VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

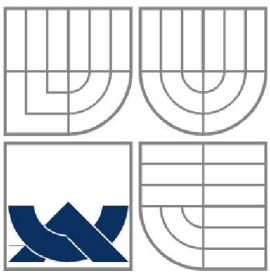
DATOVÝ SKLAD SE ZAMĚŘENÍM NA OPTIMALIZACI ETL PROCESU

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

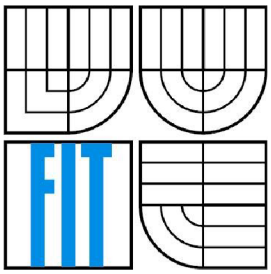
AUTOR PRÁCE
AUTHOR

Bc. IVAN VESELÝ

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

DATOVÝ SKLAD SE ZAMĚŘENÍM NA OPTIMALIZACI ETL PROCESU

DATA WAREHOUSE WITH A FOCUS ON THE ETL PROCESS OPTIMIZATION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. IVAN VESELÝ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. MICHAL ŠEBEK

BRNO 2011

Abstrakt

Cílem této práce je vytvořit datový sklad a proces jeho plnění. Datový sklad je vytvořen pro informační systém Tempo IS, z jehož provozních databází jsou data čerpána procesem ETL. Proces ETL umí pracovat se zrušenými daty ve zdrojové databázi, s tím že data jsou zachována v datovém skladu a z odmazaných dat budou ve zdrojových datech vypočteny odpovídající počáteční stavy. Celý projekt datového skladu je vytvořen jako open source pod licencí GPLv2. Datový sklad je reprezentován relační databází Firebird. Jako OLAP server je použit nástroj Mondrian a k přesunu dat do datového skladu procesem ETL nástroj PDI. Mondrian i PDI jsou součástí většího balíku nástrojů Pentaho. Uživatelské rozhraní je implementováno pomocí nástroje Jpivot a ovládáno z webového prohlížeče.

Abstract

This thesis deals with creating a data warehouse and process of its implementation. The data warehouse is created for the economic information system Tempo IS, from its databases is gathered by ETL process. ETL process allows working with deleted data from the source database; deleted data are stored in data warehouse together with creating new data in the source database which are recalculated corresponding to the initial records. The entire project repository is created as open source under GPLv2. The data warehouse is represented by a relational database Firebird. As an OLAP server is used Mondrian and for transferring data into the data warehouse by ETL process is used PDI. Mondrian and PDI are a part of a larger package called Pentaho. Web-based user interface is implemented using Jpivot.

Klíčová slova

Datový sklad, Datový trh, OLAP, ETL, Dimenze, Fakta

Keywords

Data warehouse, Data mart, OLAP, ETL, Dimension, Facts

Citace

Ivan Veselý: Datový sklad se zaměřením na optimalizaci ETL procesu, diplomová práce, Brno, FIT VUT v Brně, 2011

Datový sklad se zaměřením na optimalizaci ETL procesu

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Michala Šebka
Další informace mi poskytli Ing. Petr Chmelař a Ing. David Petrík
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Ivan Veselý
Datum 7.2.2011

Poděkování

Děkuji Ing. Michala Šebkovi a Ing. Petru Chmelařovi za jejich pomoc a rady. Dále děkuji Ing. Davidu Petríkovi za jeho rady při návrhu datového skladu.

Obsah

1 Úvod

Uchovávání vědomostí a nalézání spojitostí v těchto vědomostech je jednou z činností, která činí člověka jedinečným a poskytuje mu konkurenční výhodu v prostředí, kde žije. Tento předpoklad lze aplikovat i na větší společenství jako jsou společnosti obchodní. Tyto obchodní společnosti se většinou nalézají v konkurenčním prostředí, kde rychlé a správné rozhodnutí, rychlá a správná reakce na změny, představuje podstatnou výhodu na trhu. Tohoto lze dosáhnout zejména tak, že máme ve správnou chvíli správné informace. Takovéto informace můžeme nazvat strategické. Naskýtá se ale otázka, kdo má takové informace znát. Zřejmě by to měli být ti, kteří rozhodují nebo nesou zodpovědnost.

Společnosti data uchovávají v různých provozních systémech, přičemž mohou být uloženy v různých podobách, v papírové podobě (různé šanony), což je v dnešní době pro větší společnosti nemyslitelné, nebo v podobě elektronické či kombinované. Takovýchto „hrubých" dat společnost za dobu svého fungování vyprodukuje mnoho, avšak jsou většinou využitá v nedostatečné míře, tedy množství strategických informací získaných z těchto dat je minimální a jedná se většinou o informace viditelné na první pohled.

Jako příklad dat ukládaných do provozního informačního systému obchodní společnosti mohou sloužit data o prodaném zboží. Na první pohled tato data pouze obsahují informaci o tom, kolik bylo prodáno různého sortimentu. Pokud se však na tato data podíváme z globálního pohledu, můžeme v souvislosti s jinými daty vysledovat různé trendy, které by jinak zůstaly skryty. Ještě více informací získáme, nabídneme li zákazníkům například klubovou kartu, kterou budou předkládat při placení. Nyní máme relaci mezi zákazníkem a prodaným zbožím a díky tomu můžeme vysledovat jaké zboží a v jakém množství zákazník nakupuje s jiným zbožím v daném období, nebo jaké zboží nakupuje zákazník z dané oblasti či jiné důležité informace.

Abychom tyto dotazy mohli zodpovědět dostatečně přesně, je nutné mít tato data v dostatečně dlouhém historickém horizontu a tedy velice rozsáhlou databázi. Provozní informační systém není vytvořen k tomu, aby tato historická data, která během svého fungování nasbíral, archivoval a nakonec je ještě interpretoval uvedeným způsobem. Je určen k tomu, aby rychle a správně prováděl operace denní potřeby. Jeho běh by byl výrazně zpomalen, pokud by měl splňovat všechny výše uvedené požadavky. Proto je k této interpretaci dat nutné ukládat je ve vhodné struktuře, která ale není vhodná pro uložení dat provozního informačního systému. Pro získávání těchto komplexních informací slouží nástroje pro podporu rozhodování (DSS) neboli nástroje business inteligence (BI). Tyto nástroje získávají data ze speciálního úložiště, fyzicky i logicky odděleného od databází provozních informačních systémů. Tímto úložištěm je datový sklad (dále jen DS), který je plněn daty z těchto provozních informačních systémů a dalších zdrojů tak, aby z něj bylo možné pomocí speciálních nástrojů získat relevantní informace. Tato diplomová práce se zabývá vytvořením a plněním takového datového skladu.

V úvodní kapitole je uvedeno, proč je vhodné budovat datový sklad a kde se vyplatilo datové sklady budovat. V druhé kapitole je vysvětlen termín OLAP a jeho vztah k datovému skladu. Kapitola tři pojednává o architekturách datových skladů a kapitola čtvrtá pak informuje o způsobech jak k budování datového skladu přistupovat. V páté kapitole je pojednáno jaká, data jsou v datovém skladu uchovávána a v šesté kapitole je vysvětleno, jak datový sklad těmito daty plnit. V sedmé kapitole jsou popsány nástroje, které byly použity při implementaci DS. Kapitola osm popisuje samotnou implementaci DS a kapitola devět implementaci procesu ETL. Zhodnocení celé diplomové práce je v kapitole desáté.

1.1 Definice datového skladu

Datový sklad v originále Data Warehouse (DS) je definován mnoha způsoby dle [4], tři z nich uvedu zde:

- Barry Devlin popisuje datový sklad jako jediný, kompletní a konzistentní sklad dat získaných z různých zdrojů dostupný koncovým uživatelům tak, aby jim rozuměli a dokázali je použít v kontextu společnosti.
- Další definice datového skladu dle Billa Inmona je taková, že DS je kolekce subjektově orientovaných, integrovaných, stálých a časově proměnných dat k podpoře rozhodování.
- Jako poslední zde uvedu definici datového skladu od Kena Orra, který tvrdí, že DS je prostředek k získání snadného přístupu ke kvalitním a integrovaným datům společnosti kvalifikovaným i nekvalifikovaným koncovým uživatelům.

V diplomové práci se pak budu držet zejména druhé definice.

1.2 Datové skladování

V literatuře je často pojem datový sklad používán jak pro samotné úložiště dat, tak pro celý proces zahrnující plnění datového skladu, skladování dat a jejích následné získávání z datového skladu. V této práci se budu striktně držet toho, že datový sklad je úložiště dat a samotný komplex procesů plnění datového skladu, skladováni dat a získávání dat z datového skladu budu nazývat datové skladování.

Pojem datové skladování v originále data warhousing je dle [5] celkové řešení zahrnující návrh a implementaci nástrojů, procesů a služeb k zajištění doručení přesných srozumitelných, kompletních a včasných dat potřebných k tvorbě rozhodnutí.

1.3 Historie datového skladování

V této kapitole věnující se historii databází a datových skladů, jejichž vývoj se navzájem prolíná, čerpám informace zejména z [1] a [4].

V roce 1961 vyvinul Charles Bachman ve společnosti General Electric první úspěšný systém správy databáze (IDS). Systém fungoval pouze na počítačích této společnosti, přičemž všechna data byla uložena v jednom souboru (flat file) a veškeré generování datových tabulek bylo nutné provádět ručně. Jeden ze zákazníku General Electric, společnost BF Goodrich Chemical se pokusila upravit systém IDS tak, aby byl více použitelný, a výsledek nazvala integrated data management systém (IDMS) .

Roku 1968 IBM spustila hierarchickou databázi IMS pro jejich sálové počítače a výzkumník této společnosti Edgar F. Codd pracoval na vylepšení organizace databáze. V roce 1969 přišel s myšlenkou relační databáze organizované do tabulek. IBM spustila projekt s kódovým názvem Systém/R. Jelikož se IBM věnovala přednostně systému IMS, byl Systém/R do roku 1980 bez reálného produktu. Mezitím na univerzitě v Berkeley Michael Stonebraker a Eugene Wong použily veřejně dostupné informace z projektu Systém/R k započetí práce na jejich vlastní relační databázi a projektu Ingres. Na rozdíl od IBM, ingres zveřejnil zdrojové kódy projektu na kazetách a tyto zdrojové kódy byly použity jako základ mnoha komerčních relačních databází.

V pozdních létech 1960 se také pracovalo na vývoji nového druhu databázového softwaru, systému pro podporu rozhodování (DSS). Smyslem tohoto systému bylo usnadnit manažerům jejich rozhodování na základě dat.

V roce 1985 byl vyvinut první systém „business inteligence“ pro společnost Procter & Gamble společností Metaphor Computer Systems. V tomto roce se na univerzitě v Berkeley změnil projekt ingres na Postgres s cílem vyvinout první objektově orientovanou databázi.

Během roku 1988 výzkumníci IBM, Barry Devlin a Paul Murphy vytvořili termín „information warehouse“. V roce 1991 W. H. "Bill" Inmon uvedl datové sklady do praxe, když publikoval knihu Building the Data Warehouse (John Wiley & Sons).

Datové skladování se tedy vyvinulo skrze potřebu společností dělat strategická rozhodnutí. Během let 1970 až 1980 společnosti ve velkém implementovaly provozní systémy. Vlivem zrychlování a zvětšování kapacity medií (DASD) společnosti uchovávají ohromná množství dat za dlouhý časový úsek. Brzy však zjišťují, že mají mnoho různorodých dat, ale neumějí je správně pospojovat a získat z nich informace důležité pro rozhodování.

V letech 1980 a začátkem let 1990 již jsou dostupné osobní počítače, jazyky 4 generace a nástroje Online Analytical Processing (OLAP). Uživatelé požadují větší kontrolu nad jejich daty, a proto jednotlivá oddělení společností požadují data specifická pro jejich oddělení. Data jsou tedy extrahována z jednoho zdroje, ale po několika takových extrakcích byla údržbu takového systému téměř nemožná. Jedním z řešení tohoto problému se stává datový sklad.

1.4 Důvody k tvorbě datových skladů

Provozní informační systémy nasazené v podnicích dle [2] dobře reagují na otázky denní potřeby, tedy automatizace provozních činností a obchodních procesů. Příkladem takových provozních činností může být například vystavení objednávky, párování plateb od zákazníka, příjem zboží na sklad a mnoho dalších činností. Tyto činnosti generují obrovské množství dat, ze kterých je možné získat mnoho cenných informací.

Běžný informační systém bez datového skladu však tyto data neuchovává. Jakmile se stávají historickými, jsou maximálně přehrávána do archivu, kde je již jejich využití minimální. Toto se děje zejména z důvodu výkonnosti informačního systému a také proto, že běžný uživatel takového provozního systému tato historická data s velkou pravděpodobností nepotřebuje.

Pro rozhodování a analýzy jsou však tato data nezbytná a s tímto přístupem se stávají nedostupnými. Pokud jsou tato historická data přece jen k dispozici, pak dle [2] pravděpodobně provozní informační systém neposkytuje dostatečné techniky a nástroje pro zpracování těchto dat. Běžný provozní systém disponuje pouze sadou předem připravených sestav, které se spíše orientují na jednotlivé transakce než na globální pohled.

Další problém nastává, když máme data uložena i mimo informační systém. Pokud bychom chtěli data analyzovat i s daty z těchto externích zdrojů, bylo by nutné je nějakým způsobem do systému naimportovat, nehledě na to, že není možné pokrýt a dohledat, která data budou v budoucnu potřebná pro analýzy a připravit pro ně vhodné mechanismy pro import.

Tyto problémy kladou velké požadavky na výkon provozního systému a bylo by nevhodné ho jimi zatěžovat. Proto je dobré vytvořit fyzicky a logicky oddělený datový sklad, který uchovává data důležitá pro analýzy včetně historických dat a který poskytuje vhodné nástroje pro jejich zpracování. Data jsou do tohoto datového skladu přesunována pomocí speciálního souboru procesů, které se starají o jejich správnost.

1.4.1 Důvody ke zbudování datového skladu z pohledu uživatele

Dle [7] jsou důvody ke zbudování DS z pohledu jeho uživatele tyto:

- **Všechna data na jednom místě** – není nutné procházet mnoho zdrojů dat k získání informací, není ani nutné kombinovat množství dat ručně, protože všechna potřebná data již jsou integrovaná.
- **Aktuální informace** – Data v DS jsou automaticky aktualizována, což znamená, že uživatel má vždy poslední informace.
- **Rychlý přístup** – DS je optimalizován pro rychlé čtení dat.
- **Bez omezení na množství dat** – tabulkové procesory jsou schopné pojmout omezené množství dat a mnohdy musí být rozděleny na několik částí. Datový sklad pojme téměř neomezené množství záznamů v závislosti na použitém databázovém systému.

- **Dostupná historie** – DS neobsahuje pouze aktuální informace, ale informace získané během let. To znamená, že datový sklad je ideální k zjišťování různých trendů a porovnávání historických dat. Jak bude později ukázáno, data z datového skladu nejsou téměř nikdy mazána. DS mnohdy obsahuje i více informací než zdroj, protože DS udržuje i změny nad daty.
- **Jednoduchý k porozumění** – Výstupy DS jsou tvořeny tak, aby odpovídaly termínům, které se používají v cílové organizaci.
- **Jednotné definice** – Všichni ve společnosti používají stejné definice, což vylučuje diskuse a problémy v pohledu na pochopení smyslu použitého výrazu.
- **Standardizovaná data** – všechna data v datovém skladu jsou vždy daného typu. Vylučuje problémy typu: Muž/Žena, m/w, 0/1.

Tento seznam výhod DS dle [7] staví na předpokladu, že je datový sklad budován správným způsobem. I když je implementace technické části (prvních pět bodů seznamu) perfektní, může být celý projekt budování DS neúspěšný z pohledu uživatele (poslední tři body seznamu), protože jsou například přehlíženy. Toto nastává v případě, když je DS implementován bez dostatečné vazby a účasti uživatele.

1.4.2 Oblasti, ve kterých se společnostem vyplatilo použití datového skladování

Společnosti, které na začátku devadesátých let minulého století začaly budovat datové sklady, si dle [6] musely odpovědět na otázku, co přesně znamená strategická informace? Jsou to takové informace, díky kterým bude vedení společnosti schopno formulovat obchodní strategie, stanovit cíle a sledovat výsledky. Obchodním cílem může být dle [6] například udržení nynější základny zákazníků, zvýšení základny zákazníků o 15% během příštích 5 let, získání dalších 10% podílu na trhu během 3 let a mnoho dalších strategických cílů.

Tabulka 1.1 dle [6] ukazuje příklad odvětví, kde datové skladování přineslo reálné výsledky v podobě strategických informací.

Odvětví	Nasazení procesů datového skladování
Prodej	Věrnost zákazníků, analýza nákupního košíku
Finančnictví	Plánování rizika, zjišťování finančních podvodů
Aerolinky	Analýza zisků, zjištění návratnosti leteckých tras
Výroba	Snížení ceny výroby, řízení logistiky
Služby	Řízení aktiv, řízení zdrojů
Vláda	Plánování pracovních sil, cenová regulace

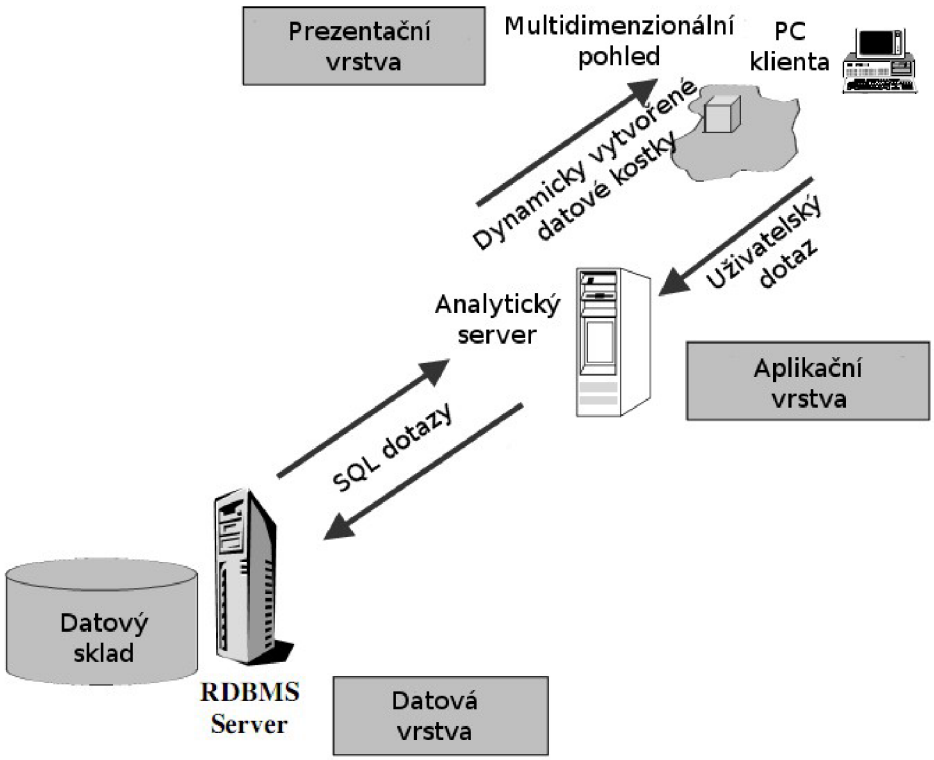
Tabulka 1.1: Použití datového skladování v různých odvětvích dle [6]

2 Online Analytical Processing (OLAP)

Tato kapitola se zabývá porovnáním jednotlivých přístupů k analytickému zpracování OLAP, který je někdy zaměňován za datový sklad. Edgar Codd definoval OLAP jako kategorie softwarových nástrojů umožňující analýzu dat uložených v databázi dle [11]. Datový sklad je tedy součástí nástrojů OLAP zprostředkovávající data pro analýzy. Podle toho jak jsou data fyzicky uložena, rozlišujeme ROLAP, MOLAP a HOLAP.

Analytické zpracování dle [6] (analytical processing), někdy také nazývané informační zpracování nebo zpracování pro podporu rozhodování, je přístup ke správě dat, zajišťující podporu pro řízení a strategické rozhodování. Při analytickém zpracování v datech často hledáme různé vzory a trendy přes několik období. Data tedy musí být dostupná za tato zkoumaná období a jsou tedy ze své podstaty historická. Takováto data může poskytovat například datový sklad.

2.1 ROLAP (Relační OLAP)

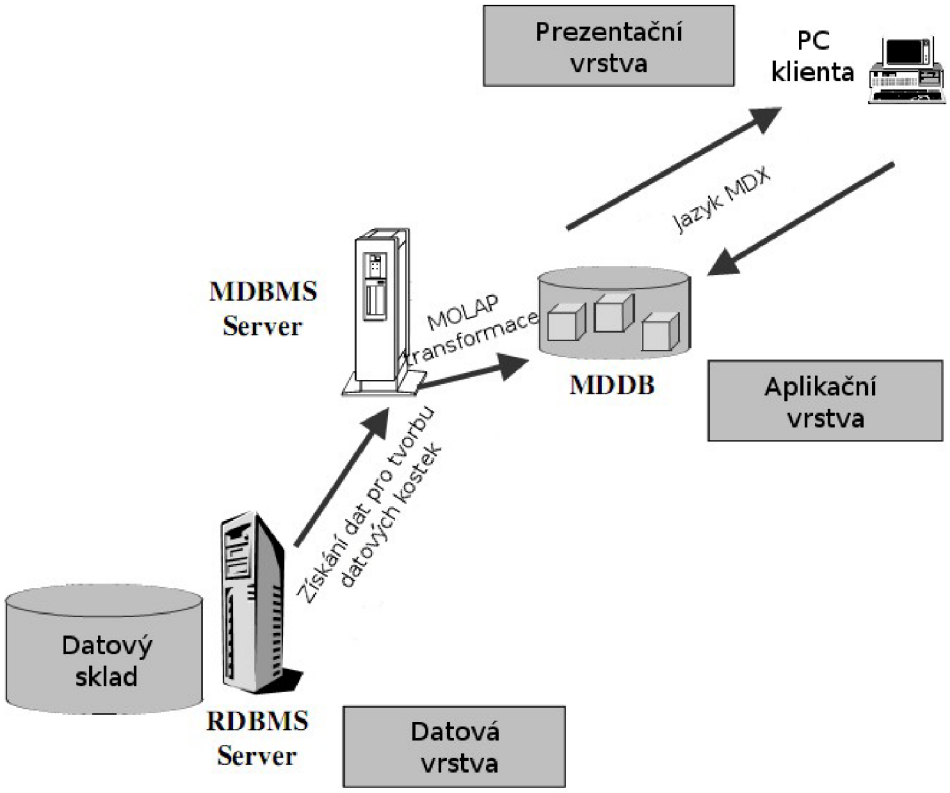


Ilustrace 2.1: Model ROLAP dle [6]

ROLAP používá pro analýzy data z relační databáze. V tomto modelu jsou uživateli prezentována data ve formě více-dimenzionálních kostek. Aby toto bylo možné je dle [6] nutná speciální vrstva metadat, která mapuje tabulky relační databáze na dimenze kostky. Dle dalších metadat je pak určeno kdy a jak použít uložené pohledy pro urychlení různých sumarizací a agregací. Ilustrace 2.1 zobrazuje architekturu ROLAP modelu. Analytický server v aplikační vrstvě architektury vytváří multidimenzionální pohledy nad daty z datového skladu. K tomu používá výše zmíněná metadata. V prezentační vrstvě jsou pak tyto pohledy různým způsobem zobrazovány uživatelům. Když uživatel chce s tímto pohledem manipulovat, je jeho dotaz pomocí analytického serveru

transformován na SQL dotaz. SQL dotaz je datovým skladem zodpovězen a předán serveru, který jej následně opět převede na multidimenzionální pohled a předá prezentační vrstvě.

2.2 MOLAP (Multidimenzionální OLAP)



Ilustrace 2.2: Model MOLAP dle [6]

Tento formát uložení je dle [7] původní OLAP. Data jsou dle [12] načtena z datového skladu a následně jsou mechanismem MOLAP transformována a nahrána do vlastních multidimenzionálních struktur. Během tohoto procesu je spočítáno maximum předběžných výsledků možných z technického hlediska. Data jsou následně uložena v multidimenzionálním formátu spolu se všemi agregovanými hodnotami a indexy uvnitř datové kostky. Dle [11] jsou analýzy nad tímto způsobem uložení rychlejší oproti ROLAP, protože odpadá nutnost transformovat MDX dotazy na SQL. Tento formát uložení dat je výhodnější pro analytické zpracování, oproti ROLAPu je však nutné data z relační databáze transformovat do požadovaného multidimenzionálního formátu.

2.3 HOLAP (Hybridní OLAP)

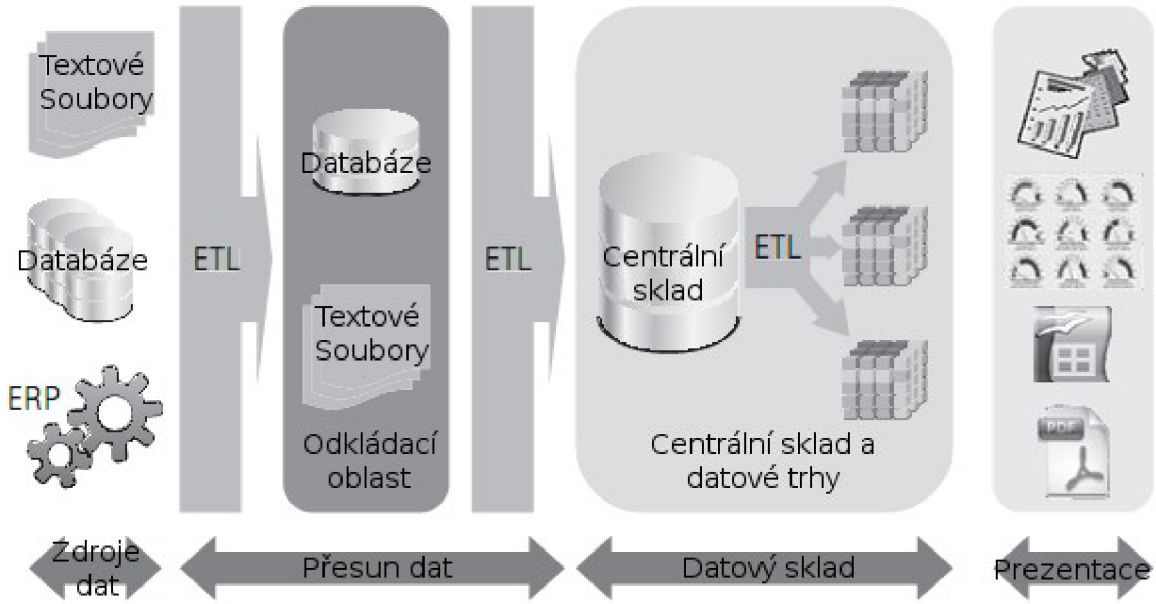
Jde o kombinaci dvou předchozích přístupů MOLAP a ROLAP, agregovaná data jsou uložena uvnitř datové kostky, ale detailní data jsou uložena v relačních tabulkách. Tento přístup dle [12] eliminuje nevýhody obou předchozích přístupů například tak, že při dotazování dochází k ukládání údajů do multidimenzionální paměti cache. Tato data pak mohou být využita při dalších obdobných dotazech.

3 Datové skladování

Tato kapitola se zabývá komponentami procesu datového skladování, jeho architekturou a strukturou samotného datového skladu.

3.1 Vztah komponent datového skladování

Architektura je dle [7] plán, jak přistupovat k problému. Jelikož je proces datového skladování složen ze složitého komplexu komponent, je použití vhodné architektury jeden z předpokladů k jeho úspěšnému vybudování. Obrázek 3.1 znázorňuje logickou strukturu a vztah jednotlivých komponent datového skladování. Neukazuje však fyzickou strukturu, protože některé datové sklady nemusí vůbec obsahovat datové trhy, které mohou být plněny procesem ETL přímo z datových zdrojů atd..



Ilustrace 3.1: Znázornění procesu datového skladování, zdroj [7]

Část zdroje dat na ilustraci znázorňuje několik různých zdrojů pro plnění DS. V části přesun dat dochází k extrakci, transformaci a nahrání dat ze zdrojů do DS. Často tato část obsahuje odkládací oblast, sloužící k dočasnému uskladnění, provedení počátečního vyčištění a transformaci dat před samotným přesunem do datového skladu. Odkládací oblast nemusí být nutně databáze, ale i soubor, který v některých případech dovoluje rychlejší zpracování. Oblast Datový sklad obsahuje centrální datový sklad a případné datové trhy. Část prezentace pak zahrnuje nástroje pro práci se sestavami, prezentacemi obsahu datového skladu, což představuje prezentační vrstvu datového skladování. Kromě uvedeného dělení můžeme rozdělit proces datového skladování dle [7] na takzvaný front office, což je souhrn nástrojů k analýze a zobrazení dat a část takzvané back office, neboli procesu ETL a samotném datovém skladu. Procesu ETL podrobně pojednám v kapitole 6.

3.1.1 Zdroje dat

Zdrojů dat pro datový sklad může být velké množství. Každý ze zdrojů může být na různých médiích v jiném formátu a v jiném kódování. Jedním z hlavních problémů dle [6] je, jak tato rozdílná data sjednotit, transformovat, vyčistit či jinak upravit tak, aby je bylo možné uložit v datovém skladu v takovém formátu, aby poskytoval relevantní informace. Je tedy nutné zvolit vhodné zdroje dat, ze

kterých pak pomocí ETL procesu data získáme, následně je transformujeme a přesuneme do datového skladu. Pokud používáme odkládací oblast, je tento proces ještě rozšířen o ukládání dat do této odkládací oblasti. Zdrojová data můžeme dle [6] rozdělit takto:

- **Produkční data** – samotná zdrojová data můžeme rozdělit na produkční data, což jsou data z různých provozních systémů. Hlavním problémem těchto dat pocházejících z různých provozních systémů je dle [6] jejich nesourodost.
- **Interní data** – dalším zdrojem dat jsou interní data společnosti, což jsou data, která společnost nezískává z provozního informačního systému, ale například z dotazníků. Problémem interních dat je dle [6] zejména to, že bývají uložena ve formě textových souborů a není jednoduché je ve vhodném formátu uložit do datového skladu.
- **Archivní data** – archivní data jsou dalším možným zdrojem, jedná se o data, která jsou pro chod provozního systému již nepotřebná, ale z jiného důvodu je potřebujeme mít k dispozici. Tato data jsou většinou v pravidelných intervalech odstraněna z provozního systému a archivována. Archivovaná data jsou pro datový sklad velice důležitá, protože datový sklad již ze své definice historická data uchovává.
- **Externí data** – posledním zdrojem dat jsou externí data, tedy data získaná z externích zdrojů. Může se jednat například o statistiky získané od společností, které se zaobírají jejich výpočty, či jiné externí data.

3.1.2 Odkládací oblast

Odkládací oblast slouží k dočasnému uložení dat a provedení operací transformace před nahráním do datového skladu. Odkládací oblast obsahuje pouze data, která jsou získána ze zdrojových souborů a po provedení ETL procesu jsou odstraněna. Zbudování odkládací oblasti může být provedeno tak, že se pouze zduplikují struktury zdrojových tabulek bez cizích klíčů a indexů. K vytvoření odkládací oblasti vede dle [7] několik důvodů, jsou to zejména tyto:

- Zatížení zdrojového systému získáváním dat by mělo být zkráceno na minimální možnou dobu, takže je vhodné zkopírovat původní data beze změny a úpravy provádět v odkládací oblasti.
- Použití odkládací oblasti umožňuje lépe pracovat na specifické podmnožině dat, která je potřeba pro aktuální přesun do DS.
- Vyhrazená odkládací oblast umožňuje použití specifických algoritmů třídění, indexování pro podporu ETL procesu.
- Pokud proces transformace z nějakého důvodu zhavaruje, je možné jej znovu spustit od uložené pozice, aniž by bylo nutné provádět znovu extrakci dat z různých zdrojů. Toto by nebylo možné bez odkládací oblasti, protože zdrojové soubory se mohly během doby procesu extrakce a transformace změnit a nebylo by možné použít již spočítané výsledky.

3.1.3 Datový sklad a datový trh

Datový sklad z pohledu Billa Inmona je centralizované úložiště bez redundantních dat. Datový trh je také úložiště dat, které však uchovává informace pouze pro určité zaměření podniku. Příkladem mohou být prodeje zboží nebo náklady na výrobu. Na obsah datového trhu se pak dá nahlížet z různých perspektiv zvaných dimenze. Každá dimenze obsahuje informace, vztahující se k danému pohledu zkoumání, jako jsou čas, zákazníci, produkty a další. Tyto dimenze jsou připojeny k tabulkám faktů obsahujících měřitelné hodnoty jako příjem, cena, počet zaměstnanců a další.

Efektem tohoto je, že uživatel je schopný získávat informace o prodejích za oddělení v určitém čase. Naopak není schopen zjistit, jaké byly náklady na výrobu daného výrobku v daném provozu za minulý rok. Na zjištění této informace již potřebuje jiný datový trh, který může použít některé

dimenze použité v jiných trzích, takzvané sdílené dimenze[1] neboli dle [8] všeobecně přijatých dimenzích.

Datový trh tedy neposkytuje celkový pohled na podnik, ale pouze na jeho část. Datový sklad je pak dle [8] sjednocení těchto datových trhů, avšak toto je splněno pouze za podmínky, že před započetím budování datových trhů je proveden celkový návrh datového skladu a jednotlivé datové trhy jsou budovány z všeobecně přijatých dimenzí a faktů s ohledem na provedený návrh.

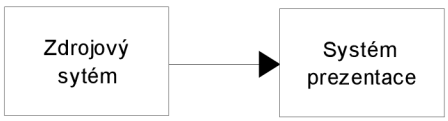
3.1.4 Prezentační vrstva

Prezentační vrstva slouží k zobrazení informací koncovému uživateli ve vhodném formátu. Pokud je datový sklad využíván například nástroji OLAP, jsou informace prezentovány například ve formě multidimenzionálních kostek.

3.2 Architektury datového skladovaní dle komponent

Striktní pravidla, jak by měl proces datového skladování přesně vypadat, neexistují, ale existuje několik architektur, dle kterých můžeme tento proces navrhnout. Jedno z možných členění je podle použitých komponent procesu datového skladování. Tyto architektury prezentují, jaké komponenty proces datového skladování obsahuje a jak jsou propojeny.

3.2.1 Jednovrstvá architektura



Ilustrace 3.2: Jednovrstvá architektura

Jednovrstvá architektura na ilustraci 3.2 neobsahuje fyzický datový sklad, dotazy jsou prováděny přímo nad operačními daty. Tato architektura datového skladování tedy neobsahuje fyzický datový sklad, namísto toho jsou data extrahována přímo ze zdrojových systémů, následně jsou upravena a prezentována uživateli.

3.2.2 Dvouvrstvá architektura

Ilustrace 3.3: Dvouvrstvá architektura

Architektura dvouvrstvá na ilustraci 3.3 již má fyzický datový sklad, ten je ale plněný přímo procesem ETL. V těchto architekturách tedy neexistuje fyzická odkládací oblast. Data jsou pomocí procesu ETL extrahována ze zdrojových systémů, transformována do požadovaného tvaru a následně uložena do datového skladu. Nepřítomnost úložiště při chybě během procesu ETL znamená, že je nutné ho celý opakovat.

---

1 V originále conformed dimension's.

3.2.3 Třívrstvá architektura

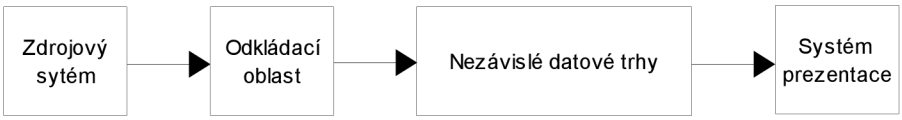
Ilustrace 3.4: Třívrstvá architektura

Přidáním fyzické odkládací oblasti do dvouvrstvé architektury získáme architekturu třívrstvou uvedenou na ilustraci 3.4. Odkládací oblast dle [10] slouží k uložení předem připravených dat a z ní je teprve plněn datový sklad. Toto řešení s odkládací oblastí zajišťuje, že při případné chybě, během ukládání dat do datového skladu není nutné celý proces extrakce a transformace opakovat a tak zbytečně zatěžovat zdrojové systémy poskytující data k plnění datového skladu.

3.3 Architektury datového skladovaní dle struktury datového skladu

Dále pak můžeme architektury dělit podle toho, zda používají datové trhy a v jakém vztahu jsou k datovému skladu. Komponenty v ilustracích 3.5 až 3.7 odpovídají komponentám na ilustraci 3.1 zobrazujícím proces datového skladování a reprezentují třívrstvé architektury.

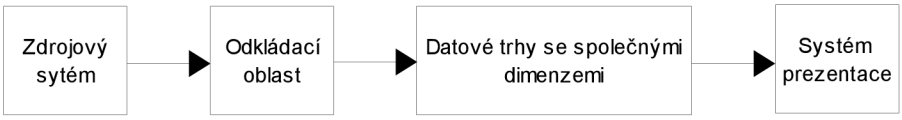
3.3.1 Independent Data Marts



Ilustrace 3.5: Architektura nezávislých datových trhů dle [7]

Každý datový trh je zbudován a plněn nezávisle na ostatních, jednotlivé trhy neví nic o ostatních datových trzích, takže neexistují společná metadata. Jednotlivé datové trhy obsahují atomická a sumarizovaná data. Tento přístup k datovému skladu není příliš vhodné používat, protože datový sklad jako celek prakticky neexistuje. Existují jen nezávislé datové trhy, které nejsou schopny zodpovědět dotazy, které se týkají několika datových trhů zároveň. Tento přístup je uveden na ilustraci 3.5.

3.3.2 Data Mart Bus

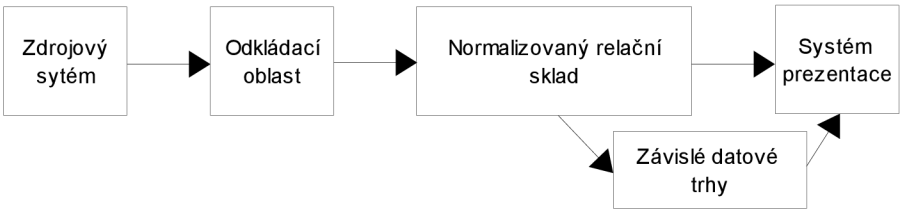


Ilustrace 3.6: Architektura sběrnice dle [7]

Řešení Ralpha Kimbala na ilustraci 3.6 je přesným opakem architektury s oddělenými datovými trhy. Využívá sdílených dimenzí a jednotlivé datové trhy obsahují atomická a sumarizovaná data. Toto řešení je také použito v diplomové práci. I když je nutná jistá počáteční analýza celého datového

skladu, je možné budovat datový sklad této architektury po jednotlivých datových trzích a tak mít dostupné některé výsledky již v průběhu budování datového skladu.

3.3.3 Hub and Spoke



Ilustrace 3.7: Architektura s centrálním datovým skladem a přidruženými trhy dle [7]

Toto je řešení Billa Inmona, obsahuje centrální datový sklad a z něj případně plněné datové trhy. Normalizovaný relační sklad obsahuje atomická data. Volitelné datové trhy pak data atomická a sumarizovaná. Příklad této architektury je uveden na ilustraci 3.7.

3.3.4 Centralized Data Warehouse

Ilustrace 3.8: Architektura s centrálním datovým skladem dle [7]

Tento vzor, uvedený na ilustraci 3.8, je v podstatě stejný jako Hub and Spoke, ale neobsahuje přidané datové trhy. Koncový uživatel je přímo odkazován na centrální datový sklad. Tento přístup je velice podobný architektuře Data Mart Bus, hlavní rozdíl je v normalizaci tabulek použitých v normalizovaném relačním skladu.

3.4 Struktura datového skladu

Ať již používáme pro datové skladování architekturu Sběrnice podle Kimbala nebo Hub and Spoke dle Imnona, je obecně používaným přístupem k modelování struktury datového skladu použití takzvaného schématu hvězdy dle [7]. Dle [13] se jedná o jednoduchou strukturu zejména dvou typů tabulek, mezi nimiž je jasně definované spojení. Tato struktura slouží především k rychlému a přehlednému zobrazení dat v datovém skladu, na rozdíl od normalizovaných struktur používaných v systémech transakčního zpracování, které slouží zejména k rychlému provádění operací vkládání a úpravy záznamů.

Ilustrace 3.9: Schéma struktury hvězdy dle [7]

Název hvězda vychází z tvaru schématu, který ve svém středu obsahuje tabulku faktů, k ní jsou připojeny tabulky dimenzí, podle kterých na obsah tabulky faktů nahlížíme. Toto rozdělení tabulek má svůj důvod v rozdílném obsahu a způsobu použití tabulek faktů a tabulek dimenzí. Tento rozdíl nemusí být vždy patrný a někdy se může stát, že v jedné struktuře hvězdy je jedna tabulka tabulkou faktů a v jiné struktuře tabulkou dimenzí. Ilustrace 3.9 znázorňuje typickou strukturu schématu typu hvězda s tabulkou faktů uprostřed a tabulkami dimenzí v ramenech hvězdy. Dle [6] je velkou výhodu této struktury její snadná pochopitelnost a vhodnost pro zpracování pomocí dotazů. Pokud jsou tabulky ve struktuře hvězdy normalizované, nazývá se taková struktura sněhová vločka. Dle [6] může být v některých případech vhodné provést určitou normalizaci na dimenzích dosahujících velké množství záznamů, čímž je možné ušetřit místo a zrychlit provádění některých dotazů.

3.4.1 Tabulky faktů

Tabulky faktů uchovávají dle [6] počitatelné hodnoty a to na nejnižší možné úrovni agregace. Pokud tabulka faktů obsahuje sumární data, nazývá se tato tabulka agregovanou. Dále tabulky faktů zejména obsahují cizí klíče na všechny tabulky dimenzí, dle kterých na položku tabulky faktů můžeme nazírat. Granularita dat neboli zrnitost hodnot v tabulce faktů určují minimální detail, se kterým jsme schopni na data nahlížet. Odtud plyne požadavek, aby byly data na nejvyšší možné úrovni zrnitosti, protože je vždy možné data sumarizovat, ale není je již vždy možné rozložit na nižší úroveň sumarizace.

Hodnoty v tabulkách faktů mohou být vzhledem k dimenzím plně aditivní, semiaditivní nebo neaditivní. Pokud je hodnota plně aditivní, znamená to, že je agregovatelná podle všech zvolených

dimenzí. Můžeme tedy například sčítat množství výrobků daného typu, které se prodalo za týden, měsíc rok, či kolik se prodalo množství dvou, třech, atd. typů výroků za týden, měsíc atd. Semiaditivní míry jsou agregovatelné jen podle některých dimenzí, naopak míry neaditivní nejdou agregovat vůbec.

Tabulky faktů mohou obsahovat takzvané degenerované dimenze. To jsou atributy, které obsahují hodnoty získané například ze zdrojových tabulek během procesu plnění skladu, pro něž nebylo vhodné tvořit zvláštní dimenzi. Tyto atributy mohou být například pořadové číslo, číslo faktury či jiné. Tabulky faktů nebývají veliké, co se týče množství atributů, ale bývají obrovské, co se týče počtu záznamů.

Agregované tabulky faktů jsou tabulky, které byly předem agregovány na určitou granularitu dle jedné nebo několika dimenzí. Tohoto přístupu se využívá zejména pro urychlení výpočtu nejčastěji používaných dotazů. Při procházení hierarchiemi je pak možné tuto předem spočítanou tabulku využít.

3.4.2 Tabulky dimenzí

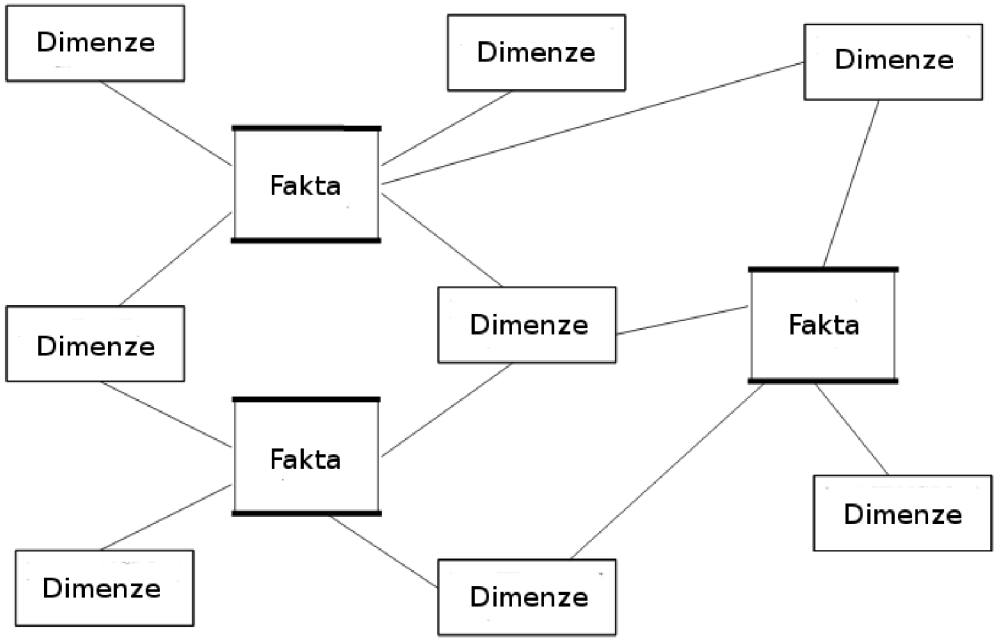
Dimenzionální tabulky představují klíčové metriky, podle nichž jsou posuzovány hodnoty v tabulkách faktů. Určují tedy jejich kontext. Hodnoty z tabulky faktů vždy odkazují na jednu hodnotu z tabulky dimenzí, přičemž více hodnot z tabulky faktů může odkazovat jednu hodnotu v tabulce dimenzí.

Tabulky dimenzí obsahují dle [6] zejména textové atributy, které popisují hodnoty z tabulek faktů. Dále jsou tyto tabulky většinou velice rozsáhlé z pohledu počtu atributů a jednotlivé atributy spolu nemusí mít přímou vazbu. Příkladem mohou být atributy pobočka a barva, toto je způsobeno tím, že tabulky dimenzí nesou normalizované. Tabulky dimenzí obsahují hierarchie, což je další členění v rámci dimenze. Například dimenze pro datum obsahuje nejen datum, ale i atributy jako týden či čtvrtletí. Díky tomu je možné provádět takzvané „drill down“ a „roll up“ operace nad daty, tyto operace provádějí pohyb v těchto hierarchiích. Tato vlastnost významně souvisí s granularitou hodnot v tabulkách faktů. Velikost tabulky dimenzí se pohybuje od několika desítek záznamů až po miliony, ale průměrně jich bývá dle [6] několik set.

Speciálním případem dimenzí jsou takzvané „junk“ dimenze neboli odpadní dimenze, tyto dimenze uchovávají hodnoty, dle kterých sice můžeme nahlížet na data v tabulkách faktů, ale nejsou zařaditelné do žádné „běžné“ dimenze.

3.4.3 Architektura sběrnice

Tuto architekturu dle [7] vyvinul Ralph Kimbal a její podstatou je, že jednotlivé tabulky faktů jsou propojeny všeobecně přijatými dimenzemi. Na rozdíl od schématu hvězdy může obsahovat architektura sběrnice několik tabulek faktů, což hvězda nemůže. Termín všeobecně přijaté dimenze znamená, že při vytváření datového skladu byly dimenze navrženy tak, aby mohly být využity více tabulkami faktů a nebyly nezávislé v rámci jednoho datového trhu. Příkladem dle [6] může být například dimenze produktů mezi dvěma tabulkami faktů, význam atributů této dimenze produktů musí být stejný pro obě tabulky faktů. Dle [7], je nejlepším přístupem při vytváření datového skladu s všeobecně přijatými dimenzemi rozepsat fakta a dimenze do tabulky, kdy na jednotlivých řádcích jsou fakta a ve sloupcích dimenze. Na místech křížení je pak vyznačeno, zda je daná tabulka dimenzí ve vztahu s danou tabulkou faktů. Toto znázorňuje tabulka 3.1.



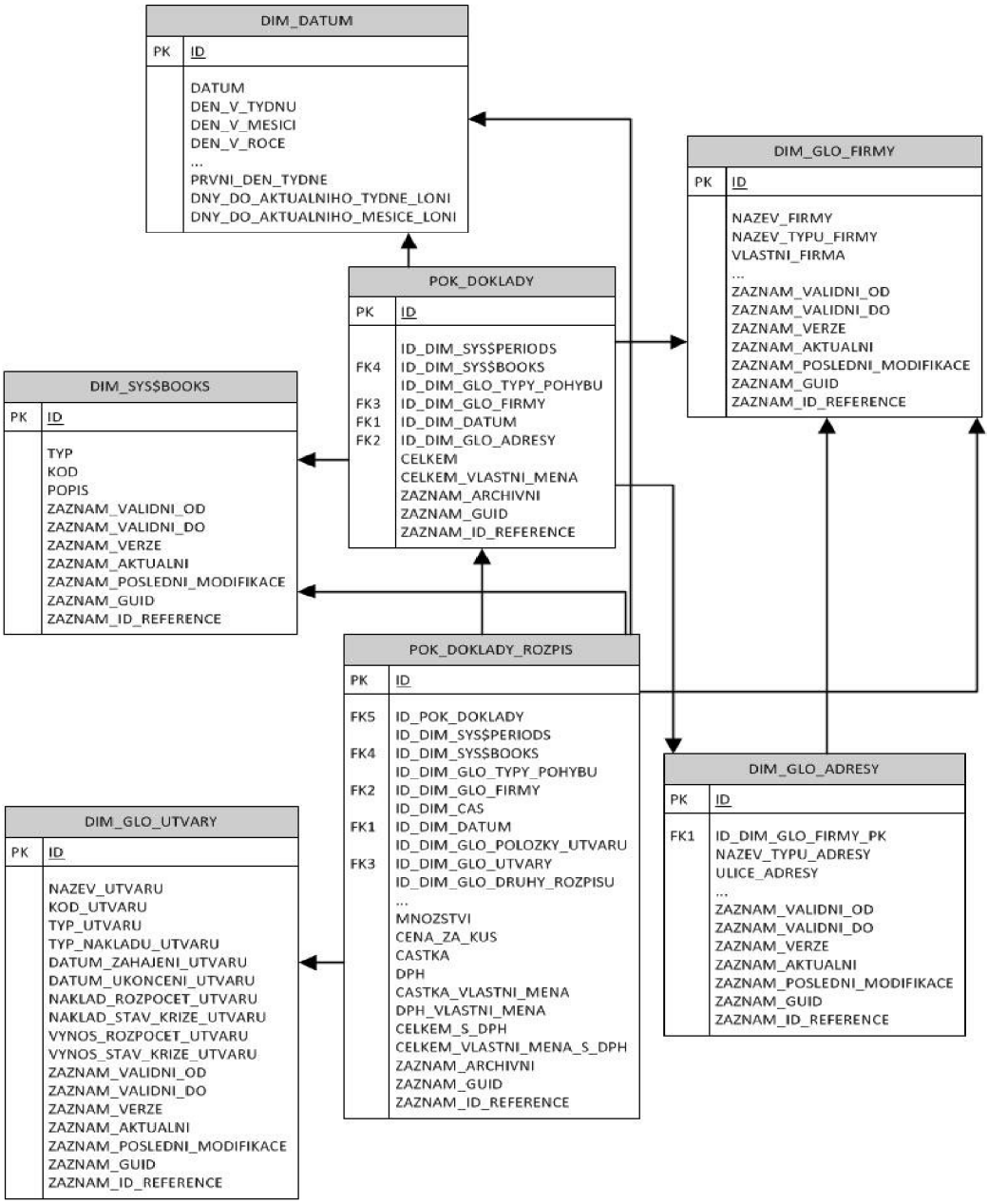
Ilustrace 3.10: Ukázka architektury sběrnice dle [6]

	DATUM	ADRESY	KNIHY	ÚTVARY	FIRMY
POK_DOKLADY	X	X	X		X
POK_DOKLADY_ROZPIS	X		X	X	X

Tabulka 3.1: Tabulka návrhu DS s všeobecně přijatými dimenzemi

Dle [6] téměř žádný datový sklad není složen pouze z jedné struktury typu hvězda, protože samotná hvězda zachycuje většinou pouze určitou oblast, jíž se společnost zabývá a nikoli všemi jeho aspekty. Proto se datový sklad modeluje z více tabulek faktů, které se zaměřují vždy na danou problematiku společnosti. Tyto tabulky faktů jsou pak spojeny všeobecně přijatými dimenzemi, které zajišťují, že při pokládaní dotazů zasahujících do více oblastí společnosti, bude datový sklad tento dotaz schopen zodpovědět.

Na ilustraci 3.11 je znázorněn příklad architektury sběrnice pro tabulku 3.1. Zobrazuje dvě tabulku faktů: pokladní doklady, rozpis pokladních dokladů a tabulky dimenzí: firmy, datum, knihy, útvary a adresy. Společné dimenze pro zobrazené tabulky faktů jsou datum, knihy a firmy.



Ilustrace 3.11: Příklad architektury sběrnice použité v diplomové práci

Použití architektury sběrnice s všeobecně přijatými dimenzemi umožňuje z datových trhů vytvořit opravdový datový sklad. Každá dimenze je v tomto datovém skladu pouze jednou a existuje pouze jeden proces, který se stará o její plnění a úpravy.

4 Postupy budování datového skladu

Postupy ke zbudování datového skladu jsou veskrze dva, přístup Billa Inmona a přístup Ralpha Kimbala. Zatímco Bill Inmon staví na centralizované normalizované architektuře, Ralph Kimbal řeší DS pomocí datových tržišť. V této kapitole jsem zejména čerpal z [5] a [7].

4.1 Centralizovaný přístup

Centralizovaný přístup neboli budování shora dolů je přístupem Billa Inmona [5]. Tato implementace shora dolů vyžaduje v počátku budování DS více plánování a navrhování. S tímto přístupem k budování datového skladu je nutné již v počátku spolupracovat s lidmi z jednotlivých oddělení společnosti, kteří budou DS využívat. Rozhodnutí, které datové zdroje budou použity k plnění, bezpečnost, struktura datového skladu, kvalita dat, standardy a celkový model dat, musí být s přístupem shora dolů hotový ještě před tím, než začne implementace. Je tedy nutné zafixovat požadavky a podle nich DS implementovat. Tento přístup k vývoji by mohl být připodobněn k modelu životního cyklu vodopád, kdy je nejprve vše naplánováno, navrženo, všechny požadavky na systém zjištěny a teprve potom dochází k implementaci. Architektura datového skladu vytvořeného tímto přístupem většinou odpovídá spíše architektuře centralizované. Pokud jsou v návrhu zahrnuty datové trhy, jsou typicky budovány později a plněny z centrálního datového skladu, než přímo z externích zdrojů. Výhodou tohoto přístupu je, že většinou získáme konzistentní definice dat a specifikovaná pravidla užití pro celou společnost již od počátku nasazení. Další výhodou je, že bezpečnost a konzistence dat je řešena pouze nad datovým skladem. Naopak nevýhodou je počáteční cena, která je určena rozsáhlým počátečním plánováním a návrhem. Tato počáteční fáze zabere dlouhý čas a oddálí samotné nasazení a tím i návratnost investic na budování DS. Další nevýhodou je, že se požadavky mohou v průběhu budování měnit a tento přístup není dostatečně pružný, aby na změny reagoval. Centralizovaný přístup budování DS může pracovat dobře tam, kde je centralizovaný informační systém, který používají všechna oddělení společnosti.

4.2 Decentralizovaný přístup

Opačným způsobem budování je iterativní/decentralizovaný přístup. Tento postup někdy nazývaný zdola nahoru, dle [5] nejprve buduje datové trhy pro jednotlivá oddělení společnosti. Datový sklad může být budován inkrementálně spolu s jednotlivými datovými trhy. Pro každý datový trh jsou specifikovány požadavky, je navržen a implementován specificky pro určité oddělení společnosti. Tyto datové trhy pak mohou být plněny z centrálního datového skladu nebo přímo z externích zdrojů. Obecně by se iterativní postup budování dal vysvětlit jako několik shora dolů postupů na jednotlivá oddělení. Tento přístup budovaní je dle [5] obecně používanější a to zejména pro rychlejší návratnost investic a nižší počáteční náklady. Toto je způsobeno tím, že návrh a implementace datových trhů je zpravidla méně náročná než budování komplexního centrálního datového skladu. Mezi nevýhody tohoto přístupu ke zbudování DS patří zejména to, že datové trhy na rozdíl od centrálního DS obsahují redundantní data. Doba plnění jednotlivých datových trhů je delší než u centralizovaného datového skladu, protože je nutné provádět více operací extrakce dat. Největším problémem tohoto přístupu však je, že při nevhodném nebo žádném návrhu datového skladu, jako celku, skládajícího se z datových trhů, nebude možné toto sjednocení používat jako datový sklad, poskytující informace o celé společnosti.

4.3 Kombinovaný přístup

Dle [5], je ideálním způsobem budování datového skladu kombinace obou předchozích postupů. Tento způsob doporučuje i Ralph Kimbal v [10]. Hlavním klíčem je nalezení vhodného poměru mezi plánováním a návrhem potřebným pro centrální datový sklad postupem shora dolů a přístupem zdola nahoru pro budování datových trhů. Pro centrální datový sklad se navrhuje základní definice infrastruktury z obchodního pohledu. Příkladem může být zjištění hlavních oblastí podnikání, které budou na zbudování datového skladu participovat. Tento pohled na problém také poskytne elementy vhodné ke zbudování datových trhů. Když jsou datové trhy implementovány, je nutné vytvořit plán, jak řídit data potřebná jednotlivými datovými trhy. Toto je okamžik, kdy je vhodné začít práce na centrálním datovém skladu nebo centrálním datovém úložišti dostupném pro všechny datové trhy. Tento přístup je dle [5] nejvhodnější z pohledu zabraného místa, přístupnosti a redundance dat.

4.4 Normalizovaný versus dimenzionální model

Posledním z velkých rozdílů v přístupu budování datového skladu je to, zda mít tabulky normalizované nebo nenormalizované. Přístup Billa Inmona předpokládá normalizované tabulky datového skladu, nad kterými je relační aritmetikou prováděno dotazování. Naopak přístup Ralpha Kimbala předpokládá, že tabulky dimenzí a faktů nejsou normalizované. Denormalizované schéma sice ukládá data v jedné tabulce a tedy obsahuje redundantní data, ale na druhou stranu ostatní problémy spojené s denormalizovanými tabulkami se datových trhů netýkají. To je způsobeno tím, že do tabulek dimenzí a faktů se záznamy pouze přidávají nebo se pouze mění příznak aktivity záznamu. Při reálné implementaci je však většinou použit alespoň částečně normalizovaný model.

4.5 Postup tvorby datového trhu

Při návrhu a implementaci datového trhu postupujeme dle [10] takto: analyzujeme a sjednotíme datové zdroje, analyzujeme požadavky na datový trh a navrhneme koncept, jak by datový trh měl vypadat. Dále odhadneme nejčastější dotazy na datový trh a validujeme koncept. Pokud návrh konceptu odpovídá požadavkům, provedeme logický návrh, definujeme proces plnění a nakonec implementujeme datový trh. Jednotlivé fáze budování datového trhu dle [10] podrobněji obsahují toto:

1. **Analýza a sjednocení dat** – zahrnuje zejména analýzu zdrojů a vytvoření sjednoceného schématu ze zdrojových tabulek. Toto schéma bude použito pro fázi koncepčního návrhu a vytvoření odkládací oblasti. Tato fáze slouží k vytvoření dobrého povědomí o zdrojích. Vstupem této fáze jsou schémata zdrojových dat a výstupem této fáze je sjednocené schéma dat zdrojů.
2. **Analýza požadavků** – slouží zejména k nalezení faktů, které budeme pomocí datového trhu analyzovat a k odhadu nejčastějších dotazů na datový trh. Pomocí tohoto můžeme určit základní granularitu dat a agregace na tabulkách faktů. Vstupem této fáze jsou požadavky na datový trh a výstupem je specifikace požadavků a předběžný návrh dotazů.
3. **Koncepční návrh** – vytváří na základě sjednoceného schématu zdrojů dimenzionální model faktů. Tento model obsahuje schémata pro každý z faktů požadovaných uživatelem a k němu připojené dimenze. Vstupem této fáze je sjednocené schéma zdrojů a specifikace požadavků, výstupem této fáze je dimenzionální model faktů.
4. **Redefinice dotazů na trh a validace návrhu obecného schématu** – slouží k upravení poznatků a dokumentů získaných v předchozích fázích. Postup v této fázi je takový, že podle dimenzionálního modelu faktů zkoumáme, zda jsme schopni odpovědět na dotazy specifikované v dokumentu s předběžnými návrhy dotazů na datový trh. Tato fáze má na svém vstupu dimenzionální model faktů a předběžné dotazy na datový trh a výstupem je validované dimenzionální schéma faktů a dotazy pokládané na datový trh.

5. **Logický návrh** – slouží zejména k volbě modelu uložení dat a návrhu struktury tabulek podle dimenzionálního modelu. V této fázi již definujeme, jaká databázová pole jednotlivé tabulky dimenzí a faktů obsahují a jak vypadá struktura datového trhu. Tato fáze budování datového trhu má na svém vstupu zvolený model uložení dat, dimenzionální model faktů a dotazy na datový trh, výstupem je logické schéma datového trhu.
6. **Proces plnění** – definuje, jak se data ze zdrojů přenášejí do odkládací oblasti, pokud existuje nebo do přímo do datových trhů. Definujeme, jak často by tento proces měl být spouštěn, jak se bude provádět integrace dat, jejich čištění a další. Tato fáze bere na svém vstupu schémata zdrojových dat, sjednocené schéma dat, logické schéma datového trhu a výstupem je definice procesu ETL.
7. **Implementace datového trhu** – provádíme na základě logického schématu datového trhu, zvoleného databázového systému a definice dotazů.

5 Obsah Datového skladu

Tato kapitola se zejména zabývá obsahem datového skladu, dle definice Billa Inmona, následně je provedeno rozdělení dat dle způsobu použití a nakonec je pojednáno o kvalitě dat.

5.1 Požadavky na obsah datového skladu

Datový sklad je tedy dle [4] kolekce subjektově orientovaných, integrovaných, stálých a časově proměnných dat určených k podpoře rozhodování. I když tato definice specifikuje datový sklad z pohledu Bila Inmona, je vhodné, aby subjektovou orientaci, integrovanost, stálost a časovou variabilitu splňoval i přístup Ralpha Kimbala, proto je v této kapitole rozebereme podrobněji.

5.1.1 Subjektová orientace

Subjektová orientace znamená, že datový sklad obsahuje data rozdělená podle individuální oblasti zaměření společnosti, například produkt, zákazník a jiné. Naproti tomu provozní informační systém dělí data spíše podle způsobu použití nebo podle jejich funkce. Například pro bankovní instituci jsou data děleny na spořicí účty, běžné účty a jiné.
Naproti tomu v DS jsou tedy data zaměřena podle subjektu, to znamená dle [6], že například pro zpracovatelskou společnost jsou to prodeje, dodávky a zásoba zboží na skladě. Toto jsou kritické podnikové subjekty. Pro maloobchod jsou například kritické subjekt položky na pokladním bloku.
Tabulka 5.1 zobrazuje dle [6] funkční zaměření na data pro operační databázi a k nim ekvivalentní subjektové zaměření na data pro DS.

Aplikační zaměření	**Subjekty datového skladu**
Zpracování objednávek	Prodeje
Zákaznické platby	Zákazník
Nezaplacené půjčky	Půjčky

Tabulka 5.1: Porovnání aplikačního a subjektového zaměření dat v operační databázi a datovém skladu

Dle tabulky je vidět, že data v DS nejsou přímo zaměřená na nějaké specifické použití na rozdíl od dat v operační databázi.
Než začneme budovat datový sklad, je velice vhodné specifikovat tyto subjekty, protože podle nich jsme schopni lépe definovat požadavky na datový modelu a specifikovat jaká data má datový sklad obsahovat. Toto usnadní implementaci jednotlivých částí DS.
Pro usnadnění specifikace subjektů datového skladu můžeme použít dle [5] pravidlo 5W1H (when, where, who, what, why, how), tedy kdy, kde, kdo, proč a jak. Tyto otázky položíme na obchodní zájmy společnosti, tedy například na otázku „kdo“ můžeme odpovědět zákazník, dodavatel, řidič, dlužník a další. Pomocí této metody můžeme vytvořit seznam kandidátních subjektů, ze kterých pak snadněji vytvoříme seznam subjektů datového skladu.
Pokud máme seznam subjektů, je dle [5] vhodné mezi nimi definovat vztahy ve společnosti. Tato definice nám usnadní vytvoření modelu dimenzí datového skladu.

5.1.2 Integrovanost

Mnoho společností dle [6] nemá svá data jednotně uložena, případně je získala nebo získává z externích zdrojů, různých operačních systémů a v různých formátech. Dále tyto různé zdroje obsahují data jinak kódována, sloupce jinak pojmenovány a mnoho dalších problémů. Příkladem může být uložení muž/žena v jednom zdroji a v jiném X/Y, dalším příkladem může být uložení čísla účtu, které může být ve formě mezinárodního čísla účtu IBAN nebo v standardním formátu.
Předtím než mohou být data z těchto různorodých zdrojů uložena do datového skladu, je dle [6] nutné vyřešit tyto inkonzistence, standardizovat jednotlivé různé elementy a být si jistý významem jednotlivých dat v různých zdrojích. Tabulka 5.2 ukazuje příklad procesu datové integrace z různých zdrojů.

Zdroje	**Transformace**				**Subjekt datového skladu**
Spořící účet	Úprava kódování	Sjednocení pojmenování	Úprava atributů	Sjednocení měrných jednotek	Účet
Běžný účet					
Debetní účet					

Tabulka 5.2: Příklad sjednocení datových zdrojů dle [6]

5.1.3 Stálost

Data v datovém skladu jsou stálá, to znamená, že nejsou mazána nebo upravována podle toho, jak se mění data v provozním systému. Pokud dojde ke změně dat v provozním systému, je v DS založen nový záznam, který se stává aktuálním. Příkladem může být změna názvu společnosti. Mějme v datovém skladu uloženy názvy společností, s nimiž obchodujeme, u některé z nich dojde ke změně názvu. V datovém skladu není nalezen záznam této společnosti a její jméno nahrazeno novým jménem. Místo toho se v DS založí nový záznam, který bude označen jako aktivní a předchozímu aktivnímu záznamu se příznak aktivity zruší. Pokud v provozním systému dojde ke smazání určitého záznamu, neznamená to, že by nikdy neexistoval. V datovém skladu se u takového záznamu pouze zruší aktivita. Na příkladu s názvy společností to znamená, že poslední aktivní záznam názvu dané společnosti je změněn na neaktivní. V praxi se, tento princip používá pouze na záznamy, u nichž nás zajímá vztah historických dat k záznamům tabulky faktů.

5.1.4 Časová proměnlivost

Datový sklad ze své povahy slouží k provádění analýz, a aby tyto analýzy podávaly relevantní data, je nutné mít v DS uložena data v dostatečně dlouhém časovém horizontu oproti operační databázi ukládající aktuální data. Když uživatel DS zkoumá vzorek nákupního košíku určitého uživatele, potřebuje nejen aktuální nákup, ale i minulé nákupy. Takto může uživatel DS zkoumat změny v chování zákazníka při nákupu zboží, zjišťovat informace o aktuálním nákupu, ale dokonce i predikovat co si zákazník koupí příště. Dle [6] každá datová struktura v datovém skladu obsahuje časovou složku, data jsou uložena jako snímky dat z operační databáze, za období jejího fungování. Tento aspekt DS je důležitý jak pro návrh datového skladu, tak pro jeho implementaci.
Jako příklad mohou posloužit prodeje určitých druhů zboží. Množství prodaného druhu zboží je potom odvislé od určitého období (dni, týdnu, měsíci nebo roku), ke kterému je vtahováno.

5.1.5 Granularita dat

Data v provozní databázi jsou uložena většinou na nejnižší úrovni detailu. Mějme například provozní systém prodejce, který do databáze ukládá data na úrovní položky prodeje. Potřebujeme-li zjistit, kolik se prodalo zboží za den, týden, rok, či měsíc je nutné tato data agregovat. Naproti tomu v datovém skladu je vhodné mít data připravená na různých úrovních agregace. Tento přístup pak výrazně urychlí výpočty dotazů, protože je možné použít předem spočítaná data. Provádět přepočítání dat je vhodné pouze pro nejčastěji používané dotazy. Obecně čím máme v datovém skladu nižší granularitu, tím více snižujeme množství dotazů, které nám může datový trh zodpovědět. Naopak pokud je granularita dat vysoká a nepoužíváme předem připravená agregovaná data, je rychlost reakce datového skladu na snížena.

5.2 Rozdělení dat v datovém skladu

Při návrhu datového skladu je třeba uvážit použití dat, pohled na data a jejich cenu. Dle [5] můžeme v DS dělit data na aktuální data[2], odvozená data a konsolidovaná data. Celý datový sklad pak obsahuje právě tyto tři typy dat.

- **Aktuální data** – Dle [5] představují reálný stav. Typicky existují v provozních systémech a vyznačují se vysokou granularitou. K tomu, aby mohla být použita v datovém skladu, je nutné je většinou nejprve vyčistit, případně sjednotit a transformovat, aby byla relevantní a byla snadněji použitelná pro analýzy.
- **Odvozená data** – jedná se o data, která dle [5] byla vytvořena pomocí sumarizací, průměrováním či jinými operacemi nad aktuálními daty. Odvozená data mohou být dále agregována či jinak upravována a slouží především pro analýzy a rozhodování. V datovém skladu je používáme zejména proto, že analytici nepotřebují velké množství detailních dat, ale spíše menší množství sumarizovaných dat. Tato data jsou také rychleji zpracována při dotazování, protože je jich výrazně méně.
- **Konsolidovaná data** – dle [5] speciální typ odvozených dat, která byla upravena tak, aby byla vhodná pro rozhodování.

5.3 Kvalita dat

Jednou z největších výzev dle [7] je udržení kvality dat v datovém skladu. Problémy s jejich kvalitou v datovém skladu a při jeho plnění mohou mít mnoho podob, ale mezi nejčastěji se vyskytující patří dle [7] tyto:

- **Duplicitní data** – představují problém, kdy je v systému vícekrát uloženo totéž, případně jsou tyto hodnoty uloženy v různých systémech, ale nemohou být propojeny z důvodu neexistence vzájemných odkazů nebo klíčů.
- **Nekompletní data** – data jsou uložena, ale neobsahují kompletní informaci, příkladem může být číslo popisné v adrese a podobné problémy.
- **Chybná data** – jedná se o data, která jsou sice kompletní, ale nejsou správně zadaná. Příkladem mohou být různé překlepy při jejich zadávaní ručně.
- **Data v konfliktu** – data jsou uložena v různých tabulkách nebo různých zdrojových systémech a navzájem se vylučují.
- **Nejednoznačná metadata** – definice dat v systému není jednoznačná, takže data v systému nemají jasný význam.

2 V originále Real-Time data

- **Chybějící data** – toto je extrémní verze nekompletních dat, kdy chybějí celé záznamy, které by v systému měly být. Tento problém je dle [7] jedním z nejzávažnějších, protože mnohdy není poznat, že data chybí.
- **NULL hodnoty** – pole v databázi nemají vyplněné hodnoty, což může znamenat, že hodnota chybí, je neznáma nebo nebyla požadována.

6 Plnění datového skladu

Plnění datového skladu je jednou z nejdůležitějších částí datového skladování. Proces plnění datového skladu (ETL) představuje získání dat ze zdrojových oblastí (Extraction), transformaci těchto dat do formy ve které je uložíme (Transformation) a konečně přesunutí těchto upravených dat do datového skladu (Loading). Každá z těchto částí je složitý proces, který vyžaduje řešení specifických problémů, a proto se zde budeme každou z těchto částí zabývat detailně.

ETL proces dle [6] získá a transformuje zdrojová data a nakonec je uloží v datovém skladu ve tvaru, který je vhodný pro získávání strategických informací. Dle [6] také vyvstává otázka, proč nevzít všechny zdrojové soubory a jednoduše je nevložit do datového skladu. V takovém datovém skladu by všechny informace ležely, ale jejich získání, zvláště pokud je zdrojů velké množství a jsou různorodé, by bylo extrémně náročné. Pokaždé, když bychom chtěli informace z takového skladu získat, museli bychom absolvovat celý proces datového skladování. Proto je výhodnější plnit datový sklad jen vhodnými a upravenými daty procesem ETL, čímž minimalizujeme náročnost získávání informací z dobře navrženého datového skladu.

Hlavní problémy, které musí proces ETL řešit jsou dle [6] tyto:

- Zdrojové systémy mohou být velice různorodé.
- Mnohdy jsou zdrojová data uložena na různých platformách a operačních systémech.
- Některá zdrojová data mohou být uložena pomocí zastaralých databázových technologií.
- Kvalita zdrojových dat je nejistá.
- Zdrojové systémy se mohou měnit v průběhu času v závislosti na požadavcích, ETL proces se tomuto musí přizpůsobit.
- Zdrojové systémy jsou většinou nekonzistentní, některá data mohou být reprezentována různě v různých zdrojových systémech.
- I když jsou nekonzistentní data objevena, nedostatek informací jak tato data sjednotit eskaluje problém nekonzistence.
- Mnoho zdrojových souborů uchovává data ve formátu nepochopitelném pro uživatele.

Všechny tyto problémy musí proces ETL řešit což je velice náročné jak na čas, tak na výkon systému, na němž proces ETL běží. Správně navržený proces ETL je tedy jednou z kritických součástí procesu datového skladování.

6.1 Získání dat ze zdrojových systémů

Tato část procesu ETL nazývaná extrakce má na starosti získaní dat z množství zdrojů, které máme k dispozici a chceme je ukládat v datovém skladu. Její součástí je dle [6] zejména identifikace zdrojů, definice metod extrakce pro každý ze zdrojů, volba frekvence provádění a specifikování časového okna pro provádění extrakce. Dále určení pořadí provádění jednotlivých metod extrakce, jejich vzájemné závislosti a nastavení chování procesu extrakce pro případy, kdy nejde data extrahovat.

Během identifikace zdrojů je dle [6] důležité identifikovat data, která budou sloužit k plnění tabulek faktů a tabulek dimenzí. Pro každou položku těchto tabulek je nutné nalézt položku ve zvolených zdrojích, která bude sloužit k jejich plnění. Může se stát, že nalezneme více takových položek, které by mohly sloužit k plnění položek tabulek datového skladu. V takovém případě zvolíme zdroj, který považujeme za nejvhodnější, a stanovíme pravidla, jimiž se budeme řídit v případě dalších vícenásobných zdrojů. V případě, že máme jednu zdrojovou položku pro více položek tabulek datového skladu, je nutné určit pravidla, kterými se budeme řídit při rozdělovaní hodnot. Dále je nutné nadefinovat výchozí hodnoty, které budeme při plnění datového skladu používat. Nakonec je nutné ve zdrojových datech vyhledat položky, které nejsou vyplněné, a tyto

položky v průběhu plnění doplnit vhodnou hodnotou. Moderní databáze umí velice dobře pracovat s prázdnou hodnotou NULL a není tedy nutné pro každou prázdnou položku definovat speciální hodnotu.

Identifikace zdrojů je kritický proces v části extrakce, při kterém je nutné projít kompletně zdrojová data a vyhledat každou informaci, kterou budeme v datovém skladu ukládat. Tato část extrakce spotřebovává velké množství času a klade vysoké nároky na přesnost.

6.2 Metody extrakce

K tomu, abychom dle [9], [6] mohli definovat metody extrakce, je nejprve nutné klasifikovat zdrojová data. Ta mohou být ukládána jako takzvaná nestálá data[3], kdy jednotlivé hodnoty jsou v případě změny přepsány novou hodnotou a původní hodnota je zahozena. Druhou možností je ukládat zdrojová data jako takzvaná periodická data. Tento způsob uložení uchovává předchozí hodnoty, takže při změně nad daty, je u původní hodnoty zapsána doba platnosti a nová hodnota se stává aktivní. Tento způsob uchovává historické hodnoty, ale extrakce dat z takového zdroje je složitější. Metody extrakce pak můžeme dělit na takzvanou statickou extrakci a inkrementální extrakci.

6.2.1 Statická extrakce

Nejjednodušším přístupem k extrakci je takzvaná statická extrakce. Čte veškerá data ze zdrojů a ta jsou následně dále zpracována procesem ETL. Tento přístup se dle [6] užívá zejména při zavádění datového skladu do provozu. Jelikož je u tohoto přístupu čteno a zpracováváno velké množství dat, je její použitelnost velice omezená a většinou ji není vhodné používat jindy jak při prvotním plnění, nebo pokud chceme kompletně přenačíst některou z tabulek dimenzí.

6.2.2 Inkrementální extrakce

Inkrementální extrakce využívá technik sloužících k detekci změn v zdrojových souborech a proces ETL dále zpracovává pouze tato změněná data. Podle toho, kdy data přesouváme do datového skladu, rozdělujeme inkrementální extrakci dle [9] na okamžitou a opožděnou. Dle [6],[9] existují pro okamžitou extrakci tři přístupy.

Přístupy k okamžité extrakci

Prvním je použití logů, ve kterých jsou uchovávány změny nad daty, z těchto logů jsou pak data čtena procesem extrakce. Tento přístup nefunguje, pokud máme data uložena v systému, který neposkytuje logování.

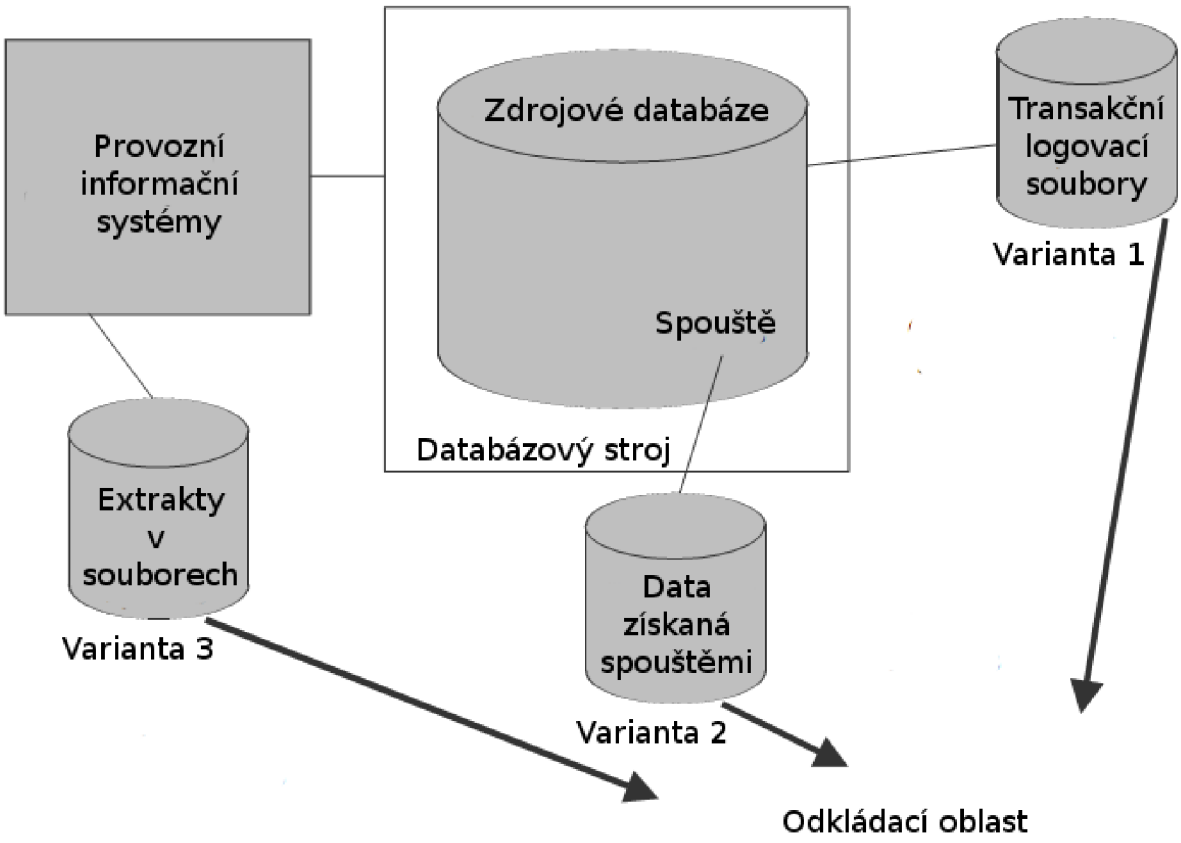
Dalším přístupem je použití spouští[4], ty ukládají změněná data do speciální tabulky nebo souboru, odkud jsou následně získávány pro proces extrakce. Tento přístup opět nefunguje pro databázové systémy, které nepodporují spouště nebo obyčejné soubory.

Posledním okamžitým přístupem extrakce je použití zachytávání změn za pomocí asistující aplikace. Tento přístup vyžaduje úpravu aplikací, které poskytují zdrojová data. Takto upravené aplikace pak při změně, zakládaní nebo mazání dat automaticky ukládají data do oblasti, odkud mohou být získána procesem extrakce. Dle [9] je toto řešení velice efektivní, ale vyžaduje změny v existujících aplikacích, což není vždy možné. Ilustrace 6.1 znázorňuje tři přístupy okamžité extrakce.

---

3 V originále dle [9] transient

4 Spoušť je uložená procedura, která je vyvolána při určité události v databázi



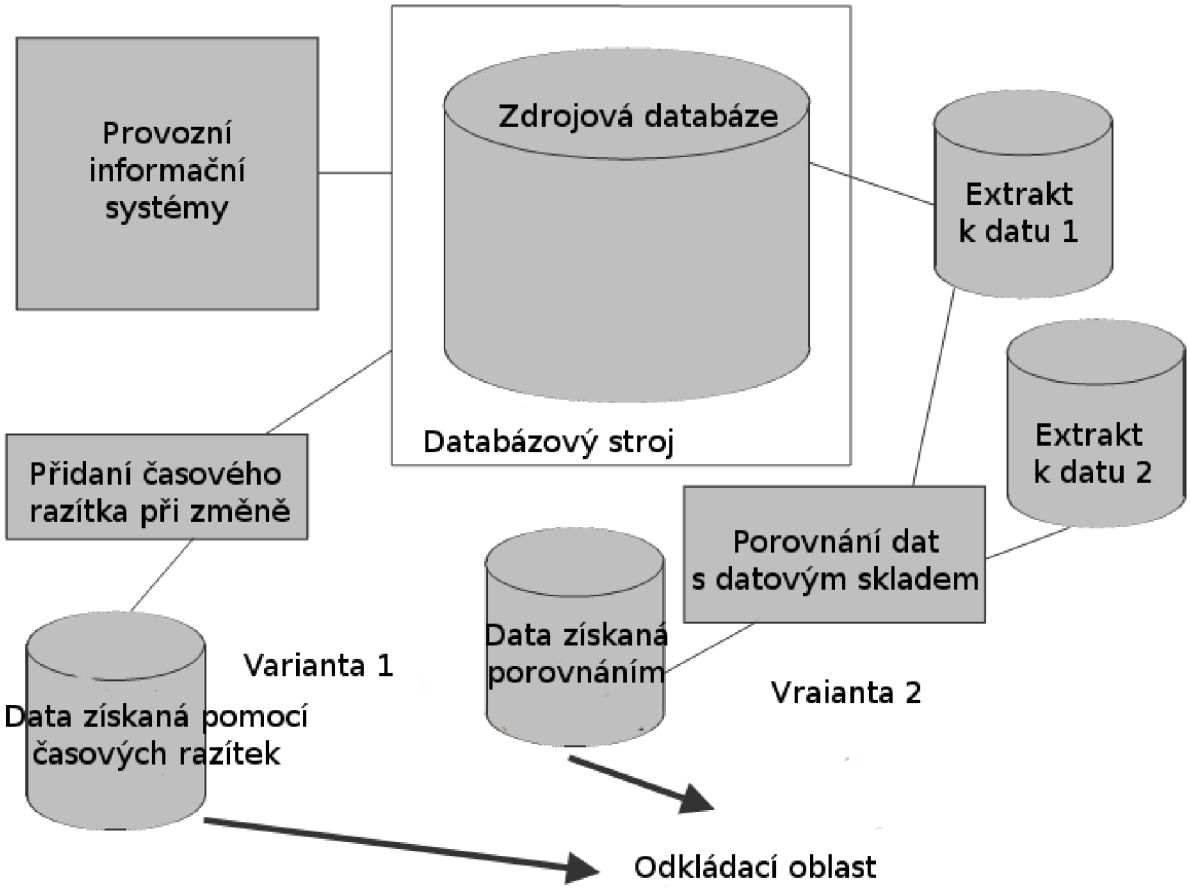
Ilustrace 6.1: Tři způsoby okamžité extrakce dle [6]

Pro odloženou extrakci dle [6],[9] existují dva přístupy, pomocí časových razítek a extrakce porovnáním. Odložená extrakce neprovádí zachytávání změněných dat, ale snaží se tuto činnost provádět najednou.

Přístup pomocí časových razítek označí každý změněný záznam časovou značkou, dle které je extrakční program schopen zjistit, že nad danými daty došlo ke změně. Tento přístup pak při extrakci čte pouze data, která mají datum a čas časového razítka vyšší jak datum a čas poslední extrakce. Tento přístup může fungovat pro libovolný zdroj dat, ale předpokládá, opatření dat časovými razítky.

Druhým typem odložené extrakce je extrakce dat pomocí porovnání. Tu je nutné použít tam, kde není možné použít jiný přístup extrakce. Tento přístup využívá porovnání zdrojů z minulé extrakce a aktuálních zdrojů. Po porovnání se extrahují pouze rozdílné záznamy. Tento přístup je aplikovatelný na libovolný zdrojový soubor a nevyžaduje žádné změny v aplikacích, poskytujících zdrojová data.

Na ilustraci 6.2 jsou znázorněny přístupy k odložené extrakci.



Ilustrace 6.2: Přístupy k odložené extrakci dle [6]

6.3 Transformace dat ke skladování

Data pro DS většinou získáváme z různých zdrojů, to znamená, že data nejsou většinou ve zdrojových systémech uložena jednotným způsobem, některé záznamy mohou chybět nebo mohou být chybné. Úkolem procesu transformace je upravit data získaná pomocí extrakce do tvaru, který je vhodný k uložení v datovém skladu a tedy zvýšení jejich kvality.

Proces transformace dat zahrnuje dle [6] zejména tyto úkony:

- **Výběr** – tato část procesu transformace je někdy již součástí extrakce, jejím úkolem je zejména zvolit data, která budeme dále transformovat a ukládat v DS. Jde například o výběr několika slov z jednoho záznamu či několika záznamů.
- **Spojování a dělení** – úkolem tohoto úkonu je rozdělit nebo naopak spojit data ze záznamů načtených extrakcí. Nejčastěji jsou data spojována z více záznamů různých zdrojů dat.
- **Konverze** – jde zejména o sjednocení dat daného typu do jednoho formátu v rámci celého datového skladu a úprava dat do tvaru srozumitelného pro uživatele.
- **Sumarizace** – v datovém skladu někdy není nutné uchovávat data na nejnižší úrovni zrnitosti, protože uživatel DS potřebuje pro analýzy pouze data s vyšší zrnitostí. Při sumarizaci jsou data sečtena či jinak seskupena do větších celků. Toto je vhodné použít pouze u případů, kdy řešíme problémy s nedostatkem paměti. Jak již bylo uvedeno dříve, existují techniky jak přistoupit k datům s vysokou granularitou tak, aby byla rychlost získání sumarizovaných dat srovnatelná s daty uloženými jako sumarizovaná a přitom jsme měli stále možnost pracovat s daty na původní úrovni granularity.

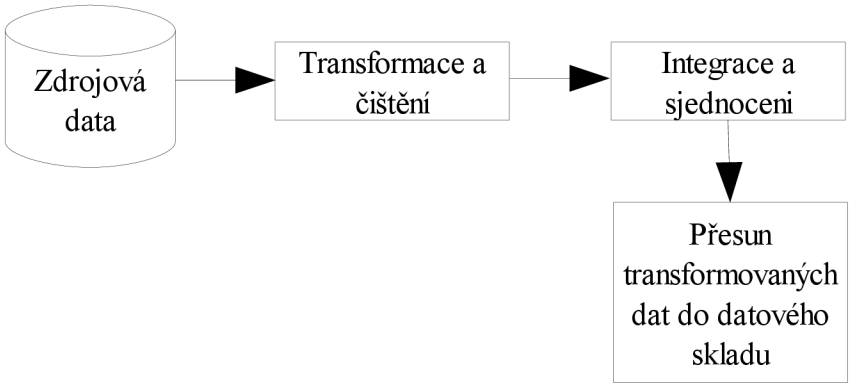
- **Obohacení** – úkolem tohoto úkonu je zvýšení množství informací které poskytuje položka datového skladu. Může jít například o vytvoření nové položky z několika položek získaných během extrakce.

Hlavní výzvou transformace je sjednocení dat z mnoha různorodých systémů do jednoho správně fungujícího celku.

Zde se dle [6] zejména vyskytuje problém identifikace shodných záznamu, kdy máme ve zdrojových systémech uložená stejná data, například o zákazníkovi. V datovém skladu však chceme mít záznam o daném zákazníkovi uložen pouze jednou. Je tedy nutné identifikovat tyto shodné záznamy, vybrat patřičná data o shodném záznamu z jednotlivých systémů a do datového skladu uložit pouze jeden relevantní záznam. Jedním z používaných přístupů dle [6] je použití kombinace automatického porovnání a manuální verifikace. Většina záznamů je převedena do datového skladu automaticky. Záznamy u nichž nebylo možné pomocí automatického porovnání rozhodnout, zda jsou duplicitní, se porovnají ručně.

Dalším problémem, který se vyskytuje během transformace, je dle [6] problém více zdrojů. Tento problém je podobný problému identifikace a nastává v případě, že máme více zdrojů dat pro jednu položku v datovém skladu, například cenu výrobku. Jedním z řešení tohoto problému je přiřazení vyšší priority jednomu ze zdrojů a ten je pak preferován v případě konfliktu.

Proces transformace dle [6] končí ve chvíli, kdy jsou pomocí úkonů transformace vytvořena data, která budou zapsána do datového skladu. Na ilustraci 6.3 jsou znázorněny úkony procesu



Ilustrace 6.3: Úkony procesu transformace

transformace, přesun dat do datového skladu je poslední část procesu ETL a jím se bude zabývat další kapitola.

6.4 Přesun dat do datového skladu

Přesun dat do datového skladu se dá rozdělit dle způsobu, jakým jsou data do skladu přesouvána. Dle [6] je to počáteční plnění, inkrementální plnění a znovu načtení. Počáteční plnění je prováděno při zavádění datového skladu do provozu. Jde o kompletní naplnění datového skladu daty získanými v době, kdy datový sklad nebyl v provozu.

Inkrementální plnění je prováděno průběžně během provozu datového skladu. Při tomto procesu jsou do datového skladu vkládány pouze změny nad zdrojovými daty od minulého běhu procesu ETL. Data pro inkrementální plnění jsou získávána pomocí technik, uvedených v kapitole 6.2.2.

Znovu načtení je obdoba počátečního plnění, ale nemusí být provedeno pro celý datový sklad, ale například pouze pro jednu dimenzi. Dalším rozdílem oproti počátečnímu plnění je, že v datovém skladu již většinou data existují a ta jsou při znovu načtení kompletně přepsána novými daty.

Po počátečním naplnění datového skladu je nutné zvolit, jakým způsobem budeme přistupovat k propagování změn na zdrojových datech do datového skladu. Můžeme zvolit inkrementální plnění nebo znovu načtení. Druhá metoda je technicky mnohem méně náročná, ale v případě, že datový sklad již obsahuje velké množství záznamů, bude doba trvání procesu plnění pomocí znovu načtení velice dlouhá. Doba trvání také závisí na množství změn provedených nad zdrojovými daty. Od jistého počtu změn, vzhledem k množství dat, je z pohledu času výhodnější použít znovu načtení, před inkrementálním plněním. Dle [6] je ještě vhodný poměr počtu změn vzhledem k množstvím dat pro použití inkrementálního plnění dvacet pět procent. Znovu načtení naopak nelze použít, pokud používáme techniky SCD 2,3,6 uvedené dále, protože historie získaná těmito technikami nemusí být dostupná.

6.4.1 Přístupy k ukládání dat v dimenzích

Při plnění dimenzí datového skladu musíme zvážit, jakým způsobem se postavíme ke změnám na jednotlivých položkách dimenzí. Pokud se například změní jméno firmy ve zdrojovém souboru, je k této změně nutné v datovém nějakým způsobem přistoupit. Dle [7] jsou toto možné způsoby.

Typ SCD 1

První možností je jméno firmy přepsat. Není nutné zakládat v dimenzi nový záznam a celkově není nutné provádět výraznější změny nad záznamy v dimenzi. Nevýhodou tohoto přístupu je ztráta historického pohledu na data. Pokud bychom tento přístup volili například na položce město, v dimenzi prodejců zboží, ztratili bychom přehled o tom, jak probíhal prodej zboží v jednotlivých městech. Při takovéto změně ukazují případné nové položky tabulek faktů na původní záznamy dimenzí. Tento přístup je vhodný dle [6] použít pokud opravujeme chybu v hodnotě pole dimenze, nebo nám nezáleží na historii.

Typ SCD 2

Dalším možným přístupem je postup, kdy při změně položky v dimenzi založíme nový záznam. Tento nový záznam nastavíme jako aktuální a původní aktuální záznam nastavíme jako historický. Abychom mohli tento přístup ukládání změn v datovém skladu provozovat, je nutné mít v dimenzi speciální atributy určující dobu platnosti záznamu a to zda je aktuální. Při vkládání nových záznamů do tabulek faktů pak cizí klíče ukazují na tyto nové aktuální záznamy v dimenzi. Původní záznamy tabulek faktů odkazují na historické položky dimenzí, takže pro příklad s prodejcem zboží dokáže tento přístup zachytit prodeje v různých městech. Naopak může vznikat problém například se změnou jména firmy, kdy budeme chtít zjistit prodeje dané firmy od počátku její existence v datovém skladu. Tento přístup komplikuje výpočet, protože je nutné zahrnout i záznamy za dobu kdy se firma jmenovala jinak.

Typ SCD 3

Tato varianta ukládání změn do dimenzí pro změnu zdrojových dat přepíše původní hodnotu jako typ SCD 1, ale také uchovává původní hodnotu ve speciálním poli. Tento přístup je tedy schopen uchovávat jednu historickou hodnotu, ale není schopen zajistit členění položek tabulky faktů jako přístup SCD 2. Obecně může být typ SCD3 použitelný pokud nás nezajímá členění položek tabulky faktů podle historických údajů a požadujeme znalost původní hodnoty.

6.4.2 Další přístupy k ukládání dat v dimenzích

Následující přístupy se v různých publikacích liší a jsou prezentovány pod různým označením, uvedeme zde nejvíce užitečné, zejména pak hybridní přístup, který je použit v praktické části diplomové práci.

Minidimenze

Tento přístup se používá v případě velmi rozsáhlých dimenzí, které mají velké množství záznamů. Jeho podstatou je vyčlenění části atributů do samostatné dimenze, používající intervaly místo přesných hodnot. Jde tedy spíše o optimalizaci již existujících dimenzí. Mějme dimenzi s atributy stát, město, datum narození, pohlaví a příjem. Vyplnit tuto dimenzi všemi hodnotami je takřka nemožné. Proto je vhodné dimenzi rozdělit na dvě dimenze, jednu obsahující geografická data a druhou obsahující demografická data. I když je původní dimenze rozdělená, stále by nová dimenze demografických dat obsahovala neúměrné množství dat, proto se v minidimenzích používají intervaly, jež jsou jeho podstatnou částí. S tímto přístupem se již dá dosáhnout únosného množství záznamů v dimenzi a přehledněji zobrazitelných.

Historické tabulky

Princip spočívá ve využití oddělené tabulky historie. Nemůže být dle [7] snadno použit k provádění analytických dotazů, stejně jako typy SCD 1 a 3, protože samy o sobě neposkytují možnost členit záznamy tabulky faktů. Tento přístup poskytuje historii pro změny nad záznamy dimenze tak, že samotná dimenze vždy uchovává aktuální data a do tabulky historie se ukládají původní záznamy. Tabulka faktů pak cizími klíči odkazuje na hodnoty dimenze, takže tento přístup také neposkytuje přímé členění záznamů podle změn v historii. Je sice možné podle hodnot v tabulce historie data tabulky faktů rozčlenit do datumových intervalů, ale tento přístup k tomuto není přímo určen, a proto to není triviální.

Hybridní přístup

Poslední přístup k načítání dat do dimenzí je vlastně kombinací přístupů 1,2,3, proto je také nazýván jako hybridní přístup. Pro hodnoty, pro které nechceme uchovávat historii, použijeme přístup 1. Pro hodnoty, pro které historii uchovávat chceme, přístup 2. A pro hodnoty, na které se chceme dívat jak z aktuálního pohledu, tak z pohledu historického členění, použijeme přidání speciálního sloupce, který uchovává aktuální hodnotu atributu, tedy upravený přístup typu 3.

Tabulka 6.1 prezentuje použití přístupu SCD 6. Tabulka zachycuje historii dvou záznamů.

ID	...	PLATNOST_OD	PLATNOST_DO	VERZE	AKTUALNI	POSLEDNI_ZMENA	GUID	ID_REFERENCE
1	...	31.12.99	31.12.9999	1	1	31.12.2010	#1	1
2	...	31.12.1999	31.12.2010	1	0	30.12.2010	#2	2
3	...	31.12.2010	31.12.9999	2	1	11.4.2011	#2	2

Tabulka 6.1: Implementace hybridního přístupu použitého v diplomové práci

Záznam s ID 1 je platný po celou dobu jeho existence, protože nedošlo ke změně na atributu, pro nějž by se uchovávala historie. Naopak záznam s ID 2 je platný pouze do 31. 12. 2010, k tomuto datu došlo na záznamu ke změně, která byla archivována. Byla vytvořena nová verze záznamu se stejným ID_REFERENCE 2, ale změnila se jeho platnost. Dále tabulka obsahuje sloupce GUID určující odkud byla data získána a POSLEDNI_ZMENA.

Pro každý atribut dimenze je tedy vhodné zvolit jaký typ úpravy dat použít v závislosti na požadovaném pohledu na data tabulek faktů.

6.4.3 Ukládání dat v tabulkách faktů

Hlavním problémem při plnění tabulek faktů je provázat záznamy se správnými záznamy tabulek dimenzí. Proto jsou dimenze plněny dříve než tabulky faktů. Pro každý nový záznam tabulek faktů je potřeba zapsat všechny hodnoty cizích klíčů odkazujících na správné záznamy dimenzí. Jestliže to není možné, je vhodným řešením mít v tabulkách dimenzí speciální záznam, na který budeme odkazovat záznamy, pro které nebudeme schopni dohledat vazbu cizího klíče tabulek faktů na určitý záznam dimenze.

Dalším velkým problémem při ukládání hodnot v tabulkách faktů je možnost přenesení chybných záznamů ze zdrojového sytému do datového skladu. Pokud například provádíme synchronizaci datového skladu každý den, je velice pravděpodobné, že může dojít k přenesení záznamů, které budou následně smazány jako chybné. Proto je nutné rozlišit, zda byl záznam ze zdroje dat odstraněn jako chybný nebo prostě proto, že je již byl nepotřebný. K této problematice je možné použít například následující postup. Do tabulky faktů přidáme speciální pole, určující zda je záznam stále ve zdrojovém systému. Toto pole je nastaveno na pravdivou hodnotu tehdy, dojde-li ve zdrojovém systému k rušení již neaktuálních záznamů, které by zbytečně zpomalovaly provoz informačního systému. V případě, že je záznam ze zdroje dat pro datový sklad odstraněn, aniž by došlo k nastavení tohoto systémového pole, jde o chybný záznam a má být také odstraněn z datového skladu, protože by zkresloval výsledky analýz.

7 Implementované řešení

Tato kapitola se zabývá nástroji zvolenými k implementaci datového skladu a důvody jejich volby. Na trhu s nástroji pro datové skladování je vcelku velký výběr jak open source řešení, tak i komerčních variant.

Při volbě nástrojů pro datové skladování bylo vycházeno zejména z diplomových prací studentů, kteří se zabývali přímo porovnáním těchto nástrojů [15]. Jako nejvhodnější řešení bylo zvoleno řešení společnosti Pentaho, které nabízí kompletní řešení datového skladovaní, kromě relační databáze. Tou byla zvolena databáze Firebird stavějící na relační databázi InterBase. Tento databázový systém byl zvolen z důvodu jeho použití na primárním zdroji datového skladu (informačním systém Tempo). Mezi další důvody patří dostatečný výkon při vhodném nastavení, možnost provozování databáze na různých platformách a také volná dostupnost tohoto řešení. V této části diplomové práce jsem zejména čerpal z dokumentace společnosti Pentaho [17] a [7].

7.1 Databáze Firebird

Jak již bylo zmíněno, jako databázový systém pro datový sklad byla zvolena relační databáze Firebird. Jedná se o databázový systém typu klient server. Pro komunikaci mezi klientem a serverem používá protokol TCP/IP. Minimální instalační požadavky na místo na disku se pohybují v rozmezí od 9MB do 12MB dle platformy, pro plnohodnotnou práci však nároky rostou. Mezi podporované platformy patří Windows, Solaris, Linux, FreeBSD, MacOS X a další. Firebird podporuje SQL standard z roku 1992 (SQL92) a nabízí navíc některá rozšíření jako uložené procedury, spouště, uživatelské role a různé další. Podstatnou vlastností je také možnost použití externích funkcí uložených v knihovnách DLL nebo SO. Pro datový sklad je využita volně dostupná knihovnu Rfunc, které nabízí množství funkcí. Co se týče schopnosti uchovávat data, dokáže databáze Firebird dle [14] efektivně pracovat s databázemi o velikosti desítek terabajtů a uchovávat více než dvě miliardy záznamů v každé tabulce, což je pro implementovaný datový sklad dostačující. Na běžné konfiguraci je databáze schopna obsluhovat přibližně 150 současných připojení. Při nasazení na výkonném HW a vhodnou konfigurací serveru je možné obsloužit řádově více, dle [14] to je až 3000 současných uživatelů ve špičce.

7.2 Nástroje business intelegence společnosti Pentaho

Společnost Pentaho nabízí kompletní balík nástrojů pro oblast business inteligence[5]. V oblasti open source existují další nástroje, ale většinou neposkytují tak komplexní řešení. Jedná se například o nástroje Jaspersoft BI, Palo, SpagoBI a další. V komerční oblasti pak jde zejména o nástroje BusinessObjects společnosti SAP, Business Intelligence společnosti Oracle, nástroje BI společnosti Microsoft a další.

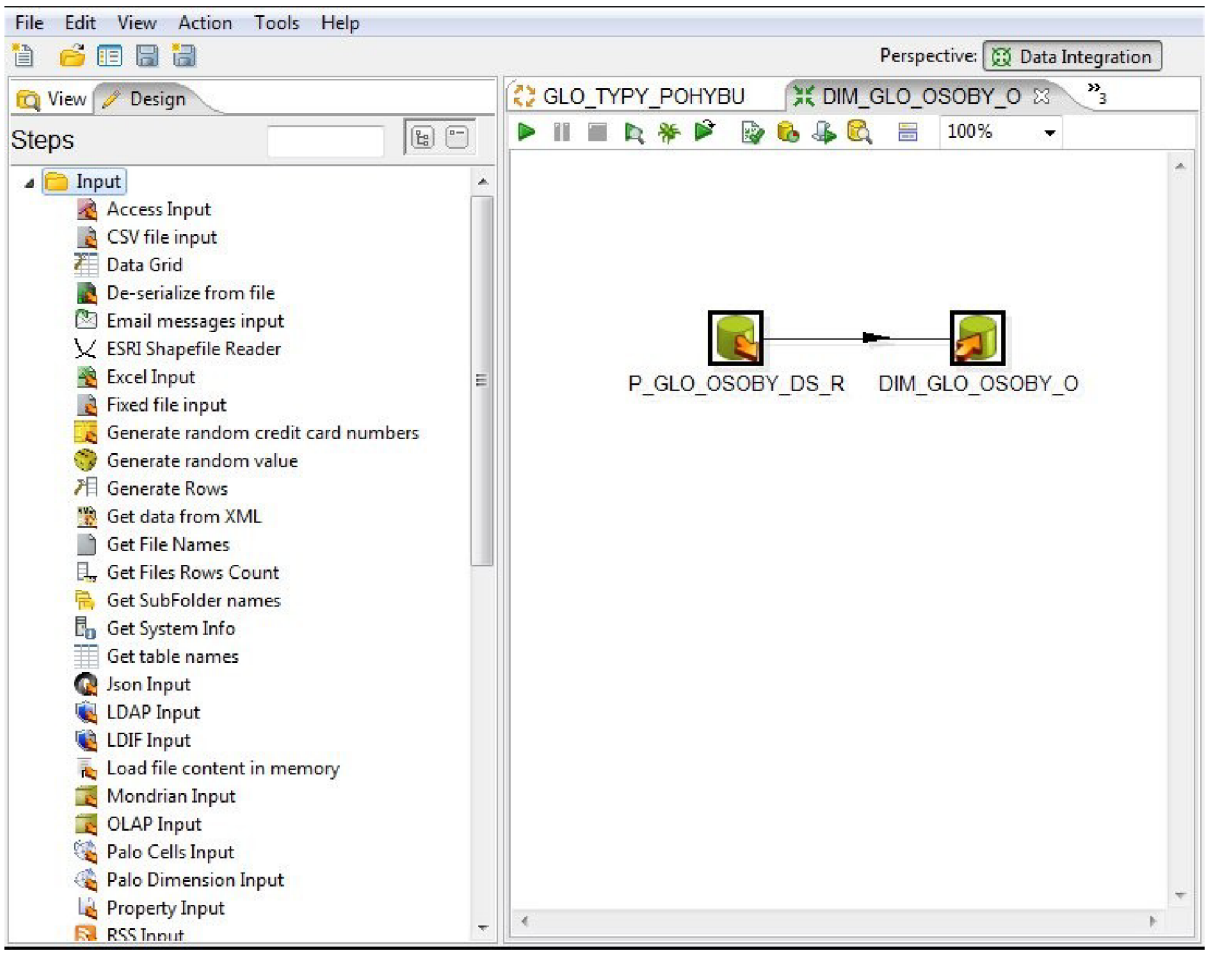
Mezi nástroje pro BI, které nabízí společnost Pentaho, patří Pentaho data integration (PDI), který slouží k provádění procesu ETL a Pentaho analysis service (PAS), sloužící k OLAP analýze dat v relačním datovém skladu. Ten je součástí většího balíku Pentaho BI server. Dále společnost Pentaho nabízí nástroje na generování sestav, ale ty nejsou pro tuto diplomovou práci důležité.

5 Jde o nástroje sloužící k podpoře rozhodování.

7.2.1 Pentaho data integration

PDI je nástroj vytvořený v javě sloužící k definici ETL procesů a je sestaven z několika částí. Z pohledu této diplomové práce jsou zejména důležité tyto: Spoon, Pan, Kitchen a Carte.

Spoon představuje grafické rozhraní nástroje, umožňuje definovat toky dat a operace nad nimi. Tyto operace nad tokem mají podobu komponent, kterých PDI poskytuje velké množství a je možné definovat vlastní. Na ilustraci 7.1 je znázorněno prostředí nástroje PDI a část ETL procesu plnění datového skladu. ETL proces vytvořený v nástroji je možné hierarchicky strukturovat do takzvaných úkolů[6] a transformací. Ilustrace 7.1 zobrazuje transformaci vykonávající přesun dat ze zdrojové databáze do odkládací oblasti datového skladu.



Ilustrace 7.1: Prostředí nástroje PDI

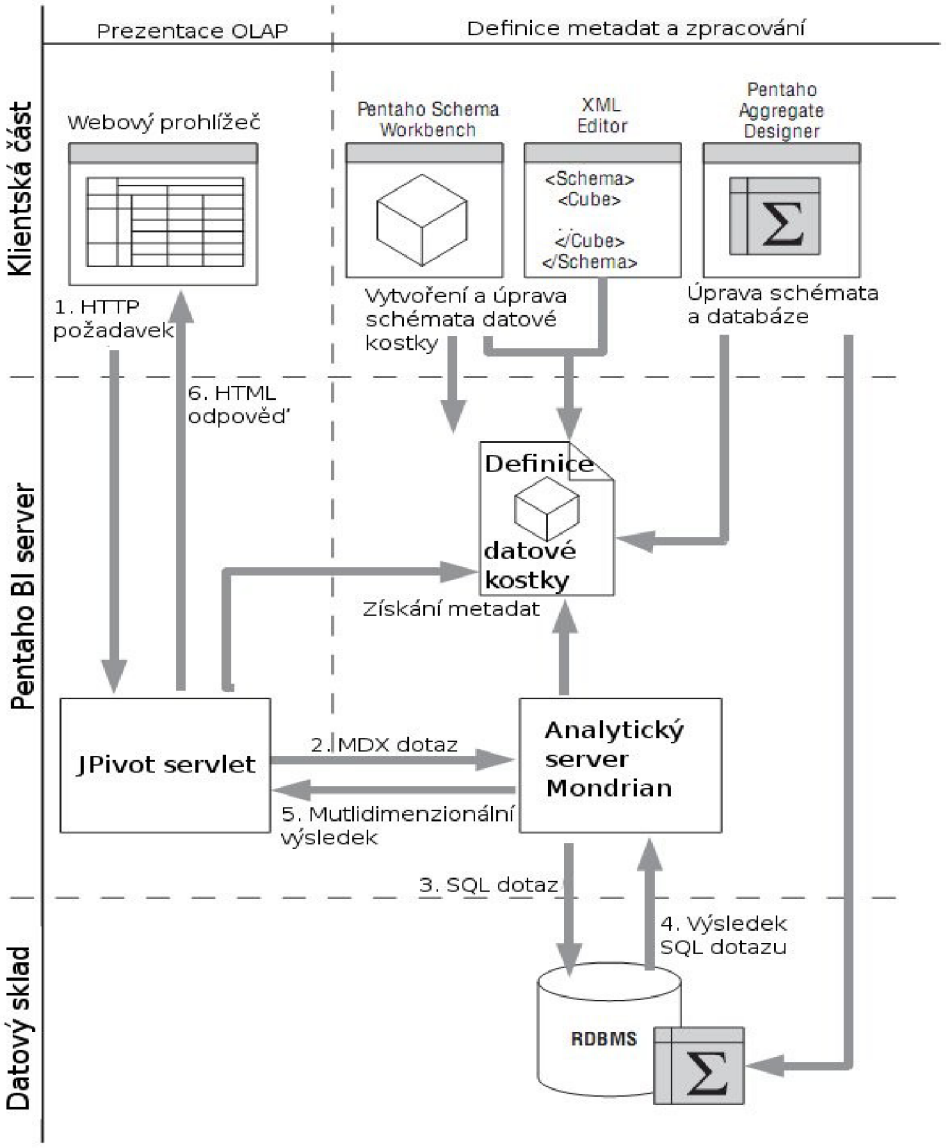
Transformace typicky představuje vytvoření datového toku, úpravu dat v datovém toku pomocí komponent transformace a jeho následné uložení. Typickými komponentami transformace je například načtení dat z datového zdroje, porovnání dvou datových toků a další. Tyto komponenty transformace zejména slouží k řízení a úpravě dat datového toku. Úkol potom obsahuje tyto transformace, podřízené úkoly a komponenty úkolu, které mají na starosti jiné úkony než komponenty transformace. Komponenty úkolů mohou například odeslat email při chybě běžícího procesu, komprese výstupních souborů či přístup na FTP. Na rozdíl od komponent transformací nemají na starosti úpravu dat. Takto lze vytvořit hierarchickou strukturu, která definuje ETL proces.

6 V originále Jobs

PDI dále umožňuje tyto transformace a úkoly spouštět, případně krokovat pokud potřebujeme odhalit chybu v námi vytvořeném procesu ETL. Ke spouštění transformací slouží nástroj Pan a pro spouštění úkolů pak nástroj Kitchen. Standardně se používá ve spolupráci s plánovači, například nástrojem Kron. Nástroj Carte pak představuje webový server, který slouží jako monitor spuštěných procesů.

7.2.2 Pentaho analysis service

Jádrem PAS (Pentaho analysis service) je relační OLAP server Mondrian napsaný v javě dle [16]. Server zpracovává MDX dotazy nad datovou kostkou a ty transformuje na běžné SQL dotazy. Ty jsou prováděny nad relační databází představující datový sklad. Jazyk MDX byl zaveden



Ilustrace 7.2: Komponenty PAS dle [7]

společností Microsoft jako součást jejich produktu Microsoft SQL server pro analýzu multidimenzionálních dat.

O prezentaci získaných dat se stará komponenta JPivot, která zprostředkovává rozhraní mezi uživatelem a datovou kostkou. Společnost Pentaho pracuje na jiném rozhraní PAT (Pentaho analysis tool), který by v budoucnu měl nahradit JPivot, ale zatím je nestabilní a používá se JPivot.

Proto, aby Mondrian věděl, jak mapovat MDX dotazy na relační databázi, potřebuje definici datové kostky v podobě metadat. Ta představují schema datové kostky uloženého v XML. Vytvoření tohoto schematu má na starosti nástroj Schema workbench, který slouží k jejich návrhu a testování.

Poslední součástí balíku nástrojů PAS je nástroj Agregate designer, který slouží k vytváření agregačních tabulek. Jejich účel je popsán v kapitole 5.1.5.

Ilustrace 7.2 představuje architekturu PAS a průběh zpracování dotazu na OLAP server. Uživatel zadá OLAP dotaz, přes rozhraní JPivot. Ten dále předa MDX dotaz serveru Mondrian, který transformuje MDX na SQL. K této transformaci je použito XML schéma datové kostky. SQL dotaz je zpracován datovým skladem v podobě relační databáze a vrátí výsledek zpět ROLAP serveru. Tato relační data jsou transformována na multidimenzionální a přestavují výsledek MDX dotazu. Tento výsledek je prezentován uživateli pomocí JPivot ve webovém prohlížeči.

8 Implementace datového skladu

Kapitola se zabývá požadavky a implementací datového skladu. Dále jsou zde popsány jednotlivé dimenze a tabulky faktů implementovaného datového skladu a systémová pole nutná pro jeho správnou funkčnost.

8.1 Požadavky

Hlavním požadavkem na implementovaný datový sklad je archivace záznamů informačního systému Tempo IS podle zákona o účetnictví 563/1991 Sb. §31. Podle tohoto zákona je nutné uchovávat účetní doklady, účetní knihy, odpisové plány a další položky uvedené v zákoně po dobu 5 let počínajících koncem účetního období, kterého se týkají.

V souvislosti s tímto požadavkem je možnost definovat data, která budou v databázi informačního systému zrušena a uchovávána pouze v datovém skladu.

Posledním požadavkem je, aby datový sklad dokázal poskytovat přehledy nad moduly informačního systému. Pro informační systém je v rámci diplomové práce vytvořen prototyp datového skladu nad modulem pokladna, který bude s ostatními tabulkami faktů nad ostatními moduly propojen pomocí společných dimenzí. Dle tohoto prototypu pak budou vytvořeny další datové trhy, které budou využívat již existujících společných dimenzí. Tato část již není součástí diplomové práce.

8.2 Návrh datového skladu

Pro implementaci datového skladu byla navržena tabulku všeobecně přijatých dimenzí, z níž pak bylo vycházeno při implementaci. Tabulka 8.1 představuje navržený datový sklad. Jednotlivé tabulky faktů jsou navrženy podle kořenových tabulek modulů informačního systému Tempo, což jsou tabulky, v nichž jsou ukládány hlavní data pro každý modul. Tabulky dimenzí jsou pak navrženy podle cizích klíčů na tyto kořenové tabulky a dále sjednocením tabulek rozdělených při normalizaci či sjednocením různých číselníků.

Při návrhu datového skladu nebyl kladen důraz na hledání určitých dotazů, které budou na datový sklad kladeny, ale na obecnou kostru a zachycení důležitých informací, které je nutné uchovávat. Nad navrženým datovým skladem lze samozřejmě provádět OLAP analýzu, což je předvedeno v poslední kapitole, ale sklad není k tomuto optimalizován. Jeho hlavním úkolem je skladovat data, které není nutné uchovávat v databázi provozního informačního systému Tempo. Následné optimalizace datového skladu budou prováděny po konzultaci se zákazníkem, tak aby mu byl datový sklad přizpůsoben na míru. Samotný datový sklad je tedy prototyp s plnou funkčností archivace, bez optimalizace pro analýzy. Přizpůsobením datového skladu na míru zákazníkovi je zejména vytváření agregačních tabulek pro zrychlení dotazů, které jsou pro zákazníka důležité. Případně různé jiné optimalizace, které není možné vytvořit dopředu, a jsou závislé na přání zákazníka.

Fakta / Dimenze	BIL_BILANCE	POK_DOKLADY	POK_DOKLADY_ROZPIS	POK_BILANCE	BAN_POHYBY_VYPISU	BAN_POHYBY_VYPISU_ROZPIS	BAN_BILANCE	POH_POHLEDAVKY	POH_POHLEDAVKY_ROZPIS	POH_BILANCE	SKL_BILANCE	UCT_BILANCE	UCT_UCETNICTVI	VPR_BILANCE	VPR_VYKAZ_PRACE	VST_BILANCE	VST_VYKAZ_STROJU	VVO_BILANCE	ZAV_BILANCE	ZAV_ZAVAZKY	ZAV_ZAVAZKY_ROZPIS
DIM_BIL_AGENDY	X			X			X			X	X	X	X	X		X		X	X		
DIM_CAS	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
DIM_DATUM	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
DIM_GLO_ADRESY		X	X					X	X												
DIM_GLO_DRUHY_ROZPISU			X			X															X
DIM_GLO_FIRMY	X	X	X	X	X		X	X	X	X	X	X	X	X	X	X		X	X	X	
DIM_GLO_MENY		X	X			X		X	X											X	
DIM_GLO_MERNE_JEDNOTKY	X		X	X		X	X		X	X	X	X		X	X	X	X	X	X		X
DIM_GLO_OSOBY	X	X	X	X										X	X		X				
DIM_GLO_POLOZKY_UTVARU	X		X	X		X	X		X	X	X	X		X	X	X	X	X	X		X
DIM_GLO_TYPY_NAKLADU	X										X			X							
DIM_GLO_TYPY_POHYBU		X		X	X		X			X		X							X		
DIM_GLO_UTVARY	X		X	X		X	X		X	X	X	X		X	X	X	X	X	X		X
DIM_SKL_SKLADY			X			X			X		X										X
DIM_SYS$BOOKS	X	X	X	X	X	X	X	X	X	X	X	X		X	X	X	X	X	X	X	X
DIM_SYS$PERIODS	X	X	X	X	X	X	X	X	X	X	X	X		X	X	X	X	X	X	X	X
DIM_UCT_ROZVRH			X	X		X	X	X	X	X	X	X	X						X	X	X
DIM_UCT_SAZBY_DPH			X			X			X												X
DIM_UCT_TYPY_DPH			X			X			X												X
DIM_VVO_VOZIDLA	X																	X			
DIM_VST_STROJE	X																X				

Tabulka 8.1: Všeobecně přijaté dimenze datového skladu

8.3 Struktura datového skladu

Pro datový sklad byly vytvořeny tyto všeobecně přijaté dimenze podle tabulky 8.1. Obsah jednotlivých tabulek dimenzí i faktů je uveden v příloze. Následuje popis systémových polí, která mají vliv na funkčnost procesu ETL a samotného datového skladu. Implementovaná část datového skladu tedy obsahuje tyto dimenze:

- DIM_CAS
- DIM_DATUM
- DIM_GLO_ADRESY
- DIM_GLO_BANKOVNI_UCTY
- DIM_GLO_DRUHY_ROZPISU
- DIM_GLO_FIRMY
- DIM_GLO_MENY
- DIM_GLO_MERNE_JEDNOTKY
- DIM_GLO_OSOBY
- DIM_GLO_POLOZKY_UTVARU
- DIM_GLO_TYPY_POHYBU
- DIM_GLO_UTVARY
- DIM_SKL_SKLADY
- DIM_SYS$BOOKS
- DIM_SYS$PERIODS
- DIM_UCT_ROZVRH
- DIM_UCT_SAZBY_DPH

- DIM_UCT_TYPY_DPH

Kromě dimenzí *DIM_CAS* a *DIM_DATUM* obsahují všechny ostatní dimenze systémová pole, která jsou nutná pro správnou funkčnost datového skladu. Dle těchto polí jsou pomocí spouští dohledávány aktuální záznamy a zejména propojovány dimenze s tabulkami faktů. Systémová pole použitá v tabulkách dimenzí jsou tato:

- ZAZNAM_VALIDNI_OD – určuje počátek platnosti záznamu,
- ZAZNAM_VALIDNI_DO – určuje konec platnosti záznamu,
- ZAZNAM_VERZE – specifikuje verzi záznamu, vytváří číselnou řadu při archivaci záznamu,
- ZAZNAM_AKTUALNI – definuje, zda je záznam platný, tedy zda stále existuje ve zdrojové, databázi
- ZAZNAM_POSLEDNI_MODIFIKACE – určuje datum a čas poslední modifikace záznamu,
- ZAZNAM_GUID – jednoznačná identifikace zdroje dat,
- ZAZNAM_ID_REFERENCE – identifikace záznamu, v případě že záznam má více verzí, je hodnota tohoto pole stejná,
- ID – unikátní hodnota v rámci dimenze, přes toto pole jsou záznamy v dimenzi vázány na tabulku faktů.

DIM_GLO_FIRMY	
ID	INTEGER
NAZEV_FIRMY	VARCHAR(25)
UPLNY_NAZEV_FIRMY	VARCHAR(83)
CISLO_FIRMY	INTEGER
NAZEV_TYPU_FIRMY	VARCHAR(25)
ICO_FIRMY	VARCHAR(10)
DIC_FIRMY	VARCHAR(25)
ZEME_REGISTRACE_FIRMY	VARCHAR(50)
AKTIVNI_FIRMA	SMALLINT
VLASTNI_FIRMA	SMALLINT
INFO_O_ZAPISU_FIRMY_DO_OR	VARCHAR(100)
FINANCNI_URAD_FIRMA	VARCHAR(60)
CINNOST_FIRMY	VARCHAR(100)
PRIJMENI_JEDNATELE_FIRMY	VARCHAR(25)
JMENO_JEDNATELE_FIRMY	VARCHAR(25)
VZTAH_JEDNATELE_FIRMY	VARCHAR(100)
ICO_JEDNATELE_FIRMY	VARCHAR(10)
KOD_JEDNATELE_FIRMY	VARCHAR(10)
ZAZNAM_VALIDNI_OD	TIMESTAMP
ZAZNAM_VALIDNI_DO	TIMESTAMP
ZAZNAM_VERZE	INTEGER
ZAZNAM_AKTUALNI	SMALLINT
ZAZNAM_POSLEDNI_MODIFIKACE	TIMESTAMP
ZAZNAM_GUID	CHAR(16)
ZAZNAM_ID_REFERENCE	INTEGER

Ilustrace 8.1: Dimenze DIM_GLO_FIRMY

Dimenze *DIM_CAS* a *DIM_DATUM* jsou generované vlastními procedurami v jazyce PSQL a uvedené systémová pole pro ně nemají význam. Ilustrace 8.1 prezentuje tabulku dimenzí *DIM_GLO_FIRMY*. Kompletní struktura všech tabulek dimenzí a tabule faktů je uvedena v příloze.

Pro modul pokladna informačního systému Tempo byly vytvořeny tyto tabulky faktů: *POK_DOKLADY* a *POK_DOKLADY_ROZPIS*.

Ilustrace 8.2 představuje tabulku faktů *POK_DOKLADY*. Pro další moduly pak budou dle vzoru vytvořeny další tabulky faktů s využitím již existujících všeobecně přijatých dimenzí. Tabulky faktů obsahují tyto systémové záznamy:

- ZAZNAM_GUID, ZAZNAM_ID_REFERENCE – pole mají stejný význam jako u tabulek dimenzí
- ZAZNAM_ARCHIVNI – specifikuje, zda je záznam archivní, detailní popis archivace je popsán v následující kapitole

POK_DOKLADY	
ID	INTEGER
ID_DIM_SYS$PERIODS	INTEGER
ID_DIM_SYS$BOOKS	INTEGER
ID_DIM_GLO_TYPY_POHYBU	INTEGER
ID_DIM_GLO_FIRMY	INTEGER
ID_DIM_GLO_OSOBY	INTEGER
ID_DIM_GLO_MENY	INTEGER
ID_DIM_CAS	INTEGER
ID_DIM_DATUM	INTEGER
ID_DIM_GLO_ADRESY	INTEGER
CELKEM	NUMERIC(16,2)
CELKEM_VLASTNI_MENA	NUMERIC(16,2)
ZAZNAM_ARCHIVNI	SMALLINT
ZAZNAM_GUID	CHAR(16)
ZAZNAM_ID_REFERENCE	INTEGER

Ilustrace 8.2: Tabulka faktů POK_DOKLADY

9 ETL

Tato kapitola se zabývá implementací procesu ETL. Nejprve je pojednáno o extrakci dat na úrovni databáze, dále je vysvětlena implementace ETL pomocí nástroje PDI a nakonec je vysvětlen přesun dat z odkládací oblasti do datového skladu.

9.1 Extrakce dat

Pro získání dat, kterými je datový sklad plněn, bylo přistoupeno způsobem minimalizujícím dopady na databázi informačního systému Tempo, která slouží jako primární zdroj dat. Implementovaná extrakce dat se pak dá rozdělit na dvě části, v první části se jedná o upravení zdrojové databáze tak, aby dokázala jednoduše poskytovat data vhodná pro datový sklad. Dále část mimo zdrojovou databázi, která byla vytvořena pomocí nástroje PDI (Pentaho data integration) a zejména slouží k přenosu dat do odkládací oblasti. Při plnění datového skladu se nepředpokládá použití více zdrojů dat, ale přesto byla při návrhu skladu zohledněna tato možnost.

9.1.1 Změny nad zdrojovou databází

V implementovaném datovém skladu jsou data získávána převážně z jednoho typu zdroje dat, jímž je databáze informačního systému Tempo.

Před zavedením datového skladu existovala ve zdrojové databázi informačního systému tabulka *SYS$REFERENCES* do níž byly ukládány informace o založení nového záznamu, zejména se jedná o datum a čas založení. V této tabulce je záznam identifikovatelný svým jedinečným identifikátorem *ID_REFERENCE* v rámci celé databáze. Pro minimalizaci změn v databázi informačního systému byla tato tabulka rozšířena o položku identifikující datum a čas změny záznamu a sadu spouští, která v případě změny záznamu zapíše datum a čas změny.

Poslední úpravou nad zdrojovou databází, která slouží pro získání dat, bylo založení speciálních extrakčních procedur s parametrem datum extrakce, pro každou tabulku datového skladu. Procedury jsou napsány v jazyku PSQL, který je součástí databáze Firebird. Tato procedura vrací množinu záznamů, které slouží k plnění tabulek dimenzí a faktů v závislosti na vstupu a obsahu tabulky *SYS$REFERENCES*. Pokud je čtecí proceduře předán parametr *NULL*, vrací procedura veškeré záznamy a je ekvivalentem prvotního plnění, v případě, že je proceduře předáno určité datum, procedura v závislosti na obsahu tabulky *SYS$REFERENCES* vrátí pouze záznamy, které mají datum vytvoření nebo změny větší jak datum na vstupu. Toto je ekvivalentní inkrementálnímu plnění. Kromě hodnot pro plnění DS daty, vrací tyto procedury také systémové záznamy nutné pro správnou funkčnost datového skladu. Jde o položky *ZAZNAM_ZMENEN*, která uchovává datum a čas poslední změny záznamu, *ZAZNAM_EXTRAHOVAN*, uchovávající, kdy byl záznam získán z databáze. Dále jde o pole *ZAZNAM_GUID*, které jednoznačně identifikuje zdroj dat pro datový sklad, a *ZAZNAM_ID_REFERENCE*, který je pro každý záznam ve zdrojové databázi jedinečný. Kombinace záznamů *ZAZNAM_GUID* a *ZAZNAM_ID_REFERENCE* pak umožňuje jednoznačně identifikovat původ záznamu v datovém skladu. Toto řešení pomocí extrakčních procedur také umožňuje provádět některé transformace dat již při jejich získávání. Jelikož je zdrojová databáze dobře navržena, nenastává tato situace příliš často. Ukázka kódu 9.1 prezentuje jednu z extrakčních procedur, která musí být pro správnou funkčnost extrakce založena ve zdrojové databázi.

V případě zrušení záznamu ve zdrojové databázi je záznam z tabulky *SYS$REFERENCES* zrušen. Tento přístup byl ponechán původní, jak byl před implementací datového skladu. Toto je zejména z důvodu, že datový sklad nemusí být vždy součástí dodávaného řešení informačního systému Tempo. Případné úpravy tohoto mechanismu by zbytečně zatěžovaly informační systém bez datového skladu. Toto vede k jistým komplikacím při zjišťování smazaných záznamů ve zdrojové

```
CREATE PROCEDURE P_SYS$BOOKS_DS_R (
    POSLEDNI_EXTRAKCE TIMESTAMP)
RETURNS (
    ...
    ZAZNAM_ZMENEN TIMESTAMP,
    ZAZNAM_EXTRAHOVAN TIMESTAMP,
    ZAZNAM_GUID CHAR(16),
    ZAZNAM_ID_REFERENCE INTEGER)
AS
DECLARE VARIABLE IDENTIFIKATOR_DATABAZE VARCHAR(16);
BEGIN
  SELECT FIRST 1 GUID FROM SYS$DB_VERSION INTO :IDENTIFIKATOR_DATABAZE;
  FOR SELECT
          ...
          :IDENTIFIKATOR_DATABAZE,
          MTBL.ID_REFERENCE
      FROM SYS$BOOKS MTBL
      LEFT JOIN SYS$BOOKS_TYPES SBT ON (SBT.ID = MTBL.ID_BOOK_TYPE)
      INTO
          ...
          :ZAZNAM_EXTRAHOVAN,
          :ZAZNAM_GUID,
          :ZAZNAM_ID_REFERENCE
  DO
  BEGIN
    SELECT FIRST 1
        COALESCE(SR.UPDATE_TIMESTAMP, SR.CREATE_TIMESTAMP) AS DATUM_ZMENENO
    FROM SYS$REFERENCES SR
    WHERE SR.ID = :ZAZNAM_ID_REFERENCE
    ORDER BY DATUM_ZMENENO DESC
    INTO ZAZNAM_ZMENEN;
    IF ((ZAZNAM_ZMENEN>POSLEDNI_EXTRAKCE)OR(POSLEDNI_EXTRAKCE IS NULL))THEN
      SUSPEND;
  END
END
```

Ukázka kódu 9.1: Příklad extrakční procedury

databázi, protože je při každé extrakci nutné porovnat obsah tabulky datového skladu a zdrojové databáze. Řešením tohoto problému bylo založení již zmíněného systémového záznamu *ZAZNAM_ID_REFERENCE* v každé tabulce datového skladu. Hodnota tohoto záznamu je plněna pomocí extrakčních procedur a její hodnota je získána z pole *ID_REFERENCE* zdrojové databáze. Pro získání zrušených záznamů pak stačí porovnat seznam hodnot *ZAZNAM_ID_REFERENCE* z datového skladu a *ID_REFERENCE* ze zdrojové databáze. Protože se jedná o indexované číselné identifikátory, je nalezení chybějících záznamů velice rychlé.

Implementovaná extrakce je tedy kombinací obou typů odložené extrakce, jak pomocí časových razítek, tak pomocí porovnání. V konečném důsledku je tedy připojení datového skladu k databázi informačního systému Tempo podmíněno pouze založením extrakčních procedur, které mají na starosti získávaní dat. Ostatní změny jsou pak využívány i samotným informačním systémem a jsou tedy potřebné oboustranně.

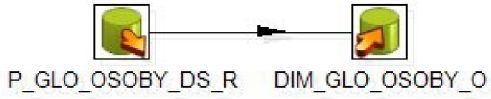
9.1.2 Proces extrakce vytvořený pomocí PDI

Celý proces ETL pro plnění implementovaného datového skladu je řízen pomocí transformací a úkolů vytvořených v nástroji PDI. Tento vytvořený proces budu dále nazývat přenos.

Ilustrace 9.1: Řídící vrstva přenosu

Ilustrace 9.1 představuje nejvyšší vrstvu přenosu, která má na starosti jeho řízení. Komponenta *PROMENNE_PRENOSU* má na starosti získání hodnot, které přenos řídí a mohou být pro každý přenos různé. Jde zejména o datum a čas minulé extrakce, který je předán extrakční proceduře. Ta již rozhodne, zda se jedná o prvotní plnění či jde o inkrementální plnění datového skladu. Dále je zde získáno *GUID* zdroje dat.

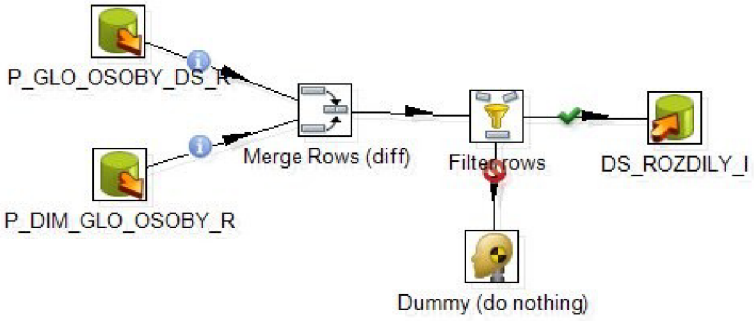
Komponenta *NAPLN_ODKLADACI_OBLAST* přenáší data ze zdroje dat získaná pomocí extrakčních procedur a plní jimi odkládací oblast. Dále je zde provedeno nalezení zrušených záznamů ve zdrojové databázi.



Ilustrace 9.2: Plnění odkládací oblasti dimenze osoby

Ilustrace 9.2 zobrazuje plnění odkládací oblasti pro dimenzi osob. Komponenta *P_GLO_OSOBY_DS_R* volá stejnojmennou extrakční proceduru, která získá data a následně je předá komponentě *DIM_GLO_OSOBY_O*, která se postará o jejich zápis do odkládací oblasti.

Ilustrace 9.3 ukazuje získání smazaných záznamů ze zdrojové databáze, nejprve jsou porovnány seznamy položek *ZAZNAM_ID_REFERENCE* a *ID_REFERENCE* v komponentě *Merge Rows*, dále jsou vyfiltrovány pouze položky, které jsou v datovém skladu navíc oproti zdrojové



Ilustrace 9.3: Získání smazaných záznamů

databázi. Tyto záznamy jsou komponentou *DS_ROZDILY_I* zapsány do odkládací oblasti.

Do této fáze je postup stejný jak pro dimenze, tak pro tabulky faktů. Rozdíl nastává až ve fázi, kdy jsou data přenášena z odkládací oblasti do datového skladu. Ten je vyvolán komponentou *NAPLN_DATOVY_SKLAD* z ilustrace 9.1, která volá procedury uložené v datovém skladu. Tyto procedury mají na starosti přesun dat z odkládací oblasti do tabulek datového skladu.

9.2 Transformace a přesun dat

Jelikož je datový sklad budován pro databázi aplikace Tempo IS, která je navržena tak, aby co nejvíce splňovala třetí normální formu a byla zajištěna integrita dat, nebyla transformace přesouvaných dat příliš rozsáhlá. Malá nutnost transformací je také způsobena tím, že jsou data pro datový sklad

získávána z jednoho zdroje. Transformaci, kterou bylo nutné provést, bylo zejména nahrazení hodnot cizích klíčů číselníků vlastními hodnotami, tedy provedení denormalizace. Toto je provedeno přímo v extrakčních procedurách popsaných v kapitole 9.1.1. Dalším úkolem transformace bylo sjednocení volných zadání například adresy, zadávané přímo na pokladní doklad a adres v jejich číselníku. Zde vyvstává problém identifikace shodných záznamů, popsaný v kapitole 6.3. Tento problém je řešen v průběhu přesunu dat z odkládací oblasti do datového skladu.

Pro tento účel byly napsány procedury provádějící transfer dat v jazyku PSQL. Tyto procedury jsou volány z komponenty *NAPLN_DATOVY_SKLAD* z ilustrace 9.1. Ta definuje pořadí, v jakém jsou transferové procedury volány. Toto je nutné, protože datový sklad má strukturu sběrnice a některé dimenze jsou závislé na jiných dimenzích. Příkladem takové závislé dimenze je dimenze adres, která je závislá nejen na tabulce faktů, ale zároveň je použita dimenzemi firem a osob. Dále je nutné nejprve plnit dimenze před tabulkami faktů, protože zde existuje závislost již z podstaty datového skladu. Nejprve jsou tedy naplněny dimenze a aktualizovány vazby mezi závislými dimenzemi. Následně jsou plněny tabulky faktů a jednotlivé záznamy připojovány na záznamy dimenzí.

9.2.1 Systémové tabulky

Datový sklad pro svoji správnou funkčnost obsahuje systémové tabulky, které slouží k nastavení datového skladu. Jde o tabulky *SYS$ARCHIVOVANA_POLE*, *SYS$LOG_PRESUNU*, *SYS$NASTAVENI* a *SYS$SMAZANE*. Tabulka *SYS$ARCHIVOVANA_POLE* definuje, jak se přistupuje k ukládání dat v tabulkách dimenzí. Pokud je v tabulce specifikován záznam určité tabulky, je použit hybridní přístup, jinak je implicitně použit typ SCD1. *SYS$LOG_PRESUNU* uchovává všechny operace nad datovým skladem provedené pomocí ETL, je zde uložen datum a čas zahájení operace, identifikace zpracovaného záznamu, název tabulky a operace. Tabulka *SYS$NASTAVENI* uchovává různá nastavení pro datový sklad, zejména jde o minimální a maximální hodnoty datumů, které mohou být v datovém skladu použity, dále jsou to výchozí hodnoty pro záznamy, které nebyly vyplněny, ale v datovém skladu musí být. Poslední tabulka *SYS$SMAZANE* slouží k zachycení záznamů, které byly smazány ze zdrojové databáze a mají být smazány z datového skladu.

9.2.2 Plnění tabulek dimenzí

Při plnění dimenzí je nejprve nutné dohledat, zda vkládaný záznam již existuje, nebo se jedná o nový. V případě, že se jedná o nový záznam, je založen a vyplněn dle odkládací oblasti. Toto aktivuje spoušť, která má na starosti plnění systémových záznamů nutných pro správnou funkčnost datového skladu. Ukázka kódu 9.2 zobrazuje toto přednastavení.

V případě, že již záznam v datovém skladu existuje, je zjištěno, zda se změnila položka, pro kterou chceme uchovávat historii. V případě, že historii nad změněnou položkou chceme uchovávat, je založen nový záznam, čímž se opět vyvolá spoušť z ukázky kódu 9.2. To zajistí správné vyplnění verze a dalších systémových položek. Jestliže se nejedna o změnu nad položkou, pro kterou chceme historii uchovávat, je záznam pouze aktualizován a je mu nastaveno datum poslední modifikace. Pro plnění dimenze je tedy podle nastavení použit buď přístup, hybridní nebo přístup SCD1 uvedený v kapitole 6.4.1. To zda pro záznam dimenze chceme uchovávat historii je definováno v systémové tabulce *SYS$ARCHIVOVANA_POLE*. Zde se specifikuje tabulka a záznamy pro které chceme v datovém skladu používat hybridní přístup.

```
CREATE OR ALTER TRIGGER T_DIM_GLO_ADRESY_BI_Z FOR DIM_GLO_ADRESY
ACTIVE BEFORE INSERT POSITION 1
AS
DECLARE VARIABLE ID_POSLEDNIHO_ZAZNAMU_V_HISTORII TYPE OF COLUMN DIM_GLO_ADRESY.ID;
DECLARE VARIABLE POSLEDNI_VERZE_V_HISTORII TYPE OF COLUMN DIM_GLO_ADRESY.ZAZNAM_VERZE;
DECLARE VARIABLE NEJMENSI_DATUM TYPE OF COLUMN DIM_GLO_ADRESY.ZAZNAM_VALIDNI_OD;
BEGIN
  NEW.ZAZNAM_AKTUALNI = 1;
  IF (NEW.ZAZNAM_VALIDNI_OD IS NULL) THEN
    NEW.ZAZNAM_POSLEDNI_MODIFIKACE = CURRENT_TIMESTAMP;
  ELSE
    NEW.ZAZNAM_POSLEDNI_MODIFIKACE = NEW.ZAZNAM_VALIDNI_OD;

  SELECT PM.NEJVETSI_DATUM, PM.NEJMENSI_DATUM
  FROM SYS$NASTAVENI PM
  INTO NEW.ZAZNAM_VALIDNI_DO, :NEJMENSI_DATUM;

  ID_POSLEDNIHO_ZAZNAMU_V_HISTORII = NULL;
  SELECT FIRST 1 DA.ID, DA.ZAZNAM_VERZE
  FROM DIM_GLO_ADRESY DA
  WHERE (NEW.ZAZNAM_GUID IS NOT DISTINCT FROM DA.ZAZNAM_GUID) AND
        (NEW.ZAZNAM_ID_REFERENCE = DA.ZAZNAM_ID_REFERENCE)
  ORDER BY DA.ZAZNAM_VERZE DESC
  INTO :ID_POSLEDNIHO_ZAZNAMU_V_HISTORII, :POSLEDNI_VERZE_V_HISTORII;

  IF (ID_POSLEDNIHO_ZAZNAMU_V_HISTORII IS NULL) THEN
  BEGIN --Prvni zaznam v historii
    NEW.ZAZNAM_VALIDNI_OD = :NEJMENSI_DATUM;
    NEW.ZAZNAM_VERZE = 1;
  END ELSE
  BEGIN --pridava dalsi zaznam do historie
    NEW.ZAZNAM_VERZE = POSLEDNI_VERZE_V_HISTORII + 1;
    IF (NEW.ZAZNAM_VALIDNI_OD IS NULL) THEN
      NEW.ZAZNAM_VALIDNI_OD = CURRENT_TIMESTAMP;
    UPDATE DIM_GLO_ADRESY
    SET ZAZNAM_VALIDNI_DO = NEW.ZAZNAM_VALIDNI_OD,
        ZAZNAM_AKTUALNI = 0
    WHERE ID = :ID_POSLEDNIHO_ZAZNAMU_V_HISTORII;
  END
END
```

Ukázka kódu 9.2: Naplnění systémových položek záznamu

Tabulky dimenzí, které jsou vázané na jiné dimenze, navíc obsahují oproti ostatním spoušť, která v případě založení nového záznamu upozorní připojené dimenze, aby se vazba obnovila. Toto prezentuje ukázka kódu 9.3.

Poslední činností, která se děje při přenosu dat z odkládací oblasti do datového skladu, je deaktivace zrušených záznamů. Ta se provádí tak, že se zrušenému záznamu nastaví *ZAZNAM_AKTUALNI* na neaktuální a upraví se mu doba platnosti. Samotné procedury starající se o přenos jsou uvedeny v příloze.

```
CREATE OR ALTER TRIGGER T_DIM_GLO_FIRMY_AI_V FOR DIM_GLO_FIRMY
ACTIVE AFTER INSERT POSITION 0
AS
DECLARE VARIABLE ID INTEGER;
BEGIN
  -- prepojim zanznamy ktere odkazuji na neaktualni zaznam.
  --ADRESY
  FOR SELECT
          MTBL.ID
      FROM DIM_GLO_ADRESY MTBL
      WHERE (MTBL.ZAZNAM_AKTUALNI = 1) AND
            (MTBL.ID_DIM_GLO_FIRMY_PK IN
            (SELECT TBL.ID
             FROM DIM_GLO_FIRMY TBL
             WHERE (NEW.ZAZNAM_GUID IS NOT DISTINCT FROM TBL.ZAZNAM_GUID) AND
                   (NEW.ZAZNAM_ID_REFERENCE = TBL.ZAZNAM_ID_REFERENCE)))
      INTO
          :ID
  DO
  BEGIN
    UPDATE DIM_GLO_ADRESY
    SET ID_DIM_GLO_FIRMY_PK = NEW.ID
    WHERE ID = :ID;
  END
...
```

Ukázka kódu 9.3: Přepojení vazeb dimenzí

U tabulek dimenzí tedy není rozlišeno, zda byl záznam smazán z důvodu chyby při zadávání záznamu nebo proto, že již není potřeba ve zdrojové databázi. Tento přístup byl zvolen, protože v případě, že uživatel udělal chybu při zadávání a tento záznam se dostal do datového skladu, nastává několik variant řešení. S největší pravděpodobností ho uživatel opraví a tato oprava se přenese i do datového skladu. Druhá varianta je, že záznam smaže. V dimenzi datového skladu tedy bude tento záznam deaktivován, ale i když v dimenzi zůstane, nijak neovlivní výsledky dotazů, protože nebude odkazován z tabulky faktů.

9.2.3 Plnění tabulek faktů

Při přenosu dat z odkládací oblasti do tabulek faktů je hlavním problémem dohledání záznamů v dimenzích. Pro tento přenos byly obdobně jako u plnění tabulek dimenzí napsány procedury v jazyku PSQL, které mají plnění tabulek faktů na starosti. Aby proběhlo plnění tabulek faktů řádně, musí být prvně naplněny dimenze, což je zajištěno nástrojem PDI. Jakmile začne přesun faktů z odkládací oblasti, je nejprve zjištěno, zda záznam v tabulce faktů existuje, následně jsou připojovány jednotlivé záznamy v dimenzích. Ty jsou připojovány dle systémového pole *ZAZNAM_GUID* a *ZAZNAM_ID_REFERENCE*, přičemž jsou připojeny pouze ty záznamy, které jsou aktuální a existují i ve zdrojové databázi. V těchto procedurách jsou dále dohledávána volná zadání, což jsou hodnoty, které byly vyplněny přímo v tabulce, z níž jsou získávány data pro tabulky faktů a nebyly zvoleny z číselníku. Toto je provedeno tak, že se procedura pokusí identifikovat volné zadání např. adresy v dimenzi adres. Pokud je takový záznam objeven, je k zpracovávanému řádku tabulky faktů připojena tato nalezená dimenze. V opačném případě se nejprve dané volné zadání založí do dimenze a následně se připojí k záznamu tabulky faktů. Jakmile jsou dohledány všechny dimenze a volná zadání, je záznam zapsán do tabulky faktů jako nový nebo úprava stávajícího, podle toho zda byl zakládaný záznam identifikován v tabulce faktů.

V případě, že se nepodaří dohledat vazbu mezi tabulkou faktů a dimenzí, je použit implicitní záznam, který je založen v každé dimenzi. Tento speciální záznam říká, že pro daný záznam hodnota v dimenze neexistuje.

Na rozdíl od tabulek dimenzí jsou z tabulek faktů mazány záznamy, které byly smazány ve zdrojové databázi. Toto je z důvodu mazání chybných záznamů ve zdrojové databázi. Při zadání

chybného záznamu je tento chybný záznam přenesen do datového skladu. Pokud ho uživatel následně smaže, je tento záznam smazán i z datového skladu. Pokud by se toto nedělo, hromadily by se nám v datovém skladu chybné záznamy a výsledky OLAP analýz by neodpovídaly skutečnosti.

Zde ovšem nastává problém s archivací záznamů, protože v požadavcích je uvedeno, že chceme v datovém skladu uchovávat i archivní záznamy, které jsme z informačního systému odstranily za účelem odlehčení databáze systému. Proto byl do tabulek faktů přidán systémový záznam *ZAZNAM_ARCHIVNI*, podle kterého datový sklad pozná, že daný záznam nemá odstraňovat z datového skladu, protože byl z provozního systému odstraněn úmyslně. Odstraňování záznamů z informačního sytému, dle kterých jsou plněny tabulky faktů, musí být prováděno tak, že než je záznam odstraněn, musí být v datovém skladu danému ekvivalentu rušeného záznamu nastaven příznak archivace.

9.2.4 Rušení dat ze zdrojové databáze

Pro rušení dat ze zdrojové databáze komplexním způsobem tak, aby nebyl porušený celkový stav, byl navržen nástroj, který je implementován současně s datovým skladem. Jeho úkolem je odstranit data ze zdrojové databáze a založit nové počáteční stavy. Zároveň musí tento nástroj mazaným záznamů nastavit v datovém skladu příznak *ZAZNAM_ARCHIVNI*. Mechanismus archivace záznamů v datovém skladu je popsán v předchozí kapitole. Nástroj funguje na principu vytvoření stromové struktury závislostí mezi tabulkami. V této struktuře je jazykem SQL definováno, co se má stát s jednotlivými záznamy tabulky v hierarchii. Následně je možné pomocí vyvíjeného nástroje tuto definici spustit a ta se začne vykonávat od listů směrem ke kořeni. Nástroj má zejména sloužit k rušení záznamů ze zdrojové databáze a zápisu pole *ZAZNAM_ARCHIVNI* do datového skladu.

9.3 Zhodnocení implementovaného ETL procesu

Proces ETL byl testován na PC s procesorem Intel(R) Core(TM)2 Duo CPU E6750 @2,66GHz se 2 GB operační paměti. Datový sklad i provozní databáze byly nainstalovány na jednom PC. Největší režii při plnění datového skladu má dohledávání záznamů dimenzí pro tabulky faktů. Doba přesunu z provozní databáze do odkládací oblasti je zanedbatelná vzhledem k době, po kterou jsou vykonávány transformační procedury. Toto je způsobené i tím, že je provozní databáze a datový sklad nainstalována na jednom pc. Při návrhu ETL procesu bylo snahou minimalizovat množství přenášených dat po síti. Rychlost přenosu by se proto neměla příliš lišit pro případ, kdy bude datový sklad umístěn na samostatném serveru. Aby bylo toto splněno, musí být samozřejmě dostatečně rychlé propojení mezi servery, na nichž je umístěn DS a provozní databáze.

Rychlost ETL procesu byla měřena pro databázi o velikosti 1,08 GB, přičemž tabulka ve zdrojové databázi, v níž jsou dohledávány záznamy pro tabulku faktů *POK_DOKLADY,* obsahovala 5500 záznamů. Tabulky, z kterých jsou plněny dimenze datového skladu, obsahovaly od jednotek do tisíců záznamů. Doba prvotního plnění datového skladu z této provozní databáze činila 13 minut a 50 sekund, z toho doba přenosu dat mezi provozní databází a datovým skladem byla přibližně 45 sekund. Následně bylo do provozní databáze přidáno 300 záznamů, a bylo provedené rozdílové plnění. Doba trvání rozdílového plnění byla 1 minuta 42 sekund. Po zdvojnásobení množství záznamů v tabulkách faktů, byla doba plnění DS 24 minut 12 sekund. Z toho doba přesunu do odkládací oblasti zabrala 1 minutu 11 sekund. Doba rozdílového plnění po přidání šesti set záznamů zabrala 2 minuty 40 sekund. Doby trvání jednotlivých plnění jsou uvedeny v tabulce 9.1 a 9.2. Z uvedených časů je vidět, že rozdílové plnění trvá výrazně kratší dobu. Po implementaci ostatních tabulek faktů, by neměla pravidelná rozdílová plnění trvat déle než desítky minut. Testované množství záznamů pro rozdílové plnění může být samozřejmě různé dle zákazníka. Množství změněných záznamů bylo přibližně stanoveno jako množství záznamů, které zákazník provede denně nad tabulkou, ze které se plní jedna tabulka faktů.

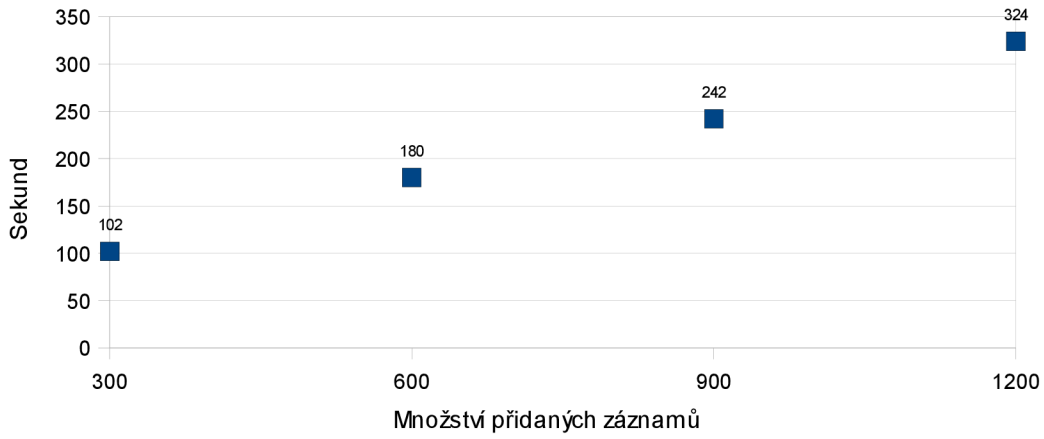
Množství přidaných záznamů do tabulek faktů	Doba plnění celého skladu
5500	13 minut 50 sekund
12000	24 minut 12 sekund
18000	40 minut 49 sekund
22000	55 minut 14 sekund

Tabulka 9.1: Doby počátečního plnění DS pomocí ETL procesu

Množství přidaných záznamů do tabulek faktů	Doba plnění celého skladu
300	1 minuta 42 sekund
600	2 minuty 40 sekund
900	4 minuty 2 sekundy
1200	5 minut 24 sekund

Tabulka 9.2: Doby rozdílového plnění DS pomocí ETL procesu

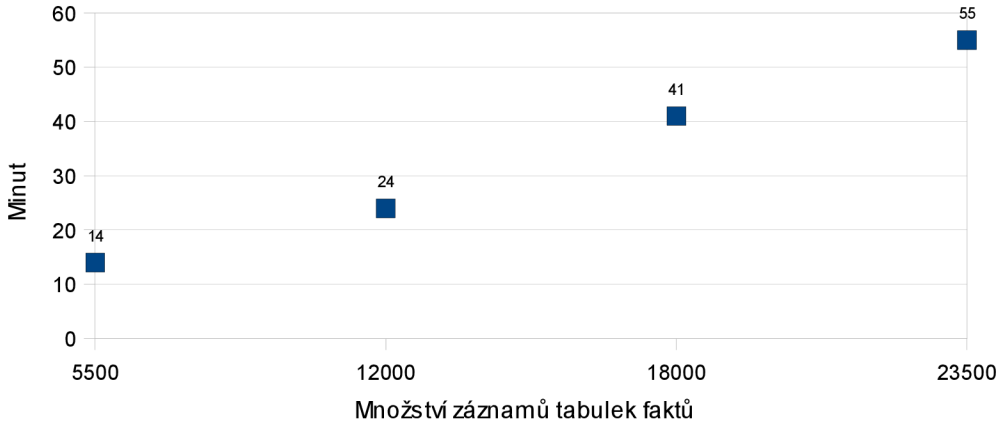
Jednou ze zkoumaných variant plnění datového skladu, bylo provedení transformací v nástroji PDI. Toto řešení, bylo výrazně pomalejší, oproti verzi, kdy transformace probíhají na serveru. Na stejné databázi zde trvalo pouze plnění dimenzí bez tabulek faktů zhruba 35 minut. Další nevýhodu tohoto přístupu byla nemožnost jednoduše definovat pole, pro která chceme udržovat historii. Proto došlo k implementaci transformačních procedur v jazyku PSQL přímo na straně databázového serveru. Graf 9.1 prezentuje dobu trvání rozdílového plnění DS. Z grafu je patrný lineární nárůst doby plnění.



Graf 9.1: Doba trvání rozdílového plnění DS

Graf 9.2 ukazuje dobu trvání počátečního plnění DS, z grafu je opět patrný lineární nárůst doby potřebné k naplnění datového skladu.

Je nutné brát v potaz, že do datového skladu budou přidány další tabulky faktů a doba plnění tomu bude úměrně narůstat. Samotný datový sklad by proto měl být umístěn na výkonnějším stroji, než byl použit k testování.



Graf 9.2: Doba trvání počátečního plnění DS

9.4 OLAP analýza nad datovým skladem

Nad implementovaným datovým skladem byla provedena OLAP analýza, prezentující jeho možnosti. K jejímu provedení byla vytvořena definice datové kostky, která je nutná pro správnou funkci analytického serveru Mondrian. Na ilustraci 7.2 je prezentován vztah této definice datové kostky k analytickému serveru a nástroji JPivot, ve kterém byly výsledky OLAP analýzy prezentovány. Samotná definice datové kostky je napsána v XML a obsahuje definice dimenzí připojených k tabulkám faktů. Pro dimenze jsou zde definovány hierarchie, které jednotlivé dimenze obsahují a míry tabulek faktů.

Mondrian server byl provozován na aplikačním serveru Apache Tomcat 7 a k prezentaci dat byl použit již zmíněný nástroj JPivot. Více informací o jeho zprovoznění lze nalézt v [3]. Dotazy na OLAP server byly kladeny v jazyku MDX. Jednotlivé dotazy se aplikačnímu serveru předávají

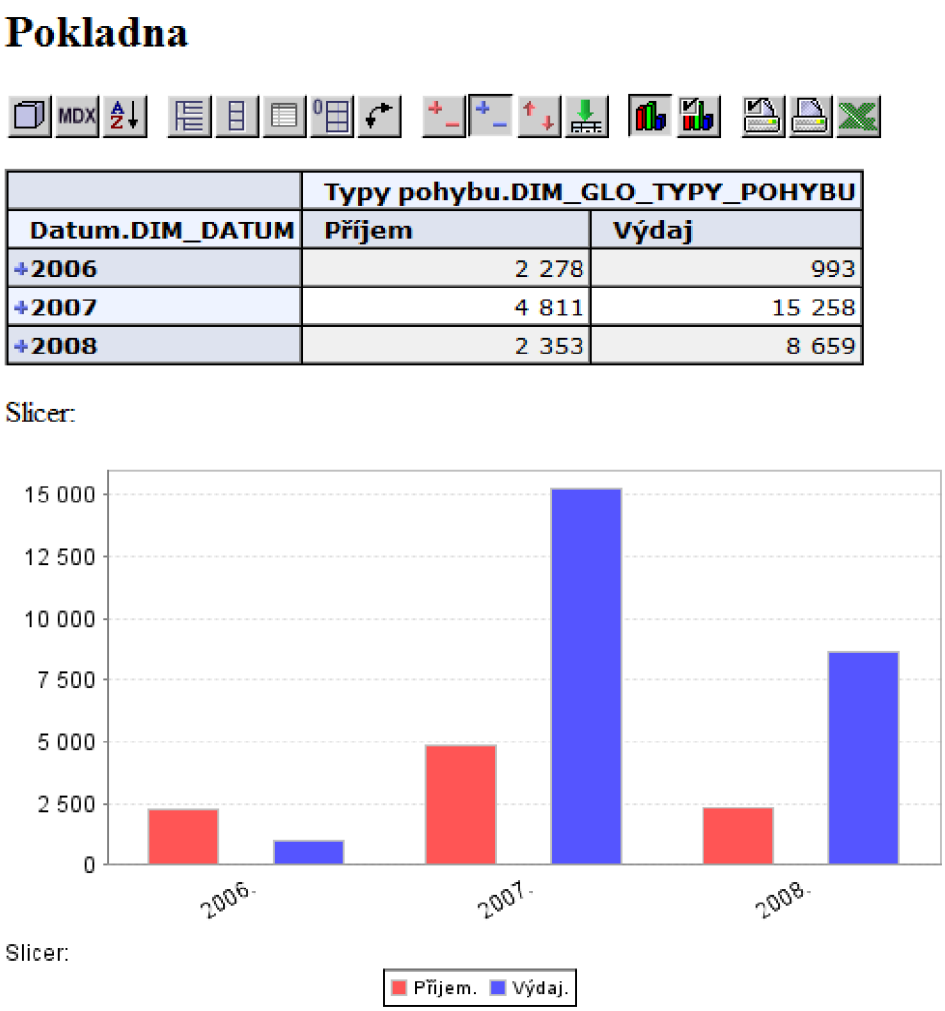
```
<%@ page session="true" contentType="text/html; charset=WINDOWS-1250" %>
<%@ taglib uri="http://www.tonbeller.com/jpivot" prefix="jp" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core" %>

<jp:mondrianQuery id="query01" jdbcDriver="org.firebirdsql.jdbc.FBDriver"
jdbcUrl="jdbc:firebirdsql:localhost/3050:dw?lc_ctype=WIN1250" catalogUri="/WEB-
INF/queries/Tempo.xml"
   jdbcUser="sysdba" jdbcPassword="masterkey" connectionPooling="false">

select {[TYPY POHYBU].[Příjem], [TYPY POHYBU].[Výdaj]} ON COLUMNS,
  {[DATUM].[ROK].[2006],[DATUM].[ROK].[2007],[DATUM].[ROK].[2008]} ON ROWS
from [POK_DOKLADY]

</jp:mondrianQuery>

<c:set var="title01" scope="session">Pokladna</c:set>
```

Ukázka kódu 9.4: MondrianQuery realizující MDX dotaz

pomocí mondrianQuery. Ta je uložená v JSP servletu, jejíž obsah je uveden v ukázce kódu 9.4. Tento dotaz obsahuje specifikaci JDBC driveru, cestu k datovému skladu, definici datové kostky a MDX dotaz, který má analytický server zodpovědět.

Datový sklad byl pro provedení těchto dotazů naplněn testovacími daty. Problémem při dotazování analytického serveru bylo kódování dat. Tento problém byl vyřešen specifikací znakové sady v dotazu takto: jdbc:firebirdsql:localhost/3050:dw?lc_ctype=WIN1250.

Výsledkem tohoto dotazu nad implementovaným datovým skladem je zobrazen na ilustraci 9.4. Ta představuje příjmy a výdaje fiktivní společnosti za roky 2006, 2007,2008.



Ilustrace 9.4: Výsledek MDX dotazu nad datovým skladem

10 Závěr

V počátečních kapitolách diplomové práce byla popsána problematika datových skladů, zejména jednotlivé komponenty celého procesu datového skladování a architektury datového skladu. Následně byly vysvětleny principy ETL procesu a podrobně popsány jeho jednotlivé fáze. Celá teoretická část diplomové práce byla psána tak, aby i laik se zájmem o datové sklady problematice porozuměl.

Cílem práce bylo navrhnout a implementovat datový sklad se zaměřením na proces ETL, který slouží k plnění implementovaného datového skladu. Datový sklad byl navržen pro informační systém Tempo společnosti Systemart s.r.o. tak, aby mohl být zákazníkovi dodáván jak s datovým skladem, tak i bez něj. Při koupi informačního systému s datovým skladem zákazník obdrží jeho základní funkcionalitu, což je archivace, možnost odlehčení provozní databáze a základní možnosti OLAP analýzy. V případě dalších požadavků může být datový sklad optimalizován na míru, což znamená zavedení agregačních tabulek pro nejčastěji používané dotazy, přidání specifických tabulek faktů a jiné. Dále byl implementován funkční prototyp, který musí být doplněn o zbývající datové trhy, dle vzoru. Proces ETL byl navržen tak, aby veškeré transformace probíhaly na úrovni databáze, zejména při přesunu dat z odkládací oblasti do datového skladu. Toto řešení přináší dobrý výkon a snadnou modifikovatelnost. ETL proces je dále navržen tak, aby správně reagoval na mazání dat ve zdrojové databázi a tyto změny se projevovaly v datovém skladu dle jeho nastavení. Datový sklad tedy dokáže rozlišovat mezi záznamy smazanými kvůli chybě zadávajícího a záznamy smazanými kvůli odlehčení provozní databáze. Tímto je splněn požadavek na archivaci záznamů. Rychlost procesu ETL byla ověřena měřeními, z kterých vyplývá předpokládaná doba počátečního plnění a rozdílového plnění. Pro rozsáhlé zdrojové databáze může být doba počátečního plnění i v řádu hodin. Rozdílové plnění datového skladu pro databáze, kde počet změn nad jednotlivými tabulkami nepřesáhne řády tisíců, pak v desítkách minut. Nad datovým skladem byla provedena jednoduchá OLAP analýzu, což již nebylo součástí zadání, ale posloužilo to k prezentaci funkčnosti samotného datového skladu.

V budoucnu se dokončí zbývající tabulky faktů a řádně se odladí funkčnost celého datového skladu, aby mohl být prodáván s informačním systémem. Dále je vhodné dokončit nástroj starající se o automatické mazání záznamů ze zdrojové databáze, čímž bude možné automatizovat proces archivace. Ten by měl probíhat přibližně jednou za dva roky, případně častěji v závislosti na rychlosti plnění provozního informačního systému zákazníkem. Tato automatická archivace musí být otestována ještě důkladněji jako samotný datový sklad a předpokládám, že na tuto činnost bude i tak nutné dohlížet. Případné chyby, vzniklé při tomto procesu, by byly pravděpodobně kritické a přinesly by společnosti mnoho problémů. Dále vidím jako vhodné do informačního systému implementovat modul, který bude data z datového skladu prezentovat prostřednictvím aplikace v pokročilejším uživatelském rozhraní. Nynější možnost prezentovat obsah datového skladu pomocí webové aplikace, vidím jako dostatečný např. pro mobilní zařízení, což jistě bude ceněné. Poslední rozšíření či úpravu vidím v možnosti nabízet datový sklad jako službu, což by také někteří zákazníci mohli ocenit.

Literatura

[1] HAYES, F. *Computerword* [online]. 2002 [cit. 2011-01-15]. The Story So Far. Dostupné z WWW: <http://www.computerworld.com/s/article/70102/The_Story_So_Far?taxonomyId=009>.

[2] Hlavní principy datových skladů a proces jejich vytváření. SystemOnline [online]. 2000, 11, [cit. 2011-01-22]. Dostupný z WWW: <http://www.systemonline.cz/clanky/hlavni-principy-datovych-skladu-a-proces-jejich-vytvareni.htm>. ISSN 1802-615X.

[3] JPivot. *JPivot* [online]. 2003 [cit. 2011-05-1]. JPivot . Dostupné z WWW: <http://jpivot.sourceforge.net/>.

[4] PODCAMENI, S, et al. *Data Warehousing with DB2 for OS/390* [online]. [s.l.] : Redbooks, 1997 [cit. 2011-05-11]. Dostupné z WWW: <http://www.redbooks.ibm.com/abstracts/sg242249.html>. ISBN 0738402486.

[5] BALLARD, C, et al. *Data Modeling Techniques for Data Warehousing* [online]. [s.l.] : Redbooks, 1998 [cit. 2011-01-20]. Dostupné z WWW: <http://www.redbooks.ibm.com/abstracts/sg242238.html?Open>. ISBN 0738402451.

[6] PONNIAH, P. *Data Warehousing Fundamentals: A Comprehensive Guide for IT Professionals*. [s.l.] : John Wiley & Sons, 2001. 544 s. ISBN 978-0-471-41254-0.

[7] BOUMAN, R; DONGEN, J. *Pentaho Solutions*. Indianopolis : WileyPublishing, 2009. 604 s. ISBN 978-0-470-48432-6.

[8] KUČERA, M. *SystemOnline* [online]. 2001 [cit. 2011-02-9]. Dva způsoby budování datového skladu. Dostupné z WWW: <http://www.systemonline.cz/clanky/dva-zpusoby-budovani-datoveho-skladu.htm>.

[9] GOLFARELLI, M; RIZZI, S. *Data Warehouse Design: Modern Priniples and Metodologies*. McGraw-Hill : Osborne Media, 2009. 480 s. ISBN 978-0071610391.

[10] KIMBALL, R, et al. *The Data warehouse Lifecycle Toolkit*. [s.l.] : WileyPublishing, 1998. 800 s. ISBN 978-0471255475.

[11] DANIEL, R. *Homel.vsb.cz* [online]. 2010 [cit. 2011-05-11]. Co je to OLAP a kdy se používá?. Dostupné z WWW: <http://homel.vsb.cz/~dan11/is_skripta/IS%202010%20-%20Danel%20-%20OLAP.pdf>.

[12] LACKO, L. *Datové sklady analýza OLAP a dolování dat*. Brno : Computer Press, 2003. 486 s. ISBN 80-7226-969-0.

[13] POE, V. *Building a data warehouse for decision support*. New Jersey : Prentice-Hall, 1996. 285 s. ISBN 0-13-371121-8.

[14] CÍSAŘ, P. *InterBase/FireBird Tvorba, administrace a programování databází*. Brno : Computer Press, 2003. 464 s. ISBN 80-7226-946-1.

[15] Lašek, I. *Integrace BI open source nástrojů*. Praha, 2010. 87 s. Diplomová práce. České vysoké učení technické.

[16] Pentaho. *Pentaho* [online]. 2002 [cit. 2011-04-20]. Mondrian Documentation. Dostupné z WWW: <http://mondrian.pentaho.com/documentation/>.

[17] Pentaho. *Pentaho* [online]. 2002 [cit. 2011-04-25]. Community Wiki Home. Dostupné z WWW: <http://mondrian.pentaho.com/documentation/>.

Seznam příloh