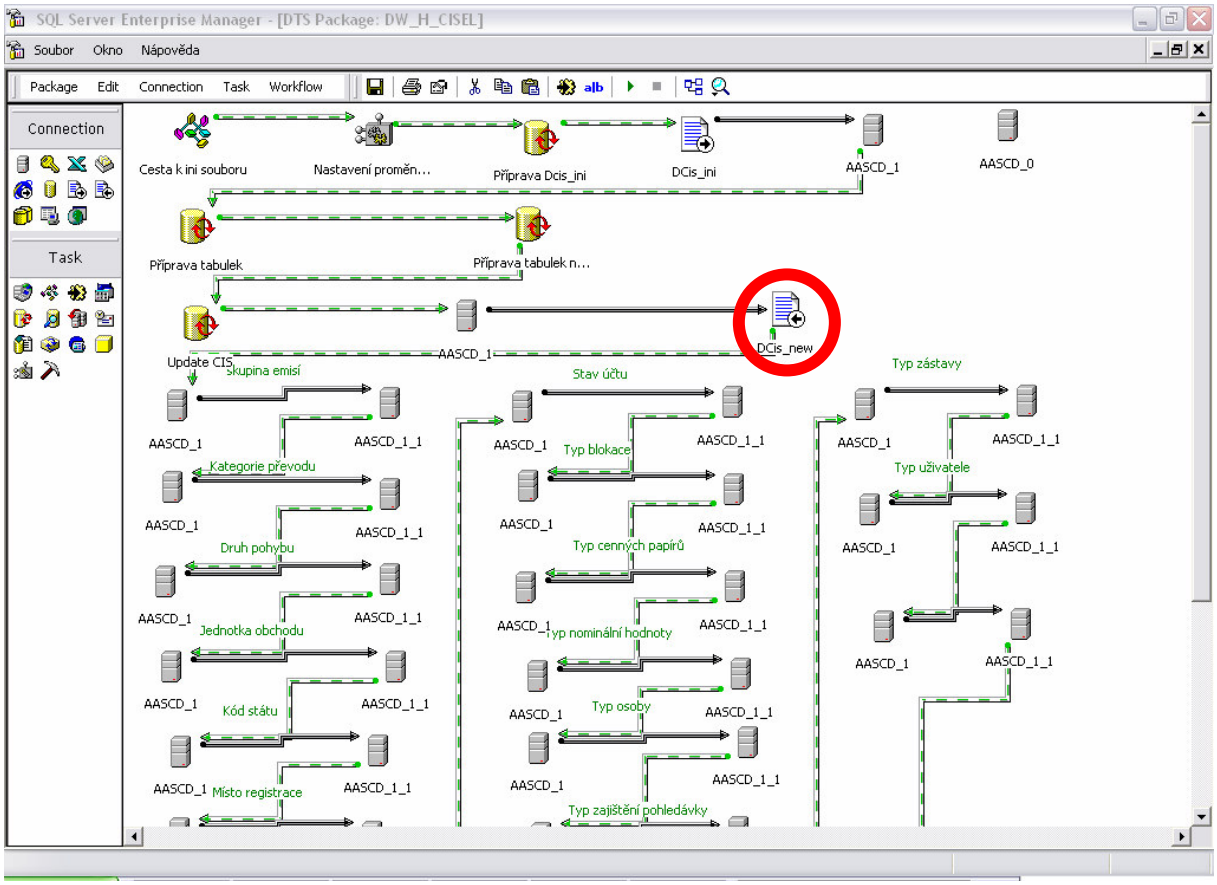
Jak již bylo řečeno výše, jelikož jsou zdrojové číselníky ploché, je nutné zajistit správné začlenění nově vložených členů dimenze do hierarchie. Tuto činnost by neměl v žádném případě provádět počítač samostatně neboť nezná pravidla pro jednoznačné

zařazení. Těmito znalostmi disponuje pouze člověk, proto by to měl být on, kdo nové prvky dimenze do hierarchie zařadí.

Počítač mu však může pomoci. Především v rámci řešení problémů měnících se dimenzí na takové nové prvky přijde, zároveň je může uživateli předložit v takové formě, která bude přehledná, snadno editovatelná a jednoduše aplikovatelná zpět do metadat datového skladu.

V praxi se tato nutnost začlenit člověka do procesu aktualizace dimenzionálních dat řeší exportem dat, která potřebují ruční úpravy do vhodného formátu pro úpravu, např. list MS Excelu nebo textový dokument s oddělovači polí. Tento export je vhodné začlenit do ETL procesu načítání dimenzionálních dat do datového skladu, jak je patrné z obrázku č.21, konkrétní krok je zvýrazněn kruhem.

Obrázek č. 21: Export dat do textového souboru



Zdroj: [PVT]

V druhém příkladu byla u nově vkládaných prvků dimenze do pole popisující hierarchii dimenze nastavena hodnota na 'Neudáno', takže uživatel na první pokus vidí, které prvky dimenze je třeba do hierarchie zařadit.

Po provedených úpravách je znovu updatována tabulka dimenze, jako zdroj slouží zeditovaný textový soubor nebo list MS Excel. Update fatkových tabulek není nutný, neboť všechny klíče zůstaly nedotčeny.

5. Závěr

Myslím si, že v úvodu vytyčené cíle mé diplomové práce byly splněny. Nejprve byl představen teoretický rámec, z kterého vycházelo podrobně popsané nasazení měnících se dimenzí v konkrétních situacích.

Za největší úspěch považuji to, že výsledky mé diplomové práce - navržené postupy řešení problémů měnících se dimenzí na úrovni datového skladu a OLAP analýzy, které vycházejí z posledních teoretických poznatků, byly opakovaně a úspěšně převedeny do praxe. Výstupy projektů, resp. zpětná vazba od uživatelů analytických výstupů vývojových a implementačních projektů, v rámci nichž se ošetření problémů měnících se dimenzí nasadilo, to potvrzují.

Seznam citovaných a použitých zdrojů

[GAR]: H. Dresner, A. Linden, F. Buytendijk, T., Friedman, K. Strange, M. Knox, M. Camm: The Business Intelligence Competency Center: An Essential Business Strategy; Gartner, Inc.; 2002

[KIM]: Kimball, R., Ross, M.: The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data; Wiley Publishing, Inc.; 2004

[MS1]: Microsoft: Course 2074A, Designing and Implementing OLAP Solutions with Microsoft SQL Server 2000; Microsoft Training and Certification Official Curricilum; 2001

[MS2]: Amir Netz: OLAP Services: Managing Slowly Changing Dimensions; msdn.microsoft.com; Microsoft Corporation

[NOV]: Ing. Ota Novotný, Ph.D., Doc. Ing. Jan Pour, CSc., Ing. David Slánský: Business Intelligence – Jak využít bohatství ve vašich datech; Grada Publishing; 2005

[PVT]: Interní dokumenty PVT, a.s.

Další použité zdroje

[1] Kimball, R., Ross, M.: The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling, 2nd Edition; John Wiley & Sons; 2002

[2] Bahník, V.: Konsolidace informačního systému a nasazení nástrojů Business Intelligence v PVT, a.s. – bakalářská práce; TU Liberec, 2004

[2] Humphries, M.: Datawarehousing – návrh a implementace; Computer Press; 2001,

[3] Jacobson, R.: Microsoft SQL Server 2000 Analysis Services – Step by Step; Microsoft Press; 2000

[4] Joe Luedtke: Implementing Slowly Changing Dimensions; SQL Server magazine; www.sqlmag.com

Příloha č.1: Kód ETL transformací v jazyce Transact-SQL

Krok „Region_castiobce“:

```
declare @Nazev_castiobce_new varchar(150)
declare @Kod_castiobce_new varchar(8)
declare @Nazev_obce_new varchar(150)
declare @Kod_obce_new varchar(8)
declare @Nazev_POU_new varchar(150)
declare @Kod_POU_new varchar(8)
declare @Nazev_ORP_new varchar(150)
declare @Kod_ORP_new varchar(8)
declare @Nazev_okresu_new varchar(150)
declare @Kod_okresu_new varchar(8)
declare @Nazev_kraje_new varchar(150)
declare @Kod_kraje_new varchar(8)
declare @Nazev_oblasti_new varchar(150)
declare @Kod_oblasti_new varchar(8)
declare @Datum_platnosti_new smalldatetime

declare @Nazev_castiobce_old varchar(150)
declare @Kod_castiobce_old varchar(8)
declare @Nazev_obce_old varchar(150)
declare @Kod_obce_old varchar(8)
declare @Nazev_POU_old varchar(150)
declare @Kod_POU_old varchar(8)
declare @Nazev_ORP_old varchar(150)
declare @Kod_ORP_old varchar(8)
declare @Nazev_okresu_old varchar(150)
declare @Kod_okresu_old varchar(8)
declare @Nazev_kraje_old varchar(150)
declare @Kod_kraje_old varchar(8)

declare @Pocet_shodnych_radku int

declare @ID_Region_SCD bigint

declare @CRC_Castiobce_old int
declare @CRC_Castiobce_new int

declare @ID_region_old bigint
declare @MDX_string varchar(1000)
```

```
declare Region_new cursor for

SELECT [Nazev_castiobce]
      ,[Kod_castiobce]
      ,[Nazev_obce]
      ,[Kod_obce]
      ,[Nazev_POU]
      ,[Kod_POU]
      ,[Nazev_ORP]
      ,[Kod_ORP]
      ,[Nazev_okresu]
      ,[Kod_okresu]
      ,[Nazev_kraje]
      ,[Kod_kraje]
      ,[Nazev_oblasti]
      ,[Kod_oblasti]
      ,[Datum_platnosti]
FROM  [KU_0].[dbo].[SU_CI_Dregiony]

open Region_new

fetch next from Region_new into @Nazev_castiobce_new, @Kod_castiobce_new,
@Nazev_obce_new, @Kod_obce_new, @Nazev_POU_new, @Kod_POU_new,
@Nazev_ORP_new, @Kod_ORP_new, @Nazev_okresu_new, @Kod_okresu_new,
@Nazev_kraje_new, @Kod_kraje_new, @Nazev_oblasti_new, @Kod_oblasti_new,
@Datum_platnosti_new

while (@@fetch_status<>-1)

BEGIN

        SELECT @Pocet_shodnych_radku = COUNT(*)
        FROM [KU_1].dbo.SU_CI_DObce
        WHERE Kod_castiobce = @Kod_castiobce_new

        IF @Pocet_shodnych_radku = 0

        BEGIN

                SELECT @ID_Region_SCD = POSLEDNI_CISLO_CASTIOBCE
                FROM KU_1.dbo.SU_CI_CISLA

                INSERT INTO [KU_1].dbo.SU_CI_DObce (ID_region_castiobce,
                Nazev_castiobce, Kod_castiobce, MDX_castiobce, Nazev_obce,
                Kod_obce, Nazev_POU, Kod_POU, Nazev_ORP, Kod_ORP,
                Nazev_okresu, Kod_okresu, Nazev_kraje, Kod_kraje, Nazev_oblasti,
```

```
        Kod_oblasti, Datum_platnosti, Od_data_castiobce,
        Do_data_castiobce, Stav_castiobce)
        VALUES(@ID_Region_SCD, @Nazev_castiobce_new,
        @Kod_castiobce_new, '$Original', @Nazev_obce_new,
        @Kod_obce_new, @Nazev_POU_new, @Kod_POU_new,
        @Nazev_ORP_new, @Kod_ORP_new, @Nazev_okresu_new,
        @Kod_okresu_new, @Nazev_kraje_new, @Kod_kraje_new,
        @Nazev_oblasti_new, @Kod_oblasti_new,  @Datum_platnosti_new,
        @Datum_platnosti_new, '2049', '1')

        UPDATE KU_1.dbo.SU_CI_CISLA SET POSLEDNI_CISLO_CASTIOBCE
        = POSLEDNI_CISLO_CASTIOBCE + 1

END

ELSE

BEGIN

        SET @CRC_Castiobce_new =
        BINARY_CHECKSUM(@Nazev_castiobce_new, @Kod_castiobce_new,
        @Nazev_obce_new, @Kod_obce_new, @Nazev_POU_new,
        @Kod_POU_new, @Nazev_ORP_new, @Kod_ORP_new,
        @Nazev_okresu_new, @Kod_okresu_new, @Nazev_kraje_new,
        @Kod_kraje_new, @Nazev_oblasti_new, @Kod_oblasti_new)

        SELECT @CRC_Castiobce_old  = CRC_Castiobce
        FROM dbo.SU_CI_DObce
        WHERE Kod_castiobce = @Kod_castiobce_new
                AND Stav_castiobce = '1'

        IF      @CRC_Castiobce_old <> @CRC_Castiobce_new


        ---     SELECT @Nazev_castiobce_old = Nazev_castiobce,
        ---     @Kod_castiobce_old = Kod_castiobce, @Nazev_obce_old =
        ---     Nazev_obce, @Kod_obce_old = Kod_obce, @Nazev_POU_old
        =
        ---     Nazev_POU, @Kod_POU_old = Kod_POU, @Nazev_ORP_old =
        ---     Nazev_ORP, @Kod_ORP_old = Kod_ORP, @Nazev_okresu_old
        =
        ---     Nazev_okresu, @Kod_okresu_old = Kod_okresu,
        ---     @Nazev_kraje_old = Nazev_kraje, @Kod_kraje_old =
        Kod_kraje

        ---     IF      @Nazev_castiobce_old <> @Nazev_castiobce_new
```

```
---             OR @Nazev_obce_old <> @Nazev_obce_new
---             OR @Kod_obce_old <> @Kod_obce_new
---             OR @Nazev_POU_old <> @Nazev_POU_new
---             OR @Kod_POU_old <> @Kod_POU_new
---             OR @Nazev_ORP_old <> @Nazev_ORP_new
---             OR @Kod_ORP_old <> @Kod_ORP_new
---             OR @Nazev_okresu_old <> @Nazev_okresu_new
---             OR @Kod_okresu_old <> @Kod_okresu_new
---             OR @Nazev_kraje_old <> @Nazev_kraje_new
---             OR @Kod_kraje_old <> @Kod_kraje_new


BEGIN

        SELECT @MDX_string = ''

        declare MDX_string cursor for
        SELECT ID_region_castiobce FROM [KU_1].dbo.SU_CI_DObce
        WHERE Kod_castiobce = @Kod_castiobce_new

        open MDX_string

        fetch next from MDX_string into @ID_region_old

        while (@@fetch_status<>-1)

        BEGIN

            IF LEN(@MDX_string)>0 SELECT @MDX_string =
            @MDX_string + ','

            SELECT @MDX_string = @MDX_string +
            '[Dimenze].[Level].&[' +
            RTRIM(CONVERT(char(1000),@ID_region_old)) + ']'

        fetch next from MDX_string into @ID_region_old

        END

        close MDX_string

        deallocate MDX_string

SELECT @ID_Region_SCD = POSLEDNI_CISLO_CASTIOBCE
FROM KU_1.dbo.SU_CI_CISLA
```

```
                SELECT @MDX_string = '{[Dimenze].[Level].&[' +
                RTRIM(CONVERT(char(1000),@ID_Region_SCD)) + '],' +
                @MDX_string + '}'

                UPDATE [KU_1].dbo.SU_CI_DObce SET Do_data_castiobce =
                convert(smalldatetime,'31.12.'+convert(varchar(4),(YEAR(@Datum_
                platnosti_new)-1)),104), Stav_castiobce = '2', MDX_castiobce =
                '$Moved'
                 WHERE Kod_castiobce = @Kod_castiobce_new

                INSERT INTO [KU_1].dbo.SU_CI_DObce (ID_region_castiobce,
                Nazev_castiobce, Kod_castiobce, MDX_castiobce, Nazev_obce,
                Kod_obce, Nazev_POU, Kod_POU, Nazev_ORP, Kod_ORP,
                Nazev_okresu, Kod_okresu, Nazev_kraje, Kod_kraje, Nazev_oblasti,
                Kod_oblasti, Datum_platnosti, Od_data_castiobce,
                Do_data_castiobce, Stav_castiobce) VALUES(@ID_Region_SCD,
                @Nazev_castiobce_new, @Kod_castiobce_new,  @MDX_string,
                @Nazev_obce_new, @Kod_obce_new, @Nazev_POU_new,
                @Kod_POU_new, @Nazev_ORP_new, @Kod_ORP_new,
                @Nazev_okresu_new, @Kod_okresu_new, @Nazev_kraje_new,
                @Kod_kraje_new, @Nazev_oblasti_new, @Kod_oblasti_new,
                @Datum_platnosti_new, @Datum_platnosti_new, '2049', '1')

                UPDATE KU_1.dbo.SU_CI_CISLA SET POSLEDNI_CISLO_CASTIOBCE
                = POSLEDNI_CISLO_CASTIOBCE + 1


               END

        END

fetch next from Region_new into @Nazev_castiobce_new, @Kod_castiobce_new,
@Nazev_obce_new, @Kod_obce_new, @Nazev_POU_new, @Kod_POU_new,
@Nazev_ORP_new, @Kod_ORP_new, @Nazev_okresu_new, @Kod_okresu_new,
@Nazev_kraje_new, @Kod_kraje_new, @Nazev_oblasti_new, @Kod_oblasti_new,
@Datum_platnosti_new

END

close Region_new

deallocate Region_new
```

```
UPDATE [dbo].[ST_KR_FKROK]
SET [dbo].[ST_KR_FKROK].[ID_region] = [dbo].[SU_CI_Dobcef].[ID_region]
FROM [dbo].[ST_KR_FKROK] LEFT JOIN [dbo].[SU_CI_Dobcef] ON
        [dbo].[ST_KR_FKROK].[krok] = [dbo].[SU_CI_Dobcef].[Kod_obce]
                and  ([dbo].[SU_CI_Dobcef].Od_data_obce
                                <=convert(smalldatetime,'01.01.'+rok,104)
                and convert(smalldatetime,'01.01.'+rok,104)
                                <=[dbo].[SU_CI_Dobcef].Do_data_obce)
GO

UPDATE [dbo].[ST_KR_FKROK_MOS]
SET [dbo].[ST_KR_FKROK_MOS].[ID_region] = [dbo].[SU_CI_Dobcef].[ID_region]
FROM [dbo].[ST_KR_FKROK_MOS] LEFT JOIN [dbo].[SU_CI_Dobcef] ON
        [dbo].[ST_KR_FKROK_MOS].[krok] = [dbo].[SU_CI_Dobcef].[Kod_obce]
                and  ([dbo].[SU_CI_Dobcef].Od_data_obce
                                <=convert(smalldatetime,'01.01.'+rok,104)
                and convert(smalldatetime,'01.01.'+rok,104)
                                <=[dbo].[SU_CI_Dobcef].Do_data_obce)
```

Krok „Update_CIS“:

```
declare @Asl_id_new int
declare @TypAas_new varchar (9)
declare @Klc_new varchar(6)
declare @Hdn1_new varchar(25)
declare @DTZM_new datetime

declare @Typ_old varchar (25)
declare @Name_old varchar(25)

declare @Pocet_shodnych_radku int

declare @Asl_id_max_old int
declare @Asl_id_max_new int

declare @Cis_id_SCD int


SELECT @Asl_id_max_old = [LastAsl_Cis]
FROM [AASCD_1].[dbo].[DW_PIMPORT]

declare Ciselnik_new cursor for

SELECT CDCis.Asl_id, TypAas, Klc, Hdn1, DatUct
FROM dbo.CDCis JOIN dbo.CDCisTyp ON
        CDCis.CisTyp_id = CDCisTyp.CisTyp_id JOIN dbo.CDAsl ON
        CDCis.Asl_id = CDAsl.Asl_id
        WHERE TypAas IS NOT NULL
                AND CDCis.Asl_id > @Asl_id_max_old

open Ciselnik_new

fetch next from Ciselnik_new into @Asl_id_new, @TypAas_new, @Klc_new,
@Hdn1_new, @DTZM_new

while (@@fetch_status<>-1)

BEGIN

        SELECT @Pocet_shodnych_radku = COUNT(*)
        FROM AASCD_1.dbo.DW_DCis
        WHERE Cis_nid = @Klc_new AND Cis_name = @TypAas_new

        IF @Pocet_shodnych_radku = 0
```

```
BEGIN

        SELECT @Cis_id_SCD = LastCis_SCD
        FROM AASCD_1.dbo.DW_PIMPORT

        INSERT INTO AASCD_1.dbo.DW_DCis ([Asl_id], [Cis_name],
        [Cis_id], [Cis_nid], [Typ], [Nazev], [Datum_od], [Datum_do],
        [Datum_run])
        VALUES(@Asl_id_new, @TypAas_new, @Cis_id_SCD, @Klc_new,
        'Neudáno', @Hdn1_new, @DTZM_new, '2199-01-01', getdate())

        UPDATE AASCD_1.dbo.DW_PIMPORT SET LastCis_SCD =
        LastCis_SCD + 1

END

ELSE

BEGIN

        SELECT @Name_old = Nazev
        FROM AASCD_1.dbo.DW_DCis
        WHERE [Cis_nid] = @Klc_new
                AND [Cis_name] = @TypAas_new
                AND [Datum_do] = '2199-01-01'

        IF      @Hdn1_new <> @Name_old

        BEGIN

        SELECT @Cis_id_SCD = LastCis_SCD FROM
        AASCD_1.dbo.DW_PIMPORT

        SELECT @Typ_old = Typ FROM AASCD_1.dbo.DW_DCis WHERE
        [Cis_nid] = @Klc_new AND [Cis_name] = @TypAas_new AND
        [Datum_do] = '2199-01-01'

        UPDATE AASCD_1.dbo.DW_DCis SET [Datum_do] = @DTZM_new -
        1 WHERE [Cis_nid] = @Klc_new AND [Cis_name] = @TypAas_new
        AND [Datum_do] = '2199-01-01'

        INSERT INTO AASCD_1.dbo.DW_DCis ([Asl_id], [Cis_name],
        [Cis_id], [Cis_nid], [Typ], [Nazev], [Datum_od], [Datum_do],
        [Datum_run]) VALUES(@Asl_id_new, @TypAas_new, @Cis_id_SCD,
        @Klc_new, @Typ_old, @Hdn1_new, @DTZM_new, '2199-01-01',
        getdate())
```

```
                UPDATE AASCD_1.dbo.DW_PIMPORT SET LastCis_SCD =
                LastCis_SCD + 1

                END

        END

fetch next from Ciselnik_new into @Asl_id_new, @TypAas_new, @Klc_new,
@Hdn1_new, @DTZM_new

END

close Ciselnik_new
deallocate Ciselnik_new

SELECT @Asl_id_max_new = MAX(Asl_id)
FROM CDCis

UPDATE AASCD_1.dbo.DW_PIMPORT SET LastAsl_CIS = @Asl_id_max_new
```

```
UPDATE AASCD_1..DW_FEMISEOBD
SET PravniForma_id = IsNull(AASCD_1..DW_DPravniForma.PravniForma_id,9999)
FROM AASCD_1..DW_FEMISEOBD LEFT JOIN AASCD_1..DW_DPravniForma ON
        AASCD_1..DW_FEMISEOBD.PravniForma_nid=AASCD_1..DW_DPravniFor
        ma.PravniForma_nid
        AND
        AASCD_1..DW_FEMISEOBD.DaTZM>=AASCD_1..DW_DPravniForma.Datu
        m_Od
        AND
        AASCD_1..DW_FEMISEOBD.DaTZM<=AASCD_1..DW_DPravniForma.Datu
        m_Do
GO

UPDATE AASCD_1..DW_FEMISEOBD
SET TypCP_id = IsNull(AASCD_1..DW_DTypCP.TypCP_id,9999)
FROM AASCD_1..DW_FEMISEOBD LEFT JOIN AASCD_1..DW_DTypCP ON
      AASCD_1..DW_FEMISEOBD.TypCP_nid=AASCD_1..DW_DTypCP.TypCP_nid
      AND
      AASCD_1..DW_FEMISEOBD.DaTZM>=AASCD_1..DW_DTypCP.Datum_Od
      AND
      AASCD_1..DW_FEMISEOBD.DaTZM<=AASCD_1..DW_DTypCP.Datum_Do
GO

UPDATE AASCD_1..DW_FEMISEOBD
SET PodobaCP_id = IsNull(AASCD_1..DW_DPodobaCP.PodobaCP_id,9999)
FROM AASCD_1..DW_FEMISEOBD LEFT JOIN AASCD_1..DW_DPodobaCP ON
      AASCD_1..DW_FEMISEOBD.PodobaCP_nid=AASCD_1..DW_DPodobaCP.Pod
      obaCP_nid
      AND
      AASCD_1..DW_FEMISEOBD.DaTZM>=AASCD_1..DW_DPodobaCP.Datum_O
      d
      AND
      AASCD_1..DW_FEMISEOBD.DaTZM<=AASCD_1..DW_DPodobaCP.Datum_D
      o
GO

UPDATE AASCD_1..DW_FEMISEOBD
SET MistoRegistrace_id =
IsNull(AASCD_1..DW_DMistoReg.MistoRegistrace_id,9999)
FROM AASCD_1..DW_FEMISEOBD LEFT JOIN AASCD_1..DW_DMistoReg ON
      AASCD_1..DW_FEMISEOBD.MistoRegistrace_nid=AASCD_1..DW_DMistoReg
      .MistoRegistrace_nid
      AND
      AASCD_1..DW_FEMISEOBD.DaTZM>=AASCD_1..DW_DMistoReg.Datum_O
      d AND
```

```
        AASCD_1..DW_FEMISEOBD.DaTZM<=AASCD_1..DW_DMistoReg.Datum_D
        o
GO

UPDATE AASCD_1..DW_FEMISEOBD
SET TypNomHodnota_id =
IsNull(AASCD_1..DW_DTypNomHod.TypNomHodnoty_id,9999)
FROM AASCD_1..DW_FEMISEOBD LEFT JOIN AASCD_1..DW_DTypNomHod ON
        AASCD_1..DW_FEMISEOBD.TypNomHodnota_nid=AASCD_1..DW_DTypNo
        mHod.TypNomHodnoty_nid
        AND
        AASCD_1..DW_FEMISEOBD.DaTZM>=AASCD_1..DW_DTypNomHod.Datum
        _Od
        AND
        AASCD_1..DW_FEMISEOBD.DaTZM<=AASCD_1..DW_DTypNomHod.Datum
        _Do
GO

UPDATE AASCD_1..DW_FEMISEOBD
SET SkupEms_id = IsNull(AASCD_1..DW_DKodSkuEm.KodSkuEm_id,9999)
FROM AASCD_1..DW_FEMISEOBD LEFT JOIN AASCD_1..DW_DKodSkuEm ON
        AASCD_1..DW_FEMISEOBD.SkupEms_nid=AASCD_1..DW_DKodSkuEm.Kod
        SkuEm_nid
        AND
        AASCD_1..DW_FEMISEOBD.DaTZM>=AASCD_1..DW_DKodSkuEm.Datum_
        Od AND
        AASCD_1..DW_FEMISEOBD.DaTZM<=AASCD_1..DW_DKodSkuEm.Datum_
        Do
GO
UPDATE AASCD_1..DW_FEMISEOBD
SET Prevoditelnost_id = IsNull(AASCD_1..DW_DPrevod.Prevoditelnost_id,9999)
FROM AASCD_1..DW_FEMISEOBD LEFT JOIN AASCD_1..DW_DPrevod ON
        AASCD_1..DW_FEMISEOBD.Prevoditelnost_nid=AASCD_1..DW_DPrevod.Pr
        evoditelnost_nid
        AND
        AASCD_1..DW_FEMISEOBD.DaTZM>=AASCD_1..DW_DPrevod.Datum_Od
        AND
        AASCD_1..DW_FEMISEOBD.DaTZM<=AASCD_1..DW_DPrevod.Datum_Do
GO

UPDATE AASCD_1..DW_FEMISEOBD
SET Stat_id = IsNull(AASCD_1..DW_DKodStatu.KodStatu_id,9999)
FROM AASCD_1..DW_FEMISEOBD LEFT JOIN AASCD_1..DW_DKodStatu ON
        AASCD_1..DW_FEMISEOBD.Stat_nid=AASCD_1..DW_DKodStatu.KodStatu_
        nid AND
        AASCD_1..DW_FEMISEOBD.DaTZM>=AASCD_1..DW_DKodStatu.Datum_O
```

```
        d AND
        AASCD_1..DW_FEMISEOBD.DaTZM<=AASCD_1..DW_DKodStatu.Datum_Do
GO

UPDATE DW_FEMISEOBD
SET JednotkaObchodu_Id =
IsNull(DW_DJednotkaObchodu.JednotkaObchodu_ID,9999)
FROM DW_FEMISEOBD LEFT JOIN DW_DJednotkaObchodu ON
        DW_FEMISEOBD.JednotkaObchodu_nid=DW_DJednotkaObchodu.Jednotka
        Obchodu_nid
        AND DW_FEMISEOBD.DaTZM>=DW_DJednotkaObchodu.Datum_Od
        AND DW_FEMISEOBD.DaTZM<=DW_DJednotkaObchodu.Datum_Do
```